UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DANUZA PRADO DE FARIA ALCKMIN

Algoritmo Genético Híbrido para resolver o problema de agrupamento de dados

Vitória - ES, Brasil

31 de agosto de 2009

Danuza Prado de Faria Alckmin

Algoritmo Genético Híbrido para resolver o problema de agrupamento de dados

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo para obtenção do título de Mestre em Informática.

Orientador:

Flávio Miguel Varejão

Programa de Pós-Graduação em Informática
Deparamento de Informática
Centro Tecnológico
Universidade Federal do Espírito Santo

Vitória – ES, Brail

31 de agosto de 2009

Dissertação de Mestrado sob o título "Algoritmo Genético Híbrido para resolver o problema de agrupamento de dados", defendida por Danuza Prado de Faria Alckmin e aprovada em 31 de agosto de 2009, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos doutores:

_____

Prof. Flávio Miguel Varejão, D. Sc.
Orientador

_____

Profª. Maria Cláudia Silva Boeres, D. Sc.
Membro Interno

_____

Profª. Simone de Lima Martins, D. Sc.
Membro Externo

“A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.”

(Albert Einstein)

Dedico este trabalho aos meus pais.
Que todo o meu esforço possa retribuir a dedicação
e a confiança que eles sempre depositaram em mim

Agradecimentos

Agradreço a Deus, acima de tudo.

Agradeço ao meu filho e ao meu marido, pela compreensão e incentivo durante a realização deste curso. Pelos dias, horas e minutos nos quais minha atenção, que deveria ser de vocês, foi dispensada à responsabilidade de elaborar este trabalho.

Agradeço a meu orientador pelo apoio, incentivo, sabedoria, compreensão, empenho e, prinicpalmente, pela confiança em mim depositada.

Aos membros da banca examinadora pela disposição em analisar este trabalho.

Aos professores do Departamento de Informática da Universidade Federal do Espírito Santo pela oportunidade de crescimento e aprendizado.

Aos colegas de curso que compartilharam as dificuldades e as conquistas na realização deste curso e, principalmente, pela amizade que construímos.

Aos colegas de trabalho, em especial àqueles que sempre se disponibilizaram a acertos necessários de horário para que eu pudesse cumprir com todas as minhas obrigações profissionais e acadêmicas.

A todos os meus amigos e amigas que, direta ou indiretamente, sempre estiveram presentes me aconselhando e incentivando.

Resumo

Agrupamentos de dados é uma tarefa que divide um conjunto de dados em subconjuntos de forma que elementos associados a um mesmo grupo sejam mais similares entre si do que em relação a elementos de outros grupos. Ao organizar os dados em grupos é possível identificar similaridades e diferenças entre eles, extrair informações relevantes e inferir conclusões úteis a respeito das características dos dados. O problema de agrupamento de dados pode ser considerado como uma tarefa de otimização, uma vez que se pretende encontrar a melhor combinação de partições dentre todas as combinações possíveis. Uma abordagem que pode ser aplicada para resolver o problema de agrupamento é o uso de metaheurísticas, que são procedimentos capazes de escapar de ótimos locais, pois o uso de métodos exatos se torna computacionalmente inviável. Entretanto, a maioria das metaheurísticas aplicadas ao problema de agrupamento não são escalonáveis para bases reais e comerciais, são mais efetivas nos casos em que a instância do problema é menor. O custo computacional necessário para calcular as soluções se torna maior em instâncias maiores do problema. Por esse motivo, procedimentos híbridos que exploram a combinação de metaheurísticas representam uma abordagem promissora para a resolução do problema de agrupamento. Este trabalho apresenta uma proposta de Algoritmo Genético Híbrido de Agrupamento que associa ao processo de busca global uma heurística de busca local e cuja população inicial é gerada por técnicas de agrupamento. Tais melhorias têm como objetivo direcionar a busca para soluções mais próximas do ótimo global. É realizada uma avaliação experimental em bases de dados reais e sintéticas com o objetivo de verificar se a abordagem proposta apresenta uma melhoria em relação aos algoritmos avaliados. O resultado dessa análise mostra que o algoritmo proposto apresenta um desempenho melhor do que quatro entre os seis algoritmos avaliados. Para complementar a análise é realizada uma avaliação do tempo de execução, cujo objetivo é quantificar a diferença entre a abordagem proposta e os demais algoritmos avaliados. O resultado mostra que o tempo de execução da abordagem proposta é viável, porém é consideravelmente maior do que os tempos de execução dos algoritmos considerados de rápida convergência.

Abstract

Clustering is a task that divides a data set in subgroups such that elements associated to one exactly group are more similar between itself than elements of other groups. Organizing data in groups make it possible to identify similarities and differences between them, to extract useful information and conclusions regarding the data features. Clustering may be considered an optimization problem because it is intended to find the best combination of partitions among all the possible combinations. An approach that can be applied to solve the clustering problem is the use of metaheuristics, which are procedures capable of escaping from local optima, once the use of exact methods is computationally impracticable. However, the majority of the metaheurísticas applied to clustering problem is not scalable for real or commercial bases. They are more effective in the cases where the instance of the problem is small. The computational cost necessary to calculate the solutions becomes greater in larger instances of the problem. For this reason, hybrid procedures that explore the combination of metaheuristics represent a promising approach for the clustering problem solving. This work shows a proposal of Hybrid Genetic Clustering Algorithm that associates the process of global search to a local search heuristic and also initializes the population by different grouping techniques. Such improvements aims to direct the search for solutions next to the global optimal one. An experimental evaluation in real and synthetic databases is performed aiming to verify if the proposed approach presents an improvement in relation to the other evaluated algorithms. The result of this analysis shows that the proposed algorithm presents a better performance in four among the six evaluated algorithms. In addition, an analysis of the execution time shows that the execution time of our proposal is viable, even though it is considerably longer than the execution times of the fast convergence algorithms.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Agrupamento de dados consiste na divisão de um conjunto de padrões em grupos cujos elementos sejam mais similares entre si do que em relação a elementos de outros grupos.

Segundo (XU, 2005), dado um conjunto de dados, $j = 1,..., N$, algoritmos de agrupamento tem como objetivo organizar os dados em K partições $\{C_1,..., C_K\}$ que otimize alguma função de custo. Essa abordagem requer a definição de uma função que associa um custo a cada grupo. O objetivo é encontrar o conjunto de partições que otimize a soma dos custos de todos os grupos. Portanto, o problema pertence à classe de complexidade *NP-hard*.

Uma abordagem promissora que pode ser aplicada para resolver o problema de agrupamento é o uso de procedimentos híbridos que exploram a combinação de metaheurísticas.

(PERIM, 2008) apresenta uma abordagem híbrida que usa métodos de inicialização combinados á metaheurística Simulated Annealing para resolver o problema de agrupamento. Tal combinação tem como objetivo aprimorar os resultados obtidos com a versão padrão da metaheurística (inicializada aleatoriamente) e com o algoritmo *K-means*. Os resultados apresentados pelo Simulated Annealing inicializado com o método *PCA_Part* (*Principal Component Analysis)*, método de divisão baseado na análise das componentes principais, são superiores aos resultados do *K-means* inicializado aleatoriamente.

O presente trabalho propõe um Algoritmo Genético Híbrido de Agrupamento que associa ao processo de busca global uma heurística de busca local e cuja população inicial é gerada por técnicas de agrupamento. Tais melhorias têm como objetivo direcionar a busca para soluções mais próximas do ótimo global. Por conseqüência, espera-se obter com essa abordagem resultados melhores do que os obtidos com as técnicas de agrupamento e heurísticas implementadas e avaliadas neste trabalho: Hierárquico Aglomerativo, *K-means* inicializado aleatoriamente*, K-means* inicializado pelo *K-means*++, *K-means* inicializado pelo *PCA_Part*, *Tabu Search* e Algoritmo Genético de Agrupamento.

Este trabalho apresenta uma análise experimental dos sete algoritmos adaptados para o problema de agrupamento citados. A análise é realizada em oito bases reais e quinze bases sintéticas. As bases reais são divididas em bases de treino, com 10% do total de exemplos, e

bases de teste com 90% do total de exemplos. Tal divisão tem a finalidade de utilizar as bases de treino para ajustar os parâmetros dos algoritmos que usam esses valores. O resultado dessa análise mostra que o Algoritmo Genético Híbrido de Agrupamento proposto apresenta os melhores resultados e, estatisticamente, é considerado melhor do quatro entre os seis algoritmos avaliados.

Este trabalho está organizado em seis capítulos, sendo este o primeiro e os demais são como se segue. O Capítulo 2 apresenta uma definição formal do problema de agrupamento de dados, bem como os aspectos principais. Esse capítulo também apresenta algumas abordagens para solucionar o problema, com ênfase maior naquelas que abordam o problema de agrupamento como uma tarefa de otimização.

O Capítulo 3 apresenta quatro técnicas de agrupamento de dados: o algoritmo Hierárquico Aglomerativo, um método determinístico que organiza os dados em uma estrutura hierárquica, e três versões do *K-means*, heurística iterativa que objetiva associar os padrões à partição mais próxima. As três versões do *K-means* se diferem no método de inicialização. A primeira versão é inicializada aleatoriamente, a segunda é inicializada pelo *K-means++* e a terceira é inicializada pelo *PCA_Part*. Com as duas últimas versões espera-se contribuir para contornar o problema da sensibilidade à escolha da solução inicial do *K-means*, que pode convergir para um mínimo local caso a solução inicial não seja escolhida adequadamente. Outra contribuição importante é que métodos de inicialização ajudam a acelerar o processo de convergência do *K-means*.

O Capítulo 4 mostra as características gerais que diferenciam as metaheurísticas dos procedimentos heurísticos. Apresenta em detalhes o Algoritmo Genético adaptado ao problema de agrupamento de dados. Além disso, esse capítulo apresenta em detalhes o Algoritmo Genético Híbrido de Agrupamento proposto e avaliado no presente trabalho e, por fim, apresenta o algoritmo *Tabu Search* adaptado ao problema de agrupamento.

O Capítulo 5 mostra a avaliação experimental da proposta apresentada, comparando o desempenho dos resultados e dos tempos de execução dos diferentes algoritmos avaliados, em bases reais e sintéticas.

Por fim, o Capítulo 6 apresenta os principais resultados alcançados com a abordagem proposta e os possíveis trabalhos futuros.

2 *Agrupamento de Dados: Tarefa de Otimização*

Agrupamento é a classificação não-supervisionada de padrões em grupos ou clusters. O problema de agrupamento de dados ou clustering consiste na divisão de um conjunto de padrões em grupos cujos elementos sejam os mais similares entre si do que em relação a elementos de outros grupos. Neste trabalho, os padrões que formam o conjunto de dados também serão denominados por exemplos, elementos, pontos ou itens.

A partir dos agrupamentos é possível identificar e inferir conclusões relevantes sobre os dados. Segundo (FAYYAD, 1996), ao organizar padrões em grupos, é possível identificar similaridades e diferenças entre eles e inferir conclusões úteis a respeito das características dos dados.

Por serem capazes de agrupar dados similares as técnicas de agrupamentos são empregadas em diversas áreas (HARTIGAN, 1975), (EVERITT; LANDAU; LEESE, 2001). Dependendo da área de aplicação o conceito de agrupamento ou particionamento, pode ser empregado de várias formas.

Na mineração de dados o resultado do agrupamento tem como objetivo contribuir para a descoberta de informações úteis sobre os dados. A partir de semelhanças e diferenças entre os padrões é possível descobrir novas informações inerentes aos dados. Essa abordagem pode ser vista em (FAYYAD et al., 1996).

No aprendizado de máquina cada partição pode ser considerada uma classe, permitindo classificar novos padrões de acordo com a partição em que ele for inserido. (HASTIE; TIBSHIRANI; FRIEDMAN, 2003) empregam o agrupamento para classificar tipos de câncer. (GOLUB, 1999) usa o agrupamento para analisar tumores e descobrir automaticamente duas categorias de leucemias.

No reconhecimento de padrões, devido à similaridade dos objetos de cada partição, é possível categorizar diferentes padrões. Novos objetos, após associados a uma das partições, podem ser identificados pelas mesmas características dos padrões pertencentes a essa partição. (LEVRAT et al., 1992) utiliza algoritmos de particionamento de dados para segmentar imagens com o objetivo de detectar transições entre regiões.

Na análise estatística o agrupamento de dados permite elaborar estatísticas descritivas sobre o conjunto de dados e identificar, por exemplo, a distribuição amostral desses dados. (BANFIELD; RAFTERY, 1993) e (KAUFMAN; ROUSSEEUW, 1990) apresentam trabalhos nessa área.

Segundo (XU; WUNSCH, 2005), na teoria de grafos, os conceitos e as propriedades permitem descrever convenientemente problemas de agrupamentos. Os nós de um grafo correspondem aos dados, representados por pontos, e o peso de cada aresta reflete a similaridade entre cada par de pontos. (ASSUNÇÃO; FURTADO, 2008) apresentam um método heurístico para particionamento de grafos com o objetivo de melhorar a delimitação de áreas de patrulhamento policial.

2.1 Definição Formal e Aspectos Principais

Formalmente o problema de agrupamento de dados pode ser definido como: dado um conjunto de dados X com N padrões $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}$, onde cada padrão $x_i = [x_{i1}, x_{i2}, ..., x_{id}]^t$ possui d dimensões (características ou atributos), deseja-se encontrar K partições (*clusters*)$\{C_1, ..., C_K\}$, de forma que padrões pertencentes a uma mesma partição são mais similares entre si do que em relação aos padrões pertencentes a outras partições, bem como, cada partição deve atender às seguintes condições (XU; WUNSCH, 2005):

- $C_j \neq \varnothing$; $j = 1, ..., K$
- $\bigcup_{j=1}^{K} C_j = X$
- $C_i \cap C_j = \varnothing$; $i \neq j$; $i, j = 1,..., K$

A terceira condição citada acima, indicando que cada padrão pertence a uma única partição, é a que define os agrupamentos chamados *hard*. Entretanto, existe outra definição de agrupamento, onde um padrão pode estar associado a mais de uma partição de acordo com um coeficiente de pertinência. Este tipo de agrupamento, definido como *fuzzy clustering,* não faz parte do escopo deste trabalho. Neste trabalho será abordado o particionamento *hard*, utilizado em diversas áreas.

Outra maneira de particionar os dados é através do agrupamento hierárquico que tenta construir uma hierarquia de partições aninhadas $H = \{H_1,..., H_Q\}$ ($Q \leq N$) sobre X, tal que (XU; WUNSCH, 2005):

$$((C_i \in H_m) \wedge (C_j \in H_l) \wedge (m > l)) \Rightarrow (C_i \in C_j) \vee (C_i \cap C_j = \varnothing)$$
$$\forall\, i; j \neq i; m, l = 1,..., Q.$$

Na identificação das partições e de quais padrões pertencem a determinadas partições está um elemento fundamental no processo de agrupamento de dados, a informação de quão similares ou diferentes são os padrões entre si.

Um padrão é descrito por um conjunto de características, representado geralmente como um vetor multidimensional. As características podem ser quantitativas ou qualitativas, contínuas ou binárias, nominais ou ordinais, e determinam a escolha da medida de similaridade ou dissimilaridade (XU; WUNSCH, 2005). A medida de similaridade avalia a semelhança entre os padrões e a medida de dissimilaridade avalia a diferença entre os padrões. Dois padrões são considerados semelhantes ou próximos quando uma medida de similaridade entre eles for grande ou quando uma medida de dissimilaridade for pequena. Segundo (XU; WUNSCH, 2005), geralmente medidas de dissimilaridade são usadas para medir características contínuas, enquanto medidas de similaridade são mais importantes para características qualitativas.

A figura 2.1 apresenta um exemplo de agrupamentos do tipo *hard*. São representados sete pontos {A, B, C, D, E, F, G} agrupados em três partições. A partição 1 indica que os padrões {A, B, C} são mais similares entre si do que em relação aos demais, o mesmo ocorre com as partições {D, E} e {F, G}.

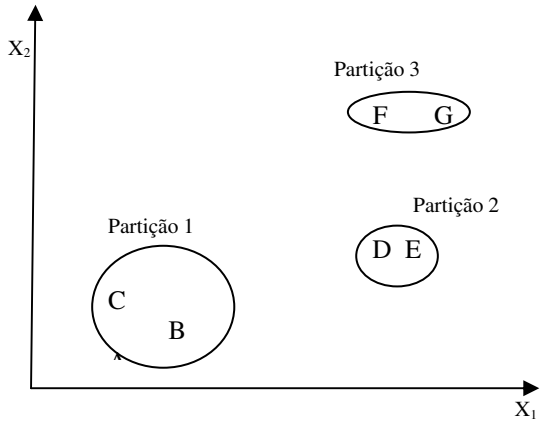

Figura 2.1 Sete pontos agrupados em três partições (JAIN; MURTY; FLYNN, 1999).

A figura 2.2 apresenta o mesmo exemplo com os setes pontos agrupados pelo algoritmo de agrupamento hierárquico aglomerativo representados em um estrutura conhecida como dendograma. O algoritmo inicialmente distribui os dados de maneira que cada exemplo represente um cluster e, então, esses clusters são recursivamente agrupados considerando alguma medida de dissimilaridade, até atingir o critério de parada. Dessa maneira, um agrupamento hierárquico agrupa os dados de modo que se dois exemplos são agrupados em algum nível, nos níveis superiores eles continuam fazendo parte do mesmo grupo, construindo uma hierarquia de clusters. O capítulo 3 apresenta maiores detalhes sobre agrupamento hierárquico.

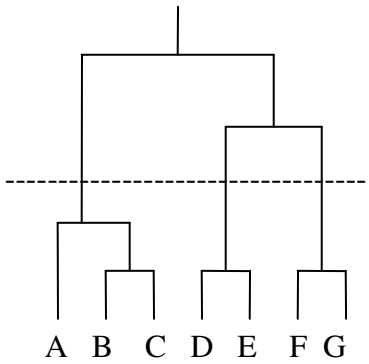

Figura 2.2 Sete pontos agrupados hierarquicamente representados por um dendograma. A linha tracejada representa o corte para se obter três partições.

Conforme mencionado anteriormente as medidas de dissimilaridade ou funções de distâncias são usadas para medir características contínuas.

Uma métrica muito utilizada, que é aplicada a padrões com atributos contínuos, é a distância de Minkowski, definida por (XU; WUNSCH, 2005):

$$D_{ij} = \left(\sum_{l=1}^{d} \left| x_{il} - x_{jl} \right|^{1/n} \right)^{n} \tag{2.1}$$

Outra métrica comumente utilizada, um caso especial da métrica de Minkowski quando $n = 2$, é a distância Euclidiana (XU; WUNSCH, 2005):

$$D_{ij} = \left(\sum_{l=1}^{d} \left| x_{il} - x_{jl} \right|^{1/2} \right)^{2} \tag{2.2}$$

Outros dois casos especiais da métrica de Minkowski são a distância de *City-blok* (XU; WUNSCH, 2005), também conhecida como distância de Manhattan, quando n = 1

$$D_{ij} = \sum_{l=1}^{d} \left| x_{il} - x_{jl} \right| \tag{2.3}$$

e a distância de Chebychev ou distância *Sup* (XU; WUNSCH, 2005), que calcula o máximo da diferença absoluta em coordenadas, obtida quando $n = \infty$:

$$D_{ij} = \max_{1<l<d} \left| x_{il} - x_{jl} \right| \tag{2.4}$$

Outro exemplo é a distância de Mahalanobis, definida por (XU; WUNSCH, 2005)

$$D_{ij} = (x_i - x_j)^T S^{-1} (x_i - x_j) \tag{2.5}$$

onde *S* é a matriz de covariância.

Esta medida associa diferentes pesos a diferentes características com base em suas variâncias e a correlação linear entre pares de padrões (JAIN; MURTY; FLYNN, 1999). Quando as características não são correlacionadas, a distância de Mahalanobis quadrada é equivalente à distância Euclidiana quadrada (XU; WUNSCH, 2005).

As medidas mais comuns para conjuntos de dados cujos atributos são contínuos são as distâncias baseadas na métrica de Minkowski, como a distância Euclidiana, a distância de Manhattan e a distância *Sup* ou *Supremum*.

Medidas de dissimilaridade diferentes aplicadas sobre o mesmo conjunto de dados podem conduzir a resultados diferentes. Por isso, a escolha de uma métrica adequada para determinado tipo de problema torna-se uma tarefa muito importante no processo de agrupamento. Não existe uma resposta exata sobre a escolha da métrica ou medida de dissimilaridade adequada, portanto sugere-se que a escolha seja guiada pelo tipo e escala dos atributos dos padrões, levando-se em conta a intuição e experiência do investigador.

2.2 Métodos de Resolução para o Problema de Agrupamento

Muitos problemas práticos são modelados da seguinte forma: dado um conjunto S de variáveis discretas s (soluções) e uma função objetivo (ou função avaliação) f : S $\leftarrow$ R, que associa cada solução s $\in$ S a um valor real f (s), encontre a solução s'$\in$ S, considerada ótima, para a qual f (s) é mínima.

Muitos desses problemas são classificados na literatura como *NP-hard*, isto é, são problemas para os quais não existem algoritmos que os resolvam em tempo polinomial. Tais problemas são classificados como problemas de otimização combinatória.

Segundo (XU; WUNSCH, 2005), dado um conjunto de dados $x_j \in \Re^d$, $j = 1, ..., N$, algoritmos de agrupamento tem como objetivo organizar os dados em K partições $\{C_1,...,C_K\}$ que otimize alguma função de custo. O número de possibilidades de particionar N padrões em K grupos é dada pela fórmula:

$$P(N,K) = \frac{1}{K!}\sum_{m=1}^{k}(-1)^{k-m}C_k^m m^N \tag{2.6}$$

Mesmo para N e K pequenos, o custo computacional é alto e para instâncias maiores o problema torna-se computacionalmente inviável. Portanto, algoritmos que realizam busca por soluções aproximadas são comumente utilizados pra resolver problemas de agrupamento e apresentam soluções satisfatórias se comparadas ao alto custo computacional da resolução baseada na enumeração.

Alguns algoritmos de agrupamento, também chamados de particionais, dividem um conjunto de dados X em um número apropriado de subconjuntos K com objetivo de otimizar um critério de avaliação. Em muitos algoritmos de agrupamento o valor de K é fornecido como parâmetro de entrada, mas em outros casos é necessário escolher o valor adequado para esse parâmetro. A escolha adequada depende intrinsecamente do problema a ser resolvido. Segundo (XU; WUNSCH, 2005), uma divisão com muitos subconjuntos complicam o resultado, dificultando interpretação e análise. Uma divisão com poucos subconjuntos pode provocar a perda de informação e mascarar o resultado final. Alguns exemplos a respeito da escolha dos valores adequados para o número de subconjuntos K são mostrados em (XU; WUNSCH, 2005).

Conforme mencionado anteriormente o objetivo dos algoritmos de agrupamentos é otimizar um critério de avaliação (ou função de custo) para K partições $\{C_1,..., C_K\}$. O critério mais comumente usado é a soma das distâncias Euclidianas quadradas (*Sum of the Squared Euclidian distances*) que utiliza como medida de dissimilaridade entre dois padrões a distância Euclidiana (EVERITT; LANDAU; LEESE, 2001). O critério *SSE* é definido por

$$SSE = \sum_{j=1}^{K}\sum_{x_i \in C_j}\left\|x_i - \mu_j\right\|^2 \tag{2.7}$$

onde $\left\|x_i - \mu_j\right\|$ é a distância Euclidiana entre o padrão x_i e o centróide μ_j.

O centróide $\mu_j = [\mu_{j1}, \mu_{j2},..., \mu_{jd}]^t$ é o ponto representativo da partição C_j e é calculado como centro de massa da partição

$$\mu_j = \frac{1}{n_j} \sum_{x_i \in C_j} x_i \tag{2.8}$$

onde n_j é o total de padrões que pertencem à partição C_j.

Outro critério de avaliação é a soma das distâncias entre padrões da mesma partição (*Within cluster point scatter*) (HASTIE; TIBSHIRANI; FRIEDMAN, 2003). Esse critério é definido por:

$$W = \frac{1}{2} \sum_{j=1}^{K} \sum_{x_i \in C_j} \sum_{x_i' \in C_j} \delta\left(x_i, x_i'\right) \tag{2.9}$$

Um outro critério, que mede a dispersão dos dados, é calculado pela soma das distâncias entre padrões pertencentes a partições diferentes (*Between cluster point scarret*) (HASTIE; TIBSHIRANI; FRIEDMAN, 2003). Esse critério é definido por:

$$B = \frac{1}{2} \sum_{j=1}^{K} \sum_{x_i \in C_j} \sum_{x_i' \notin C_j} \delta\left(x_i, x_i'\right) \tag{2.10}$$

Os critérios *SSE* e *W*, por indicarem a coesão entre os padrões pertencentes a mesma partição, devem ser minimizados. O critério *B*, por indicar a dispersão entre os padrões de partições diferentes, deve ser maximizado. (EVERITT; LANDAU; LEESE, 2001) apresentam uma revisão mais detalhada sobre critérios de avaliação.

2.2.1 Abordagens de Agrupamento

Os algoritmos de agrupamento podem ser classificados por meio de diferentes aspectos. Uma classificação bastante utilizada é apresentada por (JAIN; MURTY; FLYNN, 1999), onde os algoritmos são classificados de acordo com o método adotado para definir os clusters.

Um algoritmo particional comumente utilizado para resolver o problema de agrupamento é o *K-means* (FORGY, 1965) que realiza uma busca heurística para determinar o ponto representativo de cada partição. O ponto não precisa ser necessariamente um ponto do conjunto de dados e é calculado como o centro de massa da partição (centróide). Após o cálculo dos pontos representativos os padrões são associados à partição de maior similaridade (ou menor dissimilaridade) com o ponto representativo. Os pontos representativos são então recalculados e o processo se repete iterativamente até atingir um critério de parada. No

entanto, o algoritmo *K-means* pode convergir para um mínimo local (SELIM; ISMAIL, 1984).

Outra abordagem de algoritmo particional é a baseada em grafos. Os padrões podem ser representados pelos vértices de um grafo e as distâncias entre dois padrões pelos pesos das arestas. O método mais conhecido para essa abordagem é baseado na construção da árvore geradora mínima do grafo (ZAHN, 1971).

Outro método de agrupamento é o hierárquico, procedimento determinístico que agrupa os dados em uma seqüência de partições aninhadas. Seja uma seqüência de partições de *N* amostras em *K clusters*, em que o nível 1 corresponde a *N clusters* de um elemento e o nível *N* corresponde a um cluster com todos os elementos. Um agrupamento hierárquico (DUDA; HART; STORK, 2001) agrupa os dados de forma que se dois exemplos são agrupados em algum nível, nos níveis mais altos eles continuam fazendo parte do mesmo grupo, construindo uma hierarquia de *clusters*. O agrupamento hierárquico pode ser dividido em duas abordagens: a aglomerativa, que começa com *N clusters* com um único exemplo e forma a seqüência de partições agrupando os clusters sucessivamente e a divisiva, que começa com um cluster com todos os exemplos e forma a seqüência dividindo os clusters sucessivamente. O capítulo 3 apresenta maiores detalhes sobre o agrupamento hierárquico.

Problemas de agrupamento de dados apresentam complexidade combinatorial exponencial e características *NP-hard* e (RARDIN; UZSOY, 2001) sugere a utilização de procedimentos aproximados ou heurísticos para resolvê-los. Heurística é a técnica que procura boas soluções (próximas da solução ótima) a um custo computacional razoável, sem, no entanto, garantir a solução ótima, bem como garantir quão próximo uma determinada solução está de uma solução ótima. Esse procedimento geralmente encontra soluções aproximadas que correspondem a ótimos locais. Para escapar de ótimo locais, métodos mais complexos da busca (por exemplo, Algoritmos Evolucionários, o *Simulated Annealing* e o *Tabu Search*) podem explorar o espaço de solução de forma mais flexível e eficiente (XU; WUNSCH, 2005). Esses métodos são metaheurísticas que podem ser aplicadas ao problema de agrupamento de dados para realizar a busca pela solução ótima (ou subótima) do critério de avaliação. Maiores detalhes sobre metaheurísticas são apresentados no capítulo 4.

Uma metaheurística inspirada no processo de evolução natural são os Algoritmos Genéticos (AGs) que otimizam a estrutura de uma população usando um conjunto de operadores genéticos. O princípio básico consiste em selecionar bons indivíduos (soluções do problema) para reprodução e recombiná-los com o objetivo de obter soluções melhores que das populações anteriores.

Na sua forma mais simples um AG é um processo iterativo que aplica uma série de operadores genéticos como seleção, *crossover* (recombinação) e mutação aos elementos de uma população. Esses elementos, chamados cromossomos ou indivíduos, representam possíveis soluções para o problema. Os cromossomos iniciais são selecionados randomicamente de um espaço de solução. Operadores genéticos combinam a informação genética dos elementos para formar uma nova geração da população, este processo é conhecido como reprodução. Cada cromossomo tem um valor *fitness* associado que representa a qualidade da solução para o problema. Um cromossomo que representa a melhor solução deve ter o valor *fitness* mais alto. O cromossomo compete para reproduzir de acordo com o valor *fitness*, assim cromossomos que representam soluções melhores têm uma alta chance de sobrevivência (COLE, 1998). O capítulo 4 apresenta maiores detalhes sobre o Algoritmo Genético.

Segundo (RAYWARD-SMITH, 2005), a maioria das metaheurísticas aplicadas ao problema de agrupamento não são escalonáveis para bases reais e comerciais, são mais efetivas nos casos em que a instância do problema é menor. O custo computacional necessário para calcular as soluções se torna maior em instâncias maiores do problema.

Por esse motivo, procedimentos híbridos que exploram a combinação de meta-heurísticas representam uma abordagem promissora para a resolução do problema de agrupamento.

(PERIM, 2008) apresenta uma abordagem híbrida que usa métodos de inicialização combinados ao *Simulated Annealing* para resolver o problema de agrupamento. Nessa abordagem os resultados apresentados pelo *Simulated Annealing* inicializado com o método *PCA_Part* (*Principal Component Analysis*), método de divisão baseado na análise das componentes principais, são superiores aos resultados do *K-means* inicializado aleatoriamente.

(BABU; MURTY, 1993) aplicam algoritmos genéticos para selecionar a solução inicial para o *K-means*, abordagem que supera os resultados da aplicação direta do Algoritmo Genético.

Este trabalho realiza um estudo mais aprofundado sobre o Algoritmo Genético e metaheurísticas aplicadas ao problema de agrupamento de dados. Propõe um Algoritmo Genético Híbrido que associa ao processo de busca global uma heurística de busca local e cuja população inicial é gerada por métodos de agrupamento, bem como realiza uma análise experimental dos algoritmos avaliados.

3 *Técnicas de Agrupamento*

Este capítulo descreve quatro técnicas de agrupamento de dados: o algoritmo Hierárquico Aglomerativo e três variações do *K-means* que se diferem no método de inicialização (aleatório, *PCA_Part* e *K-means++*). As três variações do *K-means* são utilizadas como métodos de inicialização gerando a população inicial utilizada pelo Algoritmo Genético Híbrido de Agrupamento proposto, com o objetivo de aprimorar o resultado do mesmo.

3.1 Algoritmo Hierárquico Aglomerativo

O agrupamento hierárquico é um método determinístico que organiza os dados em uma estrutura hierárquica de acordo com alguma medida de similaridade (ou dissimilaridade) entre eles. Um agrupamento hierárquico agrupa os dados de forma que se dois exemplos são agrupados em algum nível, nos níveis mais altos eles continuam fazendo parte do mesmo grupo, construindo uma hierarquia de clusters (Duda, 2001).

O agrupamento hierárquico pode ser dividido em duas abordagens: a aglomerativa, que começa com n clusters com um único exemplo e forma a seqüência de partições agrupando os clusters sucessivamente e a divisiva, que começa com um cluster com todos os exemplos e forma a seqüência dividindo os clusters sucessivamente. Entretanto, a abordagem divisiva não é comumente usada na prática e não será utilizada neste trabalho. Dois algoritmos hierárquicos divisivos, chamados MONA e DIANA são descritos em (KAUFMAN; ROUSSEEUW, 1990).

No agrupamento Hierárquico Aglomerativo as soluções são geralmente representadas por uma árvore binária ou um dendograma, que consiste em um tipo especial de estrutura de árvore, conforme pode ser visto na figura 3.1. A raiz do dendograma representa todo o conjunto de dados e cada folha representa um objeto. Os nós intermediários descrevem o grau de proximidade dos objetos, a altura do dendograma usualmente descreve a distância entre cada par de objetos ou grupos, ou um objeto e um grupo.

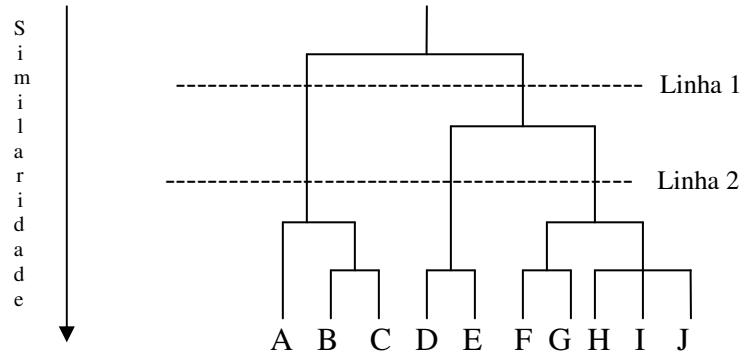

Figura 3.1 Dendograma resultante do agrupamento Hierárquico Aglomerativo. A linha 1 representa o corte para se obter duas partições e a linha 2 três partições.

As partições resultantes podem ser obtidas através de cortes no dendograma em diferentes níveis. Essa representação oferece descrições e visualizações informativas sobre as possíveis partições dos dados, especialmente quando existe uma relação hierárquica entre os dados (XU; WUNSCH, 2005).

Segundo (XU; WUNSCH, 2005), geralmente o método hierárquico aglomerativo pode ser representado pelo seguinte procedimento:

1. Começar com *N clusters*, onde *N* é o número de padrões a serem agrupados, com um único padrão em cada *cluster*.
2. Calcular a distância mínima:

$$D(C_i, C_j) = \min_{\substack{1 \le m, l \le N \\ m \ne l}} D(C_m, C_l) \tag{3.1}$$

onde $D(C_i, C_j)$ é a função de distância, na matriz de proximidade, e combinam *cluster* C_i e C_j para formarem um novo cluster.

3. Alterar a matriz de proximidade para calcular a distância entre o novo cluster e os outros clusters.
4. Repetir passos 2 e 3 até todos os padrões pertencerem ao mesmo *cluster*.

Baseados em definições diferentes para calcular a distância entre dois *clusters*, existem vários algoritmos de agrupamento hierárquico. Os mais simples e populares incluem os métodos *single linkage* e *complete linkage*.

Para o método *single linkage*, a distância entre dois *clusters* é determinada pela menor distância entre os dois objetos mais próximos nos diferentes *clusters*. Ao contrário, no método *complete linkage* é usada a maior distância de um par de objetos para definir a distância *inter-cluster*.

Alguns algoritmos hierárquicos aglomerativos mais complexos podem incluir outros métodos como *average linkage*, *centroid linkage* e *Ward´s method*.

No caso *average linkage*, a proximidade entre dois clusters é dada pela média aritmética das distâncias entre todos os pares de objetos.

A maior diferença entre os algoritmos hierárquicos aglomerativos é a definição da função de custo para fundir dois *clusters* S_i e S_j denotados por $C(S_i, S_j)$. A figura 3.1 sumariza as funções de custo mais conhecidas (FASULO, 1999).

Método	**Função de custo**
Single linkage	$\min_{x_i \in S_i, x_j \in S_j} d(x_i, x_j)$
Average linkage	$\frac{1}{\lvert S_i \rvert \lvert S_j \rvert} \sum_{x_i \in S_i} \sum_{x_j \in S_j} d(x_i, x_j)$
Complete linkage	$\max_{x_i \in S_i, x_j \in S_j} d(x_i, x_j)$

Figura 3.1 Modelos de agrupamentos hierárquicos (FASULO, 1999).

Single linkage, *average linkage* e *complete linkage* consideram todos os pontos de um par de *clusters*, quando calculam a distância *inter-cluster*, e são chamados de métodos gráficos. Os outros são chamados métodos geométricos, pois usam centros geométricos para representar *clusters* e determinar a distância entre eles (XU; WUNSCH, 2005).

Além dos métodos citados na figura 3.1, o *Ward´s method* também é usado freqüentemente. O *Ward´s method* pode ser considerado como o agrupamento hierárquico equivalente ao critério da soma dos quadrados dos algoritmos particionais. Como nos métodos citados acima, em cada etapa do algoritmo há uma lista dos *clusters* $\{S_1,..., Sn\}$ e o objetivo é escolher um par de *clusters* para fundir. Como no método da soma dos quadrados, o custo de cada S_i é definido como:

$$C(S_i) = \sum_{r=1}^{|S_i|} \sum_{s=1}^{|S_i|} (d(x_r^i, x_s^i))^2 \tag{3.2}$$

O custo da solução atual é então definido como $\sum_{i=1}^{n} C(S_i)$, e os pares de *clusters* a fundir são escolhidos com o objetivo de minimizar o custo da solução total. Igualmente como no método

da soma dos quadrados, algoritmos que executam *Ward´s method* tipicamente possuem a habilidade de calcular o centróide de cada conjunto (XU; WUNSCH, 2005).

No método *Centroid linkage* os agrupamento são representados por um vetor que armazena a média de cada atributo. Assim, a distância entre os *clusters* é definida em termos da distância entre o ponto representativo de cada cluster, o centróide. Conseqüentemente, os atributos devem ser numéricos.

O algoritmo Hierárquico Aglomerativo implementado neste trabalho utilizou o método *centroid linkage* para determinar a dissimilaridade entre os *clusters.*

O algoritmo 1 apresenta o procedimento do Hierárquico Aglomerativo implementado nesse trabalho.

Algoritmo 1: calcula os centróides $\{\mu_1,..., \mu_k\}$ de um conjunto de dados *X,* usando o método hierárquico aglomerativo com abordagem baseada na técnica *centroid linkage.*

Entrada: *X*, conjunto de *N* padrões com *d* dimensões
K, número de centróides
Saída: conjunto de *K* centróides

1. Faça cada padrão representar um cluster cujo centróide é o próprio padrão $\{C_1,..., C_n\}$
2. **Para todo** C_i **faça**
3. **Para todo** C_j **quando** $i < j$ **faça**
4. Matriz $M_{ij} \leftarrow$ distância entre C_i e C_j
5. **Fim Para**
6. **Fim Para**
7. **Enquanto** (o número de clusters for maior do que *K*) **faça**
8. **Para todo** C_i **faça**
9. Faça a fusão de C_i com o cluster C_j que apresentar a menor distância
10. Recalcule o centróide do novo cluster como sendo o centro de massa de todos os pontos associados a ele: $\mu_j = \frac{1}{n_j}\sum_{x \in C_j} x$
11. Recalcule a linha/coluna *i* e elimine a linha/coluna *j* na matriz M_{ij}
12. **Fim Para**
13. **Fim Enquanto**
14. **Retorne** $\{\mu_1,..., \mu_k\}$

Segundo (JAIN; MURTY; FLYNN, 1999), uma desvantagem comum dos algoritmos hierárquicos é que eles são sensíveis ao ruído e aos *outliers*. Uma vez que um objeto é atribuído a um conjunto, ele não é considerado novamente, isto significa que os algoritmos hierárquicos não são capazes de corrigir uma possível associação incorreta. A complexidade computacional para a maioria dos algoritmos hierárquicos é pelo menos $O(n^2)$ e este custo elevado limita sua aplicação em séries de dados em grande escala.

Como vantagem os algoritmos hierárquicos não requerem a especificação do número de *clusters* e seus resultados correspondem a taxonomias muito comuns nas ciências biológicas, além de possibilitarem a análise dos diferentes níveis de granularidade ou densidade dos agrupamentos.

3.2 Algoritmo *K-means*

O *K-means* é um algoritmo particional não-determinístico que realiza a busca pela melhor solução (ou por uma solução subótima) para o problema de agrupamento de dados. O objetivo é encontrar a melhor combinação de K partições $\{C_1,..., C_k\}$ em um conjunto de dados $X = \{x_1,..., x_n\}$, em que os d atributos de cada variável x_i são valores numéricos.

É uma técnica amplamente utilizada em diversas aplicações, devido à sua simplicidade, facilidade de implementação e rápida convergência. Seu tempo de complexidade é $O(n)$, onde n é o número de padrões. O maior problema desse algoritmo é a sua sensibilidade à escolha da solução inicial. O algoritmo pode convergir para um mínimo local caso a solução inicial não seja escolhida adequadamente (JAIN; MURTY; FLYNN, 1999).

A figura 3.2 mostra sete padrões bi-dimensionais. Neste exemplo o algoritmo precisa encontrar três partições. Com a solução inicial composta pelos pontos A, B e C, o resultado do *K-means* é dado pelos grupos {A}, {B, C} e {D, E, F, G}, como mostram as elipses na figura. Porém, o resultado mais adequado é definido pelos grupos {A, B, C}, {D, E} e {F,G}, mostrados nos retângulos. Esse resultado seria obtido caso a solução inicial fosse composta pelos pontos A, D e F.

No *k-means* a busca pela solução é realizada iterativamente associando os padrões à partição mais próxima. Para isso, minimiza-se um critério derivado da medida de dissimilaridade entre cada padrão do conjunto de dados e o centróide de uma determinada partição. O centróide é o ponto representativo de uma partição. Por isso, minimizar a dissimilaridade entre um padrão e um centróide significa associar o padrão à partição mais próxima.

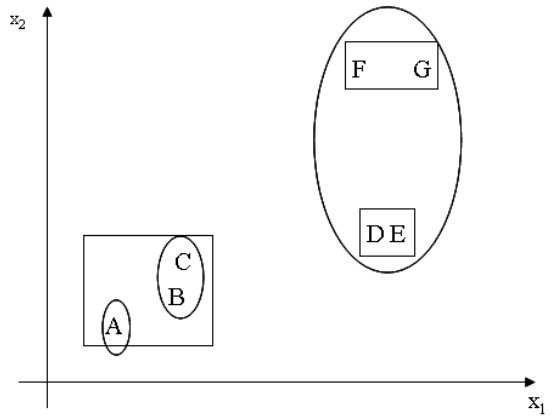

Figura 3.2 Sensibilidade do *K-means* à partição inicial (JAIN; MURTY; FLYNN, 1999).

A avaliação das partições depende de um critério baseado na medida de dissimilaridade. Entre os diferentes critérios de avaliação (EVERITT; LANDAU; LEESE, 2001), o mais comumente utilizado no algoritmo do *K-means* é a soma das distâncias Euclidianas quadradas (*SSE*).

O algoritmo padrão do *K-means* (FORGY, 1965) gera, inicialmente, um conjunto aleatório de *K* padrões a partir do conjunto de dados, os centróides, que representam a solução inicial do problema. Após a inicialização, o algoritmo gera, a cada iteração, um novo conjunto de centróides, ou seja, uma nova solução. Cada padrão do conjunto de dados é associado ao centróide mais próximo da solução corrente. Os centróides são então recalculados após todos os padrões serem associados. O algoritmo termina ao atingir um critério de convergência. Alguns exemplos de critérios de convergência são: número máximo de iterações, estabilidade da solução entre duas iterações consecutivas ou decréscimo pouco significativo no valor do critério de avaliação.

O Algoritmo 2 apresenta o procedimento de busca de solução do *K-means*, segundo (FORGY, 1965).

Algoritmo 2: calcula os centróides $\{\mu_1,..., \mu_K\}$ de um conjunto de dados *X* usando seleção aleatória inicial.

Entrada: *X*, conjunto de *N* padrões com *d* dimensões;
K, número de centróides

Saída: conjunto de *K* centróides

1. Escolha aleatoriamente *K* centróides iniciais $\{\mu_1,..., \mu_K\}$ a partir de *X*
2. **Enquanto** (critério de convergência não for atingido) **faça**
3. Associe cada padrão x_i ao centróide μ_j mais próximo
4. Recalcule cada centróide de um agrupamento como sendo o centro de massa de todos os pontos associados a ele: $\mu_j = \frac{1}{n_j}\sum_{x \in C_j} x$

5. **Fim Enquanto**
6. **Retorne** $\{\mu_1, ..., \mu_K\}$

Outra versão do *K-means* é apresentada por (MACQUEEN, 1967) e difere da versão de Forgy na atualização dos centróides. Na versão de MacQueen os centróides são recalculados depois de cada associação do padrão ao centróide mais próximo, na versão de Forgy os centróides são recalculados após todos os padrões serem associados aos centróides. Neste trabalho, conforme descrito no Algoritmo 2, o procedimento utilizado segue a versão de Forgy.

3.3 Algoritmo *K-means* Inicializado pelo *K-means++*

Um método de inicialização é um procedimento que tem como objetivo escolher uma melhor solução inicial para o *k-means*. Desta forma espera-se contornar o problema da sensibilidade à escolha da solução inicial do *k-means*, que pode convergir para um mínimo local caso a solução inicial não seja escolhida adequadamente. Outra característica importante é que métodos de inicialização ajudam a acelerar o processo de convergência do *k-means*.

(ARTHUR; VASSILVITSKII, 2007; SU; DY, 2004) e outros, têm apresentado trabalhos nessa área com resultados importantes, comprovando teórica e empiricamente que métodos de inicialização contribuem para melhoria da solução obtida com o *k-means*.

O *K-means* inicializado pelo *K-means++* é um algoritmo particional não-determinístico que realiza a busca pela melhor solução (ou por uma solução subótima) para o problema de agrupamento de dados. Seu objetivo é encontrar a melhor combinação de *K* partições em um conjunto de dados.

Com o objetivo de minimizar o problema da sensibilidade à escolha da solução inicial do *k-means*, que gera arbitrariamente partições muito ruins, o método de inicialização *k-means++* realiza esse processo de maneira diferente. (ARTHUR; VASSILVITSKII, 2007) sugerem um processo de inicialização baseado em uma escolha aleatória dos centróides com probabilidade específica, onde um ponto *p* é escolhido como centróide com probabilidade proporcional à contribuição desse ponto para o critério *SSE*.

Em (ARTHUR; VASSILVITSKII, 2007) o método proposto de escolha dos centróides

seleciona inicialmente um ponto aleatório no conjunto de dados para representar o primeiro centróide. Os demais $K-1$ centróides são escolhidos iterativamente, selecionando pontos do conjunto de dados com probabilidade proporcional à sua contribuição para o critério SSE.

Segundo (ARTHUR; VASSILVITSKII, 2007), considerando que $D(x)$ representa a menor distância entre o ponto $\boldsymbol{x}$ e o centróide mais próximo a ele. O processo de escolha da solução inicial do algoritmo *k-means*++ é definido da seguinte maneira:

1. Escolha de X um centróide inicial c_1 aleatoriamente.
2. Escolha o próximo centróide c_i, selecionando $c_i = x' \in X$ com probabilidade

$$\frac{D(x')^2}{\sum_{x \in X} D(x)^2} \tag{3.4}$$

3. Repita o passo 2 até ter escolhido os K centróides.

Desta forma, quanto maior a contribuição do ponto para o critério *SSE*, maior é a probabilidade desse ponto ser selecionado como centróide. Assim, o método seleciona os centróides de forma a minimizar o critério de avaliação. O algoritmo 3 apresenta o método de inicialização *K-means*++ utilizado neste trabalho.

(OSTROVSKY et al., 2006) propõem um método de inicialização semelhante, a diferença consiste em escolher aleatoriamente dois pontos como centróides com probabilidade proporcional à distância Euclidiana entre eles. Em seguida, são escolhidos os demais K−2 centróides, selecionando os pontos com probabilidade proporcional à distância Euclidiana entre o ponto e o centróide mais próximo. Entretanto, (OSTROVSKY et al., 2006) mostram que esse método encontra soluções próximas ao ótimo na ordem de O(log(1)), em dados que possuem um determinado critério de separação, enquanto que (ARTHUR; VASSILVITSKII, 2007) provam que o método em questão encontra soluções próximas ao ótimo na ordem de O(log(K)) para todos os conjuntos de dados.

Algoritmo 3: calcula os centróides $\{\mu_1, ..., \mu_K\}$, dado um conjunto de dados X, usando probabilidade proporcional à contribuição para o critério *SSE*.

Entrada: X, conjunto de N padrões com d dimensões;
K, número de centróides

Saída: conjunto de K centróides

1. Escolha aleatoriamente um centróide inicial μ_1 a partir de X
2. **Enquanto** (o número de centróides ($K-1$) não for gerado) **faça**

3. Escolha o próximo centróide $\mu_j = x_i \in X$, com probabilidade $\frac{D(x_i)^2}{\sum_{x \in X} D(x)^2}$
4. **Fim Enquanto**
5. **Retorne** $\{\mu_1, ..., \mu_K\}$

(ARTHUR; VASSILVITSKII, 2007) mostram, que o método proposto, além de rápido e simples, melhora a qualidade dos resultados do *K-means*, provando que a razão entre o valor esperado do *SSE* encontrado com o uso desse método e o *SSE* ótimo é limitada por uma constante na ordem de $O(log(K))$:

$$E(SSE) \leq 8(ln\ k + 2)SSE_{OPT} \tag{3.5}$$

(ARTHUR; VASSILVITSKII, 2007) apresentam experimentos onde o *K-means++* supera consideravelmente, em qualidade e velocidade, o *K-means* inicializado aleatoriamente, em bases artificiais e reais.

3.4 Algoritmo *K-means* Inicializado pelo *PCA_Part* (*Principal Components Analysis*)

O *PCA_Part* (*Principal Components Analysis*) é um método determinístico que apresenta uma abordagem de divisão dos dados baseado na direção do primeiro eixo principal no espaço de características. O método envolve uma análise estatística de correlação entre as componentes que formam o objeto, nesse caso, as características dos padrões a serem agrupados.

O *PCA_Part* consiste em reescrever as coordenadas de um conjunto de dados em um outro sistema de eixos que seja mais conveniente para a análise dos dados. Estas novas coordenadas são o resultado da combinação linear das variáveis originais e são representadas sobre eixos ortogonais, sendo obtidas em ordem decrescente de variância. Portanto, a primeira componente principal detém mais informação sobre os dados do que a segunda componente principal que não detém informações contabilizadas anteriormente (na primeira componente principal) e assim sucessivamente. Em função da ortogonalidade dos eixos, as componentes principais não são correlacionadas.

O número total das componentes principais é igual ao número total de variáveis originais e apresenta a mesma informação estatística que estas variáveis. Porém, este método

permite a redução do número total de variáveis, pois freqüentemente as primeiras componentes principais detêm a maioria das informações estatísticas dos dados originais.

O *PCA_Part* é muito usado em vários tipos da análises pois é um método simples, não-paramétrico de extrair informação relevante de conjuntos de dados complexos. Com mínimo esforço o método fornece uma redução da dimensão complexa das características dos dados para uma dimensão menos complexa que permite revelar informações às vezes escondidas (SHLENS, 2005).

O método do *PCA_Part* foi introduzido em (HUANG; HARRIS, 1993) e (BOLEY, 1998). No primeiro trabalho, o algoritmo foi chamado de *Directed-Search Binary-Splitting* (*DSBS*) e foi utilizado para inicializar *code-vectors*. No segundo, o procedimento foi chamado de *Principal Direction Divisive Partitioning* (*PDDP*) e foi usado para agrupar documentos. Essas abordagens baseadas em *PCA* são similares. A diferença principal é que o *DSBS* considera apenas os pontos do conjunto de dados que estão dentro de duas vezes o desvio padrão da média para excluir possíveis ruídos.

O procedimento de construção da solução inicial do método *PCA_Part* gera, inicialmente, uma única partição formada por todos os dados. Após a inicialização, o algoritmo seleciona, a cada iteração, a partição C_j com o maior *SSE* e a divide em duas partições C_{j1} e C_{j2}, com centróides μ_{j1} e μ_{j2}, respectivamente, na direção do maior autovetor da matriz de covariância. Essa divisão da partição C_j é feita projetando cada padrão $x_i \in C_j$ para a primeira direção principal, gerando os vetores y_i. O mesmo é feito com o centróide μ_j de C_j, gerando o vetor α_j. O processo é repetido até que K partições sejam geradas. O Algoritmo 4 apresenta o método de inicialização *PCA_Part* utilizado neste trabalho.

Algoritmo 4: calcula os centróides $\{\mu_1,..., \mu_K\}$, dado um conjunto de dados X, usando abordagem baseada na técnica *PCA*

Entrada: X, conjunto de N padrões com d dimensões;
K, número de centróides

Saída: conjunto de K centróides

7. $C_1 \leftarrow X$
8. **Enquanto** (os K centróides não forem gerados) **faça**
9. Escolha a partição C_j com o maior *SSE*
10. Projete todo $x_i \in C_j$ e o centróide μ_j para a primeira direção principal: y_i e α_j, respectivamente
11. **Para todo** $x_i \in C_j$ **faça**
12. **Se** $y_i \leq \alpha_j$ **então**
13. $C_{j1} \leftarrow C_{j1} \in \{x_i\}$
14. **Senão**
15. $C_{j2} \leftarrow C_{j2} \in \{x_i\}$
16. **Fim Se**

17. **Fim Para**
18. **Fim Enquanto**
19. **Retorne** $\{\mu_1,..., \mu_K\}$

O objetivo do método é minimizar o valor de *SSE* a cada iteração. Para tanto, o algoritmo seleciona a partição C_j com o maior *SSE* e divide essa partição na direção que minimiza o novo valor de *SSE*. O valor de *SSE* da partição C_j antes da divisão é igual a:

$$SSE_{old} = \sum_{x_i \in C_j} \left\| x_i - \mu_j \right\|^2 \tag{3.6}$$

Após a divisão, o novo valor de *SSE* é igual a:

$$SSE_{new} = \sum_{x_i \in C_{j1}} \left\| x_i - \mu_{j1} \right\|^2 + \sum_{x_i \in C_{j2}} \left\| x_i - \mu_{j2} \right\|^2 \tag{3.7}$$

A direção que minimiza SSE_{new} é a mesma que maximiza a diferença entre SSE_{old} e SSE_{new} projetados. O problema é simplificado para encontrar a direção que contribui para o maior valor de SSE_{old}, que é determinada pela direção do maior autovetor da matriz de covariância (FUKUNAGA, 1990). A Figura 3.3 mostra a abordagem de divisão do *PCA_Part*.

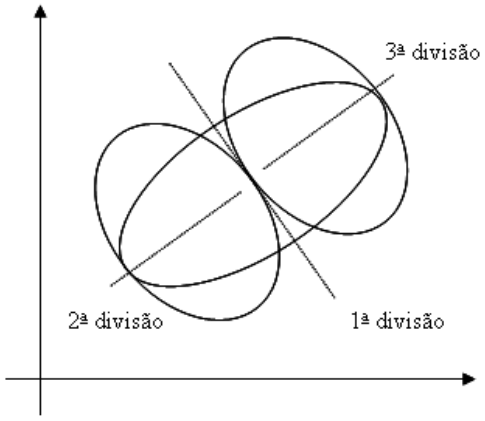

Figura 3.3 Particionamento a partir da covariância amostral em duas dimensões (SU; DY, 2007).

Uma fragilidade desse método é a escolha da primeira direção principal que somente maximiza o valor SSE_{old} na direção da componente principal sem considerar o efeito sobre SSE_{new} nessa mesma direção. Embora exista uma indicação de que essa seja uma boa direção para projetar os dados na divisão, essa escolha não é ótima nos casos em que os dados estão bem separados em direções diferentes do maior autovetor.

(SU; DY, 2007) apresentam uma extensão para o método *PCA_Part*, tal método é chamado de *PCA_Part**. O método seleciona a componente *PCA* que maximiza a diferença entre SSE_{old} e SSE_{new} como a direção para projetar os dados na divisão. Entretanto, essa solução ainda é subótima, pois considera apenas as componentes *PCA* e não o espaço infinito de possíveis direções. Além disso, o método *PCA_Part** é computacionalmente mais complexo, comparado ao *PCA_Part*. Este trabalho utiliza o método *PCA_Part*, conforme sugerido por (SU; DY, 2007).

(SU; DY, 2004) utilizaram o *PCA_Part* como método de inicialização para o *k-means* e obtiveram resultados que mostram, experimentalmente, que essa combinação, em alguns casos, apresentou uma melhoria nos resultados, obtendo valores de *SSE* próximos aos menores valores encontrados com 100 execuções do *K-means* inicializado aleatoriamente. Entretanto, em algumas bases, a combinação não obteve resultados melhores que os obtidos com a versão aleatória. Provavelmente, o motivo para tal fato se deve à violação da hipótese de que a primeira direção principal maximiza a diferença entre SSE_{old} e SSE_{new} projetados.

(PERIM, 2008) utiliza o *PCA_Part* como método de inicialização para o *Simulated Annealing* e mostra, experimentalmente, que essa combinação obteve resultados superiores aos do *K-means* inicializado aleatoriamente.

Maiores informações sobre o uso de métodos de inicialização combinados ao *K-means* e ao *Simulated Annealing* podem ser encontrados em (PERIM, 2008; PERIM; VAREJÃO, 2008; PERIM; WANDEKOKEM; VAREJÃO, 2008).

4 *Metaheurísticas Aplicadas ao Problema de Agrupamento*

Conforme descrito no capítulo 2 problemas de otimização podem ser modelados da seguinte forma: dado um conjunto S de variáveis discretas s (chamadas soluções) e uma função objetivo $f : S \rightarrow R$, que associa cada solução $s \in S$ a um valor real $f(s)$, encontre a solução $s^* \in S$, dita ótima, para a qual $f(s)$ é mínima ou máxima dependendo do contexto. Tais problemas são classificados como problemas de otimização combinatória.

Muitos desses problemas pertencem à classe *NP-hard*, ou seja, não existem algoritmos que os resolvam em tempo polinomial. Portanto, em problemas dessa natureza, o uso de métodos exatos que garantem encontrar a solução ótima se torna bastante restrito. Por isso, na prática, em geral, é suficiente encontrar uma boa solução para o problema, no lugar do ótimo global, o qual somente pode ser encontrado após um considerável esforço computacional.

Com o objetivo de encontrar soluções próximas ao ótimo global a um custo computacional aceitável, menor que dos métodos exatos, aplicam-se procedimentos chamados algoritmos de aproximação. Tais algoritmos se subdividem em aproximativos, que garantem resultados com proximidade percentual ao ótimo global, e aproximados ou heurísticos, que obtêm resultados aproximados sem garantia de proximidade percentual.

Métodos heurísticos procuram boas soluções factíveis aos problemas de otimização em circunstâncias onde a complexidade do problema ou o tempo disponível limitado para sua solução não permitem a solução exata (RARDIN, R. L.; UZSOY, R, 2001).

Heurísticas muito eficientes foram desenvolvidas para diversos problemas, entretanto a maioria dessas heurísticas é muito específica para um problema particular, não sendo eficientes (ou mesmo aplicáveis) na resolução de uma classe mais ampla de problemas. Neste contexto surgem procedimentos heurísticos gerais aplicáveis a uma variedade de problemas conhecidos como metaheurísticas, termo introduzido por (GLOVER, 1986).

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico (RIBEIRO, 1996).

Neste capítulo são apresentadas em detalhes as metaheurísticas: *Tabu Search,* Algoritmo Genético de Agrupamento e o Algoritmo Genético Híbrido de Agrupamento, abordagem proposta neste trabalho motivada na necessidade de obter resultados melhores em problemas de agrupamento, inclusive em instâncias maiores.

As metaheurísticas podem ser classificadas em duas categorias principais: baseadas em soluções parciais (ou construtivas) e baseadas em soluções completas.

A categoria construtiva tem por objetivo construir uma solução, elemento por elemento. A forma de escolha de cada elemento a ser inserido a cada passo varia de acordo com a função de avaliação adotada, que depende do problema abordado. Nas versões clássicas, os elementos candidatos são geralmente ordenados segundo uma função gulosa e apenas o melhor elemento é inserido a cada passo. O algoritmo 5 mostra o procedimento geral de métodos construtivos.

Algoritmo 5: procedimento geral de métodos construtivos.

Saída: solução *S*
1: $S \leftarrow \emptyset$
2: **Enquanto** (s não estiver completo) **faça**
3: Avalie os elementos não selecionados
4: r ← elemento melhor avaliado
5: $S \leftarrow S \cup r$
6: **Fim Enquanto**
7: **Retorne** *S*

A categoria baseada em soluções completas tem por objetivo modificar tais soluções com o intuito de encontrar uma melhor solução. A forma de modificar a solução depende da abordagem adotada que pode ser dividida em duas subcategorias: busca local e busca populacional.

A busca local parte de uma solução inicial qualquer (que pode ser obtida por um método construtivo ou gerada aleatoriamente) e de acordo com a função de vizinhança adotada tenta a cada iteração substituir a solução corrente por uma solução melhor que pertença à vizinhança da solução corrente. O algoritmo 6 mostra o procedimento geral de métodos de busca local.

Algoritmo 6: procedimento geral de métodos de busca local.

Saída: solução *S*
1: Gere solução inicial completa *S*
2: **Enquanto** (critério de parada não for satisfeito) **faça**

3: Modifique solução *S*
4: Avalie nova solução
5: **Fim Enquanto**
6: **Retorne** S

Os métodos de busca local são técnicas baseadas na noção de vizinhança. Tal método avalia a cada iteração o novo vizinho gerado com o intuito de encontrar uma solução melhor. Seja *S* o espaço de pesquisa de um problema e *N* a função que define a vizinhança de uma solução. A função *N* associa a cada solução $s \in S$, sua vizinhança $N(s) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de *s* e representa uma solução próxima à solução *s*.

Algoritmos de busca local utilizam um processo de busca que lida com uma única solução a cada instante, descrevendo uma trajetória no espaço de busca. São exemplos de algoritmos de busca local ou busca por ponto único: Busca Tabu (*Tabu Search*) (GLOVER, 1986; LAGUNA, 1994; ALSULTAN, 1995; GLOVER; LAGUNA, 1997), Resfriamento Simulado (*Simulated Annealing*) (KIRKPATRICK; GELATT; VECCHI, 1983; CERNY, 1985) e *GRASP* (*Greedy Randomized Adaptive Search Procedure*) (FEO; RESENDE, 1995).

Na seção 4.3 é descrito o algoritmo *Tabu Search,* adaptado ao problema de agrupamento, utilizado neste trabalho.

A busca populacional é baseada numa população de soluções que são combinadas para gerar nova população. Um algoritmo populacional utiliza um conjunto de indivíduos para explorar o espaço de soluções. É um processo estruturalmente paralelo onde existe a troca de informação entre as soluções à medida que efetuam a busca.

O algoritmo 7 mostra o procedimento geral de métodos de busca populacional.

Algoritmo 7: procedimento geral de métodos de busca populacional.
Saída: população *P*
1: Gere população inicial *P*
2: **Enquanto** (critério de parada não for satisfeito) **faça**
3: Modifique população *P*
4: Avalie nova população
5: **Fim Enquanto**
6: **Retorne** *P*

A Computação Evolutiva trata de sistemas para a resolução de problemas que utilizam modelos computacionais baseados na teoria da evolução natural. Para tanto, utilizam métodos de busca populacional onde mantêm uma população de soluções potenciais, assim podem

fazer a busca em diferentes áreas do espaço de solução, alocando um número de indivíduos apropriados para a busca em várias regiões. Como resultado, tais técnicas têm maior chance de atingir as áreas mais promissoras do espaço de busca.

São exemplos de algoritmos baseados em busca populacional: Algoritmos Genéticos (*Genetic Algorithm*) (GOLDBERG, 1989; DAVIS, 1991), Algoritmos Genéticos Híbridos também conhecidos como Algoritmos Meméticos (*Memetic Algorithm*) (MOSCATO, 1989; MEZ, 2000), Colônia de Formigas (*Ant Colony*) (DORIGO; CARO, 1999; DORIGO; CARO; GAMBARDELLA, 1999) e Nuvem de Partículas (*Particle Swarm*) (KENNEDY; EBERHART, 1995).

4.1 Metaheurísticas para Agrupamento

Um dos primeiros trabalhos a utilizar metaheurísticas ao problema de agrupamento foi apresentado por (RAGHAVAN; BIRCHARD, 1979). Nesse trabalho, propõe-se uma abordagem baseada no comportamento adaptativo encontrado em sistemas naturais, sugerindo uma representação simples para cada solução do problema e utilizando métodos de seleção e de mutação para encontrar soluções mais aptas.

Outros trabalhos deram continuidade ao estudo de aplicações de algoritmos genéticos ao problema de agrupamento. (MURTHY; CHOWDHURY, 1996), por exemplo, sugerem uma representação similar à de (RAGHAVAN; BIRCHARD, 1979) e operadores genéticos adequados para o problema de agrupamento. A avaliação experimental foi realizada em bases de dados mais gerais, entretanto pequenas, com menos de 100 padrões e 5 atributos.

(HALL; ÖZYURT; BEZDEK, 1999) sugerem uma técnica de busca geneticamente guiada que pode ser aplicada tanto para agrupamentos *hard* quanto *fuzzy*. Essa proposta também obteve, experimentalmente, bons resultados em bases pequenas, como a IRIS (150 padrões e 4 atributos). (FALKENAUER, 1998) e (COLE, 1998) apresentam uma discussão maior sobre o uso de algoritmos genéticos ao problema de agrupamento.

Segundo (RAYWARD-SMITH, 2005), a aplicação direta de metaheurísticas ao problema de agrupamento parece ser mais efetiva em bases pequenas. Isso acontece porque o espaço de busca cresce consideravelmente quando o número de padrões N e de partições K aumenta. Além disso, a representação de uma solução em bases maiores é muito grande e as funções de vizinhança e de avaliação são bastante custosas de serem calculadas, o que

contribui para tornar o processo de busca ainda mais demorado.

Uma estratégia mais promissora está voltada para abordagens híbridas. Um exemplo desse tipo de abordagem é a combinação de técnicas de busca usadas tanto em heurísticas quanto em metaheurísticas, como ocorre na utilização de busca local em algoritmos de busca populacional. Segundo (MEZ, 2000), uma junção de buscas populacionais e buscas locais são benéficas para o resultado final.

A escolha de uma metaheurística baseada em busca local e outra baseada em busca populacional resultou de estudos que indicam que as heurísticas baseadas em busca populacional apresentam melhor desempenho na exploração do espaço de soluções, identificando áreas promissoras, enquanto as heurísticas baseadas em busca local são mais eficientes na exploração dessas áreas (TALBI, 2002). Uma conseqüência natural, portanto, é o desenvolvimento de abordagens híbridas, que explorem as melhores características da busca populacional e da busca local.

(MAULIK; BANDYOPADHYAY, 2000) apresenta uma técnica de agrupamento baseada em algoritmo genético, utilizando, em um conjunto de dados com d dimensões, cromossomos de tamanho $d \cdot K$, sendo que as d primeiras posições representam as d dimensões do primeiro centróide, as próximas d posições representam as dimensões do segundo centróide, e assim por diante. No entanto, nesse trabalho a transição entre a população corrente e a nova geração não é realizada apenas por operadores genéticos, tais como seleção, mutação e *crossover* (recombinação). O algoritmo utiliza também a abordagem gulosa do *K-means* de associar os pontos aos centróides mais próximos, que foram encontrados previamente pelos operadores genéticos. Em seguida, a população é substituída por essas novas soluções. Experimentalmente, o algoritmo foi avaliado usando bases reais com até 871 padrões e 3 atributos, obtendo melhoria nos resultados quando comparado ao *K-means*.

Outra abordagem híbrida consiste no uso encadeado de metaheurísticas. Um exemplo dessa abordagem é o uso de algoritmos evolutivos para encontrar regiões promissoras no espaço de busca. Tais regiões seriam exploradas por algoritmos de busca local com o intuito de encontrar soluções melhores. Outro exemplo dessa abordagem consiste em aplicar uma heurística de busca local para gerar a população inicial e direcionar o processo de busca do algoritmo evolutivo. As próximas seções apresentam três metaheurísticas, adaptadas para o problema de agrupamento de dados, utilizadas neste trabalho: o Algoritmo Genético (AG), o Algoritmo Genético Híbrido (AGH), também conhecido como Algoritmo Memético

(MOSCATO, 1989) e o *Tabu Search.*

4.2 Algoritmo Genético

Abordagens Evolutivas têm se mostrado muito eficientes na obtenção de soluções para problemas de agrupamento de dados (JAIN; MURTY; FLYNN, 1999). Dentre essas abordagens, Algoritmos Genéticos (AG) são bastante utilizados, principalmente em problemas de agrupamento em *k* partições onde o valor de *k* é previamente conhecido (RAGHAVAN; BIRCHARD, 1979; BEZDEK; BOGGAVAPARU; HALL; BENSAID, 1994).

AGs tiveram projeção a partir de meados dos anos 70 depois da publicação do livro "*Adaptation in Natural and Artificial Systems*" de John Holland. São técnicas de otimização e busca global utilizadas para combinar características de possíveis soluções de bom desempenho, gerando soluções potencialmente melhores.

AGs têm se destacado na solução de problemas de agrupamento porque (COWGILL; HARVEY; WATSON, 1998; JAIN; MURTY; FLYNN, 1999):

- Executam uma busca global pelas melhores soluções, enquanto a maioria dos procedimentos conhecidos de agrupamento executa busca local.
- Utilizam procedimentos de busca probabilísticos ao invés de regras determinísticas.
- Procuram por uma população de soluções em paralelo, permitindo evitar mínimos locais.
- A função objetivo do algoritmo de agrupamento e a função de aptidão do AG correspondente são suficientes para direcionar a busca.
- Podem obter não apenas uma, mas um grupo de potenciais soluções para um dado problema.

(COLE, 1989) denomina como Algoritmo Genético de Agrupamento (AGA) (*Genetic Clustering Algorithm* – GCA) o AG adaptado para resolver problemas de agrupamento em *k* partições com o valor de *k* previamente conhecido. Tais adaptações devem acontecer na representação da solução, função de adaptação (*fitness function*), operadores e valores dos parâmetros. Nas próximas seções são apresentadas as adaptações utilizadas neste trabalho.

4.2.1 Definição e Aspectos Principais

Os AGs baseiam-se no processo de evolução dos genes das espécies. O algoritmo propõe evoluir de uma população de cromossomos para outra através de um processo de seleção usando operadores inspirados na genética.

De uma forma simples um AG é um processo iterativo que aplica uma série de operadores genéticos como seleção, *crossover* e mutação a uma população de elementos (COLE, 1998). Uma visão geral do funcionamento de um AG clássico é apresentada na figura 4.1.

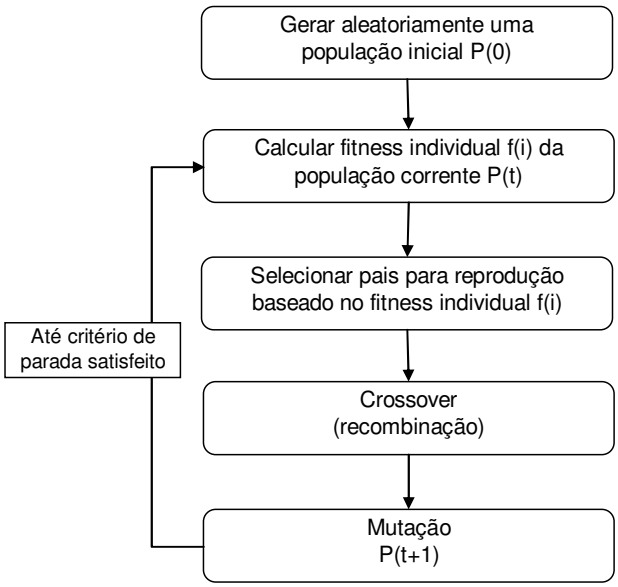

Figura 4.1 Visão geral do funcionamento de um AG simples, baseado em (COLE, 1998).

No AG cada cromossomo (indivíduo da população) está associado a uma solução do problema e cada gene está associado a uma componente da solução. Um mecanismo de reprodução, baseado em processos evolutivos, é aplicado sobre a população com o objetivo de explorar o espaço de busca e encontrar melhores soluções para o problema.

Segundo (MICHALEWICZ, 1996) um AG pode ser descrito como se segue:

- durante a iteração *t*, um AG mantém uma população de soluções potenciais (cromossomos ou indivíduos) $P(t) = \{ x_1^t, ..., x_n^t \}$;
- cada solução x_i^t é avaliada e produz uma medida de sua adaptação (*fitness*);
- uma nova população (iteração $t + 1$) é então formada privilegiando a participação dos

indivíduos mais adaptados;

- alguns membros da nova geração passam por modificações, através de *crossover* e mutação, para formar novas soluções;
- este processo se repete até que um certo número de iterações seja atingido, ou até que um nível de adaptação esperado seja alcançado.

O primeiro passo de um AG é a geração de uma população inicial que é formada por um conjunto de cromossomos que representam possíveis soluções do problema a ser resolvido. O cromossomo é composto por genes que são, portanto, parte da representação de uma solução. O valor que cada gene pode assumir é denominado alelo.

Durante o processo evolutivo, esta população é avaliada e cada cromossomo recebe um valor *fitness* (também denominado aptidão), que reflete a qualidade da solução que ele representa. Em geral, os cromossomos mais aptos são selecionados e os menos aptos são descartados. Nos cromossomos selecionados são aplicados os operadores *crossover* (recombinação) e/ou mutação. Este processo é repetido até que um critério de parada seja satisfeito, em geral, quando um certo número de populações é gerado ou quando a melhor solução encontrada atinge um certo nível de aptidão ou ainda, quando não há melhora após um certo número de iterações. O algoritmo 8 apresenta o procedimento geral de um AG clássico.

Algoritmo 8: procedimento geral do Algoritmo Genético

Entrada: tamanho da população
taxa de *crossover*
taxa de mutação

Saída: melhor solução encontrada *s*

1: $t \leftarrow 0$
2: Gere a população inicial *P(t)*
3: Avalie *P(t)*
4: **Enquanto** (critério de parada não for satisfeito) **faça**
5: $t \leftarrow t + 1$
6: Selecione *P(t)* a partir de *P(t-1)*
7: Aplique *crossover P(t)*
8: Aplique mutação *P(t)*
9: Avalie *P(t)*
10: **Fim Enquanto**
11: **Retorne** *s*

4.2.2 Inicialização

Os AGs, assim como outras estratégias evolutivas, trabalham em cima de uma população inicial. A população inicial é formada por um conjunto de cromossomos, ou indivíduos, que representam possíveis soluções do problema. Um método de inicializar uma população é produzir aleatoriamente os valores assumidos pelos cromossomos.

No AGA implementado o algoritmo usado no processo de geração da população inicial produz aleatoriamente as soluções.

4.2.3 Represetanação da Solução

Um dos aspectos fundamentais na modelagem de um AG é a representação da solução, que define a estrutura do cromossomo, com os respectivos genes que os compõem, de forma a descrever todo o espaço de busca do problema. Tal representação depende da natureza do problema em questão.

No AG clássico proposto por (HOLLAND, 1992), as soluções candidatas são codificadas em arranjos binários de tamanho fixo. Porém, outros modelos de AGs representam as soluções com outros alfabetos, por exemplo, com números reais (MICHALEWICZ, 1996).

(GOLDBERG, 1989) mostra dois princípios básicos para escolher a codificação de um AG. Primeiro, a codificação deve conter blocos significativos. Segundo, o alfabeto deve ser o menor possível que permita uma expressão natural do problema.

Representações da solução para o problema de agrupamento com número fixo e previamente conhecido de partições são baseadas em dois esquemas básicos. O primeiro aloca para cada objeto um (ou mais) inteiro ou bit, conhecido como gene, e usa o valor desse gene para indicar a qual partição o objeto pertence. O segundo esquema representa cada objeto com valores de gene, e a posição desses genes significa como os objetos são divididos entre as partições. Representações usando esses esquemas se diferem na maneira como os genes são atribuídos e como os valores dos genes são interpretados (COLE, 1989).

A representação Grupo-Número é baseada no primeiro esquema e representa um

agrupamento de n objetos como uma string de n inteiros onde o iésimo inteiro representa o número da partição do iésimo objeto. Quando há apenas duas partições (grupos) essa representação pode ser reduzida para o esquema de representação binária usando 0 e 1 como identificadores das partições (COLE,1989).

(BEZDEK; BOGGAVAPARU; HALL; BENSAID, 1994) utilizam uma matriz $M_{n\times k}$ como representação onde cada linha corresponde a um grupo e cada coluna a um objeto. Se um objeto x_i pertencer ao grupo c_j, então o conteúdo da posição $M(i, j)$ será 1, caso contrário, o mesmo será 0.

Outro tipo de representação utiliza o valor dos genes para representar os objetos e a posição dos objetos no cromossomo para representar a que grupo pertencem. Como os diferentes indivíduos representam diferentes permutações dos objetos, tal representação é denominada representação por permutação. Existem duas variações para essa representação: permutação com separadores e permutação gulosa.

Na representação de permutação com separadores (BELEW; BOOKER, 1991) utilizam valores que não representam objetos para separar um grupo do outro.

(BELEW; BOOKER, 1991) também propõem a representação que necessita de buscas locais, conhecida como permutação gulosa (*greedy permutation*), que utiliza os k primeiros genes como sementes para gerar k grupos, ou seja, cada um dos k primeiros objetos pertence a um grupo distinto e serão utilizados como sementes do mesmo. Cada um dos objetos restantes é inserido no grupo cuja semente apresentar a maior similaridade, na ordem que aparecem na permutação.

Outra representação, denominada representação por centróides, utiliza os centróides (equação 2.8) de cada grupo para compor um cromossomo. Uma matriz $M_{d\times k}$ de valores numéricos representa um grupo, onde d é a dimensão dos objetos e k é o número de grupos. Cada coluna contém o centróide c de um grupo e cada linha representa a posição do mesmo na dimensão d correspondente (HALL; ÖZYURT; BEZDEK, 1999).

A representação utilizada nesse trabalho é a Grupo-Número baseada no primeiro esquema descrito em (COLE, 1989) e na abordagem proposta por (RAGHAVAN; BIRCHARD, 1979), onde a solução é representada como um cromossomo, que corresponde a um vetor de tamanho N (número de exemplos), contendo valores no intervalo $\{1,..., K\}$, indicando a qual partição cada objeto pertence. Assim, um valor j na posição (ou gene) i indica que o elemento x_i do conjunto de dados pertence à partição j. Desta forma, um exemplo

de solução válida para um problema com 3 partições e 7 padrões seria representada pelo cromossomo

2	1	1	3	2	3	1

Figura 4.2 Representação de um cromossomo (indivíduo).

No cromossomo acima o primeiro gene indica que o primeiro padrão pertence à partição 2, o segundo padrão pertence à partição 1, e assim sucessivamente.

4.2.4 Função de Aptidão

A cada geração, o AG seleciona os melhores indivíduos para gerar uma nova população. Para a seleção é necessário atribuir a cada cromossomo uma nota ou valor de aptidão, para tanto é preciso definir uma função de aptidão.

A função de aptidão do AG depende da natureza do problema em questão. A função de aptidão deve ser a que melhor represente o problema, pois tem por objetivo oferecer uma medida de qualidade da solução (*fitness*) que cada indivíduo representa na população corrente, que irá dirigir o processo de busca.

A função objetivo de um algoritmo de agrupamento pode ser utilizada como função de aptidão do AGA. Entretanto, se o objetivo do algoritmo de agrupamento é minimizar a função objetivo para obter bons resultados é necessária uma transformação para que as soluções que apresentam menores valores para função objetivo recebam as maiores notas de aptidão (COLE, 1989).

Neste trabalho a função objetivo (ou função de avaliação) utiliza o critério *SSE* (equação 2.7) que também é utilizado como função de aptidão do AGA implementado. Essa escolha pretende tornar coerente a comparação dos resultados obtidos com o AGA e os outros algoritmos avaliados. Foi necessário implementar uma adaptação no cálculo da função de aptidão para atribuir as maiores notas de aptidão aos indivíduos que apresentam os menores valores da função objetivo. A adaptação utilizada é apresentada na seção 4.3.4.

4.2.5 Processo de Seleção

A sobrevivência dos indivíduos mais aptos resulta da competição entre os indivíduos de uma população, onde os genes dos mais aptos prevalecem em detrimento dos menos aptos, ou seja, a seleção natural leva à sobrevivência dos melhores genes.

A seleção é o processo de escolha dos cromossomos ou indivíduos mais aptos para sofrerem a ação dos operadores genéticos (*crossover* e mutação). A seleção fornece subsídios para que os melhores indivíduos (soluções) sobrevivam. Na população cada indivíduo tem um valor *fitness* associado que mede sua aptidão, ou seja, o quanto uma solução é melhor do que outra. Quanto maior for este valor, maior será a chance da solução correspondente sobreviver e reproduzir-se, e maior será sua representação nas gerações subseqüentes.

Existem várias formas de implementação da seleção como esquemas de proporção, de ordenação e de torneio, mas a idéia é dar preferência aos melhores indivíduos (GOLDBERG, 1989).

O processo de seleção utilizado neste trabalho é o método da Roleta (*Roulette Wheel*) como proposto por (GOLDBERG, 1989). O AG seleciona os melhores cromossomos da população inicial (cromossomos pais) para gerar os descendentes (cromossomos filhos). No método original da roleta os cromossomos pais são selecionados com probabilidade p_i, dada pelo quociente entre o valor da função objetivo do cromossomo i e a soma dos valores da função objetivo de todos os cromossomos da população. Valores maiores de p_i indicam cromossomos mais adaptados e com maior probabilidade de serem escolhidos para sofrerem *crossover*.

Neste trabalho foi necessário implementar uma adaptação no cálculo da função de aptidão para utilizar o método da Roleta, pois as soluções que apresentam menores valores para função objetivo devem receber as maiores notas de aptidão. Tal adaptação é apresentada na seção 4.3.4.

A seleção de indivíduos pelo método da roleta pode fazer com que o melhor indivíduo da população seja perdido, ou seja, não passe para a próxima geração. Uma alternativa é simplesmente manter sempre o melhor indivíduo da geração corrente na geração seguinte, estratégia essa conhecida como seleção elitista ou elitismo (DAVIS, 1991; FOGEL, 1994; MICHALEWICZ, 1996).

Neste trabalho é utilizado o elitismo somente para o melhor indivíduo.

Automaticamente o melhor indivíduo da população corrente é inserido na população seguinte.

Outro exemplo de método de seleção é a seleção baseada em *rank* (BACK; FOGEL; MICHALEWICZ, 1997). Esta estratégia utiliza as posições dos indivíduos quando ordenados de acordo com o *fitness* para determinar a probabilidade de seleção. Podem ser usados mapeamentos lineares ou não-lineares para determinar a probabilidade de seleção. (MICHALEWICZ, 1996) apresenta um exemplo de mapeamento não-linear. Uma variação deste método é simplesmente passar os *N* melhores indivíduos para a próxima geração.

4.2.6 Operador de *Crossover*

O mecanismo de reprodução introduz diversidade na população. O processo de evolução inicia com a combinação de dois indivíduos da população durante a reprodução, gerando novas combinações de genes e criando novos indivíduos na população. O *crossover* (recombinação) é necessário, pois se apenas o mecanismo de seleção for aplicado em AGs, o resultado é a obtenção de novas populações com apenas proporções alteradas de indivíduos da população original (GOLDBERG, 1989).

Segundo (DAVIS, 1991), os operadores genéticos devem ser adaptados ao problema. Isto é, criar estratégias novas de mutação e de *crossover* que são apropriados ao problema em questão.

Devido a importância para o funcionamento do AG, existem várias técnicas diferentes de *crossover* desenvolvidas e analisadas como, por exemplo, *crossover* aritmético, *crossover* de ponto único, *crossover* de 2 pontos, *crossover* uniforme e *crossover* heurístico (MICHALEWICZ, 1996).

Os operadores de *crossover* mais comumente utilizados são os de ponto único e dois pontos, sendo este último considerado uma evolução do *crossover* de ponto único. A escolha do número de pontos de *crossover* pode ser ajustada em 1 ou 2, de forma aleatória nos cromossomos pais, a partir dos quais é feita a troca do material genético.

Para a aplicação do *crossover* de ponto único dois indivíduos (pais) são selecionados e a partir de seus cromossomos são gerados dois novos indivíduos (filhos). Para gerar os filhos é selecionado um mesmo ponto de corte aleatoriamente nos cromossomos pais, os segmentos de cromossomo definidos a partir do ponto de corte são trocados, ou seja, é realizada a troca de material genético entre eles. Considerando, por exemplo, dois indivíduos selecionados a

partir da população inicial e supondo que o ponto de corte escolhido aleatoriamente encontra-se entre os genes 4 e 5 dos cromossomos pais, teremos:

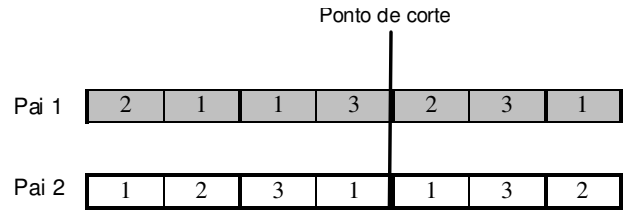

Figura 4.3 Ponto de corte do *crossover* de ponto único.

Após o *crossover* são gerados os seguintes indivíduos (filhos):

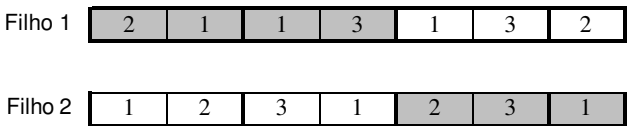

Figura 4.4 Resultado do *crossover* de ponto único.

No exemplo acima é utilizado um único ponto de corte, mas podem ser realizados cortes em mais de um ponto, caracterizando o *multi-pont crossover* (GOLDBERG, 1989; MICHALEWICZ, 1996). Quando dois pontos de corte são utilizados caracteriza-se o *crossover* de dois pontos. Neste trabalho foi utilizado o *crossover* de ponto único com seleção aleatória do ponto de corte.

Outro tipo de *crossover* também muito utilizado é o *crossover* uniforme que permite a troca de bits individuais ao longo do cromossomo, ao invés de grupos de genes (DAVIS, 1991). No *crossover* uniforme, dois indivíduos (pais) são escolhidos para produzirem dois novos indivíduos (filhos). Decide-se com alguma probabilidade fixa p qual pai contribuirá com seus valores de bit para qual filho. Para escolher os genes que serão trocados entre os pais, gera-se um mapa de bits com o mesmo tamanho do cromossomo. As posições com bit 0 são trocadas e as posições com bit 1 permanecem inalteradas. Dessa forma, o *crossover* pode combinar características independentemente da sua posição relativa no cromossomo, ou seja, a troca de genes em uma posição do cromossomo independe da troca ocorrida em outras posições.

O *crossover* aritmético é um operador que faz a combinação linear de dois vetores (MICHALEWICZ, 1996). Sejam x_1 e x_2 dois indivíduos selecionados para *crossover*, então os

dois filhos resultantes serão $x_1^{'} = ax_1 + (1-a)x_2$ e $x_2^{'} = (1-a)x_1 + ax_2$ onde a é um número aleatório pertencente ao intervalo [0,1]. Este operador é particularmente apropriado para problemas de otimização numérica com restrições, onde a região factível é convexa. Isto porque, se x_1 e x_2 pertencem à região factível, combinações convexas de x_1 e x_2 serão também factíveis. Assim, garante-se que o *crossover* não gera indivíduos inválidos para o problema em questão.

O *crossover* heurístico é um operador unário que usa os valores da função objetivo na determinação da direção da busca, e pode produzir ou não apenas um único filho (MICHALEWICZ, 1996). Dados dois cromossomos x_1 e x_2, este operador gera um cromossomo novo como $x_3 = r(x_2 - x_1) + x_2$, sendo r um número aleatório pertencente ao intervalo [0,1], e o cromossomo x_2, não pode ter um desempenho pior do que o cromossomo x_1, ou seja, considerando a função de avaliação f, $f(x_2) \geq f(x_1)$ para problemas de maximização e $f(x_2) \leq f(x_1)$ para problemas de minimização. O operador pode gerar um cromossomo que não seja viável, resultando na geração de um novo valor de r e na criação de um novo cromossomo. Se após várias tentativas a nova solução ainda não for viável, o operador desiste de produzir um novo cromossomo para a nova geração. O *crossover* heurístico contribui para a precisão da solução encontrada, por ter um caráter de sintonia local fina e busca na direção mais promissora (MICHALEWICZ, 1996).

Não há nenhum operador de *crossover* que claramente apresente um desempenho superior aos demais. Uma conclusão a que se pode chegar é que cada operador de *crossover* é particularmente eficiente para uma determinada classe de problemas e ineficiente para outras.

O *crossover* é aplicado com uma dada probabilidade denominada taxa de *crossover*. Na literatura é encontrada uma grande variedade de valores para a taxa de *crossover*, geralmente com alta probabilidade variando entre 65% a 95%.

Neste trabalho a taxa de *crossover* utilizada foi de 80% por corresponder a média aritmética entre os limites inferior e superior do intervalo da taxa de *crossover* sugeridos na literatura.

4.2.7 Operador de Mutação

O operador de mutação foi proposto por (GOLDBERG, 1989) para introduzir na população, constantemente, material genético inovador.

A mutação é outra forma de introduzir diversidade em uma população. As mutações tornam os cromossomos de filhos biológicos diferentes dos cromossomos dos pais (DAVIS, 1991).

(KROVI, 1991) usa a função de mutação implementada por (GOLDBERG, 1989). Nela cada *bit* do cromossomo é invertido com uma probabilidade igual à taxa de mutação, P_{mut}. Neste caso a representação é um cromossomo cujos genes assumem valores binários de 0 ou 1.

(JONES; BELTRAMO, 1991) trocam cada número de grupo com probabilidade, $P_{mut} = 1/n$ onde n representa o número de objetos.

A mutação introduz material genético novo na população. No contexto de agrupamento isso corresponde a mover um padrão de uma partição para outra. Como isso é feito depende da representação do cromossomo (COLE, 1998).

No presente trabalho a representação utilizada é a Grupo-Número que permite a aplicação do operador de mutação uniforme. O operador de mutação utilizado é a mutação uniforme baseada em (MICHALEWICZ, 1996). O operador de mutação uniforme seleciona aleatoriamente um componente $m \in \{1,..., n\}$ do cromossomo $\boldsymbol{x} = [x_1,... x_m..., x_n]$ e gera um indivíduo $\boldsymbol{x'} = [x_1,... x'_m..., x_n]$, onde x'_m é um número aleatório (com distribuição de probabilidade uniforme) amostrado no intervalo $[li, ls]$ onde li e ls são, respectivamente, os limites inferior e superior para o valor do alelo x_m. No problema de agrupamento de dados em questão, onde o valor de k é previamente conhecido, o limite superior é definido pelo valor de k que representa o número de partições. A figura 4.4 ilustra a aplicação do operador de mutação.

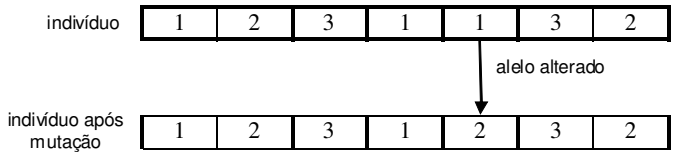

Figura 4.4 Mutação.

Nos AGAs o operador de mutação geralmente tem o efeito de mover um objeto de uma partição para outra. A operação de mutação utilizada neste trabalho contempla tal objetivo.

A mutação é aplicada com uma dada probabilidade denominada taxa de mutação. Tal taxa indica a probabilidade em que haverá a mutação de cromossomos nas populações ao longo da evolução.

Na literatura é encontrada uma grande variedade de valores para a taxa de mutação, assim como os demais parâmetros a taxa ideal dependerá do problema em questão. Entretanto, na literatura a maioria das taxas utilizadas tem baixa probabilidade e varia entre 0,01% a 2%. Neste trabalho a taxa de mutação utilizada foi de 1% por corresponder a média aritmética entre os limites inferior e superior do intervalo da taxa de mutação sugeridos na literatura.

4.2.8 Algoritmo Genético Adaptado ao Problema de Agrupamento

Conforme mencionado na seção 4.2, (COLE, 1989) denomina como Algoritmo Genético de Agrupamento (AGA) o AG adaptado para resolver problemas de agrupamento em *k* partições com o valor de *k* previamente conhecido.

Para que um AG se torne um AGA é necessário que adaptações aconteçam na representação da solução, na função de adaptação, nos operados e nos valores dos parâmetros. Tais adaptações foram descritas nas seções anteriores e são as utilizadas na implementação do AGA avaliado neste trabalho.

O algoritmo 9 apresenta o AGA utilizado neste trabalho, seguindo as características descritas nas seções anteriores.

Algoritmo 9: procedimento do Algoritmo Genético de Agrupamento

---

Entrada: X, conjunto de N padrões com d dimensões;
K, número de centróides;
T_{pop}, tamanho da população;
P_{cros} , taxa de *crossover;*
P_{mut} , taxa de mutação;
N_{ger}, número máximo de gerações;

Saída: melhor solução encontrada s^*

1: $t \leftarrow 0$
2: Gere a população inicial $P(t)$ com tamanho T_{pop}
3: Avalie $P(t)$
4: $s \leftarrow$ selecionar_melhor_individuo$(P(t))$
5: $s^* \leftarrow s$
6: $s_{sse} \leftarrow SSE$ calculado com a solução s^*
7: $sem\_melhora \leftarrow 0$
8: $P_{pai} \leftarrow \emptyset$
9: **Enquanto** ($t < N_{ger}$ e $sem\_melhora < 0.05 * N_{ger}$) **faça**

10: $t \leftarrow t+1$
11: Selecione $P_{pai}(t)$ a partir de $P(t-1)$ com taxa de *crossover* P_{cros}
12: $P(t) \leftarrow$ *crossover* $P_{pai}(t)$
13: Aplique mutação $P(t)$ com taxa de mutação P_{mut}
14: Avalie $P(t)$
15: Aplique elitismo $P(t) \leftarrow$ selecionar_melhor_individuo $P(t-1)$
16: $s' \leftarrow$ selecionar_melhor_individuo $(P(t))$
17: $s'_{sse} \leftarrow$ *SSE* calculado com a solução s'
18: $\Delta_{sse} = s'_{sse} - s_{sse}$
19: **Se** $\Delta_{sse} \geq 0$ **então**
20: $sem\_melhora \leftarrow sem\_melhora + 1$
21: **Senão**
22: $sem\_melhora \leftarrow 0$
23: s* ← s'
24: **Fim Se**
25: **Fim Enquanto**
26: **Retorne** s^*

A representação da solução escolhida para o AGA é baseada na abordagem proposta por (RAGHAVAN; BIRCHARD, 1979), onde a solução é representada por um cromossomo cujos alelos assumem valores inteiros e distintos no intervalo [1, k], sendo k o número de partições, conforme descrito na seção 4.2.3.

O AGA começa gerando a população inicial. O processo de inicialização para gerar a população inicial produz aleatoriamente as soluções. O próximo passo é a avaliação dos indivíduos. O processo de avaliação atribui o valor gerado pela função de aptidão, descrita na seção 4.2.4, para cada indivíduo.

A partir desse ponto o algoritmo entre no seu laço principal onde as populações são geradas através dos operadores genéticos. O AGA seleciona a população de pais utilizando o método da Roleta (*Roulette Wheel*) proposto por (GOLDBERG, 1989) descrito na seção 4.2.5. O *crossover* é aplicado na população pai gerando os cromossomos filhos que compõem a próxima geração. O operador de *crossover* utilizado é o de ponto único, conforme descrito na seção 4.2.6. Após o *crossover* é aplicado o operador de mutação na geração corrente. O operador de mutação utilizado é a mutação uniforme baseada em (MICHALEWICZ, 1996), descrito na seção 4.2.7. Após a mutação a população corrente é avaliada, ou seja, cada cromossomo tem seu valor da função de aptidão atualizado. O próximo passo é aplicar na população corrente o elitismo que seleciona o melhor indivíduo da geração anterior e o insere na geração corrente, caso tal indivíduo seja melhor que o melhor indivíduo da população corrente. No último passo a nova população substitui a população anterior. O AGA repete os

passos dentro de seu laço principal até que um algum critério de parada seja atingido. Como critérios de parada são utilizados o número máximo de gerações ou um dado número de gerações sem melhora no valor da função de avaliação do cromossomo de maior aptidão.

A análise e ajuste dos parâmetros tamanho da população e número máximo de gerações são descritos no capítulo 5.

4.3 Algoritmo Genético Híbrido

Apesar de sua eficiência comprovada para diversos problemas de otimização, o desempenho dos AGs na sua forma original não tem se mostrado satisfatório para muitos problemas de otimização que já possuem algoritmos heurísticos eficientes para a sua solução aproximada.

Por esse motivo, diversos pesquisadores têm proposto modificações nos AGs convencionais, incorporando neles técnicas de busca local (COLE, 1989; MAULIK; BANDYOPADHYAY, 2000; ERICSSON; RESENDE; PARDALOS, 2002; GONÇALVES; MENDES; RESENDE, 2004; RESENDE; GONÇALVES, 2004) ou incluindo nos AGs, conceitos de outras metaheurísticas tais como: *Simulated Annealing*, *Tabu Search*, obtendo versões conhecidas como Algoritmos Meméticos (MOSCATO, 1989) ou simplesmente Algoritmos Evolutivos (LORENA; FURTADO, 2001; OCHI; VIANNA; DRUMMOND, 2001; MESTER; BRAYSY, 2005).

O termo Algoritmo Memético (AM) foi usado na literatura pela primeira vez em (MOSCATO, 1989). O artigo apresenta uma heurística que usa o *Simulated Annealing* para busca local intercalado com o uso de um operador de *crossover.*

Segundo (MEZ, 2000), uma junção de buscas globais e buscas locais são benéficas para o resultado final. Esta combinação de um algoritmo evolucionário com busca local é também conhecida como AM, onde a população é localmente otimizada ao final de cada iteração. Basicamente, o método é uma variação do AG, consistindo no refinamento dos indivíduos antes de se submeterem aos operadores genéticos.

Este trabalho apresentada uma abordagem de Algoritmo Genético Híbrido aplicado ao problema de agrupamento, provido de mecanismo de melhoramento da geração inicial da população, bem como o melhoramento da exploração de regiões promissoras do espaço de

busca, utilizando para tal finalidade a heurística *k-means*.

O algoritmo proposto possui as características básicas dos AGAs, mas incorpora mecanismos adicionais a fim de obter melhores resultados do que o AGA clássico, descrito na seção 4.1.8, e do que as heurísticas de agrupamento descritas e avaliadas neste trabalho.

O Algoritmo Genético Híbrido proposto possui as seguintes particularidades em relação a um AGA básico: a população inicial não é gerada somente de maneira aleatória, uma parte dos indivíduos é gerada através de três versões distintas da heurística *k-means*, com o objetivo de gerar em baixo tempo computacional indivíduos distintos com nível de aptidão melhor do que se fossem gerados aleatoriamente. A cada geração é aplicada a heurística *k-means* nos indivíduos da população corrente, realizando uma busca local eficiente nas soluções geradas com o intuito de encontrar soluções ainda melhores.

4.3.1 Inicialização

Apesar de sua natureza populacional, onde uma grande parte do espaço de busca é avaliada, os Algoritmos Evolutivos, podem ter desempenho melhorado pela geração de uma população inicial menos aleatória. Por esse motivo, mecanismos de melhoramento podem ser introduzidos desde a geração inicial da população. Porém, cuidados devem ser tomados com relação à diversidade populacional, evitando-se que a população inicial seja pouco diversificada sobre o espaço de busca (FELTL; RAIDL, 2004).

No presente trabalho são utilizados quatro algoritmos, dentre eles três versões da heurística *K-means*, no processo de inicialização para compor a população inicial. O *K-means* foi escolhido, pois além de ser um dos algoritmos mais usados para resolver o problema de agrupamento é um procedimento simples e rápido que apresentar bons resultados principalmente quando combinado com métodos de inicialização como o *K-means++* e o *PCA_Part*.

A primeira heurística utilizada no processo de inicialização é o algoritmo *K-means* inicializado pelo método *PCA_Part*, descrito na seção 3.4. Tal heurística por realizar uma escolha determinística contribui com um único indivíduo na população inicial.

A segunda heurística é o algoritmo *K-means* inicializado pelo método *K-means++*, descrito na seção 3.3. A razão para a escolha desse método é a garantia da proximidade

percentual do ótimo global.

A terceira heurística é o algoritmo *K-means* com inicialização aleatória, descrito na seção 3.2. Tal heurística produz resultados diferentes a cada execução garantindo dessa forma a diversidade necessária para que os indivíduos gerados possam abranger o maior espaço de busca possível.

O quarto algoritmo usado no processo de inicialização da população inicial produz aleatoriamente as soluções, dessa forma também contribui para a diversidade da população garantindo um espaço de busca maior.

O processo de inicialização implementado é motivado na necessidade de obter melhores resultados em problemas de agrupamento usando metaheurísticas evolutivas, através da facilidade de convergência para uma solução ótima que uma boa população inicial pode produzir, bem como na redução do tempo de execução do algoritmo.

O objetivo de produzir soluções (indivíduos) iniciais melhores é permitir que a metaheurística comece com um espaço de busca mais próximo ao ótimo global, aumentando as possibilidades de esse ótimo ser alcançado em um número de iterações menor do que o utilizado pela metaheurística com geração da população unicamente aleatória.

4.3.2 Representação da Solução

A representação da solução utilizada no AGHA é a Grupo-Número, descrita na seção 4.2.3, onde a solução é representada por um cromossomo de tamanho N (número de objetos) cujos alelos assumem valores inteiros e distintos no intervalo [1, k], sendo K o número de partições. A figura 4.2 ilustra a representação da solução utilizada.

4.3.3 Função de Aptidão

Segundo (COLE, 1989), a função objetivo de um algoritmo de agrupamento pode ser utilizada como função de aptidão do AGA. No presente trabalho a função objetivo utiliza o critério *SSE* (equação 2.7) que também é utilizado como função de aptidão do AGHA implementado. Tal escolha torna coerente a comparação dos resultados entre o AGHA, o

AGA e os demais algoritmos de agrupamento avaliados.

O cálculo da função de aptidão é adaptado com o intuito de minimizar a função objetivo do AGHA. Tal adaptação torna função objetivo e aptidão grandezas inversamente proporcionais garantindo a atribuição de maiores notas de aptidão aos indivíduos com os menores valores da função objetivo. A adaptação utilizada é apresentada em detalhes na próxima seção.

4.3.4 Processo de Seleção

O processo de seleção utilizado no presente trabalho é o método da Roleta (*Roulette Wheel*) como proposto por (GOLDBERG, 1989). O AGHA seleciona os melhores cromossomos da população inicial (cromossomos pais) para gerar os descendentes (cromossomos filhos). No método original da roleta os cromossomos pais são selecionados com probabilidade p_i, dada pelo quociente entre o valor da função objetivo do cromossomo *i* e a soma dos valores da função objetivo de todos os cromossomos da população, tal probabilidade também é denominada aptidão relativa. Valores maiores de p_i indicam cromossomos mais adaptados e com maior probabilidade de serem escolhidos para sofrerem *crossover*.

Para utilizar o método da roleta foi necessário implementar uma adaptação, pois as soluções que apresentam menores valores para função objetivo (valor do *SSE*) devem receber as maiores notas de aptidão relativa. A adaptação implementada começa pelo cálculo da probabilidade *p* usada no método original da roleta. Tal probabilidade é dada pelo quociente entre o valor da função objetivo do cromossomo *i* e a soma dos valores da função objetivo de todos os cromossomos da população. A fórmula abaixo representada o cálculo da probabilidade *p*:

$$p_i = \frac{SSE_i}{\sum_{i=1}^{c} SSE_i} \tag{4.1}$$

onde *c* representa o número de cromossomos da população.

Depois de calculada a probabilidade *p*, a mesma é utilizada para fazer a adaptação onde os cromossomos com menor valor do critério *SSE* recebem as maiores notas de aptidão.

Tal adaptação é representada pela seguinte fórmula:

$$nota_i = \frac{(1 - p_i)/p_i}{\sum_{i=1}^{c}(1 - p_i)/p_i} \tag{4.2}$$

Onde c representa o número de cromossomos da população e $nota_i$ é o valor da aptidão relativa do cromossomo i, tal valor será usado para representar a fatia que o referido cromossomo ocupa na roleta.

Cada vez que um cromossomo é selecionado, a roleta é girada e o cromossomo correspondente à fatia apontada é escolhido.

O exemplo a seguir ilustra o método da roleta, utilizado neste trabalho, aplicado a uma população de quatro cromossomos.

Cromossomo	*SSE*	Aptidão relativa
C1	12,0	0,49
C2	34,5	0,13
C3	19,0	0,28
C4	39,0	0,10

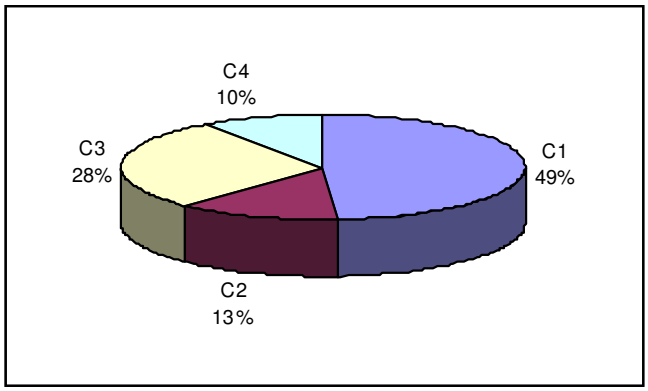

Figura 4.5 Método da roleta adaptado ao problema de agrupamento.

No exemplo acima foi utilizada a fórmula 4.2 para cálculo da função de aptidão relativa dos cromossomos.

Conforme mencionado anteriormente a seleção de indivíduos pelo método da roleta pode fazer com que o melhor indivíduo da população seja perdido, ou seja, não passe para a próxima geração. Uma boa alternativa é simplesmente manter sempre o melhor indivíduo da geração corrente na geração seguinte, estratégia conhecida como seleção elitista ou elitismo (DAVIS, 1991; FOGEL, 1994; MICHALEWICZ, 1996).

Neste trabalho é utilizado o elitismo para o melhor indivíduo. O melhor indivíduo da população corrente é automaticamente inserido na população seguinte, desde que seja melhor que o melhor indivíduo da nova população. Isto garante que a solução gerada pelo AGHA

seja, no mínimo, tão boa quanto a gerada pelos métodos utilizados na geração da população inicial.

4.3.5 Operador de *Crossover*

O *crossover* forma um novo indivíduo através da combinação de genes de outros indivíduos da população. Para tanto, é necessário selecionar os indivíduos que passarão pelo processo de *crossover*, a neste trabalho a seleção é feita conforme descrito na seção anterior, pelo método da roleta.

O *crossover* é aplicado com uma dada probabilidade denominada taxa de *crossover*, o AGHA utiliza a mesma taxa de *crossover* definida na seção 4.2.6.

4.3.6 Operador de Mutação

O operador de mutação é responsável pelo aumento da diversidade da população. No AGA o operador de mutação geralmente tem o efeito de mover um objeto de uma partição para outra. No AGHA e no AGA implementados o operador de mutação utilizado é o mesmo, mutação uniforme, baseada em (MICHALEWICZ, 1996) e descrita na seção 4.2.7.

A mutação é aplicada com uma dada probabilidade denominada taxa de mutação, o AGHA utiliza a mesma taxa de mutação definida na seção 4.2.7.

4.3.7 Algoritmo Genético Híbrido Adaptado ao Problema de Agrupamento

Segundo (DAVIS, 1991), deve-se incorporar tanto quanto possível conhecimento do domínio no AG, bem como combinar o AG com outros métodos de otimização que trabalhem bem na resolução do problema em questão. Na hipótese que há um método usado para resolver um dado problema e o AG pode ser combinado a esse método, tornando-se assim um Algoritmo Genético Híbrido, DAVIS (1991), sugere três princípios de hibridização:

1. Use a codificação corrente, isto é, represente as soluções candidatas da mesma

maneira que elas são representadas pelo método usado.

2. Torne híbrido onde for possível. Isto é, combine características benéficas do algoritmo usado com o AG onde for possível.
3. Adapte os operadores genéticos ao problema. Crie novas formas de mutação e *crossover* que sejam apropriadas à codificação natural do problema.

Nas seções anteriores foram descritos todos os componentes utilizados no AGHA implementado neste trabalho, seguindo os princípios propostos por DAVIS (1991). O algoritmo desenvolvido, além de utilizar um processo de inicialização melhorado através da implementação de quatro algoritmos (descritos na seção 4.2.1) foi combinado à heurística *k-means* para fazer o refinamento dos indivíduos a cada geração e todos os componentes foram adaptados ao problema de agrupamento de dados, tornando o algoritmo um AGHA.

Considerando que, a cada geração a população de um AG tende a convergir para uma crescente uniformidade, com perda significativa de informação sobre o espaço de busca completo. Considerando ainda que a população, em um certa geração, pode possuir uma solução suficientemente próxima, ou vizinha, à solução ótima, entende-se que uma busca baseada em alguma heurística específica poderia encontrar a solução ótima, sem depender tanto das probabilidades de transição de pontos das paisagens de aptidão dos AGs. Existe um consenso que a combinação de forma adequada de algoritmos evolutivos e heurísticas específicas tende a ser significativamente superior a versões canônicas dos AEs, (MOSCATO 1999).

Sem encontrar novas regiões de busca, nem fazer o refinamento adequado das regiões encontradas (os indivíduos), um AG tende a se estabilizar em um ponto, onde indivíduos melhores não são mais encontrados, e há baixa probabilidade de escapar dele. Com o intuito de dispor de um mecanismo efetivo de intensificação de busca, para evitar a situação anterior mencionada, no presente trabalho é inserida a heurística *k-means* como processo de refinamento dos indivíduos a cada geração. Tal procedimento é motivado no bom resultado que algoritmos de busca local obtêm na exploração de regiões específicas do espaço de busca e no consenso de que abordagens híbridas tendem a ser significativamente superiores a versões canônicas dos AEs. Para verificar a suposta superioridade são realizados testes, em bases reais e sintéticas, apresentados no capítulo 5.

No AGHA implementado, o *k-means* utilizado no refinamento dos indivíduos recebe

como centróides iniciais os centróides do indivíduo a ser refinado, desta forma evita-se atribuir centróides aleatórios para o *k-means* que é sensível à escolha inicial dos centróides, garantindo a geração de um indivíduo melhor após o refinamento.

O algoritmo 10 apresenta o AGHA utilizado neste trabalho, seguindo as características descritas nas seções anteriores.

Algoritmo 10: procedimento do Algoritmo Genético Híbrido de Agrupamento

Entrada: X, conjunto de N padrões com d dimensões;
K, número de centróides;
T_{pop}, tamanho da população;
P_{cros} , taxa de *crossover;*
P_{mut} , taxa de mutação;
N_{ger}, número máximo de gerações;

Saída: melhor solução encontrada s^*

1: $t \leftarrow 0$
2: Gere a população inicial $P(t)$ com tamanho T_{pop}
3: Avalie $P(t)$
4: $s \leftarrow$ selecionar_melhor_individuo$(P(t))$
5: $s^* \leftarrow s$
6: $s_{sse} \leftarrow$ *SSE* calculado com a solução s^*
7: $sem\_melhora \leftarrow 0$
8: $P_{pai} \leftarrow \emptyset$
9: **Enquanto** ($t < N_{ger}$ e $sem\_melhora < 0.05 * N_{ger}$) **faça**
10: $t \leftarrow t + 1$
11: Selecione $P_{pai}(t)$ a partir de $P(t-1)$ com taxa de *crossover* P_{cros}
12: $P(t) \leftarrow$ *crossover* $P_{pai}(t)$
13: Aplique mutação $P(t)$ com taxa de mutação P_{mut}
14: **Para todo** $I_i \in P(t)$
15: Aplique BuscaLocal(I_i)
16: **Fim Para**
17: Avalie $P(t)$
18: Aplique elitismo $P(t) \leftarrow$ selecionar_melhor_individuo $P(t-1)$
19: $s' \leftarrow$ selecionar_melhor_individuo $(P(t))$
20: $s'_{sse} \leftarrow$ *SSE* calculado com a solução s'
21: $\Delta_{sse} = s'_{sse} - s_{sse}$
22: **Se** $\Delta_{sse} \geq 0$ **então**
23: $sem\_melhora \leftarrow sem\_melhora + 1$
24: **Senão**
25: $sem\_melhora \leftarrow 0$
26: s* ← s'
27: **Fim Se**
28: **Fim Enquanto**
29: **Retorne** s^*

O AGHA começa gerando a população inicial, conforme descrito na seção 4.3.1. No

processo para gerar a população inicial são utilizados quatro algoritmos: três versões da heurística *K-means,* cujos métodos de inicialização são aleatório, *K-means++* e *PCA_part,* e um algoritmo que gera aleatoriamente as soluções. No próximo passo a população inicial gerada é avaliada. O processo de avaliação atribui um valor de aptidão para cada indivíduo. A função de aptidão utilizada é descrita na seção 4.2.4.

A partir desse ponto o algoritmo entre no seu laço principal onde as populações são geradas através dos operadores genéticos. O AGHA seleciona a população de pais utilizando o método da Roleta (*Roulette Wheel*) proposto por (GOLDBERG, 1989) descrito na seção 4.2.5. O *crossover* de ponto único, descrito na seção 4.2.6, é aplicado na população pai gerando os cromossomos filhos que compõem a próxima geração. Após o *crossover* é aplicado na população corrente o operador de mutação uniforme, descrito na seção 4.2.7. Neste ponto entra o algoritmo de busca local *K-means* que faz o refinamento apropriado em todas as soluções que fazem parte da população corrente. O algoritmo *K-means* utilizado no refinamento dos indivíduos é o descrito na seção 3.2, tal heurística foi escolhida por ser um método simples, rápido e que apresenta bons resultados. Após a busca local a população corrente é avaliada e cada cromossomo tem seu valor de aptidão atualizado. O próximo passo é aplicar na população corrente o elitismo que seleciona o melhor indivíduo da geração anterior e o insere na geração corrente, caso tal indivíduo seja melhor que o melhor indivíduo da população corrente. No último passo a nova população substitui a população anterior. O AGHA repete os passos dentro de seu laço principal até que um algum critério de parada seja atingido. Como critérios de parada são utilizados o número máximo de gerações ou um dado número de gerações sem melhora no valor da função de avaliação do cromossomo de maior aptidão.

A análise e ajuste dos parâmetros tamanho da população e número máximo de gerações são descritos no capítulo 5.

4.4 Algoritmo *Tabu Search*

O algoritmo *Tabu Search*, técnica originada nos trabalhos independentes de (GLOVER, 1986) e (HANSEN, 1986) é um método de busca local e contorna o problema das soluções locais através do uso de estruturas de memória. Tais estruturas de memória permitem uma análise sistemática do domínio das soluções, através do registro do percurso efetuado

pelas últimas iterações e, inclusive, aceitam movimentos de piora para escapar de ótimos locais.

A solução inicial é usada para calcular o valor da função objetivo e inicializar toda estrutura de dados. Abordagens construtivas são freqüentemente usadas no lugar de inicialização randômica, pois, geralmente, fornecem um "ponto de partida melhor" (medido pelo valor da função objetivo), Laguna (1994).

O *Tabu Search* parte de uma solução inicial completa, que pode ser gerada aleatoriamente ou obtida por um método construtivo, e, iterativamente, tenta substituir a solução corrente por uma solução melhor que pertença à vizinhança da solução corrente. A vizinhança de uma solução é definida por uma função N que associa a cada solução $s \in S$ um conjunto de novas soluções $N(s) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s e corresponde a uma solução próxima, sob algum aspecto, à solução s.

O critério de escolha do melhor vizinho é utilizado para escapar de um ótimo local. Esta estratégia, entretanto, pode fazer com que o algoritmo entre em ciclo, ou seja, que retorne a uma solução já gerada anteriormente. De forma a evitar que isto ocorra, existe uma lista tabu de movimentos proibidos. A lista tabu clássica contém os movimentos reversos aos últimos movimentos realizados e funciona como uma fila de tamanho fixo, isto é, quando um novo movimento é adicionado à lista, o mais antigo sai. Assim, na exploração do subconjunto V da vizinhança $N(s)$ da solução corrente s, ficam excluídos da busca os vizinhos que constam na lista tabu.

Através do mecanismo de memória simulado no *Tabu Search*, o algoritmo mantém, de certo modo, o caminho percorrido até a solução corrente. Esta informação é utilizada para guiar o algoritmo no movimento de uma solução para outra solução vizinha no processo de busca pela melhor solução.

A lista tabu se, por um lado, reduz o risco de ciclos, por outro, também pode proibir movimentos para soluções que ainda não foram visitadas (WERRA, 1989). Assim, existe também uma função de aspiração, que é um mecanismo que retira, sob certas circunstâncias, o *status* tabu de um movimento.

Basicamente, o método, que foi projetado para encontrar boas aproximações para a solução ótima global de qualquer problema de otimização, possui três princípios fundamentais: (i) uso de uma estrutura de dados (lista) para guardar o histórico da evolução do processo de busca, (ii) uso de um mecanismo de controle para fazer um balanceamento

entre a aceitação ou não de uma nova configuração, com base nas informações registradas na lista tabu referentes às restrições e aspirações desejadas e (iii) incorporação de procedimentos que alternam as estratégias de diversificação e intensificação (GLOVER, 1997).

A busca por soluções em uma determinada vizinhança é chamada "intensificação", quanto mais promissora a região, mais intensa a busca nesta região. Porém, quando a busca fica "presa" em uma região, a busca deve ser deslocada para uma região ainda não explorada. O procedimento de geração de uma nova solução inicial, mudando a região de busca, é chamado de "diversificação", oferecendo novas opções de soluções do problema. A diversificação permite que o algoritmo faça uma busca mais abrangente em todo o universo de soluções possíveis do problema. A geração de uma nova solução inicial pode ser feita aleatoriamente ou utilizando-se de informações das melhores soluções encontradas até o momento, chamada de "memória adaptativa" por (BERGER, 1999).

O algoritmo 11 apresenta o procedimento geral de busca realizado pelo *Tabu Search*, quando se pretende minimizar a função objetivo.

Algoritmo 11: procedimento geral do *Tabu Search*

Entrada: solução inicial s
Saída: melhor solução encontrada s^*

```
1: s* ← s
2: Enquanto (critério de parada não for satisfeito) faça
3:     Gere um conjunto de soluções vizinhas a partir da atual
4:     Escolha o vizinho de menor custo que não esteja na lista tabu
5:     Atualize lista tabu
6:     Atualize s*
7: Fim Enquanto
8: Retorne s*
```

O algoritmo começa a partir de uma solução inicial s qualquer e a cada iteração gera um conjunto de novas soluções (os vizinhos). Aceita sempre a melhor solução deste conjunto, ela pode ser melhor ou pior que a solução atual. Na memória são guardadas a melhor solução (encontrada desde o início da execução) e uma lista tabu, contendo as soluções mais recentemente visitadas. Estas soluções são proibidas para evitar ciclos. Após atingir o critério de parada (número de iterações, número de iterações sem melhorias ou quando o valor da melhor solução chega a um limite inferior conhecido – ou próximo dele) a solução dada pelo algoritmo é a melhor solução encontrada desde o início da execução.

O método pode também utilizar listas tabu dinâmicas cuja grande vantagem é

minimizar a possibilidade de ocorrência de ciclos (SCHAEFER, 1996).

Os parâmetros principais de controle do Tabu Search são a cardinalidade da lista tabu, a função de aspiração, a cardinalidade do conjunto de soluções vizinhas testadas em cada iteração e o número máximo de iterações ou o número máximo de iterações sem melhora no valor da melhor solução.

A aplicação do algoritmo *Tabu Search* a um problema específico requer uma etapa de avaliação dos valores mais adequados para os parâmetros e uma definição para o processo de geração das novas soluções (vizinhos). Tanto a adaptação do algoritmo para o problema de agrupamento quanto a escolha dos melhores parâmetros e da função de vizinhança que gera as novas soluções (vizinhos) são detalhadas a seguir.

4.4.1 Agrupamento Tabu Search Adaptado ao Problema de Agrupamento

A representação da solução do problema de agrupamento de dados adotada baseia-se na proposta apresentados por (MURTHY; CHOWDHURY, 1996). A solução é representada através de um vetor de tamanho N, onde cada posição i do vetor representa um padrão x do conjunto de dados e possui valores no intervalo $\{1,..., k\}$, indicando a qual partição o padrão x_i pertence. Desta forma, um exemplo de solução válida para um problema com 3 partições e 7 padrões seria representada pelo vetor

(2 3 2 1 3 1 2)

A primeira posição no vetor indica que o primeiro padrão pertence à partição 2, o segundo padrão pertence à partição 3, e assim sucessivamente.

Neste trabalho é utilizado como função de avaliação o critério *SSE* (*Sum of the Squared Euclidian distances*), o mesmo critério usado nos algoritmos descritos no capítulo 3. Essa escolha pretende tornar coerente a comparação dos resultados obtidos.

Outro fator que influencia no resultado do algoritmo é a escolha da função que calcula a vizinhança de uma solução, que depende diretamente do problema a ser resolvido. Um problema que possua um espaço de soluções discreto requer uma função diferente de um problema com espaço de soluções contínuo.

A função de vizinhança adotada neste trabalho escolhe aleatoriamente um padrão do conjunto de dados para ser movido para uma partição diferente da partição corrente. A nova partição para onde o padrão escolhido for movido também é selecionada aleatoriamente.

Nesta abordagem, a solução inicial é passada como parâmetro de entrada para o algoritmo proposto e pode ser escolhida por qualquer método de inicialização. Essa maneira torna o algoritmo mais flexível para funcionar com diferentes métodos de inicialização.

O algoritmo 12 apresenta o procedimento *Tabu Search* adaptado para o problema de agrupamento proposto neste trabalho, seguindo as características mencionadas nesta seção. Essa versão está baseada nos procedimentos apresentados por (GLOVER; LAGUNA, 1997) e (BERGER; GENDROM; POTVIN, 1999).

Algoritmo 12: procedimento *Tabu Search* adaptado ao problema de agrupamento de dados

Entrada: X, conjunto de N padrões com d dimensões
K, número de centróides
Número máximo de iterações
Número de vizinhos gerados a cada iteração
Tamanho da Lista Tabu

Saída: Melhor solução encontrada s^*

1: $s^* \leftarrow s$
2: $SSE^* \leftarrow SSE$ calculado com a solução s^*
3: v $\leftarrow s^*$
4: $SSE \leftarrow \infty$
5: **Enquanto** (critério de parada não for satisfeito) **faça**
6: **Enquanto** (número de máximo de vizinhos não for gerado) **faça**
7: Gere novo vizinho v' a partir de v
8: $SSE' \leftarrow SSE$ calculado com o novo vizinho v'
9: **Se** ($SSE' < SSE$) e ($v' \notin$ Lista Tabu) **então**
10: $mv \leftarrow v'$
11: $SSE_{mv} \leftarrow SSE'$
12: $SSE \leftarrow SSE'$
13: Atualizar Lista Tabu
14: **Se** $SSE_{mv} < SSE^*$ **então**
15: $SSE^* \leftarrow SSE_{mv}$
16: $s^* \leftarrow mv$
17: **Fim Se**
18: **Fim Se**
19: **Fim Enquanto**
20: $v \leftarrow mv$
21: $SSE \leftarrow \infty$
22: **Fim Enquanto**
23: **Retorne** s^*

O algoritmo começa a partir de uma solução inicial aleatória s e gera um conjunto de

novas soluções *s*'. Aceita sempre a melhor solução deste conjunto, ela pode ser melhor ou pior que a melhor solução atual, porém não pode pertencer à lista tabu. As soluções são classificadas de acordo com a função de avaliação, o critério *SSE*, neste caso pretende-se minimizar o valor da função. Quanto menor o valor do *SSE* melhor é a solução.

No algoritmo implementado é usado o conceito da lista tabu clássica que funciona como uma fila de tamanho fixo, ou seja, quando uma nova solução é adicionada à lista, a mais antiga sai.

A estratégia de diversificação utilizada é baseada na técnica de memória adaptativa sugerida por (BERGER, 1999), onde a melhor solução da iteração anterior é passada como solução inicial para a próxima iteração.

A função de vizinhança adotada seleciona aleatoriamente um padrão para ser movido para uma partição, também escolhida aleatoriamente, diferente da partição corrente.

Na memória são guardadas a melhor solução encontrada desde o início da execução e uma lista tabu. Após atingir o critério de parada, número máximo de iterações, a solução dada pelo algoritmo é a melhor solução encontrada desde o início da execução.

O algoritmo apresentado faz uso de parâmetros que precisam ser ajustados para que o processo de busca pela solução ótima seja realizado com eficiência e eficácia. Os parâmetros responsáveis por esse comportamento e utilizados nessa implementação são a cardinalidade da lista tabu, a cardinalidade do conjunto de soluções vizinhas e o número máximo de iterações. Tais parâmetros foram ajustados empiricamente utilizando os resultados da aplicação do algoritmo nas bases de treino como referência. Maiores detalhes são apresentados no capítulo 5.

5 *Experimentos Realizados*

Este capítulo apresenta a avaliação experimental do Algoritmo Genético Híbrido de Agrupamento com os seguintes algoritmos: Hierárquico Aglomerativo, *k-means* inicializado aleatoriamente, *k-means* inicializado com o *k-means++*, *k-means* inicializado com o *PCA_Part,* Algoritmo Genético de Agrupamento e *Tabu Search*.

Para realizar os experimentos foram utilizadas oito bases reais e quinze bases sintéticas. As bases reais foram separadas em bases de treino, contendo 10% do total de exemplos, e bases de teste com 90% dos exemplos. O objetivo de tal separação é utilizar as bases de treino para ajustar os parâmetros do *Tabu Search,* do Algoritmo Genético de Agrupamento e do Algoritmo Genético Híbrido de Agrupamento.

A Tabela 5.1 mostra as características das oito bases reais utilizadas na avaliação experimental, todas disponíveis em repositórios públicos (ASUNCION; NEWMAN, 2007).

Tabela 5.1 Características das bases usadas na avaliação experimental.

Nome da Base	nº de exemplos	nº de atributos	nº de classes
Iris	150	4	3
Wine	178	13	3
Vehicle	846	18	4
Cloud	1024	10	10
Segmentation	2310	19	7
Spam	4601	57	2
Pendigits	10992	16	10
Letter	20000	16	26

Entre as oitos bases reais utilizadas, *Iris*, *Wine* e *Vehicle* apresentam menor complexidade e as demais apresentam maior complexidade. Tal diversidade permite avaliar em quais tipos de problema a aplicação do AGHA é mais favorável e em quais não é.

Conforme mencionado, as oito bases reais foram divididas em bases de treino e teste. A Tabela 5.2 mostra as características das bases de treino, com 10% do total de exemplos, utilizadas para ajustes dos parâmetros do *Tabu Search*, AG e AGHA.

Tabela 5.2 Características das bases de treino.

Nome da Base	nº de exemplos	nº de atributos	nº de classes
Iris_treino	15	4	3
Wine_treino	17	13	3
Vehicle_treino	84	18	4
Cloud_treino	102	10	10
Segmentation_treino	231	19	7
Spam_treino	460	57	2
Pendigits_treino	1099	16	10
Letter_treino	2000	16	26

A Tabela 5.3 mostra as características das oito bases de teste, com 90% do total de exemplos, utilizadas na avaliação experimental.

Tabela 5.3 Características das bases de teste.

Nome da Base	nº de exemplos	nº de atributos	nº de classes
Iris_treino	135	4	3
Wine_treino	161	13	3
Vehicle_treino	762	18	4
Cloud_treino	922	10	10
Segmentation_treino	2079	19	7
Spam_treino	4141	57	2
Pendigits_treino	9893	16	10
Letter_treino	18000	16	26

Com o intuito de realizar uma maior bateria de testes e ter mais resultados para avaliar estatisticamente, foram geradas quinze bases sintéticas a partir das bases reais *Iris*, *Wine* e *Cloud*. As bases *Iris* e *Wine* foram escolhidas como bases geradoras por apresentarem menor complexidade facilitando o processo de geração das bases sintéticas. Entre as bases de maior complexidade a base *Cloud* foi escolhida por apresentar atributos cujas características são propícias à geração de novos exemplos.

Segundo (RADIN, 2001), o uso de bases sintéticas é uma alternativa muito pouco usada, que expande o poder de algumas fontes de dados reais disponíveis através de testes em variações aleatórias. A macroestrutura da aplicação real é preservada, mas os detalhes são mudados aleatoriamente para produzir novos exemplos.

Para as três bases geradoras (*Iris*, *Wine* e *Cloud)* foi utilizado o mesmo método de geração das bases sintéticas. O primeiro passo foi definir o limite inferior e superior dos valores de cada característica para no segundo passo gerar aleatoriamente os valores dentro do intervalo previamente definido. Através desse método foram geradas quinze bases sintéticas, cinco bases sintéticas derivadas de cada uma das três bases geradoras. A Tabela 5.4 mostra as características das quinze bases sintéticas utilizadas na avaliação experimental.

Tabela 5.4 Características das bases sintéticas.

Nome da base	nº de exemplos	nº de atributos	nº de classes
Iris1	540	4	3
Iris2	945		
Iris3	1350		
Iris4	1755		
Iris5	2160		
Wine1	641	13	3
Wine2	1122		
Wine3	1602		
Wine4	2083		
Wine5	2564		
Cloud1	1844	10	10
Cloud2	3687		
Cloud3	5530		
Cloud4	7373		
Cloud5	9216		

Dentre as quinze bases sintéticas utilizadas, três apresentam menor complexidade (*Iris1, Iris2* e *Wine1*) e as demais possuem complexidade maior.

A seção 5.1 descreve o procedimento utilizado para ajustar os parâmetros do *Tabu Search*, do AG e do AGHA. A seção 5.2 apresenta os resultados obtidos com os diferentes algoritmos aplicados às bases reais de teste. A seção 5.3 apresenta os resultados obtidos com os diferentes algoritmos aplicados às bases sintéticas.

5.1 Análise e Ajustes dos Parâmetros

Conforme mencionado, as bases reais foram divididas em bases de treino e teste. As bases de treino são utilizadas para ajustar os parâmetros dos algoritmos *Tabu Search*, AGA e AGHA implementados.

Para o *Tabu Search* foram realizados treinos com três valores para os parâmetros: número de iterações, número de vizinhos e tamanho da lista tabu. A Tabela 5.5 apresenta os parâmetros selecionados através dos testes realizados com as bases de treino, tais parâmetros são utilizados com as bases de teste.

Tabela 5.5 Resultados das bases de treino com parâmetros selecionados para o Tabu Search.

Nome da Base	K	nº de iterações	nº de vizinhos	tamanho lista tabu	SSE	Tempo (s)
iris_treino	3	1000	500	200	5,052	13,51
wine_treino	3	600	300	150	433640	28,46
vehicle_treino	4	400	200	80	512954	37,73
cloud_treino	10	400	200	80	2,102E+06	47,52
segmentation_treino	7	400	200	80	2,729E+06	98,38
spam_treino	2	400	200	80	4,473E+07	189,81
pendigits_treino	10	200	150	50	5,430E+06	451,72
letter_treino	26	150	100	50	61400,8	1898,26

Para o AGA e o AGHA foram realizados treinos com quatro valores para o parâmetro tamanho da população e dois valores para o parâmetro número de gerações. Nos testes realizados com as bases de treino o critério de parada utilizado inclui duas condições: número máximo de gerações ou 5% do número máximo de gerações sem melhora no valor da função de avaliação do melhor indivíduo da população. Na maioria dos testes o segundo critério de parada foi satisfeito antes do primeiro. A Tabela 5.6 apresenta os resultados obtidos com as bases de treino com o número de gerações igual a 600. Os melhores resultados estão destacados em negrito.

Tabela 5.6 Resultados com as bases de treino para número de geração igual a 600.

Nome da Base	Número de Gerações: 600							
	Número de indivíduos da população							
	20		40		80		160	
	SSE	tempo (s)	SSE	tempo (s)	SSE	tempo (s)	SSE	tempo (s)
Iris_treino	6,29833	0,05	5,052	0,08	**5,052**	0,26	**5,052**	1,30
Wine_treino	233756	0,08	233756	0,15	**233756**	0,38	**233756**	1,42
Vehicle_treino	218092	0,48	218092	1,05	**218092**	1,98	**218092**	4,67
Cloud_treino	358551	1,27	356223	3,00	354279	4,72	**353088**	10,11
Segmentation_treino	1,369E+07	48,21	1,348E+07	95,90	1,346E+07	18,95	**1,343E+07**	30,78
Spam_treino	3,623E+07	5,45	3,623E+07	10,95	**3,623E+07**	19,63	**3,623E+07**	37,28
Pendigits_treino	5,044E+06	21,99	5,024E+06	54,35	**5,024E+06**	73,66	**5,024E+06**	147,76
Letter_treino	60077,4	267,83	60066,4	670,55	60029	933,37	**60025,2**	1415,46

Nos resultados apresentados na Tabela 5.6, o número máximo de gerações é igual a 600 e o valor do número de gerações sem melhora é igual a 30, ou seja, depois de 30 gerações seguidas sem melhora no valor da função de avaliação do melhor indivíduo o algoritmo é encerrado. Foram utilizados quatro valores para tamanho da população: 20, 40, 80 e 160. Os melhores resultados foram obtidos com as populações de tamanho 80 e 160 indivíduos, sendo que 5 dos 8 resultados foram iguais para as duas populações. Apenas três resultados foram melhores para a população de 160 indivíduos, nas bases *Cloud_treino*, *Segmentation_treino* e

Letter_treino, sendo a diferença entre esses três resultados pouco significativa. Entretanto, levando-se em consideração o tempo de execução do algoritmo, o parâmetro tamanho da população igual a 160 torna a execução muito mais custosa. As diferenças entre os tempos de execução para população igual a 80 e 160 são bem significativas, portanto optou-se em utilizar tamanho da população igual a 80 no AGA e no AGHA.

Com o intuito de melhorar o ajuste do parâmetro número de gerações, foram realizados testes usando o valor 1000 para o referido parâmetro, neste caso o valor do número de gerações sem melhora é igual a 50, que representa 5% de 1000. A Tabela 5.7 apresenta os resultados obtidos com as bases de treino usando número de gerações igual a 1000.

Tabela 5.7 Resultados com as bases de treino para número de geração igual a 1000.

Nome da Base	1000 Gerações							
	Número de indivíduos da população							
	20		40		80		160	
	SSE	tempo (s)	SSE	tempo (s)	SSE	tempo (s)	SSE	tempo (s)
Iris_treino	5,052	0,055	5,052	0,12	**5,052**	0,42	**5,052**	2,28
Wine_treino	233756	0,095	233756	0,21	**233756**	0,60	**233756**	2,51
Vehicle_treino	218092	0,733	218092	1,45	**218092**	3,03	**218092**	7,58
Cloud_treino	355598	1,852	354660	3,44	353509	6,84	**352382**	13,93
Segmentation_treino	9,952E+05	4,445	9,952E+05	8,51	**9,952E+05**	14,45	**9,952E+05**	28,92
Spam_treino	3,623E+07	8,128	3,623E+07	14,43	**3,623E+07**	28,83	**3,623E+07**	57,22
Pendigits_treino	5,036E+06	36,564	5,024E+06	88,12	**5,024E+06**	123,76	**5,024E+06**	243,51
Letter_treino	60048,6	437,421	60075	897,01	**59980,5**	1235,53	60026	1564,75

Com os resultados obtidos foi possível observar que houve uma melhora nos valores apresentados pela população igual a 80, onde 6 dos 8 resultados foram iguais para as populações de 80 e 160 indivíduos e o resultado da base *Letter_treino* foi melhor para a população de 80 indivíduos. Outro ponto a favor da população de 80 indivíduos é o tempo de execução, bem menor em relação ao tempo gasto na execução da população de 160 indivíduos. Como é possível observar os resultados obtidos com o número de gerações igual a 1000 é melhor do que os apresentados com o número de gerações igual a 600 sem, no entanto, afetar de forma abrupta o tempo de execução. Para uma melhor visualização a Tabela 5.8 apresenta os resultados utilizando como parâmetros a população igual a 80 e o número de gerações iguais a 600 e 1000.

Tabela 5.8 Resultados para população igual a 80 e gerações iguais a 600 e 1000.

Nome da Base	Número de indivíduos da população: 80			
	600 Gerações		1000 Gerações	
	SSE	tempo (s)	SSE	tempo (s)
Iris_treino	**5,052**	0,26	**5,052**	0,42
Wine_treino	**233756**	0,38	**233756**	0,60
Vehicle_treino	**218092**	1,98	**218092**	3,03
Cloud_treino	354279	4,72	**353509**	6,84
Segmentation_treino	1,346E+07	18,95	**9,952E+05**	14,45
Spam_treino	**3,623E+07**	19,63	**3,623E+07**	28,83
Pendigits_treino	**5,024E+06**	73,66	**5,024E+06**	123,76
Letter_treino	60029	933,37	**59980,5**	1235,53

É possível verificar que todos os resultados para número de gerações igual a 1000 foram melhores ou iguais do que os resultados para número de gerações igual a 600. É importante destacar que o valor do número de gerações influencia tanto na qualidade dos resultados quanto no tempo de execução do algoritmo, por isso é preciso avaliar o custo *versus* benefício do valor do referido parâmetro. No presente trabalho optou-se por utilizar o valor do número de gerações igual a 1000, por entender que o aumento no tempo de execução em relação ao número de gerações igual a 600 foi pouco significativo em relação ao ganho na qualidade dos resultados.

Os parâmetros definidos e utilizados no AGA e no AGHA foram: tamanho da população igual a 80 e número máximo de gerações igual a 1000.

Os parâmetros taxa de *crossover* e de mutação foram definidos a partir dos valores propostos na literatura, conforme descrito no capítulo 4 nas seções 4.1.6 e 4.1.7. A taxa de *crossover* utilizada é de 80% e a taxa de mutação igual a 1%.

5.2 Resultados com as Bases Reais

A Tabela 5.9 apresenta os resultados obtidos com a aplicação do AG, AGHA, Hierárquico Aglomerativo, *Tabu Search* e as três versões do *K-means,* inicializado aleatoriamente, com o *K-means++* e com o *PCA_Part,* às bases citadas na Tabela 5.3. Para cada base foram realizados experimentos com três valores diferentes de *K*, sendo um dos valores igual ao número de classes existentes nos dados. Os melhores resultados estão destacados em negrito.

A variação do valor de *K* influencia diretamente no valor do critério de avaliação do agrupamento. O valor do critério de avaliação decresce naturalmente com o aumento do valor de *K*. Para o valor de *K* igual ao número de elementos do conjunto de dados, cada elemento representa um centróide, tornando o critério de avaliação igual a zero.

Com o intuito de atenuar a influência da aleatoriedade os algoritmos *Tabu Search*, AG, AGHA, *k-means* com inicialização aleatória e pelo *k-means*++, foram executados dez vezes para cada base de teste. A partir das dez execuções foram calculadas as médias dos tempos de execução, as médias dos valores do critério *SSE*, bem como o desvio padrão para cada valor do *SSE*.

Tabela 5.9 *SSE* médio obtido com as dez execuções dos algoritmos avaliados para cada base.

Base	k	Hier. Aglo.	kmeans	Tabu Search	kmeans (kmeans++)	kmeans (*PCA_Part*)	Genético	Genético Híbrido
Iris	2	142,505	**140,015**	156,687	**140,015**	**140,015**	140,606	**140,015**
	3	73,4843	78,6348	73,63	72,8742	72,90	75,7637	**72,86**
	4	62,6795	54,38	60,00	52,98	**51,45**	52,98	**51,45**
Wine	2	4,094E+06	4,079E+06	4,674E+06	4,078E+06	4,079E+06	4,088E+06	**4,077E+06**
	3	2,549E+06	2,134E+06	2,496E+06	2,237E+06	2,331E+06	2,120E+06	**2,096E+06**
	4	2,262E+06	1,202E+06	1,722E+06	1,210E+06	1,201E+06	1,224E+06	**1,195E+06**
Veh.	3	5,174E+06	4,559E+06	5,322E+06	4,612E+06	4,506E+06	4,645E+06	**4,505E+06**
	4	4,974E+06	3,489E+06	4,301E+06	3,276E+06	3,286E+06	3,459E+06	**3,226E+06**
	5	3,317E+06	2,434E+06	3,723E+06	2,257E+06	**2,138E+06**	2,756E+06	**2,138E+06**
Cloud	9	8,693E+06	7,668E+06	1,602E+07	6,576E+06	6,539E+06	9,198E+06	**6,321E+06**
	10	8,512E+06	6,762E+06	1,425E+07	5,719E+06	5,350E+06	8,466E+06	**5,228E+06**
	11	8,485E+06	6,545E+06	1,254E+07	5,169E+06	4,673E+06	7,936E+06	**4,472E+06**
Segm.	6	3,623E+07	1,519E+07	1,666E+07	1,431E+07	1,351E+07	1,849E+07	**1,339E+07**
	7	3,607E+07	1,400E+07	1,497E+07	1,297E+07	1,203E+07	1,673E+07	**1,188E+07**
	8	3,593E+07	1,190E+07	1,375E+07	1,152E+07	1,066E+07	1,640E+07	**1,052E+07**
Spam	2	1,397E+09	8,98888E+08	9,478E+08	1,074E+09	8,98888E+08	9,166E+08	**8,98883E+08**
	3	1,331E+09	5,999E+08	7,576E+08	5,314E+08	5,999E+08	7,102E+08	**5,020E+08**
	4	1,053E+09	5,065E+08	6,252E+08	3,689E+08	5,065E+08	6,025E+11	**3,109E+08**
Pend.	9	1,158E+08	4,897E+07	4,807E+07	4,788E+07	4,799E+07	5,514E+07	**4,722E+07**
	10	1,080E+08	4,574E+07	4,427E+07	4,537E+07	4,485E+07	5,163E+07	**4,422E+07**
	11	1,075E+08	4,403E+07	4,301E+07	4,285E+07	4,335E+07	4,981E+07	**4,185E+07**
Letter	25	1,308E+06	563782,0	561161,0	564735,0	561189,0	652184,0	**557366,0**
	26	1,302E+06	555925,0	551063,0	554031,0	555722,0	647331,0	**549394,0**
	27	1,304E+06	549175,0	544910,0	549874,0	548137,0	632960,0	**541938,0**

Uma análise menos rigorosa dos resultados apresentados na Tabela 5.9 permite constatar que o AGHA obteve os melhores resultados em todos os 24 testes com apenas 3 empates, indicando uma possível superioridade da metaheurística em relação aos demais algoritmos avaliados, embora não seja uma comprovação estatística.

É importante destacar que mesmo na base maior (*Letter*) o desempenho do AGHA foi sistematicamente superior aos demais algoritmos avaliados, indicando que a metaheurística

apresenta um desempenho satisfatório em bases grandes.

Para complementar a avaliação foi realizada uma análise do tempo de execução dos algoritmos, a fim de quantificar a diferença entre a aplicação do AGHA que realiza um procedimento de busca global associado com busca local e dos demais algoritmos. A Tabela 5.10 apresenta o tempo de execução dos algoritmos avaliados e a Tabela 5.11 o desvio padrão em relação ao critério *SSE*. Cada algoritmo, não determinístico, foi executado dez vezes para cada problema, assim os resultados apresentados na Tabela 5.10 se referem à média do tempo das dez execuções. O tempo é apresentado em segundos.

Tabela 5.10 Média do tempo das dez execuções dos algoritmos avaliados para cada base de teste.

Nome da Base	k	Hier. Aglo.	kmeans	Tabu Search	kmeans (kmeans++)	kmeans (*PCA_Part*)	Genético	Genético Híbrido
Iris	2	0,07	< 0,01	55,9	< 0,01	< 0,01	0,99	1,0
	3	0,06	< 0,01	57,4	< 0,01	< 0,01	1,52	1,4
	4	0,06	< 0,01	62,7	< 0,01	< 0,01	2,12	1,6
Wine	2	0,20	< 0,01	78,2	< 0,01	< 0,01	1,78	2,6
	3	0,20	0,01	81,5	< 0,01	< 0,01	2,31	3,2
	4	0,15	0,02	99,9	0,02	< 0,01	4,15	4,2
Veh.	3	15,77	0,15	145,2	0,15	0,01	10,47	25,3
	4	15,96	0,16	175,1	0,11	0,01	14,04	31,9
	5	15,41	0,13	203,2	0,14	0,02	19,18	43,5
Cloud	9	18,63	0,46	241,4	0,17	0,02	20,72	100,2
	10	18,65	0,57	260,6	0,20	0,09	19,31	106,8
	11	18,63	0,51	283,0	0,16	0,04	18,56	125,7
Segm.	6	322,16	0,57	631,7	0,43	0,09	30,00	242,2
	7	321,83	0,69	712,4	0,53	0,09	40,59	200,6
	8	322,12	0,61	798,8	0,77	0,20	38,62	214,6
Spam	2	6267,00	0,87	1541,9	0,54	0,24	92,90	357,3
	3	6120,00	1,69	2012,2	0,63	0,43	95,66	670,6
	4	6185,00	2,93	2496,0	0,87	0,56	150,03	1026,2
Pend.	9	27599,00	4,08	1334,0	4,21	0,59	220,67	1042,9
	10	30946,00	4,24	1451,9	4,72	0,37	279,85	1082,7
	11	31765,00	4,39	1580,4	4,49	0,67	239,63	1186,3
Letter	25	192900,00	42,42	3066,8	48,30	11,22	595,06	17093,2
	26	192840,00	53,24	3183,1	42,12	6,98	1000,15	19701,7
	27	192540,00	50,27	3306,4	47,52	7,78	684,17	19440,1

Tabela 5.11 Desvio padrão do critério *SSE* das bases de teste.

Nome da Base	k	kmeans	Tabu Search	kmeans (kmeans++)	Genético	Genético Híbrido
Iris	2	< 0,01	3,2	0,03	0,5	< 0,01
	3	18,2	0,2	0,02	2,5	< 0,01
	4	6,1	0,7	4,55	0,7	< 0,01
Wine	2	774613,0	83783,1	1291,0	13957,4	< 0,01
	3	75244,0	38203,2	121109,0	31039,8	< 0,01
	4	3210,5	34590,0	10244,0	36548,0	< 0,01
Veh.	3	168583,0	80684,5	224718,0	81538,7	< 0,01
	4	429268,0	92560,3	43714,1	94602,5	< 0,01
	5	311343,0	114904,0	249063,0	273549,0	< 0,01
Cloud	9	883680,0	913998,0	354413,0	1,60E+06	30144,1
	10	308225,0	1,11E+06	330037,0	902545,0	29517,3
	11	36323,7	975484,0	454038,0	1,26E+06	6451,0
Segm.	6	2,28E+06	66830,7	3,90E+05	1,83E+06	64989,0
	7	2,50E+06	205145,0	4,98E+05	1,57E+06	116930,0
	8	184308,0	185102,0	6,82E+05	1,47E+06	31339,9
Spam	2	< 0,01	2,43E+06	1,50E+08	1,46E+07	< 0,01
	3	< 0,01	2,73E+06	4,73E+07	5,95E+07	< 0,01
	4	< 0,01	4,35E+06	7,79E+07	5,74E+07	< 0,01
Pend.	9	2,44E+06	1212,0	8,57E+05	2,99E+06	0,9
	10	798905,0	911,4	1,44E+06	2,00E+06	36,8
	11	1,12E+06	4944,3	1,28E+06	1,66E+06	11,9
Letter	25	3967,9	502,8	2647,0	6490,2	398,4
	26	3963,5	399,8	2325,9	8541,5	363,9
	27	3764,9	1765,7	3010,5	7761,6	446,8

Todos os algoritmos avaliados foram implementados na linguagem C e os testes realizados em uma máquina com a seguinte configuração: Processador AMD Athlon 64 X2 Dual Core 3800+, Cache 512 KB; Placa mãe ASUS M2NPV-VM; 2 GB de memória RAM DDR2 533; HD SATA de 100 GB; Sistema operacional Ubuntu 6.4.0.

Os resultados apresentados na Tabela 5.10 mostram que o melhor tempo de execução para todas as bases é do algoritmo *K-means* inicializado com o *PCA_Part*, o tempo de execução pequeno se deve ao fato da boa escolha dos centróides iniciais através do método *PCA_Part*, pois assim o *K-means* precisa de poucas iterações para alcançar a solução final. Observa-se ainda que, para as duas bases menores (*Iris* e *Wine*) a diferença dos tempos de execução das três versões do *K-means* é muito pequena, a influência da boa escolha dos centróides iniciais começa a aparecer de forma significativa nas bases maiores.

O algoritmo Hierárquico Aglomerativo apresentou tempos de execução pequenos para as bases pequenas, porém para as bases maiores o algoritmo obteve os maiores tempo de execução entre todos os algoritmos avaliados. Tal resultado já era esperado, pois a complexidade computacional para a maioria dos algoritmos hierárquicos é pelo menos $O(n^2)$ e

este custo elevado limita sua aplicação em bases muito grandes.

A diferença entre o tempo do *Tabu Search* e os demais algoritmos se deve ao fato de ser o *Tabu Search* um método de busca local que a cada iteração altera um único elemento da solução corrente, elevando seu tempo de execução.

O AG e o AGHA apresentam tempos parecidos quando aplicados nas bases menores (*Iris* e *Letter*) a diferença significativa entre os tempos começam a aparecer a partir das bases maiores, tal fato se deve a busca local realizada em todos os indivíduos a cada nova geração no AGHA. Uma forma de tentar reduzir o tempo de execução do AGHA seria a aplicação de estratégias para identificar regiões promissoras, possibilitando limitar a aplicação de busca local a certo grupo de indivíduos da população. De um modo geral, apenas um certo percentual da população considerado de indivíduos-elite, sofre melhorias heurísticas (YEN; LEE, 1997).

Os resultados apresentados na tabela 5.11 mostram que o AGHA apresentou menor dispersão em torno da média dos valores do critério *SSE*, indicando uma maior homogeneidade entre os 10 resultados do algoritmo para cada base.

5.2.1 Avaliação Experimental com Métodos Estatísticos

Para realizar uma avaliação mais precisa e rigorosa dos experimentos, optou-se por fazer uma análise estatística dos resultados. Para realizar tal avaliação os métodos estatísticos de comparações múltiplas são os mais apropriados, pois se pretende avaliar o valor médio do critério de avaliação obtido com vários algoritmos em diferentes bases.

Em métodos de múltiplas comparações, espera-se mostrar que existem diferenças entre os algoritmos com nível de significância igual a $\alpha\%$. O nível de significância indica a probabilidade de uma amostra de dados aleatória gerar o resultado atual, supondo que os algoritmos obtêm resultados iguais. A suposição de que os algoritmos obtêm resultados iguais representa a hipótese nula para a avaliação estatística em questão.

O nível de significância representa a probabilidade de rejeitar a hipótese nula quando ela é verdadeira, ou seja, cometer um erro do Tipo I. O nível de significância deve ter um valor ajustado para garantir uma baixa probabilidade de ocorrência de erro do Tipo I. Geralmente o nível de significância utilizado é de 5%, sendo esse o valor utilizado na avaliação estatística no presente trabalho.

Segundo (DEMSAR, 2006), o método estatístico mais adequado para comparação de algoritmos em múltiplos domínios é o *Teste de Friedman* (FRIEDMAN, 1940). O método testa se em c ≥ 2 experimentos diferentes e dependentes, pelo menos dois são estatisticamente diferentes. Para tanto, o método ordena os algoritmos para cada problema separadamente, atribuindo um posto (*rank*) a cada um deles com valores de 1 a *c*. O algoritmo com melhor resultado recebe o posto 1, o segundo melhor recebe o posto 2, e assim sucessivamente. Os algoritmos que apresentam resultados iguais recebem a média dos postos que seriam atribuídos a eles.

O *Teste de Friedman* compara a média dos postos R_j, onde $j = 1, ..., c$ dos c algoritmos em todos os n problemas avaliados. Com a hipótese nula de que todos os algoritmos são iguais, a estatística de *Friedman*

$$X_r^2 = \frac{12n}{c(c+1)}\left[\sum_{j=1}^{c} R_j^2 - \frac{c(c+1)}{4}\right] \tag{5.1}$$

segue uma distribuição X^2 com $(c - 1)$ graus de liberdade, se os valores de c e n forem suficientemente grandes.

(IMAN; DAVENPORT, 1980) apresentam uma estatística derivada da estatística de *Friedman* para corrigir o excesso de conservadorismo da versão original:

$$F_r = \frac{(n-1)X_r^2}{n(c-1) - X_r^2} \tag{5.2}$$

onde F_r segue uma distribuição F_{n_1, n_2} com $n_1 = (c - 1)$ e $n_2 = (c - 1)(n - 1)$ graus de liberdade. A hipótese nula de que todos os algoritmos são iguais é rejeitada se o valor calculado por F_r for maior do que o valor tabelado de F_{n_1, n_2}, isto significa que F_r pertence à região crítica e possui probabilidade menor do que o nível de significância desejado.

Quando a hipótese nula é rejeitada, a hipótese alternativa é aceita, indicando que pelo menos dois algoritmos são estatisticamente diferentes entre si. Nesse caso, prossegue-se com o teste para verificar quais pares de algoritmos são diferentes.

O *Teste de Nemenyi* (NEMENYI, 1963) é usado para identificar a diferença significativa entre os algoritmos quando todos são comparados entre si, ou seja, não há um algoritmo como referência com o qual os demais serão comparados. Nesse teste dois algoritmos i e j são considerados significativamente diferentes se as médias dos postos correspondentes (R_i e R_j) diferem pelo menos de uma diferença crítica igual a

$$CD = q_\alpha \sqrt{\frac{c(c+1)}{6n}} \tag{5.3}$$

onde os valores críticos q_α são baseados na distribuição *t* dividida por $\sqrt{2}$.

Na análise foram utilizados os resultados de cada base obtidos quando *K* é igual ao número de classes dos dados, já que se fossem utilizados os testes sobre a mesma base, mesmo com valores diferentes de *K*, os problemas não representariam amostras independentes, prejudicando a análise do teste estatístico.

Além disso, como os algoritmos são rodados sobre as mesmas bases de teste os experimentos são considerados dependentes, satisfazendo as restrições exigidas para o *Teste de Friedman*. A análise estatística dos resultados é apresentada a seguir.

A Tabela 5.12 apresenta os valores médios do critério de avaliação *SSE*, encontrados com as dez execuções de cada algoritmo para cada base. O posto atribuído pelo *Teste de Friedman* é apresentado entre parênteses e os postos médios na última linha da tabela.

Tabela 5.12 Comparação entre os algoritmos avaliados usando diferentes bases.

Base	Hierarquico Aglomerativo	kmeans	Tabu Search	kmeans (kmeans++)	kmeans (pca_part)	Genético	Genético Híbrido
Iris	73,48 (5)	78,63 (7)	73,63 (6)	72,87 (2)	72,90 (4)	75,76 (6)	72,86 (1)
Wine	2,549E+06 (7)	2,134E+06 (3)	2,496E+06 (6)	2,237E+06 (4)	2,331E+06 (5)	2,120E+06 (2)	2,096E+06 (1)
Veh.	4,974E+06 (7)	3,489E+06 (5)	4,301E+06 (6)	3,276E+06 (2)	3,286E+06 (3)	3,459E+06 (4)	3,226E+06 (1)
Cloud	8,512E+06 (6)	6,762E+06 (4)	1,425E+07 (7)	5,719E+06 (3)	5,350E+06 (2)	8,466E+06 (5)	5,228E+06 (1)
Seg.	3,607E+07 (7)	1,400E+07 (4)	1,497E+07 (5)	1,297E+07 (3)	1,203E+07 (2)	1,673E+07 (6)	1,188E+07 (1)
Spam	1,397E+09 (7)	8,98888E+08 (2,5)	9,478E+08 (5)	1,074E+09 (6)	8,98888E+08 (2,5)	9,166E+08 (4)	8,98883E+08 (1)
Pen.	1,080E+08 (7)	4,574E+07 (5)	4,427E+07 (2)	4,537E+07 (4)	4,485E+07 (3)	5,163E+07 (6)	4,422E+07 (1)
Letter	1,302E+06 (7)	555925,0 (5)	551063,0 (2)	554031,0 (3)	555722,0 (4)	647331,0 (6)	549394,0 (1)
p.m.	6,625	4,4375	4,875	3,375	3,1875	4,875	1

A avaliação experimental com métodos estatísticos em múltiplos domínios consiste de duas etapas. A primeira etapa consiste em verificar a hipótese nula de que todos os algoritmos são estatisticamente iguais com base nos resultados apresentados na Tabela 5.11. O resultado do *Teste de Friedman* realizado com os valores da Tabela 5.11, com nível de significância de 5%, indica que a hipótese nula pode ser rejeitada. Dessa forma a hipótese alternativa é aceita, ou seja, existe pelo menos um par de algoritmos estatisticamente diferentes.

A segunda etapa consiste em identificar quais pares de algoritmos possuem uma diferença significativa. Para o *Teste de Nemenyi* com nível de significância de 5% dois algoritmos são considerados diferentes, se a diferença entre as respectivas médias dos postos forem no mínimo igual a $CD = 2{,}948\sqrt{7.8/6.8} = 3{,}185$. Assim, o teste reporta que o AGHA é

considerado significativamente melhor do que os seguintes algoritmos: Hierárquico Aglomerativo (6,625 – 1 = 5,625 > 3,185), *k-means* inicializado aleatoriamente (4,4375 – 1 = 3,4375 > 3,185), *Tabu Search* (4,875 – 1 = 3,875 > 3,185) e AG (4,875 – 1 = 3,875 > 3,185). Nos demais pares de algoritmos as diferenças são menores do que a diferença crítica *CD*, portanto nada se pode concluir, nem que são iguais e nem que são diferentes.

No *Teste de Nemenyi* quanto maior a amostra de dados do experimento, ou seja, quanto maior o número de problemas *n*, mais preciso é o resultado estatístico obtido pelos testes. Com o intuito de aumentar a eficácia do teste estatístico foram criadas 15 bases sintéticas, conforme descrito anteriormente. A seguir são apresentados os resultados com as bases sintéticas.

5.3 Resultados com as Bases Sintéticas

A Tabela 5.13 apresenta os resultados obtidos com a aplicação do AG, AGHA, Hierárquico Aglomerativo, *Tabu Search* e as três versões do *K-means,* inicializado aleatoriamente, com o *K-means++* e com o *PCA_part,* às bases sintéticas citadas na Tabela 5.4. Para cada base sintética foram realizados experimentos com três valores diferentes de *K*, sendo um dos valores igual ao número de classes existentes nos dados. Os melhores resultados estão destacados em negrito.

Os testes com as bases sintéticas foram realizados utilizando os mesmos procedimentos dos testes com as bases reais, ou seja, foram executados dez vezes para cada base sintética, utilizaram os mesmos valores dos parâmetros definidos para as bases reais e foram calculadas as médias dos tempos de execução, as médias dos valores do critério *SSE*, bem como o desvio padrão para cada valor do *SSE*.

Uma análise menos rigorosa dos resultados apresentados na Tabela 5.13 permite constatar que o AGHA obteve os melhores resultados em todos os 45 testes com apenas 7 empates, novamente indicando uma possível superioridade da metaheurística em relação aos demais algoritmos avaliados, embora não seja uma comprovação estatística.

Tabela 5.13 *SSE* médio obtido com as dez execuções dos algoritmos avaliados para cada base sintética.

Base	k	Hier. Aglo.	kmeans	kmeans (kmeans++)	kmeans (*PCA_Part*)	Tabu Search	Genético	Genético Híbrido
Iris1	2	1404,85	**1358,91**	**1358,91**	**1358,91**	1431,84	1383,34	**1358,91**
	3	1395,01	1145,87	1138,12	1169,40	1218,36	1162,89	**1133,79**
	4	1289,47	961,913	966,47	954,47	1074,54	1024,12	**954,11**
Iris2	2	2643,05	2596,79	2596,78	2596,81	2642,60	2632,16	**2596,77**
	3	2256,80	2202,53	2187,41	**2165,15**	2233,42	2230,29	**2165,15**
	4	2078,23	1831,98	1805,75	1804,36	1874,30	1954,98	**1804,30**
Iris3	2	4200,62	**3696,26**	**3696,26**	**3696,26**	3739,23	3755,74	**3696,26**
	3	3257,01	3115,96	3119,37	3091,04	3149,63	3195,31	**3091,03**
	4	3009,17	2627,43	2630,94	2606,80	2677,42	2779,43	**2606,43**
Iris4	2	4945,81	**4717,83**	**4717,83**	**4717,83**	4746,74	4776,21	**4717,83**
	3	4300,34	4013,52	4016,10	**3987,60**	4055,91	4101,62	**3987,60**
	4	3612,99	3378,29	3360,93	**3339,31**	3381,13	3540,72	**3339,31**
Iris5	2	6555,19	6046,85	6046,85	6046,89	6070,47	6106,91	**6046,75**
	3	5338,33	5080,68	5086,76	**5036,76**	5072,60	5215,89	**5036,76**
	4	4726,11	4261,60	4321,25	4247,04	4284,40	4548,62	**4246,86**
Wine1	2	2,923E+07	**2,447E+07**	**2,447E+07**	**2,447E+07**	2,973E+07	2,465E+07	**2,447E+07**
	3	1,214E+07	1,11907E+07	1,11890E+07	**1,11791E+07**	1,319E+07	1,139E+07	**1,11791E+07**
	4	8,499E+06	6,716E+06	6,708E+06	6,729E+06	1,004E+07	6,981E+06	**6,686E+06**
Wine2	2	6,325E+07	4,294E+07	4,294E+07	4,294E+07	4,582E+07	4,314E+07	**4,294E+07**
	3	2,362E+07	2,11532E+07	**2,11531E+07**	2,11532E+07	2,321E+07	2,159E+07	**2,11531E+07**
	4	1,322E+07	1,16410E+07	1,16401E+07	1,16415E+07	1,454E+07	1,236E+07	**1,16387E+07**
Wine3	2	9,007E+07	**6,895E+07**	**6,895E+07**	**6,895E+07**	7,118E+07	6,953E+07	**6,895E+07**
	3	3,080E+07	2,85257E+07	2,85264E+07	2,85267E+07	3,060E+07	2,914E+07	**2,85254E+07**
	4	1,878E+07	1,72304E+07	1,72306E+07	**1,72302E+07**	1,936E+07	1,817E+07	**1,72302E+07**
Wine4	2	1,119E+08	8,79542E+07	8,796E+07	**8,79537E+07**	8,944E+07	8,875E+07	**8,79537E+07**
	3	4,120E+07	3,97467E+07	3,97465E+07	3,97459E+07	4,135E+07	4,079E+07	**3,974E+07**
	4	2,904E+07	2,307E+07	2,308E+07	2,304E+07	2,529E+07	2,423E+07	**2,302E+07**
Wine5	2	1,302E+10	**1,203E+10**	**1,203E+10**	**1,203E+10**	1,207E+10	1,222E+10	**1,203E+10**
	3	1,124E+10	9,432E+09	9,343E+09	9,261E+09	9,344E+09	9,652E+09	**9,259E+09**
	4	8,115E+09	7,065E+09	7,065E+09	7,06418E+09	7,131E+09	7,590E+09	**7,06415E+09**
Cloud1	9	5,377E+07	4,155E+07	4,088E+07	4,012E+07	5,067E+07	4,890E+07	**4,008E+07**
	10	4,127E+07	3,930E+07	3,697E+07	3,662E+07	4,271E+07	4,258E+07	**3,658E+07**
	11	3,830E+07	3,523E+07	3,476E+07	3,532E+07	4,220E+07	4,151E+07	**3,330E+07**
Cloud2	9	1,120E+08	9,668E+07	9,639E+07	9,596E+07	1,129E+08	1,106E+08	**9,595E+07**
	10	1,055E+08	8,908E+07	8,802E+07	8,765E+07	9,673E+07	1,040E+08	**8,762E+07**
	11	1,055E+08	8,371E+07	8,274E+07	8,287E+07	9,744E+07	9,644E+07	**8,246E+07**
Cloud3	9	1,650E+08	1,503E+08	1,501E+08	1,497E+08	1,649E+08	1,822E+08	**1,494E+08**
	10	1,535E+08	1,371E+08	1,371E+08	1,374E+08	1,475E+08	1,615E+08	**1,369E+08**
	11	1,442E+08	1,291E+08	1,290E+08	1,290E+08	1,369E+08	1,564E+08	**1,284E+08**
Cloud4	9	2,243E+08	2,029E+08	2,034E+08	2,01872E+08	2,130E+08	2,290E+08	**2,01869E+08**
	10	2,038E+08	1,863E+08	1,866E+08	1,880E+08	1,955E+08	2,204E+08	**1,860E+08**
	11	2,038E+08	1,749E+08	1,755E+08	1,73983E+08	1,847E+08	2,084E+08	**1,73970E+08**
Cloud5	9	3,040E+08	2,515E+08	2,516E+08	2,521E+08	2,630E+08	2,885E+08	**2,513E+08**
	10	2,815E+08	2,315E+08	2,317E+08	2,31292E+08	2,389E+08	2,684E+08	**2,31253E+08**
	11	2,559E+08	2,190E+08	2,184E+08	2,204E+08	2,255E+08	2,608E+08	**2,176E+08**

Para complementar a avaliação também foi realizada uma análise do tempo de execução dos algoritmos, a fim de quantificar a diferença entre a aplicação do AGHA que realiza um procedimento de busca global associado com busca local e dos demais algoritmos.

A Tabela 5.14 apresenta o tempo de execução dos algoritmos avaliados e a Tabela 5.15 o desvio padrão em relação ao critério *SSE*. Cada algoritmo, não determinístico, foi executado dez vezes para cada problema, assim os resultados apresentados na Tabela 5.14 se referem à média do tempo das dez execuções. O tempo é apresentado em segundos.

Tabela 5.14 Média do tempo das dez execuções dos algoritmos avaliados para cada base sintética.

Base	k	Hier. Aglo.	kmeans	kmeans (kmeans++)	kmeans (*PCA_Part*)	Tabu Search	Genético	Genético Híbrido
Iris1	2	2,86	0,01	< 0,01	< 0,01	244,02	2,88	3,78
	3	2,49	0,02	0,01	< 0,01	282,13	2,41	4,96
	4	2,50	0,04	0,01	< 0,01	325,99	3,63	8,59
Iris2	2	13,30	0,01	0,01	< 0,01	332,96	4,73	6,51
	3	13,28	0,02	0,02	< 0,01	401,13	3,86	10,81
	4	13,30	0,06	0,04	< 0,01	476,21	6,23	12,66
Iris3	2	38,86	0,02	0,02	< 0,01	420,15	5,86	10,19
	3	38,87	0,03	0,03	< 0,01	516,39	6,51	17,99
	4	38,88	0,05	0,04	0,01	626,72	8,49	20,03
Iris4	2	85,82	0,02	0,02	< 0,01	510,96	10,24	12,12
	3	85,52	0,06	0,05	0,01	635,95	9,92	24,02
	4	85,49	0,12	0,13	0,01	777,00	10,92	25,27
Iris5	2	158,69	0,03	0,03	< 0,01	599,18	11,87	15,93
	3	158,66	0,08	0,08	0,01	759,25	11,90	29,04
	4	158,58	0,15	0,20	0,01	927,79	23,18	38,09
Wine1	2	7,36	0,03	0,02	< 0,01	99,24	4,73	9,63
	3	7,35	0,07	0,03	0,01	123,61	6,37	13,16
	4	7,38	0,15	0,06	0,01	147,10	8,79	18,27
Wine2	2	39,32	0,03	0,03	0,01	154,48	6,48	16,40
	3	39,34	0,11	0,10	0,01	197,12	10,50	24,08
	4	39,35	0,15	0,10	0,01	238,43	12,36	31,95
Wine3	2	114,40	0,05	0,04	0,01	209,70	12,50	24,00
	3	114,34	0,10	0,10	0,01	270,94	14,51	34,70
	4	114,37	0,33	0,18	0,02	329,34	16,26	47,23
Wine4	2	251,69	0,08	0,09	0,01	264,65	19,55	32,08
	3	251,64	0,16	0,15	0,03	344,54	24,47	46,86
	4	251,50	0,26	0,21	0,02	420,86	28,74	63,37
Wine5	2	471,69	0,08	0,07	0,01	320,11	26,05	40,56
	3	471,66	0,25	0,24	0,03	418,97	23,58	88,14
	4	471,69	0,46	0,55	0,03	513,36	31,04	103,05
Cloud1	9	149,8	0,62	0,40	0,07	456,55	34,12	151,75
	10	149,7	0,92	0,45	0,04	499,50	43,31	277,01
	11	149,6	1,13	0,68	0,03	536,46	58,31	337,93
Cloud2	9	1206,8	1,70	1,03	0,24	891,22	70,60	502,64
	10	1204,6	2,11	2,11	0,33	973,32	80,81	508,80
	11	1204,3	1,95	1,42	0,34	1053,10	77,63	418,64
Cloud3	9	4034,0	3,29	2,52	0,39	1251,83	159,71	493,32
	10	4039,2	5,50	4,71	1,13	1330,83	149,56	798,64
	11	4042,0	5,07	4,46	1,23	1575,23	133,51	832,25
Cloud4	9	9582,0	3,46	3,14	0,52	1766,43	175,03	1006,01
	10	9592,0	6,50	3,47	1,11	1929,87	158,54	1151,30
	11	9598,7	8,10	4,51	1,78	2091,03	193,08	1041,59
Cloud5	9	18655,2	5,01	4,57	0,54	2232,00	193,28	1427,89
	10	18678,4	6,25	5,35	0,64	2217,80	216,67	1087,18
	11	18685,6	9,29	7,76	1,52	2617,00	305,29	535,10

Tabela 5.15 Desvio padrão do critério *SSE* das bases sintéticas.

Base	k	kmeans	kmeans (kmeans++)	Tabu Search	Genético	Genético Híbrido
Iris1	2	0,23	0,02	4,70	11,84	< 0,01
	3	16,75	11,02	7,26	8,38	< 0,01
	4	10,88	16,43	12,63	21,66	< 0,01
Iris2	2	0,02	0,02	4,08	25,75	< 0,01
	3	33,08	31,36	2,99	21,88	< 0,01
	4	42,82	0,84	3,33	35,41	< 0,01
Iris3	2	0,31	0,02	2,19	36,30	< 0,01
	3	24,57	25,93	5,24	49,72	< 0,01
	4	31,19	34,02	27,75	56,12	< 0,01
Iris4	2	0,09	0,06	1,14	54,52	< 0,01
	3	25,78	19,09	19,29	31,28	< 0,01
	4	63,56	44,52	1,90	87,49	< 0,01
Iris5	2	0,07	0,07	1,57	39,11	< 0,01
	3	57,38	65,13	2,74	84,89	< 0,01
	4	46,39	78,28	0,77	64,06	< 0,01
Wine1	2	0,02	0,02	342906,00	176249,00	< 0,01
	3	7986,03	8537,42	196054,00	213503,00	< 0,01
	4	20698,50	22583,90	292118,00	150895,00	< 0,01
Wine2	2	338,68	362,06	311824,00	283232,00	< 0,01
	3	43,42	42,54	172970,00	399571,00	< 0,01
	4	595,73	981,58	982059,00	649906,00	< 0,01
Wine3	2	0,02	0,03	131709,00	464411,00	< 0,01
	3	532,86	532,86	196439,00	611552,00	< 0,01
	4	257,15	209,96	138923,00	635478,00	< 0,01
Wine4	2	932,85	932,85	117499,00	792373,00	< 0,01
	3	1062,07	1901,45	147505,00	909897,00	< 0,01
	4	51090,70	52114,90	241155,00	1,613E+06	< 0,01
Wine5	2	0,02	0,02	1,971E+06	1,176E+08	< 0,01
	3	1,49E+08	1,33E+08	4,084E+06	1,683E+08	< 0,01
	4	568852,00	519136,00	6,904E+06	1,477E+08	< 0,01
Cloud1	9	1,07E+06	707747,00	1,097E+06	3,844E+06	1012,28
	10	1,90E+06	847509,00	662999,00	2,422E+06	671,00
	11	290224,00	1,85E+06	823862,00	3,488E+06	1876,23
Cloud2	9	658472,00	557774,00	837290,00	6,439E+06	2019,20
	10	1,69E+06	1,04E+06	1,368E+06	4,864E+06	3421,77
	11	2,12E+06	190502,00	1,039E+06	5,452E+06	158,57
Cloud3	9	712762,00	685972,00	1,109E+05	1,517E+07	233,54
	10	228396,00	208576,00	1,131E+06	1,041E+07	303,46
	11	1,16E+06	818636,00	841397,00	1,012E+07	71,05
Cloud4	9	1,42E+06	1,58E+06	614101,00	1,330E+07	4215,74
	10	653198,00	977213,00	655861,00	1,349E+07	1360,45
	11	1,49E+06	1,85E+06	1,207E+06	1,041E+07	529,57
Cloud5	9	235265,00	357443,00	870392,00	1,723E+07	30,14
	10	310517,00	310204,00	3,097E+06	9,532E+06	273,78
	11	1,22E+06	1,09E+06	405240,00	1,693E+07	168,64

Os resultados apresentados na Tabela 5.14 são condizentes com o descrito na seção 5.2. Novamente, o melhor tempo de execução para todas as bases é do algoritmo *K-means* inicializado com o *PCA_Part*, reforçando a hipótese que a boa escolha dos centróides iniciais

através do método *PCA_Part* reduz o número de iterações para alcançar a solução final, conseqüentemente reduz-se o tempo de execução do algoritmo.

O algoritmo Hierárquico Aglomerativo apresentou, devido à sua complexidade computacional de $O(n^2)$, tempos de execução pequenos para as bases pequenas (*Iris1* e *Wine1*) e os maiores tempos de execução entre todos os algoritmos avaliados para as bases maiores (*Cloud2, Cloud3, Cloud4* e *Cloud5*), reforçando que seu custo elevado limita sua aplicação em bases muito grandes.

Novamente o AGHA apresenta tempo satisfatório quando aplicado nas bases menores, porém o tempo começa a aumentar a partir das bases maiores, mais uma vez indicando que seria promissora a aplicação de estratégias para limitar a aplicação de busca local a certo grupo de indivíduos da população.

Os resultados apresentados na tabela 5.15 mostram que, novamente, o AGHA apresentou menor dispersão em torno da média dos valores do critério *SSE*, indicando uma maior homogeneidade entre os 10 resultados do algoritmo para cada base.

5.3.1 Avaliação Experimental com Métodos Estatísticos

Para realizar uma avaliação mais precisa e rigorosa dos experimentos, optou-se por fazer uma análise estatística dos resultados das bases sintéticas, assim como foi feito para as bases reais. Foram utilizados os mesmos métodos estatísticos, *Teste de Friedman* para comparação de algoritmos em múltiplos domínios e o *Teste de Nemenyi* para identificar a diferença significativa entre os algoritmos quando todos são comparados entre si.

A Tabela 5.16 apresenta os valores médios do critério de avaliação *SSE*, encontrados com as dez execuções de cada algoritmo para cada base sintética. O posto atribuído pelo *Teste de Friedman* é apresentado entre parênteses e os postos médios na última linha da tabela.

Tabela 5.16 Comparação entre os algoritmos avaliados usando diferentes bases sintéticas.

Base	Hier. Aglo.	kmeans	kmeans (kmeans++)	kmeans (pca_part)	Tabu Search	Genético	Genético Híbrido
Iris1	1395,01 (7)	1145,87 (3)	1138,12 (2)	1169,40 (5)	1218,36 (6)	1162,89 (4)	1133,79 (1)
Iris2	2256,80 (7)	2202,53 (4)	2187,41 (3)	2165,15 (1,5)	2233,42 (6)	2230,29 (5)	2165,15 (1,5)
Iris3	3257,01 (7)	3115,96 (3)	3119,37 (4)	3091,04 (2)	3149,63 (5)	3195,31 (6)	3091,03 (1)
Iris4	4300,34 (7)	4013,52 (3)	4016,10 (4)	3987,60 (1,5)	4055,91 (5)	4101,62 (6)	3987,60 (1,5)
Iris5	5338,33 (7)	5080,68 (4)	5086,76 (5)	5036,76 (1,5)	5072,60 (3)	5215,89 (6)	5036,76 (1,5)
Wine1	1,214E+07 (6)	1,11907E+07 (4)	1,11890E+07 (3)	1,11791E+07 (1,5)	1,319E+07 (7)	1,139E+07 (5)	1,11791E+07 (1,5)
Wine2	2,362E+07 (7)	2,11532E+07 (3,5)	2,11531E+07 (1,5)	2,11532E+07 (3,5)	2,321E+07 (6)	2,159E+07 (5)	2,11531E+07 (1,5)
Wine3	3,080E+07 (7)	2,85257E+07 (2)	2,85264E+07 (3)	2,85267E+07 (4)	3,060E+07 (6)	2,914E+07 (5)	2,85254E+07 (1)
Wine4	4,120E+07 (6)	3,97467E+07 (4)	3,97465E+07 (3)	3,97459E+07 (2)	4,135E+07 (7)	4,079E+07 (5)	3,974E+07 (1)
Wine5	1,124E+10 (7)	9,432E+09 (5)	9,343E+09 (3)	9,261E+09 (2)	9,344E+09 (4)	9,652E+09 (6)	9,259E+09 (1)
Cloud1	4,127E+07 (5)	3,930E+07 (4)	3,697E+07 (3)	3,662E+07 (2)	4,271E+07 (7)	4,258E+07 (6)	3,658E+07 (1)
Cloud2	1,055E+08 (7)	8,908E+07 (4)	8,802E+07 (3)	8,765E+07 (2)	9,673E+07 (5)	1,040E+08 (6)	8,762E+07 (1)
Cloud3	1,535E+08 (6)	1,371E+08 (3)	1,37066E+08 (2)	1,37363E+08 (4)	1,475E+08 (5)	1,615E+08 (7)	1,369E+08 (1)
Cloud4	2,038E+08 (6)	1,863E+08 (2)	1,866E+08 (3)	1,880E+08 (4)	1,955E+08 (5)	2,204E+08 (7)	1,860E+08 (1)
Cloud5	2,815E+08 (7)	2,391E+08 (5)	2,317E+08 (3)	2,31292E+08 (2)	2,389E+08 (4)	2,684E+08 (6)	2,31253E+08 (1)
p.m.	6,600	3,567	3,033	2,567	5,400	5,667	1,167

O resultado do *Teste de Friedman* realizado com os valores da Tabela 5.16, com nível de significância de 5%, indica que a hipótese nula pode ser rejeitada. Dessa forma a hipótese alternativa é aceita, ou seja, existe pelo menos um par de algoritmos estatisticamente diferentes.

Para o *Teste de Nemenyi* com nível de significância de 5% dois algoritmos são considerados diferentes, se a diferença entre as respectivas médias dos postos forem no mínimo igual a $CD = 2{,}949\sqrt{7.8/6.15} = 2{,}326$. Assim, o teste reporta que o AGHA é considerado significativamente melhor do que os seguintes algoritmos: Hierárquico Aglomerativo (6,6 – 1,167 = 5,433 > 2,326), *k-means* inicializado aleatoriamente (3,567 – 1,167 = 2,4 > 2,326), *Tabu Search* (5,4 – 1,167 = 4,233 > 2,326) e AG (5,667 – 1,167 = 4,5 > 2,326). Nos demais pares de algoritmos as diferenças são menores do que a diferença crítica *CD*, portanto nada se pode concluir, nem que são iguais e nem que são diferentes.

É importante ressaltar que os resultados com as bases sintéticas apontaram a mesma conclusão dos testes estatísticos com as bases reais: que o AGHA é significativamente melhor que o Hierárquico Aglomerativo, o *K-means* inicializado aleatoriamente, o *Tabu Search* e o AG.

6 *Conclusões e Trabalhos Futuros*

O presente trabalho apresenta uma proposta de Algoritmo Genético Híbrido de Agrupamento, cuja população inicial é parcialmente gerada por métodos de agrupamento, que realiza um procedimento de busca global associado com busca local. Tais melhorias, gerar uma população inicial melhor e aplicar busca local em cada indivíduo da população corrente, têm como objetivo direcionar a busca para soluções mais próximas do ótimo global.

Para analisar se a solução apresentada consegue um desempenho melhor do que outros algoritmos propostos para resolver o mesmo problema, este trabalho realizou uma avaliação experimental usando oito bases reais disponíveis em repositórios públicos e quinze bases sintéticas. As bases reais foram escolhidas por apresentarem as características adequadas para os algoritmos avaliados: atributos numéricos e não nulos. As bases sintéticas foram geradas a partir de três bases reais selecionadas entre as oito utilizadas nos experimentos.

O algoritmo proposto, uma versão híbrida do Algoritmo Genético de Agrupamento, foi combinado a três versões do *K-means* (inicializado aleatoriamente, com o *K-means++* e com o *PCA_Part)* para gerar a população inicial e associou ao processo de busca global a aplicação de busca local, também utilizando o *K-means*, para melhoria dos indivíduos a cada geração. Tal associação de busca global e local é benéfica para o resultado final, uma vez que heurísticas baseadas em busca global apresentam melhor desempenho na exploração do espaço de soluções, enquanto heurísticas de busca local são mais eficientes na exploração dessas áreas.

Outros seis algoritmos foram implementados: Hierárquico Aglomerativo, três versões do *K-means* que se diferem no método de inicialização (aleatório, *K-means++* e *PCA_Part*), *Tabu Search* e Algoritmo Genético de Agrupamento.

Na avaliação realizada, os sete algoritmos foram comparados em múltiplos domínios, a fim de descobrir se há uma tendência geral de melhoria do algoritmo proposto em relação aos demais algoritmos avaliados. Nesse caso, a análise estatística mostrou que, em geral, o Algoritmo Genético Híbrido de Agrupamento proposto fornece melhores resultados que o Hierárquico Aglomerativo, o *K-means* inicializado aleatoriamente, o *Tabu Search* e a versão canônica do Algoritmo Genético de Agrupamento. Em relação aos demais algoritmos, a análise não permite concluir se existe ou não diferença entre eles. É importante destacar que a

análise estatística apresentou a mesma conclusão tanto para as bases reais como para as bases sintéticas.

Uma contribuição importante deste trabalho foi o desenvolvimento do Algoritmo Genético Híbrido de Agrupamento cuja população inicial é parcialmente gerada por três versões do *K-means*, garantido uma população inicial melhor do que se fosse gerada apenas aleatoriamente, e cujo procedimento de busca global é associado ao método de busca local proporcionando uma exploração eficiente de soluções potenciais. Outra contribuição relevante foi a ampla experimentação comparativa realizada, mostrando que a abordagem híbrida proposta apresenta os melhores resultados tanto para as bases reais como para as sintéticas.

6.1 Trabalhos Futuros

A proposta do AGHA abordada neste trabalho utiliza como métodos de geração da população inicial três versões do *k-means*, uma possível continuação deste estudo seria analisar a aplicação de outras heurísticas para gerar a população inicial.

Uma outra possível continuação deste estudo está em analisar a aplicação de outras heurísticas de busca local que seriam associadas ao processo de busca global do AGHA.

A avaliação de outros algoritmos permite melhorar a qualidade da comparação estatística, portanto uma outra continuação seria a realização de experimentos com um número maior de algoritmos, como por exemplo o *Simulated Annealing* e o *GRASP (Greedy Randomized Adaptive Search Procedure).*

Aumentar o número de problemas também garante uma comparação mais eficaz com os testes estatísticos propostos. Por isso, recomenda-se como trabalho futuro a realização de mais experimentos, usando outros problemas reais, bem como um número maior de bases sintéticas, para realizar uma avaliação mais precisa dos algoritmos estudados.

Dentre os experimentos realizados, constatou-se que na maior base (*Letter*) o AGHA obteve em todos os testes desempenho superior aos demais algoritmos, indicando que a metaheurísica apresenta um desempenho satisfatório em bases muito grandes. Uma possível continuação para este trabalho consiste em realizar experimentos para analisar o desempenho do AGHA em bases muito grandes.

Por fim, uma complementação para este trabalho seria aplicar no AGHA a abordagem

sugerida por (YEN; LEE, 1997), onde apenas um certo percentual da população considerado de indivíduos-elite, sofreriam melhorias heurísticas. Dessa forma limita-se a aplicação de busca local a um certo grupo de indivíduos diminuindo o tempo de execução do algoritmo. Seria necessária uma análise para verificar se tal abordagem não prejudica a qualidade das soluções em prol de um menor tempo de execução.

Referências Bibliográficas

ALSULTAN, K. A Tabu Search Approach to the Clustering Problem, *Pattern Recognition,* 28, 1443-1451, 1995.

ARTHUR, D.; VASSILVITSKII, S. K-means++: The Advantages of Careful Seeding. *Symposium on Discrete Algorithms (SODA)*, 2007.

ASSUNÇÃO, T.; FURTADO, V. A heuristic method for balanced graph partitioning: an application for the demarcation of preventive police patrol areas. *IBERAMA,* LNAI 5290, p. 67-72, 2008.

ASUNCION, A.; NEWMAN, D. *UCI Machine Learning Repository*. 2007. Disponível em: < http://archive.ics.uci.edu/ml/>.

BABU, G. P.; MURTY, M. N. A Near-Optimal Initial Seed Value Selection in K-means Algorithm using a Genetic Algorithm. *Pattern Recognition Letters*, New York, NY, USA, v.14, n. 10, p. 763–769, 1993.

BACK, T. Evolutionary Algorithms in Theory and Practice. Oxford University Press, 1996.

BACK, T., FOGEL, D. B. & MICHALEWICZ, Z. "Handbook of Evolutionary Computation", Institute of Physics Publishing and Oxford University Press, 1997.

BANFIELD, J. D.; RAFTERY, A. E. Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics*, v. 49, n. 3, p. 803–821, 1993.

BELEW, R. K.; BOOKER, L. B. Solving Partitioning Problems with Genetic Algorithms. Morgan Kaufmann, 1991.

BERGER, D.; GENDROM, B.; POTVIN, J. Y. "Tabu Search for a Network Loading Problem", Institute for Systems Research, Technical Report 99-23, 1999.

BEZDEK, J. C.; BOGGAVAPARU, S.; HALL, L. O.; BENSAID, A. Genetic Algorithm guided clustering. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 34-40, 1994.

BIRRU, H. K.; CHELLAPILLA, K.; RAO, S. Local search operators in fast evolutionary programming. Congress on Evolutionary Computation (CEC'99), Washington, USA. IEEE Press,v. 2, p. 1506–1513, 1999.

BOLEY, D. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, v. 2, n. 4, p. 325–344, 1998.

CERNY, V. A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. *Journal of Optimization Theory and Applications*, v. 45, n. 1, p. 41 51,1985.

COLE, R. M. Clustering with genetic algorithms. Thesis - (Master of Science), Department of Computer Science, University of Western Australia, 1998.

COWGILL, M. C.; HARVEY, R. J.; WATSON, L. T. A Genetic Algorithm approach to cluster analysis. Technical report, Virginia Polytechnic Institute & State University, Blacksburg, VA, USA, 1998.

DAVIS, L. D. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold, 1991.

DEMSAR, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, v. 7, n. 1, p. 1–30, 2006.

DORIGO, M.; CARO, G. D. *The Ant Colony Optimization Meta-heuristic*. London: McGraw-Hill, 1999.

DORIGO,M.; CARO, G. D.; GAMBARDELLA, L. Ant Algorithms for Discret Optimization. *Artificial Life*, v. 5, n. 2, p. 137–172, 1999.

DUDA, R., P. HART, & D. STORK. *Pattern Classification.* John Wiley & Sons, 2001.

EVERITT, B. S.; LANDAU, S.; LEESE, M. *Cluster Analysis*. [S.l.]: Hodder Arnold Publication, 2001.

ERICSSON, M.; RESENDE, M. G. C.; PARDALOS, P. M. A genetic algorithm for the weight setting problem in OSPF routing. *Journal of Combinatorial Optimization*, 6, 299-333, 2002.

FALKENAUER, E. *Genetic Algorithms and Grouping Problems*. New York, NY, USA: Wiley, 1998.

FAYYAD, U. et al. Advances in Knowledge Discovery and Data Mining. *MIT Press*, 1996.

FASULO, D. An Analysis of Recent Work on Clustering Algorithms. *Technical Report*, 1999.

FELTL, H.; RAIDL, G. R. An improved hybrid genetic algorithm for the generalized assignment problem. In: ACM Symposium on Applied Computing (SAC2004), 19, Nicosia, Cyprus. Proceedings. ACM Press, p. 990–995, 2004.

FEO, T.; RESENDE, M. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization,* v. 6, p. 109–133, 1995.

FOGEL, D.B. “An Introduction to Simulated Evolutionary Computation”*, IEEE Transactions on Neural Networks,* vol. 5, no. 1, pp. 3-14, 1994.

FORGY, E. Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications. *Biometrics*, 1965.

FRIEDMAN, M. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *Annals of Mathematical Statistics*, v. 11, n. 1, p. 86–92, 1940.

FUKUNAGA, K. Introduction to Statistical Pattern Recognition. Academic Press, 1990.

GLOVER, F. W. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, v. 13, n. 5, p. 533–549, 1986.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Kluwer Academic Publishers, 1997.

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

GOLUB, T. R. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, v.286, n.5439, p. 531-537, 1999.

GONÇALVES, J. F.; MENDES, J. J. M.; RESENDE, M. G. C. A hybrid genetic algorithm for the Job Shop Scheduling Problems. To appear in *European Journal of Operational Research*, 2004.

HALL, L. O.; ÖZYURT, I. B.; BEZDEK, J. C. Clustering with a Genetically Optimized Approach. *IEEE Trans. on Evolutionary Computation*, v. 3, n. 2, p. 103–112, 1999.

HANSEN, P. The steepest ascent mildest descent heuristic for combinatorial programming. *In Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

HARTIGAN, J. A. *Clustering Algorithms*. [S.l.]: John Wiley & Sons Inc, 1975.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning – Data Mining, Inference and Prediction*. [S.l.]: Springer, 2003.

HUANG, C. M.; HARRIS, R. W. A Comparison of Several Vector Quantization Codebook Generation Approaches. *IEEE Transactions on Image Processing*, v. 2, n. 1, p. 108–112, 1993.

IMAN, L.; DAVENPORT, J. M. Approximations of the Critical Region of the Friedman Statistic. *Communications in Statistics*, v. 9, n. 6, p. 571–595, 1980.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data Clustering: A Review. *ACM Computing Surveys*, New York, NY, USA, v. 31, n. 3, p. 264–323, 1999.

JONES, D. R.; BELTRAMO, M. A. Solving partitioning problems with genetic algorithms. *In Proceeding of the Fourth International Conference on Genetic Algorithms*, pages 442-9, 1991.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding Groups in Data - An Introduction to Cluster Analysis*. [S.l.]: Ed. Wiley, 1990.

KENNEDY, J.; EBERHART, R. C. Particle Swarm Optimization. *In Proceedings of the IEEE International Joint Conference on Neural Networks*, v. 4, p. 1942–1948, 1995.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, v. 220, p. 671–680, 1983.

KROVI, R. Genetic algorithms for clustering: A preliminary investigation. In V. Milutinovic, B. D. Shriver, J. F. Jr. Numaker, and R. H. Jr. Sprague, editors, *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences,* pages 540-4. *IEEE Computer Society Press*, 1991.

LAGUNA, M. A guide to implementing Tabu Search. University of Colorado, Boulder, v.4, n.1, p.5-25, 1994.

LEVRAT, E. et al. Multi-level Image Segmentation using Fuzzy Clustering and Localmembership Variations Detection. *IEEE International Conference on Fuzzy Systems*, New York, p. 221–228, 1992.

LORENA, L. A. N.; FURTADO, J. C. Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 309-327, 2001.

MACQUEEN, J. Some Methods for Classification and Analysis of Multivariate Observations. *In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

MAULIK, U. BANDYOPADHYAY, S. Genetic algorithm-based clustering technique. Pattern Recognition, 1455 – 1465, 2000.

MESTER, D.; BRAYSY, O. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 593-614, 2005.

MEZ, P. Memetic Algorithm for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies, 2000.

MICHALEWICZ, Z. Genetic Algorithms + Data Structures = Evolution Programs, 3rd edition, Springer-Verlag, 1996.

MOSCATO, P. On Evolution, Search, Optimizaton, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

MOSCATO, P. Memetic algorithms: a short introduction. In: Corne, D.; Dorigo, M.; Glover, F. ed. New Ideas in Optimization. London: McGraw-Hill. p. 219–234, 1999.

MURTHY, C. A.; CHOWDHURY, N. In search of optimal clusters using genetic algorithms. *Pattern Recogn. Lett.*, New York, NY, USA, v. 17, n. 8, p. 825–832, 1996.

NEMENYI, P. B. *Distribution-free multiple comparisons*. Tese (Doutorado) - Princeton University, 1963.

OCHI, L. S.; VIANNA, D. S.; DRUMMOND, L. M. A. An Asynchronous Parallel metaheuristic for the Period Vehicle Routing Problems. *Future Generation on Computer Systems*. Elsevier, 379-386, 2001.

OSTROVSKY, R. et al. The Effectiveness of Lloyd-Type Methods for the K-Means Problem. *IEEE Symposium on Foundations of Computer Science*, p. 165–176, 2006.

PERIM, T. G. Uso de métodos de inicialização combinados ao *Simulated Annealing* para resolver o problema de agrupamento de dados. 76f. Dissertação - (Mestrado em Informática) - UFES, Universidade Federal do Espírito Santo. Vitória, 2008.

PERIM, T. G.; VAREJÃO, F. M. Aplicação de método baseado em *pca* para inicialização do S*imulated Annealing* no problema de particionamento de dados, XL Simpósio Brasileiro de Pesquisa Operacional, 2008.

PERIM, T. G.; WANDEKOKEM D. E.; VAREJÃO, M. F. K-Means Initialization Methods for Improving Clustering by Simulated Annealing, *Lecture Notes in Computer Science, Advances in Artificial Intelligence – IBERAMIA,* 2008.

RAGHAVAN, V. V.; BIRCHARD, K. A clustering strategy based on a formalism of the reproductive process in natural systems. *SIGIR Forum*, New York, NY, USA, v. 14, n. 2, p. 10–22, 1979.

RARDIN, R. L.; UZSOY, R. *Experimental evaluation of heuristic optimization algorithms: a tutorial. Journal of Heuristics*, v.7, n.3, p.261-304, 2001.

RAYWARD-SMITH, V. J. Metaheuristics for Clustering in KDD. *IEEE Congress on Evolutionary Computation*, v. 3, p. 2380–2387, 2005.

RESENDE, M. G. C.; GONÇALVES, J. F. An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering*, 247-273, 2004.

RIBEIRO, C. C. Metaheuristics and Applications. *In Advanced School on Artificial Intelligence*, Estoril, Portugal, 1996.

SCHAEFER, A. Tabu search techniques for large high-school timetabling problems. In Proceedings of the 30th National Conference on Artificial Intelligence, pages 363–368, 1996.

SELIM, S. Z.; ISMAIL, M. A. K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Trans. Pattern Analysis and Machine Intelligence*, v. 6, n. 1, p. 81–87, 1984.

SHLENS, Jonathon. A Tutorial on Principal Component Analysis. Systems Neurobiology Laboratory, Salk Insitute for Biological Studies. Institute for Nonlinear Science, University of California, San Diego, 2005.

SU, T.; DY, J. A Deterministic Method for Initializing K-means Clustering. *16th IEEE International Conference on Tools with Artificial Intelligence*, 2004.

TALBI, E.-G. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristic*, v. 8, n. 5, p. 541–564, 2002.

XU, L; WUNSCH, D. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, vol.16, nº 3, 2005.

WERRA, D. Tabu Search Techniques: A Tutorial and an Application to Neural Networks. OR Spektrum, 11:131_141, 1989.

YEN, J.; LEE, B. A simplex genetic algorithm hybrid. *IEEE International Conference on Evolutionary Computation (ICEC'97)*, 4, Indianapolis (USA). IEEE Press, p. 175–180, 1997.

ZAHN, C. T. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, C-20, n. 1, p. 68-86, 1971.