Gabriela Trazzi Perim

Uso de métodos de inicialização combinados ao Simulated Annealing 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
DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória - ES, Brasil

08 de maio de 2008

Dissertação de Mestrado sob o título "*Uso de métodos de inicialização combinados ao Simulated Annealing para resolver o problema de agrupamento de dados*", defendida por Gabriela Trazzi Perim e aprovada em 08 de maio de 2008, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos doutores:

Prof. Flávio Miguel Varejão, D. Sc.
Orientador

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

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

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Perim, Gabriela Trazzi, 1982-

P444u Uso de métodos de inicialização combinados ao simulated annealing para resolver o problema de agrupamento de dados / Gabriela Trazzi Perim. – 2008.

75 f. : il.

Orientador: Flávio Miguel Varejão.

Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Simulated annealing (Matemática). 2. Otimização Combinatória. 3. Programação heurística. I. Varejão, Flávio Miguel. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Problemas de agrupamento de dados são definidos como a tarefa de dividir 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. Essa tarefa é bastante empregada para extrair informações relevantes dos dados, encontrar uma divisão de classes entre os elementos, ou ainda obter uma representação mais compacta para o conjunto de dados original. O problema de agrupamento pode ser visualizado como uma tarefa de otimização, na qual pretende-se encontrar a melhor combinação de partições dentre todas as combinações possíveis. Com essa formulação, algoritmos aproximados são os mais adequados para resolver o problema, já que, embora não garantam encontrar a solução ótima global, obtêm soluções próximas ao ótimo da função de avaliação e demandam um custo computacional menor do que os algoritmos exatos. O *K-means* é o algoritmo aproximado mais comumente empregado para resolver o problema de agrupamento, por apresentar um procedimento simples que converge rapidamente. Porém, é um algoritmo bastante sensível à escolha da solução inicial. Trabalhos nessa área propõem métodos de inicialização específicos para aprimorar o resultado do algoritmo. Uma forma alternativa de resolver o problema consiste em aplicar meta-heurísticas, que são algoritmos com estratégias capazes de escapar de ótimos locais. No entanto, trabalhos que empregam diretamente esses algoritmos mostram que resultados mais significativos são obtidos em instâncias menores do problema. Abordagens híbridas podem apresentar uma melhoria dos resultados em instâncias maiores. Este trabalho propõe o uso de métodos de inicialização combinados a meta-heurísticas com o objetivo de aprimorar os resultados obtidos com a versão padrão da meta-heurística (com inicialização aleatória) e com o algoritmo *K-means*. Uma avaliação experimental é realizada em um conjunto de problemas reais a fim de verificar se a solução proposta apresenta uma melhoria em relação aos algoritmos já estudados. O resultado dessa análise mostra que, em geral, a meta-heurística combinada com um dos métodos de inicialização implementado apresenta um desempenho melhor do que a versão aleatória do *K-means*.

Abstract

Clustering is defined as the task of dividing a data set in subsets such that elements within each cluster are more closely related to one another than to elements assigned to different clusters. This task is frequently employed to extract useful information in the data set, assign labels to the patterns or even get a compact representation of the whole data set. Clustering can be understood as an optimization problem that looks for the best configuration of the clusters among all possible configurations. Approximate algorithms are the most appropriate way to solve this problem. Although these algorithms cannot guarantee optimality of the solutions, they get solutions near the globally optimal solution and require less computational efforts than the exact ones. K-means is the most popular approximate algorithm used to solve the clustering problem because it is a simple procedure that converges quickly. However it is very sensitive to the selection of the start solution. Some works focus on this problem and present methods for initializing the K-means algorithm that outperform the standard version of this algorithm. Metaheuristics can also be used to solve the clustering problem. These algorithms use strategies capable of escaping from local optima. Nevertheless, the direct application of metaheuristics to the clustering problem might seem to be effective only on small data sets. Hybrid approaches may find better quality solutions on large data sets. This work suggests the use of metaheuristics and initializing methods defined to the K-means algorithm in order to outperform the standard metaheuristic (which is randomly initialized) and K-means. We carried out a series of experiments on real-world problems in order to verify whether the proposed algorithm is better than the classical ones. The results show that, in general, one of the proposed combination improves the random K-means algorithm.

1 Introdução

O estudo do problema de agrupamento ou particionamento de dados possui uma variedade de objetivos em diferentes áreas de aplicação, mas, em todos os casos, o princípio básico consiste em segmentar uma coleção de padrões em subconjuntos de tal maneira que elementos de um mesmo subconjunto ou grupo estejam mais relacionados entre si do que em relação aos elementos de outros grupos. Ao organizar padrões em grupos, é possível identificar similaridades e diferenças entre eles e inferir conclusões úteis a respeito das características dos dados.

Esse problema pode ser dividido em dois tipos básicos: agrupamento do tipo *fuzzy* e agrupamento do tipo *hard* ou *crisp*. No primeiro, os elementos estão associados a todos os grupos simultaneamente, de acordo com um grau de pertinência. No segundo, um elemento está associado a um único grupo exclusivamente.

A Figura 1.1 mostra uma coleção de dados representados pelos números {1, 2, 3, 4, 5, 6, 7, 8, 9} que é particionada tanto com o conceito de agrupamentos *fuzzy* (grupos representados pelas elipses) quanto com o de agrupamentos *hard* (grupos representados pelos retângulos). Nesse exemplo, os elementos {1, 2, 3} estão associados ao grupo F_1 com pertinência igual a 100% e ao grupo F_2 com pertinência de 0%. Já os elementos {8, 9} estão associados ao grupo F_1 com pertinência igual a 0% e ao grupo F_2 com pertinência de 100%. Além disso, os elementos {4, 5, 6, 7} estão associados tanto ao grupo F_1 quanto ao grupo F_2 com pertinência igual a 50%. Este trabalho realiza um estudo apenas de agrupamentos do tipo *hard*, que são os mais empregados na área de descoberta de conhecimento.

Uma maneira de resolver o problema de agrupamento é tratando-o como uma tarefa de otimização. Nesse caso, a melhor forma de agrupar os padrões é identificar todas as possíveis combinações de partições e escolher a que for melhor avaliada de acordo com uma função de custo. No entanto, a avaliação de todas as combinações é uma tarefa computacionalmente inviável, principalmente em instâncias maiores do problema. Por isso, os algoritmos aplicados ao problema de agrupamento realizam uma busca por soluções aproximadas, avaliando apenas algumas dentre todas as combinações possíveis.

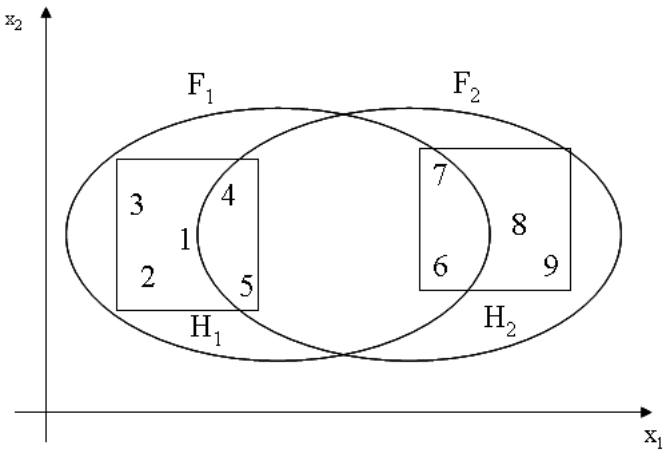

Figura 1.1: Diferença de agrupamentos *fuzzy* e *hard* (JAIN; MURTY; FLYNN, 1999)

Esses algoritmos são, portanto, procedimentos heurísticos que não garantem otimalidade das soluções encontradas, mas requerem menor esforço computacional. Um exemplo de algoritmo heurístico amplamente utilizado para resolver o problema de agrupamento é o *K-means* (FORGY, 1965; MACQUEEN, 1967). Essa preferência se deve pela simplicidade e pela rápida convergência do algoritmo.

Contudo, como será discutido mais adiante neste trabalho, o resultado obtido por esse algoritmo é fortemente dependente da solução inicial do problema escolhida como ponto de partida para o processo de busca. Se a solução inicial for escolhida sem nenhum critério (aleatoriamente), em alguns casos, a heurística pode obter resultados bem próximos ao ótimo global, mas em outros, pode encontrar resultados muito ruins. Para contornar esse problema, alguns trabalhos estudam métodos de inicialização que contribuem para melhorar a escolha da solução inicial e, conseqüentemente, aprimorar o resultado obtido pelo algoritmo *K-means*.

Algoritmos heurísticos com caráter mais geral, que aplicam estratégias para escapar de ótimos locais e procedimentos de melhoria local, são uma alternativa para resolver o problema de agrupamento. No entanto, esses algoritmos, chamados também de meta-heurísticas, mostram-se mais efetivos, na maioria das vezes, em instâncias menores do problema (RAYWARD-SMITH, 2005). A aplicação de abordagens híbridas, que exploram a combinação de técnicas de resolução de problemas de otimização, pode apresentar resultados melhores em instâncias maiores do problema se comparados aos obtidos com a aplicação direta de meta-heurísticas.

Por isso, este trabalho propõe a combinação de métodos de inicialização, que são originalmente indicados para o algoritmo *K-means*, com meta-heurísticas a fim de encontrar resultados melhores, principalmente em instâncias maiores do problema, do que os obtidos com a versão padrão das meta-heurísticas, na qual a solução inicial é escolhida aleatoriamente. Por conse-

qüência, espera-se obter com essa abordagem resultados ainda melhores do que os obtidos com a heurística *K-means*, mesmo essa sendo inicializada com algum método específico.

A meta-heurística utilizada como estudo de caso para essa abordagem foi o *Simulated Annealing*, por ser um procedimento que realiza o processo de busca pela melhor solução partindo de uma solução inicial, geralmente escolhida aleatoriamente. Além disso, essa meta-heurística aplicada ao problema de agrupamento já foi analisada em outros trabalhos, como em (KLEIN; DUBES, 1989; SELIM; ALSULTAN, 1991), e apresentou resultados positivos se comparados ao *K-means*.

A adaptação do procedimento geral para o problema de agrupamento apresentada neste trabalho baseia-se em abordagens semelhantes (KLEIN; DUBES, 1989; SELIM; ALSULTAN, 1991) e representa um algoritmo simples e intuitivo, assim como o algoritmo *K-means*.

O restante deste trabalho está dividido em mais seis capítulos. O Capítulo 2 apresenta uma definição formal do problema em análise, mencionando os conceitos básicos relacionados ao agrupamento de dados. Além disso, esse capítulo discute algumas abordagens para solucionar o problema, com ênfase maior naquelas que tratam o problema de agrupamento como uma tarefa de otimização.

O Capítulo 3 faz uma discussão detalhada do algoritmo *K-means* que é a heurística mais comumente empregada para resolver o problema de agrupamento. O capítulo também apresenta as fragilidades dessa heurística e discute alguns métodos de inicialização propostos para melhorar os resultados obtidos com ela.

O Capítulo 4 define formalmente o problema de otimização e mostra as características gerais que diferenciam as meta-heurísticas dos procedimentos heurísticos. Esse capítulo apresenta ainda o procedimento geral do *Simulated Annealing* usado para avaliar a proposta abordada nesta dissertação.

O Capítulo 5 propõe a aplicação de métodos de inicialização em conjunto com meta-heurísticas a fim de aprimorar os resultados obtidos e apresenta em detalhes a versão adaptada do algoritmo geral *Simulated Annealing* para o problema de agrupamento.

O Capítulo 6 apresenta a avaliação experimental da proposta apresentada, comparando o desempenho das diferentes combinações de métodos de inicialização com o *K-means* e com o *Simulated Annealing* adaptado, em vários problemas reais.

Por fim, o Capítulo 7 discute os principais resultados alcançados com a abordagem proposta e os possíveis trabalhos que podem surgir a partir desta dissertação.

2 *Problema de Agrupamento como Tarefa de Otimização*

O problema fundamental de particionamento (*clustering*) consiste em agrupar padrões que apresentem algum tipo de similaridade (HARTIGAN, 1975) (FASULO, 1999). Ao longo deste trabalho, os padrões que formam o conjunto de dados também são denotados por elementos, objetos, pontos, variáveis, itens ou exemplos.

A tarefa de agrupar é uma das atividades mais utilizadas pela mente humana. Para processar as inúmeras informações disponíveis, os seres humanos costumam categorizar entidades (objetos, pessoas, eventos) em "partições" e abstrair apenas as propriedades comuns às entidades de cada uma dessas partições. Por exemplo, o conceito de livro é representado na mente por um objeto com capa, folhas, título, capítulos e outras propriedades. Ao visualizar uma nova entidade da classe livro, uma pessoa consegue categorizá-la corretamente, pois essa nova entidade apresenta as mesmas características que as demais entidades da classe livro, que já foram previamente "agrupadas".

O conceito de particionamento pode ser empregado de várias formas, dependendo da área de aplicação. Alguns exemplos são citados a seguir:

- Mineração de dados: O resultado do agrupamento permite encontrar semelhanças e diferenças entre os padrões e inferir conclusões úteis sobre eles, contribuindo para a descoberta de novas informações inerentes aos dados. Exemplo do uso dessa abordagem pode ser visto em (FAYYAD et al., 1996).

- Aprendizado de Máquina: Cada partição pode ser considerada como uma representação de uma classe de padrões, permitindo classificar novos padrões de acordo com a partição em que for inserido. Como a classe dos padrões não precisa estar disponível para que esses sejam agrupados, o processo de particionamento pode ser chamado de aprendizado não-supervisionado. Um exemplo dessa aplicação é apresentado em (HASTIE; TIBSHIRANI; FRIEDMAN, 2003) que empregam o agrupamento para classificar tipos de câncer

humano.

- Reconhecimento de padrões: A similaridade dos objetos de cada partição permite categorizá-los em diferentes padrões. Novos elementos, após associados a uma das partições, podem ser identificados pelas mesmas propriedades dos padrões pertencentes a essa partição. Por exemplo, (LEVRAT et al., 1992) utilizam algoritmos de particionamento para segmentar imagens de forma a detectar transições entre regiões.

- Compressão de dados: O particionamento é usado para agrupar um grande volume de dados em um número menor de partições. Dessa forma, cada partição é processada como uma entidade singular, utilizando-se menos espaço para representação dos dados. (ZHANG; RAMAKRISHNAN; LIVNY, 1997) apresentam um método eficiente de particionamento de dados utilizado para compressão de imagem.

- Análise estatística: O uso de agrupamento permite identificar estatísticas descritivas sobre o conjunto de dados, identificando, por exemplo, a distribuição amostral desses dados. (KAUFMAN; ROUSSEEUW, 1990) e (BANFIELD; RAFTERY, 1993) apresentam aplicações nessa área.

2.1 Definição Formal para o Problema de Agrupamento

Formalmente, dado um conjunto de dados X com N padrões $\{\mathbf{x_1},...,\mathbf{x_N}\}$, sendo que cada padrão $\mathbf{x_i} = [x_{i1}, x_{i2}, \ldots, x_{id}]^t$ possui d atributos (características ou dimensões), deseja-se encontrar K partições $\{C_1,...,C_K\}$, tal que padrões pertencentes a uma mesma partição estão mais relacionados entre si e menos relacionados em relação aos padrões associados às outras partições. Além disso, as partições devem atender às seguintes condições (THEODORIDIS; KOUTROUMBAS, 2006):

- $C_j \neq \emptyset,\ j = 1, \ldots, K$
- $\bigcup_{j=1}^{K} C_j = X$
- $C_i \cap C_j = \emptyset,\ i \neq j,\ i, j = 1, \ldots, K$

Agrupamentos que atendam à terceira condição, indicando que cada padrão deva fazer parte de uma única partição, são chamados de *hard* ou *crisp*. Existe uma outra definição de agrupamento, na qual um padrão está associado a cada partição de acordo com um grau de pertinência.

Esse tipo de agrupamento, conhecido como *fuzzy clustering*, não faz parte do escopo deste trabalho, que aborda o particionamento *hard*, utilizado pela maioria das aplicações na área de Descoberta de Conhecimento em Banco de Dados - *Knowledge Discovery in Database* (*KDD*).

A Figura 2.1 mostra um exemplo de agrupamento que segue a definição acima para agrupamentos do tipo *hard*. Nesse exemplo, os sete pontos {A, B, C, D, E, F, G} foram agrupados em três partições, indicando que os padrões {A, B, C} são mais similares entre si do que em relação aos demais, assim como os padrões {D, E} e {F, G}.

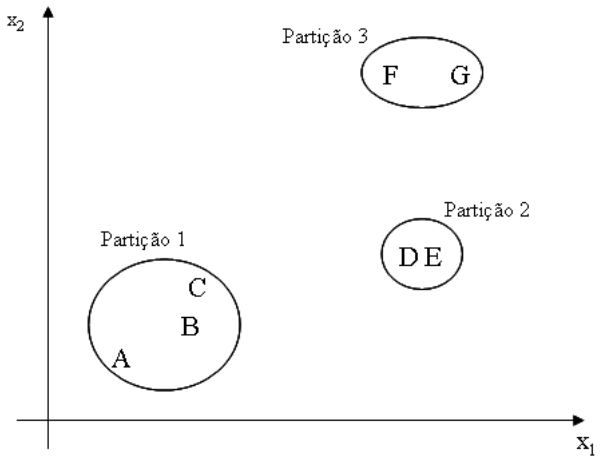

Figura 2.1: Sete pontos agrupados em três partições (JAIN; MURTY; FLYNN, 1999)

A partir desse exemplo e da própria definição de agrupamento, pode-se observar que um elemento essencial no processo de identificação de grupos em um conjunto de dados é a informação de quão semelhantes ou quão diferentes os padrões são entre si. A maioria dos algoritmos de agrupamento necessitam de uma medida quantitativa que represente a proximidade entre os padrões. Essa medida pode ser tanto uma medida de similaridade, que avalia a semelhança ou a afinidade entre os padrões, como uma medida de dissimilaridade, que avalia a diferença ou falta de afinidade entre eles. Assim, dois padrões são considerados próximos quando uma medida de similaridade entre eles for grande ou quando uma medida de dissimilaridade for pequena.

As medidas de similaridade são mais comumente utilizadas em padrões cujos atributos são todos categóricos (simbólicos) (EVERITT; LANDAU; LEESE, 2001). Como este trabalho não pretende abordar casos que apresentem esse tipo de atributo, as medidas de similaridade não serão abordadas nesta discussão.

Quando todos os atributos forem numéricos, a proximidade entre os padrões é geralmente quantificada pela medida de dissimilaridade. Uma medida de dissimilaridade δ em X é uma função definida formalmente por (THEODORIDIS; KOUTROUMBAS, 2006) como:

$$\delta : X \times X \rightarrow \Re \tag{2.1}$$

tal que

$$\exists \delta_0 \in \Re : -\infty < \delta_0 \leq \delta(\mathbf{x},\mathbf{y}) < \infty, \quad \forall \mathbf{x},\mathbf{y} \in X \tag{2.2}$$

$$\delta(\mathbf{x},\mathbf{x}) = \delta_0, \quad \forall \mathbf{x} \in X \tag{2.3}$$

$$\delta(\mathbf{x},\mathbf{y}) = \delta(\mathbf{y},\mathbf{x}), \quad \forall \mathbf{x},\mathbf{y} \in X \tag{2.4}$$

Um exemplo de medida de dissimilaridade comumente utilizada é a distância de Minkowski de ordem p (DEVIJVER; KITTLER, 1982), que é aplicada a padrões com atributos numéricos contínuos. Essa medida é definida por

$$\delta_{Minkowski}(\mathbf{x},\mathbf{y}) = \left(\sum_{l=1}^{d} |x_l - y_l|^p \right)^{1/p} \tag{2.5}$$

onde x_l, y_l são as coordenadas l de $\mathbf{x}$ e $\mathbf{y}$, respectivamente.

Um caso especial dessa medida, obtido quando $p = 2$, é a distância Euclidiana (DEVIJVER; KITTLER, 1982) que é a medida mais utilizada para atributos numéricos contínuos:

$$\delta_{Euclidiana}(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\| = \sqrt{\sum_{l=1}^{d} (x_l - y_l)^2} \tag{2.6}$$

Outros dois casos especiais da distância de Minkowski usados na prática são a distância Manhattan, também chamada de *City Block* (DEVIJVER; KITTLER, 1982), obtida com $p = 1$

$$\delta_{Manhattan}(\mathbf{x},\mathbf{y}) = \sum_{l=1}^{d} |x_l - y_l| \tag{2.7}$$

e a distância de Chebychev (DEVIJVER; KITTLER, 1982), obtida quando $p = \infty$

$$\delta_{Chebychev}(\mathbf{x},\mathbf{y}) = \max_{1 \leq l \leq d} \{|x_l - y_l|\} \tag{2.8}$$

Pode-se citar outras duas medidas de dissimilaridade, que são menos utilizadas que a de Minkowski, mas que também são aplicadas a padrões com atributos contínuos, que são as distâncias δ_G e δ_Q (SPATH, 1980):

$$\delta_G(\mathbf{x},\mathbf{y}) = -log_{10}\left(1 - \frac{1}{d}\sum_{l=1}^{d} \frac{|x_l - y_l|}{b_l - a_l}\right) \tag{2.9}$$

onde b_l e a_l são, respectivamente, os valores máximo e mínimo do l-ésimo atributo dos N padrões de X.

$$\delta_Q(\mathbf{x},\mathbf{y}) = \sqrt{\frac{1}{d}\sum_{l=1}^{d}\left(\frac{x_l - y_l}{x_l + y_l}\right)^2} \tag{2.10}$$

Um exemplo de medida de dissimilaridade aplicada entre padrões com atributos discretos é a distância de Hamming (HAMMING, 1950). Essa distância define o número de atributos no qual dois padrões **x** e **y** diferem e é calculada por

$$\delta_{Hamming}(\mathbf{x},\mathbf{y}) = \sum_{l=1}^{d} t(x_l, y_l) \tag{2.11}$$

onde $t(x_l, y_l) = 1$, se $x_l \neq y_l$ e $t(x_l, y_l) = 0$, caso contrário.

A aplicação de diferentes medidas de dissimilaridade sobre o mesmo conjunto de dados pode levar a diferentes resultados. Por isso, a escolha da medida mais adequada para uma instância do problema é uma tarefa de fundamental importância no processo de agrupamento. Como não existe uma resposta exata e única para essa escolha, sugere-se que a decisão seja guiada pelo tipo dos atributos dos padrões e pela intuição do investigador.

2.2 Abordagens de Solução para o Problema de Agrupamento

Uma forma de visualizar o problema de particionamento é considerá-lo como uma tarefa de otimização que corresponde a encontrar a melhor combinação de partições dentre todas as combinações possíveis, de acordo com um critério de avaliação da qualidade das partições (RAYWARD-SMITH, 2005). Essa abordagem requer a definição de uma função $c : \{C_j : C_j \subseteq X\} \rightarrow \Re^+$ que associa um custo a cada grupo. O objetivo, nesse caso, é encontrar o conjunto de partições que otimize a soma dos custos de todos os grupos $\sum_{j=1}^{K} c(C_j)$.

A partir dessa formulação, pode-se deduzir que esse é um problema que pertence à classe *NP-hard* (GAREY; JOHNSON, 1979). O algoritmo conhecido para resolvê-lo de maneira ótima realiza a avaliação de todas as possíveis soluções, o que torna o problema computacionalmente inviável, principalmente para instâncias maiores, já que o número de diferentes partições de N padrões em K grupos é dado por (LIU, 1968):

$$S(N,K) = \frac{1}{K!}\sum_{j=1}^{K}(-1)^{K-j}\left(\begin{array}{c} K \\ j \end{array}\right) j^N \tag{2.12}$$

Por isso, algoritmos voltados para esse tipo de problema realizam uma busca por soluções

aproximadas, que são consideradas satisfatórias se comparadas ao alto custo computacional envolvido na resolução baseada na enumeração.

Alguns algoritmos de agrupamento, chamados de particionais ou de otimização, realizam a tarefa de particionamento dividindo um conjunto de dados X em K partições $\{C_1,...,C_K\}$ com o objetivo de otimizar um critério de avaliação (função de custo). O número de partições K que devem ser formadas é fornecido como um parâmetro de entrada desses algoritmos. Escolher o valor mais adequado para esse parâmetro, caso não seja conhecido, depende intrinsicamente do problema a ser resolvido. Algumas discussões a respeito dessa decisão estão abordadas em (DUBES, 1987) e (XU, 1996).

A diferença nos algoritmos particionais é dada pela variedade de critérios que podem ser otimizados e pelas diferentes técnicas que podem ser usadas para obter o valor ótimo do critério escolhido.

O critério mais conhecido e aplicado em algoritmos particionais 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). Esse critério é definido por

$$SSE = \sum_{j=1}^{K} \sum_{\mathbf{x_i} \in C_j} \|\mathbf{x_i} - \mu_\mathbf{j}\|^2 \tag{2.13}$$

onde $\|\mathbf{x_i} - \mu_\mathbf{j}\|$ é a distância Euclidiana entre o padrão $\mathbf{x_i}$ e o centróide $\mu_\mathbf{j}$.

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

$$\mu_\mathbf{j} = \frac{1}{n_j} \sum_{\mathbf{x_i} \in C_j} \mathbf{x_i} \tag{2.14}$$

onde n_j é o total de padrões pertencentes à partição C_j.

Um critério semelhante ao *SSE* é a soma das distâncias entre padrões pertencentes à mesma partição (*Within cluster point scatter*) (HASTIE; TIBSHIRANI; FRIEDMAN, 2003) e é calculado por

$$W = \frac{1}{2} \sum_{j=1}^{K} \sum_{\mathbf{x_i} \in C_j} \sum_{\mathbf{x'_i} \in C_j} \delta(\mathbf{x_i}, \mathbf{x'_i}) \tag{2.15}$$

Outro critério utilizado em algoritmos particionais é calculado pela soma das distâncias entre padrões pertencentes a diferentes partições (*Between cluster point scatter*) (HASTIE; TIB-

SHIRANI; FRIEDMAN, 2003). Esse critério é dado por

$$B = \frac{1}{2} \sum_{j=1}^{K} \sum_{\mathbf{x_i} \in C_j} \sum_{\mathbf{x'_i} \notin C_j} \delta(\mathbf{x_i}, \mathbf{x'_i}) \tag{2.16}$$

Os critérios *SSE* e *W* devem ser minimizados para encontrar a partição ótima, pois indicam a coesão entre os elementos, e o critério *B* precisa ser maximizado para atingir o mesmo objetivo, já que mede a dispersão dos dados. Uma revisão mais detalhada sobre critérios de avaliação é feita em (EVERITT; LANDAU; LEESE, 2001).

Alguns exemplos de algoritmos particionais comumente utilizados são o *K-means* (FORGY, 1965) (MACQUEEN, 1967), o *PAM* (*Partitioning Around Medoids*) (KAUFMAN; ROUSSEEUW, 1990) e o *CLARA* (*Clustering LARge Applications*) (KAUFMAN; ROUSSEEUW, 1990). Todos eles realizam uma busca heurística para determinar o ponto representativo de cada partição.

No *K-means* o ponto representativo não necessariamente precisa ser um ponto do conjunto de dados, e é calculado como o centro de massa da partição. Já no *PAM* e no *CLARA*, o ponto representativo deve ser o elemento do conjunto de dados localizado o mais próximo possível da região central da partição, sendo que o *CLARA*, ao invés de encontrar pontos representativos para todo o conjunto de dados, como é feito pelo *PAM*, extrai múltiplas amostras dos dados, aplica o *PAM* e encontra os pontos representativos para cada amostra, retornando o melhor resultado das diferentes amostras.

Nesses algoritmos, uma vez que os pontos representativos são calculados, os padrões são associados à partição em relação à qual possuam 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. Essa busca é considerada, portanto, uma forma de minimizar o critério *SSE*.

Ainda dentro da classe dos algoritmos particionais, encontra-se a abordagem baseada em grafos. Nessa abordagem, os objetos podem ser representados por vértices de um grafo e as distâncias entre quaisquer dois objetos por arestas com peso. O método de agrupamento mais conhecido para essa abordagem é baseado na construção da árvore geradora mínima (*Minimal Spanning Tree - MST*) do grafo (ZAHN, 1971). A partir da árvore geradora mínima, retiram-se as arestas de maior peso para gerar tantas componentes conexas (*clusters*) quanto forem necessárias. Esse procedimento realiza uma construção gulosa da solução, pois faz sempre a melhor escolha pontual e não geral, e pode ser visto como um método que calcula a solução minimizando o critério *SSE*.

Algoritmos particionais são, portanto, procedimentos heurísticos voltados especificamente

para o problema de agrupamento. No entanto, esses procedimentos encontram geralmente soluções aproximadas que correspondem a ótimos locais do critério de avaliação. (SELIM; ISMAIL, 1984) mostram, por exemplo, que o algoritmo *K-means* pode convergir para um mínimo local do critério *SSE*.

Outro método de resolução de problemas de otimização está baseado em procedimentos heurísticos com caráter mais geral, chamados de meta-heurísticas. As meta-heurísticas englobam um conjunto de procedimentos heurísticos que podem ser aplicados a uma variedade de problemas combinatoriais, necessitando de poucas modificações para se adaptar ao problema específico. Além disso, esses procedimentos de resolução buscam a interação entre procedimentos de melhoria local e estratégias gerais para escapar de ótimos locais e para percorrer de forma robusta o espaço de soluções (GLOVER; KOCHENBERGER, 2003). Maiores detalhes sobre as meta-heurísticas são apresentados no Capítulo 4.

Meta-heurísticas são, portanto, métodos de otimização que podem ser aplicados ao problema de agrupamento, já que utilizam procedimentos heurísticos gerais que são facilmente adaptáveis ao problema de agrupamento para realizar a busca pela solução ótima (ou subótima) do critério de avaliação das partições.

(RAYWARD-SMITH, 2005) apresenta uma revisão de vários trabalhos que aplicam meta-heurísticas para resolver o problema de particionamento de dados. Entretanto, esse estudo conclui que a maioria das meta-heurísticas aplicadas ao problema de agrupamento não são escalonáveis para bases reais e comerciais, mostrando-se mais efetivas nos casos em que a instância do problema é menor (com número menor de padrões e de dimensões). Adicionalmente, o custo computacional necessário para calcular as soluções se torna maior em instâncias maiores do problema.

Por isso, procedimentos híbridos que exploram a combinação de técnicas de resolução de problemas de otimização representam uma abordagem promissora para resolver o problema de agrupamento. (BABU; MURTY, 1993), por exemplo, aplicam algoritmos genéticos para selecionar a solução inicial para o *K-means*, superando os resultados da aplicação direta do algoritmo genético. Os mesmos autores apresentam uma outra versão que utiliza *Simulated Annealing* para a mesma tarefa de inicialização do *K-means* (BABU; MURTY, 1994). O Capítulo 3 apresenta maiores detalhes sobre o algoritmo *K-means* e os métodos de inicialização existentes e o Capítulo 4 discute o uso de meta-heurísticas.

Como o problema de particionamento de dados pode ser considerado uma tarefa de otimização, outra forma de resolvê-lo é com o uso de programação matemática, que busca minimizar ou maximizar uma função objetivo linear ou não-linear sujeita a restrições. Nesse caso, o prob-

lema de particionamento deve ser formalizado em um modelo matemático no qual a função objetivo e as restrições devem ser definidas adequadamente.

O uso sistemático dessa abordagem teve início com os trabalhos de (VINOD, 1969) e (RAO, 1971). (HANSEN; JAUMARD, 1997) apresentam uma revisão de trabalhos posteriores a esses que estudam particionamento via programação matemática, detalhando os diferentes critérios de avaliação e formalizando alguns modelos matemáticos para esse problema. Outro trabalho mais recente (MANGASARIAN, 1997) propõe uma formalização matemática do problema de agrupamento justificada teoricamente, implementada computacionalmente e empregada em um conjunto de dados para descobrir informações úteis sobre pacientes com câncer de mama.

Vários outros algoritmos de particionamento estão disponíveis na literatura, como é o caso, por exemplo, dos algoritmos hierárquicos (FASULO, 1999; JAIN; MURTY; FLYNN, 1999). Porém, como não resolvem o problema através de um critério de otimização, esses procedimentos não estão abordados nesta dissertação. Este trabalho realiza um estudo mais aprofundado sobre o algoritmo *K-means* e meta-heurísticas aplicados ao problema de particionamento, com ênfase na melhoria dos resultados obtidos por esses algoritmos, especialmente em instâncias maiores do problema.

3 *Métodos de Inicialização Voltados para o Algoritmo K-means*

Este capítulo descreve um dos mais populares algoritmos de particionamento, o *K-means*, e apresenta alguns métodos de escolha da solução inicial que têm como objetivo aprimorar os resultados desse algoritmo, e que possam, porventura, tornar determinística essa etapa de inicialização.

3.1 O Algoritmo *K-means*

O *K-means* é uma heurística que realiza a busca pela melhor solução (ou por uma solução subótima próxima à melhor) para o problema de agrupamento de dados. O objetivo é encontrar, portanto, a melhor combinação de K partições $\{C_1,...,C_K\}$ em um conjunto de dados $X = \{\mathbf{x_1},...,\mathbf{x_N}\}$, em que os d atributos de cada variável $\mathbf{x_i}$ são valores numéricos. Tal heurística é amplamente utilizada em diversas aplicações, devido à sua simplicidade, facilidade de implementação e rápida convergência.

A busca pela solução é realizada por um algoritmo iterativo que objetiva associar 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, que indica o ponto representativo da partição. Por isso, minimizar a dissimilaridade entre um padrão e um centróide significa associar o padrão à partição mais próxima.

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 do conjunto de dados como centróides, que representam a solução inicial do

problema. Após a inicialização, o algoritmo gera, a cada iteração, uma nova solução (novo conjunto de centróides), associando cada padrão do conjunto de dados ao centróide mais próximo da solução corrente, e recalcula os centróides após todos os padrões serem associados aos centróides, até atingir um critério de convergência. Alguns possíveis 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 1 apresenta o procedimento de busca de solução do *K-means*, segundo (FORGY, 1965).

Algoritmo 1 Calcula os centróides $\{\mu_1, ..., \mu_K\}$ de um conjunto de dados X usando seleção aletória 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 $\mathbf{x_i}$ ao centróide $\mu_\mathbf{j}$ mais próximo
4: Recalcule cada centróide como sendo o centro de massa de todos os pontos associados a ele: $\mu_\mathbf{j} = \frac{1}{n_j} \sum_{\mathbf{x} \in C_j} \mathbf{x}$
5: **Fim Enquanto**
6: **Retorne** $\{\mu_1, ...\mu_K\}$

A versão do *K-means* dada por (MACQUEEN, 1967) difere da versão de Forgy apenas na atualização dos centróides. Na versão de MacQueen os centróides são recalculados depois de cada associação do padrão ao centróide mais próximo, diferente da versão de Forgy que recalcula os centróides após todos os padrões serem associados aos centróides. Neste trabalho, os resultados apresentados foram obtidos usando-se a versão de Forgy, por ser um procedimento mais eficiente.

Apesar das vantagens mencionadas, o *K-means* apresenta um problema decorrente da sua sensibilidade à seleção da solução inicial. Por ser uma heurística baseada numa abordagem gulosa, o algoritmo pode convergir para um mínimo local do critério de avaliação, se a solução inicial não for escolhida apropriadamente. Por exemplo, na Figura 3.1 são apresentados os mesmos sete objetos da Figura 2.1 do Capítulo 2, agora agrupados pelo *K-means*.

Se o algoritmo precisa encontrar três partições e a solução inicial escolhida for composta pelos objetos A, B e C, então o resultado do *K-means* é dado pelos grupos {A}, {B,C} e {D,E,F,G}, como mostram as elipses na figura. No entanto, o resultado mais adequado é definido pelos grupos {A,B,C}, {D,E} e {F,G}, mostrados nos retângulos. Esse resultado seria obtido se a solução inicial fosse composta pelos objetos A, D e F.

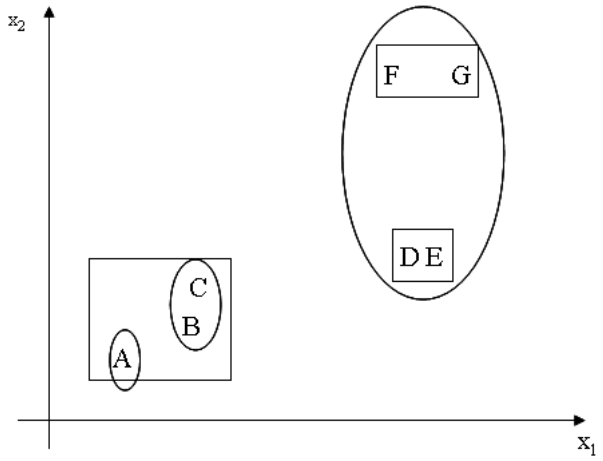

Figura 3.1: Sensibilidade do *K-means* à partição inicial (JAIN; MURTY; FLYNN, 1999)

Além disso, como o algoritmo padrão é implementado usando uma seleção aleatória da solução inicial, os resultados obtidos diferem a cada execução, ou seja, podem não ser reproduzidos novamente.

Por isso, recentes trabalhos apresentam estudos sobre métodos de inicialização com o objetivo de aprimorar o resultado do *K-means* e/ou de garantir uma seleção determinística das soluções iniciais.

3.2 Métodos de Inicialização para o *K-means*

Um método de inicialização é um procedimento de escolha ou melhoria da solução inicial do *K-means* que tem como objetivo principal fugir de mínimos locais, que são facilmente encontrados com a seleção inicial arbitrária do algoritmo padrão. Outra característica dos métodos de inicialização é que eles ajudam a acelerar o processo de convergência do *K-means*, já que fornecem soluções mais próximas da solução final.

Pesquisas nessa área, tais como (KATSAVOUNIDIS; KUO; ZHANG, 1994; SU; DY, 2004; ARTHUR; VASSILVITSKII, 2007) e outros, têm apresentado importantes resultados, comprovando teórica e empiricamente que os métodos de inicialização contribuem para a melhoria da solução obtida com o *K-means*.

Um dos primeiros trabalhos nessa área apresenta o método de inicialização *Simple Cluster Seeking* (TOU; GONZALEZ, 1974). Esse método consiste em escolher centróides que sejam distantes entre si, considerando que esses pontos têm mais chances de pertencer a grupos diferentes. Para isso, o método seleciona o centróide inicial como sendo o primeiro elemento do conjunto de dados. Para os próximos elementos do conjunto de dados, o método seleciona

como centróide o elemento que está distante dos centróides já selecionados por um valor pré-definido ρ, até encontrar K centróides. Esse método de inicialização, no entanto, é sensível ao parâmetro ρ e à ordem de apresentação dos dados.

Um trabalho similar é apresentado em (KATSAVOUNIDIS; KUO; ZHANG, 1994), que propõem um método eficiente para obter soluções iniciais que permitem acelerar o processo de convergência do algoritmo e alcançar mínimos locais melhores. Esse método seleciona o elemento com maior norma como o centróide inicial. Após isso, o método encontra, iterativamente, um novo centróide, calculando as distâncias entre os demais elementos e o centróide mais próximo e selecionando o ponto que apresentar a maior distância.

Um outro trabalho sobre métodos de inicialização do *K-means* é apresentado em (BRADLEY; FAYYAD, 1998). Esse trabalho propõe um método no qual uma solução inicial qualquer, escolhida aleatoriamente ou usando outro método de inicialização, é refinada antes de ser aplicada ao algoritmo do *K-means*. Esse método de refinamento baseia-se em uma técnica eficiente para estimar os parâmetros da distribuição dos dados, determinando-se a moda da densidade de probabilidade de cada partição. A moda corresponde ao elemento da amostra de dados que maximiza a função de densidade de probabilidade.

A técnica empregada consiste em extrair múltiplas subamostras dos dados e aplicar um algoritmo de agrupamento sobre essas subamostras. Os experimentos realizados em (BRADLEY; FAYYAD, 1998) mostram um ganho na qualidade da solução em relação ao algoritmo padrão do *K-means*, tanto em bases sintéticas quanto em bases reais.

(SUN; ZHU; CHEN, 2002) apresentam uma continuação do trabalho de (BRADLEY; FAYYAD, 1998), aplicando esse mesmo método de refinamento de soluções iniciais em conjunto com o algoritmo *K-modes* (HUANG, 1997), que é uma extensão do *K-means* para agrupar conjuntos de dados com atributos numéricos e categóricos. (SUN; ZHU; CHEN, 2002) mostram experimentalmente que o *K-modes* usando solução inicial refinada obteve resultados mais precisos e confiáveis do que os obtidos com o método de seleção aleatória sem refinamento.

(SU; DY, 2007) realizam uma análise comparativa de métodos de inicialização determinísticos que competem com o método aleatório em termos de eficiência. Nesse trabalho, três métodos determinísticos foram analisados: o método proposto por (KATSAVOUNIDIS; KUO; ZHANG, 1994), o método de divisão baseado na análise dos componentes principais *PCA_Part* (SU; DY, 2004) e o método de partição por variância *Var_Part* (SU; DY, 2007).

O método *Var_Part* é bastante similar ao *PCA_Part*. A única diferença entre eles é que o método *PCA_Part* secciona os dados do espaço euclidiano com um hiperplano ortogonal à

direção do maior autovetor que passa pelo centróide da partição a ser dividida, enquanto que o método *Var_Part* secciona da mesma maneira, porém com um hiperplano ortogonal à dimensão com maior variância. O método *PCA_Part* é descrito em detalhes na Seção 3.2.1.

(SU; DY, 2007) avaliaram experimentalmente o uso do *K-means* com os três métodos determinísticos, com o método aleatório padrão e com o método de amostragem refinada de (BRADLEY; FAYYAD, 1998) e constataram para os experimentos realizados que os métodos *PCA_Part* e *Var_Part* apresentam resultados similares, sendo, no entanto, melhores do que os outros três métodos de inicialização.

Diferente dos trabalhos citados acima que investem na melhoria dos resultados a partir da escolha da solução inicial, (GRIRA; HOULE, 2007) propõem uma melhoria no procedimento de busca do *K-means* de forma a acelerar o processo de convergência sem aumentar o custo computacional das iterações. Essa melhoria consiste em realizar um número de trocas dos centróides, semelhante ao da heurística *K-medoids* (KAUFMAN; ROUSSEEUW, 1990). Os experimentos realizados nesse trabalho mostram que o procedimento proposto apresenta uma melhoria na convergência dos resultados em relação ao *K-means*, a um custo computacional compatível.

As próximas seções descrevem em detalhes os métodos de inicialização empregados nesta dissertação. O primeiro a ser discutido é o método *PCA_Part* que apresenta uma abordagem de divisão dos dados na direção do primeiro eixo principal no espaço de características. Esse método foi escolhido por realizar uma escolha determinística e apresentar bons resultados comparado a outros métodos. O segundo método a ser abordado é o aplicado no algoritmo *K-means++* que seleciona os centróides iniciais com probabilidade proporcional à contribuição para o critério de avaliação. A razão para escolha desse método é a garantia da proximidade percentual do ótimo global.

3.2.1 *PCA_Part*: Método de Inicialização Baseado na Divisão pela Direção Principal

O método de inicialização para o *K-means*, apresentado em (SU; DY, 2004), utiliza uma abordagem hierárquica de divisão baseada na técnica *PCA* (*Principal Component Analysis*) (JOLLIFFE, 1986).

Essa abordagem de divisão foi introduzida 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 três 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 relativo ao método *PCA_Part* gera, inicialmente, uma única partição formada por todo o conjunto de 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 $\mu_{\mathbf{j1}}$ e $\mu_{\mathbf{j2}}$, respectivamente, na direção do maior autovetor da matriz de covariância. Essa divisão da partição C_j é feita projetando cada padrão $\mathbf{x_i} \in C_j$ para a primeira direção principal, gerando os vetores $\mathbf{y_i}$. O mesmo é feito com o centróide $\mu_\mathbf{j}$ de C_j, gerando o vetor $\alpha_\mathbf{j}$. O processo é repetido até que K partições sejam geradas. O Algoritmo 2 apresenta o método de inicialização *PCA_Part*.

Algoritmo 2 Calcula os centróides iniciais $\{\mu_\mathbf{1}, ..., \mu_\mathbf{K}\}$, dado um conjunto de dados X, usando abordagem hierárquica baseada na técnica PCA

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

1: $C_1 \leftarrow X$
2: **Enquanto** (os K centróides não forem gerados) **faça**
3: Escolha a partição C_j com o maior *SSE*
4: Projete todo $\mathbf{x_i} \in C_j$ e o centróide $\mu_\mathbf{j}$ para a primeira direção principal: $\mathbf{y_i}$ e $\alpha_\mathbf{j}$, respectivamente
5: **Para todo** $\mathbf{x_i} \in C_j$ **faça**
6: **Se** $y_i \leq \alpha_j$ **então**
7: $C_{j1} \leftarrow C_{j1} \cup \{\mathbf{x_i}\}$
8: **Senão**
9: $C_{j2} \leftarrow C_{j2} \cup \{\mathbf{x_i}\}$
10: **Fim Se**
11: **Fim Para**
12: **Fim Enquanto**
13: **Retorne** $\{\mu_\mathbf{1}, ...\mu_\mathbf{K}\}$

O objetivo do método *PCA_Part* é minimizar o valor de *SSE* a cada iteração. Por isso, 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:

$$SSE_{old} = \sum_{\mathbf{x_i} \in C_j} \|\mathbf{x_i} - \mu_\mathbf{j}\|^2 \tag{3.1}$$

Após a divisão, o novo valor de *SSE* passa a ser:

$$SSE_{new} = \sum_{\mathbf{x_i} \in C_{j1}} \|\mathbf{x_i} - \mu_{\mathbf{j1}}\|^2 + \sum_{\mathbf{x_i} \in C_{j2}} \|\mathbf{x_i} - \mu_{\mathbf{j2}}\|^2 \tag{3.2}$$

A direção que minimiza SSE_{new} é a mesma direção que maximiza a diferença entre SSE_{old} e SSE_{new} projetados. O problema é simplificado para encontrar a direção que contribui para o maior SSE_{old}, que é determinada pela direção do maior autovetor da matriz de covariância (FUKUNAGA, 1990). A Figura 3.2 mostra a abordagem de divisão proposta pelo *PCA_Part*.

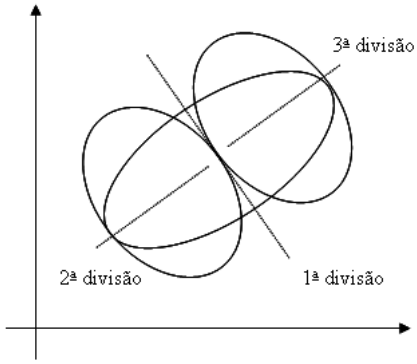

Figura 3.2: Particionamento hierárquico a partir da covariância amostral em duas dimensões (SU; DY, 2007)

O aspecto interessante desse método é que, além de, potencialmente, fornecer um resultado melhor que o aleatório, ele realiza uma escolha determinística, que pode ser reproduzida a cada execução.

No entanto, uma fragilidade desse método é que a escolha da primeira direção principal somente maximiza o valor SSE_{old} na direção da componente principal e não considera o efeito de 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 possível extensão para o método *PCA_Part*, o método chamado de *PCA_Part**, que seleciona a componente *PCA* que maximiza a diferença entre SSE_{old} e SSE_{new} como a direção para projetar os dados na divisão. Contudo, 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*, por um fator *d*, que corresponde ao número de dimensões. Por isso, este trabalho utiliza o método *PCA_Part*, como sugerem (SU; DY, 2007).

(SU; DY, 2004) mostram que, experimentalmente, o *K-means* inicializado com o método

PCA_Part apresentou, em alguns casos, uma melhoria nos resultados, obtendo valores de *SSE* próximos aos menores valores encontrados com 100 execuções do *K-means* inicializado aleatoriamente. No entanto, em algumas bases, o *K-means* inicializado com o *PCA_Part* não obteve resultados melhores que os obtidos com a versão aleatória. Provavelmente, a razão para esse fato se deve à violação da hipótese de que a primeira direção principal maximiza a diferença entre SSE_{old} e SSE_{new} projetados.

3.2.2 *K-means++*: Método de Inicialização Baseado em Probabilidade Específica

O trabalho proposto por (ARTHUR; VASSILVITSKII, 2007) sugere um método de inicialização baseado em uma escolha aleatória dos centróides com probabilidade específica, no qual um ponto *p* é escolhido como centróide com probabilidade proporcional à contribuição desse ponto para o critério *SSE*. A motivação desse método é dada pela existência de muitos problemas reais em que o algoritmo padrão do *K-means* gera arbitrariamente partições muito ruins. Nesses casos, a razão entre o critério *SSE* obtido e o ótimo não é limitada por uma constante, mesmo com *N* e *K* fixos.

O método de escolha dos centróides proposto em (ARTHUR; VASSILVITSKII, 2007) escolhe 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*.

Considerando que $D(\mathbf{x})$ denote a distância Euclidiana entre o ponto $\mathbf{x}$ e o centróide mais próximo a ele, dentre os centróides escolhidos nas iterações anteriores, e seja a probabilidade de escolha do padrão $\mathbf{x_i} \in X$ definida como

$$P(\mathbf{x_i}) = \frac{D(\mathbf{x_i})^2}{\sum_{\mathbf{x} \in X} D(\mathbf{x})^2} \tag{3.3}$$

Então, quanto maior a contribuição do ponto para o critério *SSE*, maior é a probabilidade desse ponto ser escolhido como centróide. Dessa maneira, o método seleciona os centróides de forma a minimizar o critério de avaliação. O Algoritmo 3 apresenta esse método de inicialização, chamado de *K-means++*.

Esse trabalho complementa um resultado recente de (OSTROVSKY et al., 2006) que propõem independentemente um método de inicialização semelhante. A diferença é que, no método de (OSTROVSKY et al., 2006), são escolhidos inicialmente dois pontos como centróides com probabilidade proporcional à distância Euclidiana entre eles. Em seguida, são escolhidos os

Algoritmo 3 Calcula os centróides iniciais $\{\mu_1, ..., \mu_K\}$, 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 restantes $(K-1)$ não for gerado) **faça**
3: Escolha o próximo centróide μ_j, selecionando $\mu_j = x_i \in X$ com probabilidade $\frac{D(x_i)^2}{\sum_{x \in X} D(x)^2}$
4: **Fim Enquanto**
5: **Retorne** $\{\mu_1, ...\mu_K\}$

demais $K-2$ centróides, selecionando os pontos com probabilidade proporcional à distância Euclidiana entre o ponto e o centróide mais próximo. No entanto, (OSTROVSKY et al., 2006) mostram que esse método encontra soluções próximas ao ótimo na ordem de $O(log(1))$, em dados que seguem 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.

O trabalho de (ARTHUR; VASSILVITSKII, 2007) mostra, portanto, que o método proposto, além de ser 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 $O(log(K))$:

$$E(SSE) \leq 8(ln(k)+2)SSE_{OPT} \tag{3.4}$$

Esse resultado representa um grande avanço em relação aos demais trabalhos apresentados nesta dissertação, pois obtém soluções próximas à ótima, com uma aproximação proporcional a uma constante na ordem de $O(log(K))$, enquanto os outros não possuem essa garantia.

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

4 *Uso de Meta-heurísticas para o Problema de Agrupamento*

Problemas de otimização são aqueles que, dado um conjunto S de variáveis discretas ou contínuas s (chamadas de soluções do problema) e uma função objetivo $f : S \rightarrow \Re$, que associa cada $s \in S$ a um valor real $f(s)$, deseja-se encontrar a solução ótima $s^* \in S$ tal que $f(s)$ seja minimizada ($f(s^*) \leq f(s), \forall s \in S$) ou maximizada ($f(s^*) \geq f(s), \forall s \in S$) (BLUM; ROLI, 2003; STüTZLE, 1999). Quando o problema possui um conjunto de soluções discretas, esse passa a ser chamado de problema de otimização combinatória.

O conjunto S é geralmente chamado de espaço de busca ou espaço de soluções, sendo que cada variável s pode ser vista como uma solução candidata. O subconjunto $S^* \subseteq S$ contém as soluções ótimas s^* do problema, também chamadas de ótimo global da função objetivo. Ótimos locais são soluções cujo valor da função objetivo é ótimo em relação às soluções próximas (vizinhas).

Algoritmos voltados para resolver problemas de otimização combinatoriais podem ser classificados como exatos ou de aproximação. Algoritmos exatos garantem que, para qualquer instância do problema de tamanho finito, a solução ótima pode ser encontrada. No entanto, como a maioria dos problemas de otimização combinatoriais pertencem à classe de problemas *NP-Hard*, a aplicação de algoritmos exatos a esses problemas demanda tempo exponencial no pior caso, tornando restrito o uso desses métodos. Por isso, aplicam-se procedimentos aproximados que, embora não garantam encontrar a solução ótima global, procuram obter soluções próximas ao ótimo a um custo computacional menor. Esses procedimentos são chamados de algoritmos de aproximação, que se subdividem em algoritmos aproximativos, que garantem resultados com proximidade percentual ao ótimo global, e algoritmos aproximados ou heurísticos, que obtêm resultados aproximados porém sem essa garantia de proximidade percentual.

Algoritmos de aproximação estão divididos basicamente entre métodos construtivos e métodos de busca local (STüTZLE, 1999). Métodos construtivos geram soluções a partir de uma solução vazia, adicionando elemento a elemento à solução parcial até obter a solução completa.

A escolha de cada elemento a ser inserido na solução varia de acordo com a função de avaliação adotada. Nas versões clássicas, os elementos candidatos são geralmente ordenados segundo uma função gulosa e o melhor candidato é escolhido para compor a solução. O Algoritmo 4 apresenta a versão geral dos métodos construtivos.

Algoritmo 4 Procedimento geral de métodos construtivos

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

Métodos de busca local partem de uma solução inicial completa, que pode ser obtida por um método construtivo ou gerada aleatoriamente, e, iterativamente, tentam 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 algoritmo 5 apresenta o procedimento geral de métodos de busca local.

Algoritmo 5 Procedimento geral de métodos de busca local

Saída: Solução s
1: Gere solução inicial s
2: **Enquanto** (ocorrer melhoria significativa) **faça**
3: Modifique solução s
4: Avalie nova solução
5: **Fim Enquanto**
6: **Retorne** s

A partir da década de 1980, intensificaram-se estudos para desenvolver métodos heurísticos mais gerais que pudessem ser facilmente adaptados ao problema de otimização específico e que possuíssem capacidade de explorar eficientemente o espaço de busca a fim de fugir de ótimos locais e encontrar soluções com qualidade superior (OSMAN; LAPORTE, 1996). Esses algoritmos passaram a ser chamados de meta-heurísticas, termo introduzido por (GLOVER, 1986).

Algumas propriedades fundamentais que caracterizam as meta-heurísticas são listadas a seguir:

- São estratégias que guiam o processo de busca.

- Têm como objetivo principal explorar eficientemente o espaço de busca, fugindo de ótimos locais.
- Possuem desde técnicas simples de busca local até processos de aprendizado complexos.
- São algoritmos aproximados e geralmente não são determinísticos.
- Podem incorporar mecanismos para evitar que a busca fique restrita a áreas confinadas do espaço de soluções.
- Possuem estratégias gerais que podem ser adaptáveis ao problema específico.

Existem várias maneiras de classificar os diferentes algoritmos que implementam meta-heurísticas. Uma das classificações mais comumente utilizadas é a que diferencia os algoritmos de acordo com o número de soluções usadas, separando-os em algoritmos de busca por população e algoritmos de busca por ponto único, segundo a nomenclatura adotada em (BLUM; ROLI, 2003; TALBI, 2002).

Algoritmos de busca por população se baseiam em um processo de busca que descreve a evolução de uma população de soluções no espaço de busca. Os indivíduos (soluções) da população são combinados para gerar uma nova população. Os algoritmos mais conhecidos nessa categoria são: Algoritmos Genéticos (*Genetic Algorithm*) (GOLDBERG, 1989), Colônia de Formigas (*Ant Colony*) (DORIGO; CARO, 1999; DORIGO; CARO; GAMBARDELLA, 1999) e Nuvem de Partículas (*Particle Swarm*) (KENNEDY; EBERHART, 1995).

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

A próxima seção apresenta em detalhes o algoritmo *Simulated Annealing*, que foi escolhido para a realização dos experimentos que avaliam a abordagem proposta no Capítulo 5. A escolha dessa meta-heurística se deve ao fato de ser um algoritmo tradicional, mostrando-se eficaz em diversos problemas de otimização, e de apresentar um procedimento conceitualmente simples, sendo capaz de competir com a simplicidade do *K-means*.

4.1 O Algoritmo *Simulated Annealing*

O algoritmo *Simulated Annealing* foi proposto independentemente por (KIRKPATRICK; GELATT; VECCHI, 1983) e (CERNY, 1985) como um método de busca iterativo baseado no algoritmo de simulação de processos moleculares apresentado em (METROPOLIS et al., 1953). O nome teve origem na analogia do algoritmo com o processo de modelagem de metal e vidro, no qual o material é aquecido a uma alta temperatura até ficar flexível o suficiente para ser modelado no formato desejado, e então resfriado lentamente até atingir uma temperatura final, permitindo que as moléculas se alinhem e entrem em um estado de baixa energia.

Nesse processo, o resfriamento deve ser feito lentamente, pois, caso contrário, o resultado consistirá em um sólido com irregularidades e defeitos. Por isso, a temperatura do material deve decrescer gradativamente e, a cada etapa, ser mantida constante por um período de tempo suficiente para que o sólido alcance o equilíbrio térmico.

A analogia do algoritmo associa o conjunto de soluções do problema de otimização combinatória com os diferentes estados térmicos do sólido. Além disso, a função objetivo corresponde à energia do sólido em um dado estado térmico e o ótimo global corresponde ao estado de menor energia.

O algoritmo 6 apresenta o procedimento geral de busca realizado pelo *Simulated Annealing*, quando se pretende minimizar a função objetivo. O algoritmo começa a partir de uma solução inicial qualquer e com uma temperatura inicial T_0. Em seguida, o procedimento executa várias iterações até atingir um critério de parada, que corresponde, na maioria das implementações, a alcançar valores muito pequenos da temperatura (próximos a zero).

O algoritmo realiza, para cada valor da temperatura, perturbações na solução corrente até que o equilíbrio térmico seja atingido. Esse equilíbrio é implementado geralmente como um número fixo de iterações N_{iter} sem ocorrência de melhoria nas soluções visitadas. Para realizar as perturbações, o algoritmo gera aleatoriamente um vizinho da solução corrente e avalia se ocorre ou não melhoria da função objetivo. Se houver melhoria, a nova solução é aceita. Caso contrário, a solução é aceita com probabilidade diretamente proporcional à temperatura, permitindo que o algoritmo escape de ótimos locais.

Uma vez que o equilíbrio térmico é atingido, a temperatura é reduzida por uma regra de resfriamento, para que o algoritmo realize novamente perturbações na solução corrente. À medida que a temperatura diminui, as piores soluções são aceitas com menos freqüência, até que o critério de parada seja satisfeito, atingindo soluções ótimas ou bem próximas a essas.

Algoritmo 6 Procedimento geral do *Simulated Annealing*

Entrada: Temperatura inicial T_0
Saída: Melhor solução encontrada s^*

```
1:  Gere solução inicial s
2:  s* ← s
3:  T ← T0
4:  Enquanto (critério de parada não for satisfeito) faça
5:     Enquanto (equilíbrio térmico não for atingido) faça
6:        Gere um vizinho s' ∈ N(s)
7:        Δ ← f(s') − f(s)
8:        Se Δ < 0 então
9:           s ← s'
10:          Se f(s) < f(s*) então
11:             s* ← s
12:          Fim Se
13:       Senão
14:          r ← variável aleatória no intervalo [0,1)
15:          Se r ≤ e^(−Δ/T) então
16:             s ← s'
17:          Fim Se
18:       Fim Se
19:    Fim Enquanto
20:    Diminua o valor de T
21: Fim Enquanto
22: Retorne s*
```

Resultados teóricos (AARTS; LENSTRA, 1997) mostram que o algoritmo pode convergir para um ótimo global, quando o número de iterações tende a infinito e com o uso de regras de resfriamento que atendam determinadas condições. No entanto, essas regras não são comumente aplicadas, pois convergem lentamente para o ótimo global. Por isso, regras que levam a uma convergência mais rápida são adotadas para propósitos práticos. Uma das mais conhecidas é a que segue uma forma geométrica, na qual a temperatura decresce exponencialmente por uma razão de resfriamento α, tal que $T = \alpha \cdot T$, com $0 < \alpha < 1$.

A aplicação do algoritmo *Simulated Annealing* a um problema de otimização específico requer uma etapa de avaliação dos valores mais adequados para os parâmetros e uma definição para o processo de perturbação de uma solução. Os parâmetros do algoritmo a serem ajustados são a temperatura inicial (T_0), a temperatura final (T_f), a razão de resfriamento (α) e o número de iterações para atingir o equilíbrio térmico (N_{iter}). Tanto a adaptação do algoritmo para o problema de agrupamento quanto a escolha dos melhores parâmetros e da função de movimento ou perturbação para esse problema são detalhados no Capítulo 5.

4.2 Meta-heurísticas Aplicadas ao Problema de Agrupamento

Conforme discutido no Capítulo 2, o problema de agrupamento pode ser visto como uma tarefa de otimização, definindo-se uma função de custo que avalie a qualidade das partições e escolhendo-se a combinação de partições com o valor ótimo para essa função. Essa formulação permite a aplicação de meta-heurísticas como um possível método para resolver o problema.

Um dos primeiros trabalhos a empregar meta-heurísticas ao problema de particionamento 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. Na abordagem proposta, a solução é representada como um cromossomo (termo utilizado na área da computação evolutiva), 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 $\mathbf{x_i}$ do conjunto de dados pertence à partição j.

Outros trabalhos posteriores a esse 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, porém 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). Uma discussão maior sobre o uso de algoritmos genéticos ao problema de agrupamento é apresentada em (FALKENAUER, 1998).

Além dessas abordagens baseadas em algoritmos genéticos, alguns trabalhos propõem o uso do *Simulated Annealing* para resolver o problema de agrupamento como uma tarefa de otimização. (KLEIN; DUBES, 1989) apresentam um algoritmo adaptado para o problema, utilizando uma matriz para representar a solução do agrupamento, sendo que a posição a_{ij} da matriz é igual a um se o elemento $\mathbf{x_i}$ pertencer à partição j e zero, caso contrário.

O trabalho proposto em (SELIM; ALSULTAN, 1991) também utiliza uma representação similar a essa. Nesse trabalho, os autores descrevem uma adaptação do algoritmo *Simulated Annealing* para o problema de agrupamento e apresentam uma avaliação e interpretação dos parâmetros do algoritmo. Os trabalhos feitos com o *Simulated Annealing* por (KLEIN; DUBES,

1989) e por (SELIM; ALSULTAN, 1991) também obtiveram bons resultados com bases simples.

Alguns outros trabalhos se dedicam à avaliação do uso dos algoritmos *Tabu Search* (ALSULTAN, 1995), *Particle Swarm* (MERWE; ENGELBRECHT, 2003) e *Ant Colony Optimization* (ACO) (KANADE; HALL, 2004).

No entanto, conforme apresentado em (RAYWARD-SMITH, 2005), a aplicação direta de meta-heurísticas 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 nesses casos.

Uma estratégia mais promissora para o problema está voltada para abordagens híbridas. Um exemplo desse tipo de abordagem é a combinação de técnicas de busca usadas tanto em heurísticas quanto em meta-heurísticas, como ocorre na utilização de heurísticas de busca local em algoritmos de busca populacional. Essa combinação pode apresentar bons resultados, pois algoritmos baseados em população são melhores em identificar áreas mais promissoras no espaço de soluções, enquanto que procedimentos de busca local são melhores em explorar essas áreas.

Um exemplo dessa abordagem é apresentado em (MAULIK; BANDYOPADHYAY, 2000). Esse trabalho 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 de soluções e a nova geração não é realizada apenas por operadores genéticos, tais como seleção, mutação e recombinação (*crossover*). 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, e recomputá-los como o centro de massa da partição. 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 meta-heurísticas. Um exemplo dessa abordagem é o uso de algoritmos evolutivos para encontrar regiões promissoras no vasto e

complexo espaço de busca. Após essas regiões serem localizadas, pode ser interessante aplicar algoritmos de busca local para explorar essas regiões e 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. Uma taxonomia mais formal de meta-heurísticas híbridas é apresentada em (TALBI, 2002).

(BABU; MURTY, 1993) propõem uma abordagem híbrida de encadeamento que aplica algoritmo genético com o objetivo de direcionar a busca pelo espaço de soluções seguido do *K-means* que encontra a melhor solução local. O resultado experimental mostra que essa abordagem híbrida obtém melhores resultados do que a aplicação direta do algoritmo genético. (BABU; MURTY, 1994) apresentam uma abordagem semelhante, aplicando *Simulated Annealing* para escolher a solução inicial do algoritmo *K-means*.

De maneira similar, (MERWE; ENGELBRECHT, 2003) mostram que o desempenho do algoritmo *Particle Swarm* pode ser aprimorado com uma inicialização obtida a partir do resultado do *K-means*. A abordagem foi avaliada com a aplicação do algoritmo em bases reais com até 500 padrões e 11 atributos.

O próximo capítulo apresenta a abordagem sugerida por este trabalho que está motivada na necessidade de obter resultados melhores em problemas de agrupamento, principalmente em instâncias maiores.

5 *Escolha Inicial de Soluções em Metaheurísticas Voltadas para o Problema de Agrupamento*

A discussão realizada neste trabalho mostra que o problema de agrupamento de dados pode ser visto como uma tarefa de otimização, na qual busca-se a melhor configuração de partições dentre todas as configurações possíveis de acordo com um critério de avaliação (função de custo). Algoritmos exatos realizam a enumeração de todas as soluções possíveis para encontrar a que obtém o melhor custo. No entanto, a avaliação de todas as combinações de agrupamento é uma tarefa computacionalmente inviável. Por isso, algoritmos aproximados são os mais adequados para resolver o problema, pois, embora não garantam a otimalidade da solução, eles encontram soluções próximas ao ótimo e demandam custo computacional menor.

Um dos algoritmos de aproximação mais usados para resolver o problema de agrupamento é o *K-means*, um procedimento de busca simples e rápido que utiliza estratégias gulosas para minimizar o critério *SSE* de avaliação dos agrupamentos. Uma fragilidade desse algoritmo é que o resultado obtido é sensível à escolha inicial da solução. A versão clássica do algoritmo utiliza como método de inicialização a escolha não criteriosa de K centróides dentre padrões aleatórios do conjunto de dados. Se essa escolha for ruim, então o algoritmo tende a encontrar um ótimo local como solução. Além disso, a escolha aleatória das soluções iniciais não garante a reprodução dos resultados em aplicações sucessivas.

Para resolver o problema da escolha de soluções ruins, podem ser usados métodos criteriosos que direcionem a inicialização para uma solução próxima do ótimo global. Em relação à aleatoriedade da escolha, métodos determinísticos de inicialização podem ser utilizados para que o procedimento de escolha seja sempre o mesmo, sem depender de um fator randômico.

Contudo, mesmo com essas alternativas, ainda existe o problema do *K-means* de não conseguir escapar de ótimo local, pois se a inicialização direcionada levar a uma solução próxima a um ótimo local, o resultado do algoritmo pode ser uma solução subótima.

Uma abordagem alternativa para resolver o problema de agrupamento consiste em usar meta-heurísticas, que são procedimentos heurísticos com caráter mais geral que combinam heurísticas simples e regras para encontrar soluções aproximadas melhores, além de oferecer mecanismos para escapar de ótimos locais. Algumas dessas meta-heurísticas, assim como o *K-means*, também dependem da escolha de uma solução inicial que funciona como ponto de partida para o processo de busca.

No entanto, há uma indicação de que a aplicação das meta-heurísticas para o problema de agrupamento obtém resultados melhores apenas em bases pequenas, já que o espaço de busca cresce consideravelmente quando as dimensões do problema aumentam. Porém, abordagens híbridas podem indicar uma estratégia promissora para o problema.

Baseado na tendência das abordagens híbridas em obter bons resultados para o problema de agrupamento, este trabalho investiga uma nova alternativa que utiliza métodos de inicialização definidos para o *K-means* para encontrar a solução inicial de meta-heurísticas, a fim de obter resultados melhores do que as abordagens citadas acima, principalmente em bases maiores.

O objetivo de selecionar soluções iniciais melhores para a meta-heurística é permitir que o algoritmo comece de uma solução no espaço de busca mais próxima ao ótimo global, aumentando as possibilidades de esse ótimo ser alcançado em um número de iterações menor do que o utilizado pelo algoritmo sem inicialização.

Dentre os métodos de inicialização do *K-means* estudados, optou-se por utilizar neste trabalho o método *PCA_Part* e o método de inicialização do *K-means++*. E a meta-heurística implementada neste trabalho é uma adaptação do procedimento geral do *Simulated Annealing* para o problema de agrupamento de dados.

Por fim, optou-se por avaliar o *Simulated Annealing* em relação ao algoritmo mais utilizado para o problema de agrupamento, o *K-means*. Nessa avaliação, os dois algoritmos foram inicializados tanto com o método aleatório clássico, como com os dois métodos de inicialização escolhidos. As próximas seções deste capítulo discutem em detalhes o algoritmo *Simulated Annealing* implementado.

5.1 Adaptação do *Simulated Annealing* para o Problema de Agrupamento

A representação da solução do problema de agrupamento adotada neste trabalho baseia-se nas propostas apresentadas em (RAGHAVAN; BIRCHARD, 1979) e (MURTHY; CHOWD-

HURY, 1996). A solução é representada como um vetor de tamanho N, onde cada posição i do vetor representa um padrão do conjunto de dados e possui valores no intervalo $\{1,...,K\}$, indicando de qual partição o padrão $\mathbf{x_i}$ faz parte.

Por exemplo, se o resultado do agrupamento para um problema com $N = 7$ padrões e $K = 4$ partições for igual a $C_1 = \{\mathbf{x_1}, \mathbf{x_4}\}$, $C_2 = \{\mathbf{x_3}, \mathbf{x_5}\}$, $C_3 = \{\mathbf{x_6}, \mathbf{x_7}\}$ e $C_4 = \{\mathbf{x_2}\}$, então a solução é representada pelo vetor

$$\begin{pmatrix} 1 & 4 & 2 & 1 & 2 & 3 & 3 \end{pmatrix}$$

Além da representação da solução apresentada, a abordagem proposta neste trabalho utiliza como função de avaliação o critério *SSE* (*Sum of the Squared Euclidian distances*), o mesmo critério usado no algoritmo do *K-means*. Essa escolha pretende tornar coerente a comparação dos resultados obtidos com o algoritmo baseado no *Simulated Annealing* e com o *K-means*.

Um outro fator que influencia no resultado do algoritmo *Simulated Annealing* 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 segue a proposta apresentada em (KLEIN; DUBES, 1989), que sugere escolher 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. Com essa função de vizinhança, a nova solução consegue ser bem próxima à soluçao corrente, como sugerem (SELIM; ALSULTAN, 1991).

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

Além disso, a abordagem proposta define uma maneira eficiente de recalcular os centróides das partições a cada iteração do algoritmo. Ao mover um padrão $\mathbf{x_i}$ de uma partição C_j para uma partição $C_{j'}$, apenas os centróides dessas partições precisam ser recomputados. O centróide da partição C_j, ou seja, da partição à qual o padrão $\mathbf{x_i}$ pertencia antes de ser movido, é recalculado como

$$\mu_\mathbf{j} = \frac{1}{n_j - 1}(n_j \cdot \mu_\mathbf{j} - \mathbf{x_i}), \tag{5.1}$$

onde n_j é o número de padrões da partição j antes de mover o elemento $\mathbf{x_i}$.

Já o centróide da partição $C_{j'}$ que corresponde à nova partição do padrão $\mathbf{x_i}$, é recalculado

como

$$\mu_{\mathbf{j'}} = \frac{1}{n_{j'}+1}(n_{j'} \cdot \mu_{\mathbf{j'}} + \mathbf{x_i}), \tag{5.2}$$

onde $n_{j'}$ é o número de padrões da partição j' antes de mover o elemento $\mathbf{x_i}$.

De maneira similar, não é necessário realizar o cálculo completo do critério de avaliação para verificar se uma solução vizinha é melhor do que a solução corrente. Quando um padrão $\mathbf{x_i}$ é movido de uma partição C_j para uma partição $C_{j'}$, é necessário computar apenas a diferença do critério de avaliação.

Supondo que $\mu_{\mathbf{j}(\mathbf{bef})}$ e $\mu_{\mathbf{j'}(\mathbf{bef})}$ representem os centróides das partições C_j e $C_{j'}$ antes de serem recomputados, e $\mu_{\mathbf{j}(\mathbf{aft})}$ e $\mu_{\mathbf{j'}(\mathbf{aft})}$ representem os centróides das partições C_j e $C_{j'}$ depois de serem recomputados, a diferença entre os critérios de avaliação ($\Delta SSE = SSE_{new} - SSE_{old}$) é realizado da seguinte maneira:

$$\Delta SSE = \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mu_{\mathbf{j}(\mathbf{aft})}\|^2 + \sum_{\mathbf{x'} \in C_{j'}} \|\mathbf{x'} - \mu_{\mathbf{j'}(\mathbf{aft})}\|^2 - \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mu_{\mathbf{j}(\mathbf{bef})}\|^2 - \sum_{\mathbf{x'} \in C_{j'}} \|\mathbf{x'} - \mu_{\mathbf{j'}(\mathbf{bef})}\|^2 \tag{5.3}$$

Se o valor de ΔSSE for negativo, então a solução vizinha possui um critério de avaliação menor do que o da solução corrente e, portanto, deve ser escolhida para a próxima iteração. Senão, a solução vizinha possui um critério de avaliação maior ou igual ao da solução corrente e só será aceita com probabilidade diretamente proporcional à temperatura.

Por fim, o valor do critério de avaliação para a nova solução é calculada como $SSE_{new} = SSE_{old} + \Delta SSE$.

O algoritmo 7 apresenta o procedimento *Simulated Annealing* adaptado para o problema de agrupamento proposto neste trabalho, seguindo as características mencionadas nesta seção. Essa versão está baseada nos trabalhos propostos por (KLEIN; DUBES, 1989; SELIM; AL-SULTAN, 1991).

5.2 Análise dos Parâmetros

O algoritmo apresentado na seção anterior faz uso de parâmetros que precisam ser ajustados para guiar com eficiência e eficácia o processo de busca pela solução ótima. Os parâmetros responsáveis por esse comportamento do algoritmo são a temperatura inicial (T_0), a temperatura final (T_f), a razão de resfriamento (α) e o número de iterações para atingir o equilíbrio térmico (N_{iter}).

Algoritmo 7 Procedimento *Simulated Annealing* adaptado ao problema de agrupamento

Entrada: X, conjunto de N padrões com d dimensões;
K, número de centróides;
T_0, temperatura inicial;
T_f, temperatura final;
α, razão de resfriamento;
N_{iter}, número de iterações para atingir o equilíbrio térmico;
s, solução inicial

Saída: Melhor solução encontrada s^*

1: $s^* \leftarrow s$
2: $SSE^* \leftarrow SSE$ calculado com a solução s
3: $T \leftarrow T_0$
4: **Enquanto** $(T > T_f)$ **faça**
5: $iter \leftarrow 1$
6: **Enquanto** $(iter \leq N_{iter})$ **faça**
7: $\mathbf{x_i} \leftarrow$ padrão escolhido aleatoriamente no conjunto de dados X
8: $j \leftarrow s[i]$ // *partição atual de* $\mathbf{x_i}$
9: $j' \leftarrow$ partição escolhida aleatoriamente no intervalo $[1,K]$ e $j' \neq j$
10: Gere o vizinho s' movendo o padrão $\mathbf{x_i}$ para a partição j'
11: $\mu_{\mathbf{j(bef)}} \leftarrow$ centróide corrente da partição j
12: $\mu_{\mathbf{j'(bef)}} \leftarrow$ centróide corrente da partição j'
13: $\mu_{\mathbf{j(aft)}} \leftarrow$ centróide recalculado pela Equação 5.1
14: $\mu_{\mathbf{j'(aft)}} \leftarrow$ centróide recalculado pela Equação 5.2
15: $\Delta SSE \leftarrow$ diferença calculada pela Equação 5.3
16: **Se** $\Delta SSE < 0$ **então**
17: $s \leftarrow s'$
18: $SSE \leftarrow SSE + \Delta SSE$
19: **Se** $SSE < SSE^*$ **então**
20: $s^* \leftarrow s$
21: $SSE^* \leftarrow SSE$
22: $iter = 0$
23: **Fim Se**
24: **Senão**
25: $r \leftarrow$ variável aleatória no intervalo $[0,1)$
26: **Se** $r \leq e^{\frac{-\Delta}{T}}$ **então**
27: $s \leftarrow s'$
28: $SSE \leftarrow SSE + \Delta SSE$
29: **Fim Se**
30: **Fim Se**
31: $iter \leftarrow iter + 1$
32: **Fim Enquanto**
33: $T = \alpha \cdot T$
34: **Fim Enquanto**
35: **Retorne** s^*

Conforme discutido no Capítulo 4, a temperatura é uma variável que controla, ao longo das iterações, se soluções ruins (com função objetivo maior do que a solução corrente) são

aceitas ou não. Se uma solução não promove melhoria na função de custo, então ela é aceita com probabilidade diretamente proporcional à temperatura. Dessa forma, se a temperatura for grande, a probabilidade de aceitação será maior.

Inicialmente, deseja-se aceitar soluções ruins com mais freqüência a fim de explorar um número maior de soluções. Logo, a temperatura inicial precisa ser grande o suficiente para garantir uma busca mais ampla, mas sem aumentar o número de iterações a ponto de inviabilizar o uso do algoritmo. Segundo (SELIM; ALSULTAN, 1991), quando o problema de agrupamento se torna mais difícil, isto é, quando N e K aumentam, em geral, a temperatura inicial deve ser maior. (SELIM; ALSULTAN, 1991) também mostram que a temperatura inicial depende da magnitude da função objetivo do problema. Quanto maior a magnitude da função de custo, maior deve ser a temperatura inicial.

Nas iterações finais do algoritmo, a temperatura deve alcançar valores bem pequenos para garantir que poucas soluções ruins sejam aceitas e que a convergência atinja soluções próximas ao ótimo global. Por isso, a temperatura final deve ser próxima a zero para que nas últimas iterações o algoritmo aceite quase que apenas soluções com melhoria da função objetivo.

Quando as soluções visitadas não apresentam melhoria no critério de avaliação por um certo número de iterações N_{iter}, significa que o algoritmo alcançou o equilíbrio. A temperatura deve, portanto, ser decrementada por uma razão α, tal que $0 < \alpha < 1$. À medida que a dificuldade do problema aumenta, a razão deve se aproximar de 1 (SELIM; ALSULTAN, 1991). No entanto, quanto maior o valor de α, mais iterações são necessárias para o algoritmo resolver o problema.

Por fim, no algoritmo apresentado, perturbações nas soluções são realizadas até que não haja mais melhoria no critério de avaliação por um certo número de iterações N_{iter}. Na analogia com o processo de resfriamento de sólidos, esse parâmetro corresponde ao tempo necessário para o sólido alcançar o equilíbrio térmico. Em geral, melhores resultados são obtidos se mais iterações são permitidas. Por outro lado, maior é o tempo de convergência do algoritmo. Por isso, a escolha desse parâmetro deve ser definida de acordo com o tamanho do problema, ou seja, dependendo do número de padrões N, de atributos d e de partições K.

6 *Experimentos Realizados*

Este capítulo apresenta a avaliação experimental do algoritmo *Simulated Annealing* adaptado para o problema de agrupamento e do algoritmo *K-means*, ambos combinados com diferentes métodos de inicialização. O método de inicialização mais simples e usual utilizado foi o aleatório, no qual os K centróides são escolhidos randomicamente entre os elementos do conjunto de dados original. Os outros dois métodos empregados neste trabalho foram o *PCA_PART* e o *K-means++*, apresentados no Capítulo 3.

A Tabela 6.1 mostra as características das oito bases reais utilizadas na avaliação experimental, todas disponíveis em repositórios públicos (ASUNCION; NEWMAN, 2007).

Tabela 6.1: Características das bases usadas na avaliação experimental.

Nome da base	Núm. de pontos	Núm. de atributos	Núm. 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
Pen digits	10992	16	10
Letter	20.000	16	26

Dentre as bases utilizadas, algumas apresentam menor complexidade (*Iris*, *Wine* e *Vehicle*) e as demais possuem complexidade maior. Essa diversidade permite avaliar em quais tipos de problema a aplicação de métodos de inicialização combinados a meta-heurísticas é mais favorável e em quais não é.

O critério de parada do algoritmo *K-means* usado neste trabalho inclui duas condições: número máximo de iterações atingido (igual a 200 nos testes realizados) ou estabilidade da solução entre duas iterações consecutivas. Na maioria dos casos, a segunda condição foi satisfeita antes da primeira.

A Seção 6.1 descreve o procedimento utilizado para ajustar os parâmetros do *Simulated*

Annealing nos experimentos realizados. A Seção 6.2 apresenta os resultados obtidos com os diferentes algoritmos aplicados às bases escolhidas.

6.1 Ajuste dos Parâmetros e Estimativa do Critério de Avaliação

Este trabalho propõe um procedimento baseado em validação cruzada para realizar a busca pelos melhores parâmetros do algoritmo proposto e para calcular a estimativa do critério de avaliação. Validação cruzada é o método mais empregado para avaliar classificadores, quando o tamanho da amostra de exemplos é limitado (DIETTERICH, 1998).

O procedimento proposto divide aleatoriamente o conjunto de exemplos em q conjuntos disjuntos e realiza, em q repetições, a avaliação do resultado do algoritmo aplicado a um dos conjuntos q, usando os melhores parâmetros encontrados com os demais $q-1$ conjuntos de exemplos. A estimativa do critério de avaliação é calculada como a média das q rodadas do procedimento. A divisão dos padrões em conjuntos disjuntos garante que o processo de avaliação dos parâmetros não use os mesmos exemplos da etapa de aferição do critério de avaliação. Se os padrões usados na busca dos melhores parâmetros forem os mesmos usados na avaliação do resultado, o procedimento obtém um valor subestimado do critério de avaliação.

A escolha dos melhores parâmetros é feita normalmente com o uso de heurísticas, já que o espaço de valores possíveis é extenso e a avaliação do algoritmo para cada combinação de parâmetros é uma tarefa árdua e demorada. Além disso, como o procedimento baseado em validação cruzada requer ainda mais esforços computacionais, a heurística precisa ser bastante eficiente ou o procedimento de estimativa se torna uma tarefa computacionalmente inviável.

Por isso, apenas alguns valores discretos foram selecionados para encontrar a estimativa dos melhores parâmetros. Como o algoritmo proposto é heurístico, uma mudança pequena nos valores dos parâmetros geralmente não fornece uma diferença significativa na solução final (SELIM; ALSULTAN, 1991). Os valores escolhidos para os parâmetros na etapa de ajuste foram:
$T_0 \in \{500; 5500; 10500; 15500; 20500\}$,
$T_f \in \{0.1\}$,
$\alpha \in \{0.85; 0.90; 0.95\}$ e
$N_{iter} \in \{1000; 2000; 3000\}$.

O Algoritmo 8 exibe o procedimento de ajuste dos parâmetros e de estimativa do critério de avaliação adotado neste trabalho. Esse algoritmo segue a estrutura geral do método de avaliação

sugerido por (SALZBERG, 1997).

Algoritmo 8 Estimativa do critério de avaliação com ajuste de parâmetros.

Entrada: X, conjunto com N padrões e d dimensões;
$P = \{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_{Npar}}\}\}$, vetores de valores a explorar de cada parâmetro.
Saída: SSE_{med}, critério de avaliação médio nas q execuções da validação cruzada.

1: Faça $Y \leftarrow \{Y_1, Y_2, ..., Y_q\}$ *// Subconjuntos disjuntos e aleatórios de X para validação cruzada*
2: **Para todo** $Y_i \in Y$ **faça**
3: $T_i \leftarrow Y - \{Y_i\}$ *// Conjunto para ajuste dos parâmetros da rodada*
4: Faça $R \leftarrow \{R_1, R_2, ..., R_m\}$ *// Subconjuntos disjuntos e aleatórios de* T_i
5: Determine $\mathbf{p_b}$, a configuração que minimiza o critério de avaliação médio aferido por validação cruzada em R
6: Atualize SSE_{med} usando $\mathbf{p_b}$ para determinar o critério de avaliação em Y_i
7: **Fim Para**
8: **Retorne** SSE_{med}

O algoritmo recebe como entrada o conjunto de dados X e o conjunto de valores discretos $\mathbf{p_z}$ de cada parâmetro, sendo $z = 1, \ldots, Npar$, onde $Npar$ é o número de parâmetros (no caso do *Simulated Annealing*, $Npar = 4$, pois o algoritmo possui 4 parâmetros: T_0, T_f, α e N_{iter}).

Com isso, o algoritmo de validação cruzada realiza, em cada iteração, todas as $|\mathbf{p_1}| \times |\mathbf{p_2}| \ldots \times |\mathbf{p_{Npar}}|$ combinações possíveis desses valores no conjunto T_i (conjunto de treino usado para ajuste dos parâmetros) e encontra a melhor combinação, dividindo T_i em m subconjuntos, nos quais a validação cruzada é aplicada, e a combinação dos parâmetros $\mathbf{p_b}$ que obtém o menor critério de avaliação é escolhida. A melhor configuração $\mathbf{p_b}$ é, então, usada no conjunto de dados Y_i (um dos q subconjuntos de X usado para teste) para calcular o critério de avaliação obtido com o *Simulated Annealing*.

A mesma configuração $\mathbf{p_b}$ e o mesmo conjunto Y_i são empregados com as diferentes combinações de método de inicialização e o algoritmo *Simulated Annealing* adaptado. Os resultados com o *K-means* também são obtidos com os mesmos conjuntos Y_i usados no *Simulated Annealing*.

Em todos os experimentos, o procedimento de validação cruzada foi executado com $q = 10$ e $m = 3$. A escolha de um número pequeno de divisões se deve às restrições computacionais impostas pelo ajuste exaustivo adotado.

6.2 Resultados

A Tabela 6.2 apresenta os resultados obtidos com a aplicação do *K-means* e do *Simulated Annealing* às bases citadas na Tabela 6.1. Os resultados foram obtidos com a combinação dos

algoritmos com os três métodos de inicialização adotados. Para cada base foram realizados experimentos com três valores diferentes de *K*, sendo um desses valores igual ao número de classes existentes nos dados. Os melhores resultados estão destacados em negrito.

É importante destacar que a variação do valor de *K* influencia no valor do critério de avaliação do agrupamento. Espera-se que o valor do critério decresça naturalmente com o aumento de *K*. Pode-se constatar que quando o valor de *K* é igual ao número de elementos do conjunto de dados ($K = N$), cada elemento representa um centróide, tornando o critério de avaliaçao igual a zero.

Tabela 6.2: SSE médio obtido com a combinação dos algoritmos de inicialização com *K-means* e *Simulated Annealing* (SA).

		K-means			*S.A.*		
Base	K	*Aleat.*	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Iris	2	13.9491	13.9491	**13.8699**	**13.8699**	**13.8699**	**13.8699**
	3	9.96375	6.7914	6.75412	**5.98379**	**5.98379**	**5.98379**
	4	5.00787	4.23722	3.87814	**3.46131**	**3.46131**	**3.46131**
Wine	2	369267	364926	368955	**362683**	**362683**	**362683**
	3	159229	154158	155967	151839	**145690**	150402
	4	93706.8	86061.1	107483	**76841.1**	79477.1	83833.8
Veh.	3	**463986**	471304	481356	482624	466703	480391
	4	324726	290645	318651	286907	**283006**	294798
	5	242253	221410	234724	222688	**216325**	220228
Cloud	9	834627	504282	569423	678689	**477777**	525079
	10	669664	421224	448449	451459	**392694**	397056
	11	659576	377512	388657	393348	**345548**	347743
Segm.	6	1.44e+06	1.25e+06	1.246e+06	1.36e+06	**1.17e+06**	1.22e+06
	7	1.17e+06	1.11e+06	1.08e+06	1.14e+06	1.07e+06	**1.05e+06**
	8	1.12e+06	976326	969249	943673	**921804**	937324
Spam	2	9.00e+07	9.00e+07	8.32e+07	8.07e+07	8.07e+07	**7.07e+07**
	3	5.39e+07	5.39e+07	3.0754e+07	4.17e+07	4.17e+07	**3.0753e+07**
	4	2.59e+07	2.52e+07	2.04e+07	2.40e+07	2.34e+07	**2.01e+07**
Pend.	9	5.49e+06	5.32e+06	5.38e+06	5.35e+06	**5.28e+06**	5.33e+06
	10	5.03e+06	5.02e+06	4.988e+06	**4.95e+06**	5.04e+06	4.99e+06
	11	4.91e+06	4.73e+06	4.82e+06	4.70e+06	**4.69e+06**	4.75e+06
Letter	25	62001	**61791.9**	62543.7	65033.3	63663	63532.4
	26	61347.2	**60979.2**	61519.8	63274.1	64721.6	67876.8
	27	60509.6	**60053**	60386.5	62428.4	62173.2	61355

Uma análise menos rigorosa dos resultados apresentados na Tabela 6.2 permite constatar algumas indicações das combinações avaliadas, embora não sejam comprovações estatísticas.

A primeira delas é que o *K-means* inicializado com o método *PCA_Part* obteve melhores resultados do que o *K-means++* em 14 dos 24 testes, levantando a indicação de que o método

PCA_Part pode ser tão bom quanto o método de inicialização do *K-means++* que garante alcançar soluções próximas ao ótimo global por uma constante na ordem de $O(log(K))$.

Outra constatação é que o melhor resultado com o *Simulated Annealing* foi igual ou superior ao melhor resultado com o *K-means* em 20 dos 24 testes realizados, sendo que na base *Letter* a meta-heurística obteve em todos os testes desempenho inferior ao do *K-means*. Se essa base fosse desconsiderada, o *Simulated Annealing* seria melhor em 20 dos 21 testes. Essa constatação mostra uma indicação de que o algoritmo do *Simulated Annealing* implementado é superior ao *K-means*.

Observa-se também que o *Simulated Annealing* inicializado com *PCA_Part* obteve o melhor resultado em 14 dos 24 testes, indicando que essa é a melhor combinação dentre as outras avaliadas.

Além disso, as combinações do *Simulated Annealing* tanto com o método *PCA_Part* como com o método do *K-means++* obtiveram resultado melhor ou igual ao do *K-means* com método *K-means++* em 20 dos 24 testes realizados. Isso indica que a meta-heurística implementada, quando inicializada com esses métodos, pode chegar mais perto ao ótimo global.

Pode-se constatar também que, em 17 dos 24 experimentos, o *Simulated Annealing* inicializado com *PCA_Part* foi melhor ou igual à mesma meta-heurística combinada ao método de inicialização do *K-means++*, reforçando a indicação de que a primeira combinação é melhor do que a segunda.

Embora essas observações indiquem uma melhoria obtida com a abordagem proposta neste trabalho, na maior base (*Letter*), o desempenho do *Simulated Annealing* foi sistematicamente inferior ao do *K-means*, reforçando a indicação de que meta-heurísticas apresentam um desempenho degradado em bases muito grandes.

6.2.1 Avaliação Experimental com Métodos Estatísticos em Múltiplos Domínios

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

Em métodos de múltiplas comparações, espera-se mostrar que existem diferenças entre os algoritmos com significância de α%. O nível de significância indica a probabilidade de uma amostra de dados aleatória gerar o resultado atual, supondo que os algoritmos obtêm resultados

iguais (hipótese nula). Quando a amostra aleatória gera o resultado com probabilidade menor do que o nível de significância desejado (geralmente é usado $\alpha = 5\%$), então a hipótese nula é rejeitada. O nível de significância pode ser interpretado como a probabilidade de rejeitar a hipótese nula quando ela é verdadeira, ou seja, a probabilidade de cometer um erro do Tipo I. No caso de múltiplas comparações, o nível de significância em cada comparação deve ser ajustado para garantir baixa probabilidade de ocorrência de erro do Tipo I.

O método estatístico mais adequado para comparação de algoritmos em múltiplos domínios, segundo (DEMSAR, 2006), é o *Teste de Friedman* (FRIEDMAN, 1937, 1940). Esse método testa se em $c \geq 2$ experimentos diferentes e dependentes, pelo menos dois são estatisticamente diferentes. Para isso, o método ordena os algoritmos para cada problema separadamente, atribuindo um posto a cada um deles com valores de 1 a c. O melhor algoritmo recebe o posto 1, o segundo melhor recebe o posto 2, e assim por diante. Os algoritmos que empatarem recebem a média dos postos que seriam atribuídos a eles.

O *Teste de Friedman* compara a média dos postos, R_j, $j = 1, \ldots, c$, dos c algoritmos em todos os n problemas avaliados. Sob a hipótese nula de que todos os algoritmos são iguais e, portanto, todas as médias dos postos são iguais, a estatística de *Friedman*

$$\chi_r^2 = \frac{12n}{c(c+1)} \left[\sum_{j=1}^{c} R_j^2 - \frac{c(c+1)^2}{4} \right] \tag{6.1}$$

segue uma distribuição χ^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:

$$F_r = \frac{(n-1)\chi_r^2}{n(c-1) - \chi_r^2}, \tag{6.2}$$

onde F_r segue uma distribuição F com $(c-1)$ e $(c-1)(n-1)$ graus de liberdade.

A hipótese nula de que os algoritmos são iguais é rejeitada se o valor obtido por F_r indicar 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 existe um algoritmo como referência com o qual os demais serão comparados. Com esse 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

$$CD = q_\alpha \sqrt{\frac{c(c+1)}{6n}}, \tag{6.3}$$

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

A análise estatística dos resultados é apresentada a seguir. Nessa 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 os mesmos subconjuntos obtidos pela divisão da validação cruzada, então os experimentos são considerados dependentes, satisfazendo as restrições exigidas para o *Teste de Friedman*.

A Tabela 6.3 mostra os valores médios do critério de avaliação, encontrados com a validação cruzada, de cada combinação de algoritmos. O posto atribuído pelo *Teste de Friedman* é apresentado entre parênteses e os postos médios na última linha da tabela.

Tabela 6.3: Comparações entre os algoritmos de agrupamento *K-means* e *Simulated Annealing* combinados com diferentes métodos de inicialização, usando diferentes bases.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Iris	9.96375 (6)	6.7914 (5)	6.75412 (4)	5.98379 (2)	5.98379 (2)	5.98379 (2)
Wine	159229 (6)	154158 (4)	155967 (5)	151839 (3)	145690 (1)	150402 (2)
Veh.	324726 (6)	290645 (3)	318651 (5)	286907 (2)	283006 (1)	294798 (4)
Cloud	669664 (6)	421224 (3)	448449 (4)	451459 (5)	392694 (1)	397056 (2)
Segm.	1.17e+06 (6)	1.11e+06 (4)	1.08e+06 (3)	1.14e+06 (5)	1.07e+06 (2)	1.05e+06 (1)
Spam	9.0e+07 (5.5)	9.0e+07 (5.5)	8.3e+07 (4)	8.1e+07 (2.5)	8.1e+07 (2.5)	7.1e+07 (1)
Pend.	5.03e+06 (5)	5.02e+06 (4)	4.988e+06 (2)	4.95e+06 (1)	5.04e+06 (6)	4.99e+06 (3)
Letter	61347.2 (2)	60979.2 (1)	61519.8 (3)	63274.1 (4)	64721.6 (5)	67876.8 (6)
p.m.	5.3125	3.6875	3.8125	3.0625	2.5625	2.5625

A primeira etapa da comparação múltipla dos algoritmos consiste em verificar a hipótese nula de que todos os algoritmos são estatisticamente iguais com base nos resultados da tabela. O *Teste de Friedman* reporta para esses resultados uma probabilidade de 2.04%, supondo que a hipótese nula é verdadeira. Essa hipótese pode, portanto, ser rejeitada com nível de significância de 5%. O resultado do teste indica que existe pelo menos um par de algoritmos estatisticamente

diferentes.

A segunda etapa dessa análise identifica quais os pares de algoritmos que possuem uma diferença significativa. Considerando o nível de significância de 5% e o *Teste de Nemenyi*, dois algoritmos são ditos diferentes, se a diferença entre as respectivas médias dos postos forem no mínimo igual a $CD = 2.85\sqrt{\frac{6 \cdot 7}{6 \cdot 8}} = 2.67$. Assim, o teste reporta que o algoritmo *Simulated Annealing* inicializado com o método *PCA_Part* e com o método de inicialização do *K-means++* é considerado significativamente melhor do que o algoritmo *K-means* inicializado aleatoriamente ($5.3125 - 2.5625 = 2.75 > 2.67$). Em relação aos demais pares de algoritmos, nada se pode concluir (nem que são iguais e nem que são diferentes), porque as diferenças são menores do que a diferença crítica *CD*.

Um fator importante a ser observado nesse método de comparação é que 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. Por isso, o aumento da eficácia nesses resultados estatísticos pode ser feito, obtendo um número maior de problemas. No entanto, existem poucas bases disponíveis com as características adequadas para os algoritmos utilizados neste trabalho. Para os algoritmos adotados, as bases precisam estar completas (sem valores nulos) e possuir todos os atributos numéricos. Além disso, algumas delas demandam mais tempo de execução e, por restrições diversas, não puderam ser usadas nessa avaliação experimental.

6.2.2 Avaliação Experimental com Métodos Estatísticos em cada Problema

Realizou-se ainda uma avaliação do comportamento dos algoritmos em cada problema a fim de identificar alguma tendência de melhoria significativa em instâncias maiores com o uso de meta-heurísticas combinadas a métodos de inicialização. Assim, para cada base e para cada valor de *K* avaliado, foram utilizados os resultados individuais das rodadas da validação cruzada para formar a amostra dos experimentos. Como os valores de *SSE* são obtidos a partir de subconjuntos disjuntos dos dados, então as amostras são independentes entre si. Além disso, como os algoritmos são rodados sobre os mesmos subconjuntos de padrões obtidos pela divisão da validação cruzada, então os experimentos são considerados dependentes, satisfazendo as restrições necessárias para aplicar o *Teste de Friedman*.

Os resultados individuais obtidos com a validação cruzada a partir de cada base e valor de *K* avaliado estão apresentados no Apêndice A.

Para a base *Iris*, quando $K = 2$, a probabilidade reportada pelo *Teste de Friedman* não

permite rejeitar a hipótese nula, ou seja, não é possível concluir que os algoritmos diferem significativamente ou não, nesse caso específico. Porém, nos casos em que $K = 3$ e $K = 4$, a probabilidade reportada pelo teste permite rejeitar a hipótese nula, indicando que existe pelo menos um par de algoritmos com diferença significativa. No entanto, o *Teste de Nemenyi* não consegue detectar essa diferença, quando $K = 3$. Apenas nos experimentos realizados com $K = 4$, o teste detecta que as combinações da meta-heurística com os três diferentes métodos de inicialização são melhores em relação ao *K-means* inicializado aleatoriamente.

Essa mesma análise foi realizada com a base *Wine*. Para os casos em que $K = 2$ e $K = 3$, a probabilidade reportada pelo *Teste de Friedman* não permite rejeitar a hipótese nula. Nesses casos, portanto, não é possível indicar se existe diferença significativa entre pares de algoritmos. Já para o caso em que $K = 4$, o *Teste de Friedman* indica que existe diferença entre pelo menos um par de algoritmos, porém o *Teste de Nemenyi* não consegue identificar quais são os pares que diferem significativamente.

Com os experimentos realizados com a base *Vehicle*, o *Teste de Friedman* não consegue identificar diferença entre os algoritmos para o caso em que $K = 3$, porém, para $K = 4$ e $K = 5$, o teste rejeita a hipótese nula, indicando a existência de pelo menos um par de algoritmos diferentes. Nesses dois casos, o *Teste de Nemenyi* detecta que o *Simulated Annealing* combinado com o método de inicialização *PCA_Part* apresenta um desempenho melhor se comparado ao algoritmo *K-means* inicializado aleatoriamente.

O método de comparação estatística empregado com a base *Cloud* identifica, para todos os três valores de K ($K = 9$, $K = 10$ e $K = 11$), uma diferença significativa entre pares de algoritmos. Nos experimentos realizados com $K = 9$, o *Teste de Nemenyi* constata melhoria do *K-means* inicializado com *PCA_Part* em relação à mesma heurística inicializada aleatoriamente. O teste também detecta, nesse experimento, melhoria do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* aleatório, ao *K-means++* e ao *Simulated Annealing* aleatório. O teste verifica ainda um melhor desempenho da combinação do *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório.

Quando $K = 10$, o teste constata melhoria do *K-means* inicializado com *PCA_Part*, do *Simulated Annealing* aleatório e do *Simulated Annealing* com inicialização usada pelo *K-means++*, quando esses são comparados, par a par, ao *K-means* aleatório. O teste também identifica que o *Simulated Annealing* inicializado com *PCA_Part* obtém um melhor desempenho quando comparado ao *K-means* aleatório e ao *K-means++*.

Para $K = 11$, o teste constata um melhor desempenho do *Simulated Annealing* inicializado com *PCA_Part* e do *Simulated Annealing* com inicialização usada pelo *K-means++*, ambos em

relação ao *K-means* aleatório.

Com a base *Segmentation*, também constata-se diferença entre pares de algoritmos, para todos os três valores de K avaliados ($K = 6$, $K = 7$ e $K = 8$). No caso em que $K = 6$, o teste indica uma melhor avaliação do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* aleatório e ao *K-means++*. Para $K = 7$, o teste reporta melhoria do algoritmo *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório, ao *K-means* com *PCA_Part* e ao *K-means++*. Reporta também melhoria do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* inicializado com o mesmo método. Quando, $K = 8$, o teste apresenta a mesma avaliação obtida com a base *Cloud* para $K = 11$.

Com a base *Spam*, o *Teste de Friedman* não consegue identificar diferença entre os algoritmos para os casos em que $K = 2$ e $K = 3$. Quando $K = 4$, o teste reporta a seguinte diferença: o *Simulated Annealing* inicializado com o método do *K-means++* apresenta melhores resultados do que o *K-means* aleatório.

Com a base *Pen digit*, o *Teste de Friedman* não consegue identificar diferença entre os algoritmos para os casos em que $K = 9$ e $K = 10$. Quando $K = 11$, o teste reporta as seguintes diferenças: o *Simulated Annealing* aleatório apresenta melhores resultados do que o *K-means* aleatório; o *Simulated Annealing* inicializado com o método *PCA_Part* apresenta melhores resultados do que o *K-means* aleatório.

Por fim, com a base *Letter*, o *Teste de Friedman* não consegue identificar diferença entre os algoritmos para os casos em que $K = 25$ e $K = 27$. Quando $K = 26$, o teste reporta as seguintes diferenças: o *K-means* aleatório apresenta melhores resultados do que o *Simulated Annealing* inicializado com o método do *K-means++*; o *K-means* inicializado com *PCA_Part* apresenta um melhor desempenho em relação ao *Simulated Annealing* inicializado com *PCA_Part* e com o método de inicialização do *K-means++*; o algoritmo *K-means++* apresenta melhores resultados do que o *Simulated Annealing* com inicialização usada pelo *K-means++*.

Essa avaliação mostra uma indicação de que a meta-heurística combinada a métodos de inicialização consegue superar, especialmente, o *K-means* aleatório, e, em alguns casos, o *K-means* com outros métodos de inicialização. Entre os dois métodos de inicialização sugeridos para serem combinados com o *Simulated Annealing*, o *PCA_Part* combinado a essa meta-heurística consegue superar mais vezes o *K-means* aleatório do que o *K-means++* combinado à mesma meta-heurística.

O bom desempenho apresentado pelo algoritmo *Simulated Annealing* inicializado com o método *PCA_Part* é constatado tanto em três (*Cloud* e *Segmentation Pen digit*) das cinco instân-

cias maiores do problema, como também em duas (*Iris* e *Vehicle*) das três instâncias menores, sendo, contudo, mais freqüente nos primeiros casos. Essa observação pode indicar uma evidência de que essa combinação apresenta uma melhoria mais significativa em instâncias maiores do problema de agrupamento, embora o resultado obtido com essa combinação na maior base (*Letter*) tenha sido pior do que com as combinações do *K-means*.

No entanto, essas constatações não podem ser generalizadas para qualquer problema, já que os testes foram realizados em cada base separadamente. Apenas os testes realizados com a Tabela 6.3, que avaliam amostras de problemas independentes, permitem generalizar que, para qualquer problema de agrupamento, o algoritmo *Simulated Annealing* inicializado com *PCA_Part* apresenta melhores resultados do que o *K-means* aleatório. Em relação aos demais algoritmos, nada se pode concluir.

7 Conclusões e Trabalhos Futuros

Este trabalho apresenta o estudo de métodos que geram soluções iniciais para o problema de agrupamento aplicados em conjunto com heurísticas e meta-heurísticas. Essa combinação tem como objetivo selecionar soluções iniciais mais próximas ao ótimo global, aumentando as possibilidades do algoritmo alcançar com poucas iterações a melhor solução de acordo com um critério de avaliação.

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 seis bases reais disponíveis em repositórios públicos. Essas bases foram escolhidas por apresentarem as características adequadas para os algoritmos avaliados: atributos numéricos e não nulos.

O algoritmo proposto, uma versão adaptada do *Simulated Annealing* para o problema de agrupamento, foi combinado a dois métodos de inicialização: *PCA_Part* e o método de inicialização do *K-means++*. Essas duas combinações foram comparadas à mesma meta-heurística inicializada aleatoriamente, assim como já proposto em outros trabalhos, e ao *K-means* que é a heurística mais empregada para resolver o problema. Esse último foi empregado tanto na versão padrão, na qual o algoritmo é inicializado aleatoriamente, quanto combinado com os mesmos métodos de inicialização usados com a meta-heurística.

Dois tipos de avaliação foram realizados. No primeiro, as seis diferentes combinações de métodos de inicialização e procedimentos de busca foram comparadas em múltiplos domínios, a fim de descobrir se há uma tendência geral de melhoria do algoritmo proposto em relação aos já existentes. Nesse caso, a análise estatística mostrou que, em geral, o *Simulated Annealing* inicializado com o método *PCA_Part* fornece melhores resultados do que o *K-means* inicializado aleatoriamente. Em relação aos demais algoritmos, a análise não permite concluir se existe ou não diferença entre eles.

Na segunda avaliação, as seis combinações de algoritmo de busca foram avaliadas em cada base separadamente, para indicar em quais tipos de base (instâncias menores ou maiores do

problema) a solução proposta é mais favorável, embora essa indicação não possa ser generalizada para qualquer problema. O resultado mostrou que o algoritmo *Simulated Annealing* inicializado com o método *PCA_Part* apresentou um bom desempenho tanto em duas das três bases menores quanto em duas das três bases maiores. No entanto, nas duas bases maiores, essa combinação apresentou mais diferenças em relação às demais, indicando uma evidência de que a solução proposta neste trabalho apresenta uma melhoria mais significativa em instâncias maiores do problema de agrupamento.

Por fim, uma importante contribuição deste trabalho foi o desenvolvimento do algoritmo *Simulated Annealing* adaptado ao problema de agrupamento e inicializado com dois métodos de inicialização *PCA_Part* e *K-means++*, sendo que o primeiro é um método determinístico que apresenta bons resultados quando comparados a outros métodos e que o segundo garante soluções próximas à ótima, a menos de uma constante. Outra contribuição relevante foi a ampla experimentação comparativa realizada neste trabalho, mostrando que a abordagem proposta apresenta bons resultados quando comparados às abordagens clássicas, como o algoritmo *K-means* e o *Simulated Annealing* inicializado aleatoriamente.

7.1 Trabalhos Futuros

A proposta abordada neste trabalho apresenta uma adaptação do algoritmo *Simulated Annealing* baseado em versões indicadas em outros trabalhos, tais como (KLEIN; DUBES, 1989) e (SELIM; ALSULTAN, 1991), para resolver o problema de agrupamento. No entanto, a avaliação experimental realizada não teve o objetivo de mostrar que esse algoritmo supera as adaptações do *Simulated Annealing* já propostas. Por isso, uma análise da combinação de outras versões do *Simulated Annealing* com métodos de inicialização pode aprimorar os resultados obtidos com os experimentos realizados neste trabalho.

Um outro estudo semelhante a esse avalia o uso de outros métodos de inicialização propostos para o *K-means*, que, no entanto, não foram avaliados neste trabalho. Uma análise comparativa dos métodos de inicialização propostos na literatura auxilia na escolha dos mais adequados para serem combinados com as meta-heurísticas.

É possível ainda, após essa análise, comparar os métodos de inicialização específicos para o problema de agrupamento (voltados para o *K-means*) combinados a meta-heurísticas com o algoritmo de busca *GRASP*. A meta-heurística *GRASP* gera a solução inicial usando um método construtivo de caráter geral (não é específico a um problema de otimização) e realiza o processo de busca usando um algoritmo básico de busca local (um procedimento de melhoria local, por

exemplo) ou um algoritmo de busca local mais avançado, tal como *Simulated Annealing* ou *Tabu Search*. Essa avaliação pode indicar qual tipo de método de inicialização é mais favorável para obter melhores resultados, se são os métodos específicos ao problema de agrupamento (voltados para o *K-means*) ou se são os de caráter mais geral (usados no algoritmo *GRASP*).

Uma outra possível continuação deste estudo está em analisar a aplicação de outras meta-heurísticas combinadas a métodos de inicialização, já que esta dissertação realiza uma avaliação experimental da abordagem proposta apenas com o algoritmo *Simulated Annealing* adaptado ao problema.

Um exemplo de outra meta-heurística que pode ser avaliada experimentalmente é o *Tabu Search*. Esse algoritmo, assim como o *Simulated Annealing* inicia o processo de busca pela solução ótima a partir de uma solução inicial, que, em geral, é escolhida aleatoriamente. Outra meta-heurística que pode ser combinada a métodos de inicialização é o algoritmo genético, que realiza a busca a partir de uma população de soluções iniciais. Nos exemplos de algoritmos citados, a escolha da configuração inicial das partições pode favorecer a convergência para uma solução mais próxima ao ótimo global, seguindo o comportamento do *Simulated Annealing*.

A avaliação de outros algoritmos permite melhorar a qualidade da comparação estatística apresentada no Capítulo 6. A realização de experimentos com um número maior 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, para realizar uma avaliação mais precisa dos algoritmos estudados.

Dentre os experimentos realizados, constatou-se que na maior base (*Letter*) a meta-heurística obteve em todos os testes desempenho inferior ao do *K-means*, apresentando um comportamento diferente do esperado. Uma possível continuação para este trabalho consiste em analisar a razão para esse resultado, avaliando se existe ou não alguma característica da base que leva a esse comportamento.

Por fim, uma complementação para este trabalho está em realizar a análise do tempo de execução dos algoritmos, a fim de quantificar a diferença entre a aplicação do *K-means* que realiza um procedimento heurístico de busca local e da meta-heurística *Simulated Annealing* que percorre o espaço de busca de maneira mais robusta. Embora este trabalho não tenha focado na comparação do tempo entre os algoritmos, observou-se que em todos os testes, o *K-means* apresentou um desempenho mais eficiente, exigindo menos iterações para alcançar o resultado final.

Referências Bibliográficas

AARTS, E. H. L.; LENSTRA, J. K. *Local Search in Combinatorial Optimization*. Chichester, UK: Wiley, 1997.

AL-SULTAN, K. A Tabu Search Approach to the Clustering Problem. *Pattern Recognition*, v. 28, n. 9, p. 1443–1451, 1995.

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

ASUNCION, A.; NEWMAN, D. *UCI Machine Learning Repository*. 2007. Disponível em: <http://www.ics.uci.edu/∼mlearn/MLRepository.html>.

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.

BABU, G. P.; MURTY, M. N. Simulated Annealing for Optimal Initial Seed Selection in K-means Algorithm. *Indian Journal of Pure and Applied Mathematics*, v. 3, p. 85–94, 1994.

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

BLUM, C.; ROLI, A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, v. 35, n. 3, p. 268–308, 2003.

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

BRADLEY, P. S.; FAYYAD, U. M. Refining Initial Points for K-means Clustering. *In Proceedings of the Fifteenth International Conference on Machine Learning*, San Francisco, CA, USA, p. 91–99, 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.

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

DEVIJVER, P. A.; KITTLER, J. *Pattern Recognition: A Statistical Approach*. London: Prentice Hall, 1982.

DIETTERICH, T. G. Approximate Statistical Test For Comparing Supervised Classification Learning Algorithms. *Neural Computation*, v. 10, n. 7, p. 1895–1923, 1998.

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 Discrete Optimization. *Artificial Life*, v. 5, n. 2, p. 137–172, 1999.

DUBES, R. C. How Many Clusters are Best? - An Experiment. *Pattern Recognition*, v. 20, n. 6, p. 645–663, 1987.

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

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

FASULO, D. An Analysis of Recent Work on Clustering Algorithms. *Technical Report 01-03-02*, 1999.

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

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

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

FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, v. 32, n. 200, p. 675–701, 1937.

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.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. [S.l.]: W. H. Freeman and Company, 1979.

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. W.; KOCHENBERGER, G. A. *Handbook of Metaheuristics*. [S.l.]: Springer, 2003.

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

GRIRA, N.; HOULE, M. E. Best of Both: A Hybridized Centroid-Medoid Clustering Heuristic. *In Proceedings of the 24th International Conference on Machine Learning*, Corvalis, Oregon, v. 227, p. 313–320, 2007.

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.

HAMMING, R. W. Error detecting and error correcting codes. *Bell System Technical Journal*, v. 29, n. 2, p. 147–160, 1950.

HANSEN, P.; JAUMARD, B. Cluster Analysis and Mathematical Programming. *Sixteenth International Symposium on Mathematical Programming*, v. 79, p. 191–215, 1997.

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.

HUANG, Z. Clustering Large Data Sets with Mixed Numeric and Categorical Value. *In Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conf. World Scientific*, p. 21–34, 1997.

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.

JOLLIFFE, I. T. *Principal Component Analysis*. [S.l.]: Springer-Verlag, 1986.

KANADE, P. M.; HALL, L. O. Fuzzy Ants Clustering with Centroids. *Proceedings of the International Conference on Fuzzy Systems*, 2004.

KATSAVOUNIDIS, I.; KUO, C.-C.; ZHANG, Z. A New Initialization Technique for Generalized Lloyd Iteration. *IEEE Signal Processing Letters*, v. 1, n. 10, p. 144–146, 1994.

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.

KLEIN, R. W.; DUBES, R. C. Experiments in Projection and Clustering by Simulated Annealing. *Pattern Recognition*, v. 22, n. 2, p. 213–220, 1989.

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.

LIU, G. L. *Introduction to Combinatorial Mathematics*. New York: McGraw Hill, 1968.

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

MANGASARIAN, O. L. Mathematical Programming in Data Mining. *Data Mining and Knowledge Discovery*, v. 1, n. 2, p. 183–201, 1997.

MAULIK, U.; BANDYOPADHYAY, S. Genetic algorithm-based clustering technique. *Pattern Recognition*, v. 33, p. 1455–1465, 2000.

MERWE, D. W. van der; ENGELBRECHT, A. P. Data Clustering using Particle Swarm Optimization. *Congress on Evolutionary Computation*, v. 1, p. 215–220, 2003.

METROPOLIS, N. et al. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, v. 21, p. 1087–1092, 1953.

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.

OSMAN, I.; LAPORTE, G. Metaheuristics: A bibliography. *Annals of Operations Research*, v. 63, n. 5, p. 511–623, 1996.

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.

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.

RAO, M. R. Cluster Analysis and Mathematical Programming. *Journal of the American Statistical Association*, v. 66, n. 335, p. 622–626, 1971.

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

SALZBERG, S. L. On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 3, p. 317–328, 1997.

SELIM, S. Z.; ALSULTAN, K. A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, Elsevier Science Inc., New York, NY, USA, v. 24, n. 10, p. 1003–1008, 1991.

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

SPATH, H. *Cluster Analysis Algorithms*. [S.l.]: Ellis Horwood, 1980.

STüTZLE, T. *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications*. Tese (Doutorado) — Darmstadt University, Computer Science Department, 1999.

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

SU, T.; DY, J. In Search of Deterministic Methods for Initializing K-means and Gaussian Mixture Clustering. *Intelligent Data Analysis*, v. 11, n. 4, p. 319–338, 2007.

SUN, Y.; ZHU, Q.; CHEN, Z. An Iterative Initial-Points Refinement Algorithm for Categorical Data Clustering. *Pattern Recognition Letters*, v. 23, p. 875–884, 2002.

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

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition*. Amsterdam: Elsevier/Academic Press, 2006. ISBN 0123695317.

TOU, J. T.; GONZALEZ, R. C. *Pattern Recognition Principles*. [S.l.]: Addison-Wesley, 1974.

VINOD, H. D. Integer Programming and the Theory of Grouping. *Journal of the American Statistical Association*, v. 64, n. 326, p. 506–519, 1969.

XU, L. How Many Clusters?: A Ying-Yang Machine Based Theory for a Classical Open Problem in Pattern Recognition. *IEEE International Conference on Neural Networks*, Washington, DC, USA, v. 3, p. 1546–1551, 1996.

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

ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. BIRCH: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery*, v. 1, n. 2, p. 141–182, 1997.

APÊNDICE A – Resultados Individuais da Validação Cruzada por Base

Este apêndice apresenta os resultados individuais da validação cruzada para cada base experimentada, com três diferentes valores de K estudados, sendo um deles igual ao número de classes encontradas nos dados.

As Tabelas A.1, A.2 e A.3 mostram os valores de *SSE* obtidos na base *Iris*, fazendo K = 2, 3 e 4, respectivamente. Cada linha das tabelas mostra o resultado individual no conjunto Y_i da validação cruzada para cada combinação do algoritmo. O posto atribuído pelo *Teste de Friedman* está em parênteses e os postos médios na última linha das tabelas.

Tabela A.1: Comparação estatística para a base *Iris* quando K = 2.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	17.452 (3.5)	17.452 (3.5)	17.452 (3.5)	17.452 (3.5)	17.452 (3.5)	17.452 (3.5)
Y_2	14.759 (3.5)	14.759 (3.5)	14.759 (3.5)	14.759 (3.5)	14.759 (3.5)	14.759 (3.5)
Y_3	12.904 (3.5)	12.904 (3.5)	12.904 (3.5)	12.904 (3.5)	12.904 (3.5)	12.904 (3.5)
Y_4	14.715 (3.5)	14.715 (3.5)	14.715 (3.5)	14.715 (3.5)	14.715 (3.5)	14.715 (3.5)
Y_5	13.911 (3.5)	13.911 (3.5)	13.911 (3.5)	13.911 (3.5)	13.911 (3.5)	13.911 (3.5)
Y_6	5.452 (3.5)	5.452 (3.5)	5.452 (3.5)	5.452 (3.5)	5.452 (3.5)	5.452 (3.5)
Y_7	10.446 (3.5)	10.446 (3.5)	10.446 (3.5)	10.446 (3.5)	10.446 (3.5)	10.446 (3.5)
Y_8	13.964 (3.5)	13.964 (3.5)	13.964 (3.5)	13.964 (3.5)	13.964 (3.5)	13.964 (3.5)
Y_9	15.402 (3.5)	15.402 (3.5)	15.402 (3.5)	15.402 (3.5)	15.402 (3.5)	15.402 (3.5)
Y_{10}	20.486 (5.5)	20.486 (5.5)	19.695 (2.5)	19.695 (2.5)	19.695 (2.5)	19.695 (2.5)
p.m.	3.70	3.70	3.40	3.40	3.40	3.40

Para os dados da Tabela A.1, o *Teste de Friedman* reporta uma probabilidade de 99.73%, supondo que a hipótese nula é verdadeira. Portanto, não é possível rejeitar essa hipótese com nível de significância de 5%. Com isso, não se pode afirmar se existe diferença significativa entre os algoritmos para a base *Iris* com $K = 2$.

No caso dos dados das Tabelas A.2 e A.3, o *Teste de Friedman* reporta uma probabilidade de 3.43% e 0.002%, respectivamente. Nesses casos, a hipótese nula pode ser rejeitada, indicando que existe diferença significativa entre pelo menos dois algoritmos.

Tabela A.2: Comparação estatística para a base *Iris* quando K = 3.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	5.871 (3)	5.871 (3)	6.179 (6)	5.871 (3)	5.871 (3)	5.871 (3)
Y_2	6.144 (3)	6.298 (6)	6.144 (3)	6.144 (3)	6.144 (3)	6.144 (3)
Y_3	25.673 (6)	6.452 (3)	6.452 (3)	6.452 (3)	6.452 (3)	6.452 (3)
Y_4	7.468 (4.5)	7.468 (4.5)	8.463 (6)	5.921 (2)	5.921 (2)	5.921 (2)
Y_5	3.522 (3)	3.522 (3)	4.988 (6)	3.522 (3)	3.522 (3)	3.522 (3)
Y_6	6.328 (5.5)	4.204 (4)	6.328 (5.5)	4.063 (2)	4.063 (2)	4.063 (2)
Y_7	12.607 (5.5)	12.607 (5.5)	7.894 (4)	6.772 (2)	6.772 (2)	6.772 (2)
Y_8	7.765 (3.5)	7.765 (3.5)	7.765 (3.5)	7.765 (3.5)	7.765 (3.5)	7.765 (3.5)
Y_9	11.163 (6)	7.684 (5)	7.284 (2.5)	7.284 (2.5)	7.284 (2.5)	7.284 (2.5)
Y_{10}	13.097 (6)	6.044 (3)	6.044 (3)	6.044 (3)	6.044 (3)	6.044 (3)
p.m.	4.60	4.05	4.25	2.70	2.70	2.70

Tabela A.3: Comparação estatística para a base *Iris* quando K = 4.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	5.871 (6)	5.871 (5)	6.179 (2.5)	5.871 (2.5)	5.871 (2.5)	5.871 (2.5)
Y_2	6.144 (5.5)	6.298 (5.5)	6.144 (4)	6.144 (2)	6.144 (2)	6.144 (2)
Y_3	25.673 (2.5)	6.452 (6)	6.452 (5)	6.452 (2.5)	6.452 (2.5)	6.452 (2.5)
Y_4	7.468 (3)	7.468 (3)	8.463 (6)	5.921 (3)	5.921 (3)	5.921 (3)
Y_5	3.522 (5)	3.522 (6)	4.988 (2.5)	3.522 (2.5)	3.522 (2.5)	3.522 (2.5)
Y_6	6.328 (6)	4.204 (4)	6.328 (5)	4.063 (2)	4.063 (2)	4.063 (2)
Y_7	12.607 (6)	12.607 (5)	7.894 (4)	6.772 (2)	6.772 (2)	6.772 (2)
Y_8	7.765 (6)	7.765 (3)	7.765 (3)	7.765 (3)	7.765 (3)	7.765 (3)
Y_9	11.163 (5)	7.684 (4)	7.284 (6)	7.284 (2)	7.284 (2)	7.284 (2)
Y_{10}	13.097 (6)	6.044 (4.5)	6.044 (4.5)	6.044 (2)	6.044 (2)	6.044 (2)
p.m.	5.10	4.60	4.25	2.35	2.35	2.35

O método usado para reconhecer a diferença entre um par de algoritmos subtrai os postos médios de cada coluna da tabela e compara essa diferença com a diferença crítica, que nesse caso é igual a $CD = 2.85\sqrt{\frac{6 \cdot 7}{6 \cdot 10}} = 2.38$. Apenas no caso em que $K = 4$, foi possível identificar que as três diferentes combinações com o *Simulated Annealing* foram melhores do que o *K-means* aleatório, já que $5.10 - 2.35 = 2.75 > 2.38$.

As Tabelas A.4, A.5 e A.6 mostram os valores de *SSE* obtidos na base *Wine*, fazendo K = 2, 3 e 4, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

A partir dos dados das Tabelas A.4, A.5 e A.6, o *Teste de Friedman* reporta probabilidade de 75.16%, 62.10% e 4.08%, respectivamente, supondo que a hipótese nula é verdadeira. Dessa forma, para $K = 2$ e $K = 3$, não se pode rejeitar a hipótese nula com nível de significância de 5%, indicando nada se pode afirmar em relação a diferenças entre pares de algoritmos. Para $K = 4$, a probabilidade reportada indica que existe diferença significativa em pelo menos um

Tabela A.4: Comparação estatística para a base *Wine* quando K = 2.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	536175 (5.5)	473463 (2.5)	536175 (5.5)	473463 (2.5)	473463 (2.5)	473463 (2.5)
Y_2	484555 (3.5)	484555 (3.5)	484555 (3.5)	484555 (3.5)	484555 (3.5)	484555 (3.5)
Y_3	393393 (3.5)	393393 (3.5)	393393 (3.5)	393393 (3.5)	393393 (3.5)	393393 (3.5)
Y_4	446889 (3.5)	446889 (3.5)	446889 (3.5)	446889 (3.5)	446889 (3.5)	446889 (3.5)
Y_5	472586 (3)	491889 (6)	472586 (3)	472586 (3)	472586 (3)	472586 (3)
Y_6	177746 (3.5)	177746 (3.5)	177746 (3.5)	177746 (3.5)	177746 (3.5)	177746 (3.5)
Y_7	357570 (3.5)	357570 (3.5)	357570 (3.5)	357570 (3.5)	357570 (3.5)	357570 (3.5)
Y_8	271105 (5.5)	271105 (5.5)	269509 (2.5)	269509 (2.5)	269509 (2.5)	269509 (2.5)
Y_9	277096 (5.5)	277096 (5.5)	275566 (2.5)	275566 (2.5)	275566 (2.5)	275566 (2.5)
Y_{10}	275557 (3.5)	275557 (3.5)	275557 (3.5)	275557 (3.5)	275557 (3.5)	275557 (3.5)
p.m.	4.05	4.05	3.45	3.15	3.15	3.15

Tabela A.5: Comparação estatística para a base *Wine* quando K = 3.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	120601 (5.5)	120601 (5.5)	113936 (2.5)	113936 (2.5)	113936 (2.5)	113936 (2.5)
Y_2	100137 (3)	97201 (1.5)	146266 (5)	146266 (5)	97201 (1.5)	146266 (5)
Y_3	293376 (3)	293376 (3)	293376 (3)	307755 (6)	293376 (3)	293376 (3)
Y_4	98958 (4)	98958 (4)	116424 (6)	97011 (1.5)	98958 (4)	97011 (1.5)
Y_5	181636 (3.5)	181636 (3.5)	181636 (3.5)	181636 (3.5)	181636 (3.5)	181636 (3.5)
Y_6	200927 (3)	203380 (6)	200927 (3)	200927 (3)	200927 (3)	200927 (3)
Y_7	78365 (3)	90797 (6)	78365 (3)	78365 (3)	78365 (3)	78365 (3)
Y_8	202310 (6)	139650 (3)	139650 (3)	139650 (3)	139650 (3)	139650 (3)
Y_9	167744 (5.5)	167744 (5.5)	104610 (2.5)	104610 (2.5)	104610 (2.5)	104610 (2.5)
Y_{10}	148239 (3)	148239 (3)	184485 (6)	148239 (3)	148239 (3)	148239 (3)
p.m.	3.95	4.10	3.75	3.30	2.90	3.00

Tabela A.6: Comparação estatística para a base *Wine* quando K = 4.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	192908 (6)	185507 (5)	135526 (2.5)	135526 (2.5)	135526 (2.5)	135526 (2.5)
Y_2	71625 (6)	51347 (2.5)	56467 (5)	51347 (2.5)	51347 (2.5)	51347 (2.5)
Y_3	73625 (2.5)	74180 (5)	76362 (6)	73625 (2.5)	73625 (2.5)	73625 (2.5)
Y_4	48863 (2.5)	48863 (2.5)	132347 (6)	48863 (2.5)	48863 (2.5)	131143 (5)
Y_5	108501 (6)	45968 (2.5)	91773 (5)	45968 (2.5)	45968 (2.5)	45968 (2.5)
Y_6	65272 (3.5)	65272 (3.5)	65272 (3.5)	65272 (3.5)	65272 (3.5)	65272 (3.5)
Y_7	155411 (6)	118218 (3)	118218 (3)	118218 (3)	118218 (3)	118218 (3)
Y_8	67366 (2)	79530 (4)	82632 (6)	79719 (5)	67366 (2)	67366 (2)
Y_9	87159 (2)	125389 (5)	125536 (6)	87159 (2)	122250 (4)	87159 (2)
Y_{10}	66338 (4)	66338 (4)	190697 (6)	62715 (1.5)	66338 (4)	62715 (1.5)
p.m.	4.05	3.70	4.90	2.75	2.90	2.70

par de algoritmos. No entanto, o *Teste de Nemenyi* não consegue identificar quais são esses pares, pois os postos médios não diferem de pelo menos $CD = 2.38$.

As Tabelas A.7, A.8 e A.9 apresentam os resultados obtidos com a base *Vehicle*, quando K = 3, 4 e 5, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

Tabela A.7: Comparação estatística para a base *Vehicle* quando K = 3.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	465231 (4.5)	465231 (4.5)	461741 (1.5)	573978 (6)	464455 (3)	461741 (1.5)
Y_2	430047 (1.5)	478049 (6)	461254 (5)	430047 (1.5)	460890 (3.5)	460890 (3.5)
Y_3	418458 (1.5)	454678 (3.5)	500065 (5.5)	418458 (1.5)	454678 (3.5)	500065 (5.5)
Y_4	369946 (6)	321129 (3)	321129 (3)	321129 (3)	321129 (3)	321129 (3)
Y_5	482199 (6)	481315 (3)	481315 (3)	481315 (3)	481315 (3)	481315 (3)
Y_6	612103 (3.5)	599957 (1.5)	629789 (6)	612103 (3.5)	599957 (1.5)	623498 (5)
Y_7	535929 (1)	574435 (5)	577254 (6)	574257 (3)	574257 (3)	574257 (3)
Y_8	495918 (6)	484506 (4.5)	439161 (1.5)	469931 (3)	484506 (4.5)	439161 (1.5)
Y_9	371426 (5)	395136 (6)	367238 (2)	370401 (4)	367238 (2)	367238 (2)
Y_{10}	458605 (2)	458605 (2)	574617 (5)	574617 (5)	458605 (2)	574617 (5)
p.m.	3.70	3.90	3.85	3.35	2.90	3.30

Tabela A.8: Comparação estatística para a base *Vehicle* quando K = 4.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	307812 (5.5)	307187 (4)	307812 (5.5)	307105 (2)	307105 (2)	307105 (2)
Y_2	210629 (3)	210629 (3)	350034 (6)	210629 (3)	210629 (3)	210629 (3)
Y_3	205779 (6)	185796 (3)	185796 (3)	185796 (3)	185796 (3)	185796 (3)
Y_4	526218 (6)	341076 (4)	429141 (5)	339982 (2)	339982 (2)	339982 (2)
Y_5	352980 (5.5)	352980 (5.5)	304212 (3)	331147 (4)	304066 (1.5)	304066 (1.5)
Y_6	314828 (4)	314189 (2)	357805 (5.5)	314189 (2)	314189 (2)	357805 (5.5)
Y_7	357069 (3.5)	357069 (3.5)	357069 (3.5)	357069 (3.5)	357069 (3.5)	357069 (3.5)
Y_8	413877 (6)	308897 (2)	371267 (4.5)	308897 (2)	308897 (2)	371267 (4.5)
Y_9	248940 (6)	204566 (5)	193766 (2.5)	193766 (2.5)	193766 (2.5)	193766 (2.5)
Y_{10}	309130 (2)	324066 (5)	329610 (6)	320493 (3.5)	308557 (1)	320493 (3.5)
p.m.	4.75	3.70	4.45	2.75	2.25	3.10

Pela Tabela A.7, o *Teste de Friedman* reporta probabilidade de 84.61%, indicando que não há diferença entre os algoritmos para a base *Vehicle* quando $K = 3$. No caso das Tabelas A.8 e A.9, o *Teste de Friedman* reporta probabilidade de 1.10% e 1.35%, respectivamente. Portanto, nesses casos, a hipótese nula pode ser rejeitada, indicando que existe diferença significativa entre pelo menos dois algoritmos.

Nos dois casos, a diferença constatada foi que o *Simulated Annealing* combinado com o método de inicialização *PCA_Part* apresenta um desempenho melhor se comparado ao algo-

Tabela A.9: Comparação estatística para a base *Vehicle* quando K = 5.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	249443 (5.5)	236655 (2)	249443 (5.5)	249413 (3.5)	212436 (1)	249413 (3.5)
Y_2	209984 (5)	209124 (4)	234307 (6)	208813 (2)	208813 (2)	208813 (2)
Y_3	381304 (6)	313121 (3)	379386 (5)	340639 (4)	311717 (1.5)	311717 (1.5)
Y_4	206545 (4.5)	206545 (4.5)	256852 (6)	205468 (2)	205468 (2)	205468 (2)
Y_5	253203 (6)	186388 (3)	186388 (3)	186388 (3)	186388 (3)	186388 (3)
Y_6	249567 (6)	247125 (5)	230295 (1.5)	236777 (3.5)	236777 (3.5)	230295 (1.5)
Y_7	166868 (4.5)	161371 (2)	166868 (4.5)	166868 (4.5)	160670 (1)	166868 (4.5)
Y_8	243329 (1.5)	260240 (6)	254518 (5)	243329 (1.5)	251792 (3)	254138 (4)
Y_9	278847 (6)	210087 (5)	205744 (2.5)	205744 (2.5)	205744 (2.5)	205744 (2.5)
Y_{10}	183440 (3.5)	183440 (3.5)	183440 (3.5)	183440 (3.5)	183440 (3.5)	183440 (3.5)
p.m.	4.85	3.80	4.25	3.00	2.30	2.80

ritmo *K-means* inicializado aleatoriamente. Quando $K = 4$, a diferença entre os postos médios desses dois algoritmos foi igual a $4.75 - 2.25 = 2.50 > 2.38$, e quando $K = 5$, a diferença foi de $4.85 - 2.30 = 2.55 > 2.38$.

As Tabelas A.10, A.11 e A.12 mostram os resultados obtidos com a base *Cloud*, quando K = 9, 10 e 11, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

Tabela A.10: Comparação estatística para a base *Cloud* quando K = 9.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	799094 (6)	604073 (3)	733461 (5)	661429 (4)	543522 (2)	541884 (1)
Y_2	1032130 (6)	566329 (4)	557637 (1)	968689 (5)	560110 (2.5)	560110 (2.5)
Y_3	482618 (6)	366274 (2)	399028 (4)	453024 (5)	364863 (1)	397502 (3)
Y_4	968588 (6)	567675 (2)	623181 (4)	820935 (5)	567669 (1)	612098 (3)
Y_5	696994 (6)	600349 (4)	592679 (3)	668615 (5)	530897 (1.5)	530897 (1.5)
Y_6	490165 (6)	431909 (5)	429886 (4)	415567 (3)	415412 (2)	346249 (1)
Y_7	591371 (4)	538378 (2)	609397 (5)	710160 (6)	521597 (1)	548639 (3)
Y_8	535422 (5)	496031 (2)	535191 (4)	991998 (6)	456404 (1)	530673 (3)
Y_9	1726390 (6)	461035 (2)	807179 (5)	716776 (3)	452692 (1)	803037 (4)
Y_{10}	1023500 (6)	410766 (5)	406588 (4)	379698 (2.5)	364603 (1)	379698 (2.5)
p.m.	5.70	3.10	3.90	4.45	1.40	2.45

Para essa base, nos três casos avaliados, o teste de *Teste de Friedman* rejeita a hipótese nula, reportando probabilidade de 1.26e-07%, de 1.74e-06% e de 8.81e-05%, para $K = 9$, $K = 10$ e $K = 11$, respectivamente.

As diferenças encontradas nessa base, para $K = 9$, foram as seguintes: melhoria do *K-means* inicializado com *PCA_Part* em relação à mesma heurística inicializada aleatoriamente ($5.70 - 3.10 = 2.60 > 2.38$); melhoria do *Simulated Annealing* inicializado com *PCA_Part* em

Tabela A.11: Comparação estatística para a base *Cloud* quando K = 10.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	439046 (6)	344818 (2)	412246 (5)	333810 (1)	352489 (3)	379662 (4)
Y_2	497552 (6)	438377 (5)	362462 (3)	374771 (4)	351059 (1)	360942 (2)
Y_3	696803 (5)	415453 (3)	459096 (4)	724689 (6)	397725 (1.5)	397725 (1.5)
Y_4	592961 (6)	421199 (2)	516146 (5)	464966 (4)	421129 (1)	436718 (3)
Y_5	333418 (6)	331172 (4)	332520 (5)	325210 (3)	324852 (2)	320940 (1)
Y_6	356126 (5)	344426 (2)	363749 (6)	354783 (4)	342264 (1)	349038 (3)
Y_7	428024 (6)	396213 (4)	410364 (5)	374326 (1.5)	374587 (3)	374326 (1.5)
Y_8	637461 (6)	353631 (2)	462401 (5)	350636 (1)	356717 (3)	386857 (4)
Y_9	1231740 (6)	697098 (4)	659259 (3)	762001 (5)	572818 (2)	521560 (1)
Y_{10}	1483510 (6)	469854 (4)	506243 (5)	449403 (3)	433303 (1)	442792 (2)
p.m.	5.80	3.20	4.60	3.25	1.85	2.30

Tabela A.12: Comparação estatística para a base *Cloud* quando K = 11.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	656903 (6)	417387 (3)	417942 (4)	564339 (5)	406237 (1)	413783 (2)
Y_2	654878 (6)	356068 (3)	428291 (5)	339008 (1)	340287 (2)	359938 (4)
Y_3	606979 (6)	373151 (2)	423413 (5)	409490 (3.5)	346832 (1)	409490 (3.5)
Y_4	394783 (6)	314945 (3)	328243 (4)	352514 (5)	293184 (1.5)	293184 (1.5)
Y_5	291907 (4)	333067 (6)	290913 (3)	286605 (1)	310568 (5)	287500 (2)
Y_6	1291740 (6)	348466 (3)	368602 (4)	395800 (5)	336646 (2)	310043 (1)
Y_7	641505 (6)	450027 (3)	579048 (5)	458435 (4)	396475 (1.5)	396475 (1.5)
Y_8	1346540 (6)	573806 (4)	477262 (1)	574573 (5)	491155 (2.5)	491155 (2.5)
Y_9	364548 (6)	276265 (4)	278277 (5)	253899 (3)	238227 (1.5)	238227 (1.5)
Y_{10}	345978 (6)	331943 (5)	294583 (2)	298819 (4)	295864 (3)	277631 (1)
p.m.	5.80	3.60	3.80	3.65	2.10	2.05

relação ao *K-means* aleatório ($5.70-1.40=4.30>2.38$), ao *K-means++* ($3.90-1.40=2.50>2.38$) e ao *Simulated Annealing* aleatório ($4.45-1.40=3.05>2.37$); melhoria do *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório ($5.70-2.45=3.25>2.38$).

Quando $K=10$, o teste constata as seguintes diferenças: melhoria do *K-means* inicializado com *PCA_Part* em relação ao *K-means* aleatório ($5.80-3.20=2.60>2.38$); melhoria do *Simulated Annealing* aleatório em relação ao *K-means* aleatório ($5.80-3.25=2.55>2.38$); melhoria do *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório ($5.80-2.30=3.50>2.38$); melhoria do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* aleatório ($5.80-1.85=3.95>2.38$) e ao *K-means++* ($4.60-1.85=2.75>2.38$).

Para $K=11$, o teste constata as seguintes diferenças: melhoria do *Simulated Annealing*

inicializado com *PCA_Part* em relação ao *K-means* aleatório ($5.80 - 2.10 = 3.70 > 2.38$); melhoria do *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório ($5.80 - 2.05 = 3.75 > 2.38$).

As Tabelas A.13, A.14 e A.15 mostram os resultados obtidos com a base *Segmentation*, para os valores de K = 6, 7 e 8, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

Tabela A.13: Comparação estatística para a base *Segmentation* quando K = 6.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	1692170 (4)	1829980 (6)	1693790 (5)	1650400 (2.5)	1442620 (1)	1650400 (2.5)
Y_2	1013470 (5)	967915 (2)	1038970 (6)	951662 (1)	978721 (3.5)	978721 (3.5)
Y_3	1173760 (5)	1083390 (2)	1121180 (4)	1384840 (6)	1071360 (1)	1093600 (3)
Y_4	1529540 (6)	1351380 (2)	1434690 (5)	1434580 (3.5)	1337020 (1)	1434580 (3.5)
Y_5	1894930 (6)	1270350 (2)	1397130 (4)	1794140 (5)	1222490 (1)	1344980 (3)
Y_6	2160080 (6)	1651770 (4)	1466450 (2)	2128820 (5)	1465850 (1)	1467580 (3)
Y_7	997879 (5)	961538 (4)	1020890 (6)	958374 (2)	958374 (2)	958374 (2)
Y_8	929554 (1)	994839 (6)	930418 (2)	940968 (5)	938830 (3.5)	938830 (3.5)
Y_9	1796990 (6)	1130270 (2)	1146050 (4)	1149780 (5)	1119250 (1)	1146000 (3)
Y_{10}	1229410 (5)	1262750 (6)	1215370 (4)	1207260 (2)	1194870 (1)	1212450 (3)
p.m.	4.90	3.60	4.20	3.70	1.60	3.00

Tabela A.14: Comparação estatística para a base *Segmentation* quando K = 7.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	947125 (4)	971496 (5)	1027360 (6)	944725 (2.5)	944725 (2.5)	939111 (1)
Y_2	909497 (5)	889877 (4)	946277 (6)	883891 (1)	889482 (3)	888528 (2)
Y_3	2070960 (5)	1396880 (3)	12818506 (6)	1982250 (4)	1270650 (1)	1273890 (2)
Y_4	939971 (1)	1027510 (6)	1002520 (5)	975718 (2)	977849 (3.5)	977849 (3.5)
Y_5	1210390 (5)	1226170 (6)	1152610 (4)	1151790 (3)	1141730 (2)	1139450 (1)
Y_6	1002540 (1)	1052800 (5)	1091420 (6)	1003790 (2)	1031480 (4)	1027440 (3)
Y_7	1065180 (6)	1055840 (4)	1051230 (3)	1058220 (5)	1037600 (1.5)	1037600 (1.5)
Y_8	1439530 (5)	1432330 (4)	1303690 (2)	1444320 (6)	1430450 (3)	1285750 (1)
Y_9	866783 (5)	909587 (6)	862152 (4)	833990 (2)	833990 (2)	833990 (2)
Y_{10}	1207130 (6)	1142620 (5)	1093390 (3)	1120170 (4)	1093200 (1.5)	1093200 (1.5)
p.m.	4.30	4.80	4.50	3.15	2.40	1.85

Para a base *Segmentation*, nos três casos avaliados, o teste de *Teste de Friedman* rejeita a hipótese nula, reportando probabilidade de 0.082%, de 0.014% e de 0.36%, para $K = 6$, $K = 7$ e $K = 8$, respectivamente.

Quando $K = 6$, o teste reporta as seguintes diferenças: melhoria do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* aleatório ($4.90 - 1.60 = 3.30 > 2.38$) e ao

Tabela A.15: Comparação estatística para a base *Segmentation* quando K = 8.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	1557430 (6)	972815 (5)	964237 (4)	898626 (1)	934667 (3)	898854 (2)
Y_2	1207320 (6)	961949 (5)	935195 (4)	886606 (2)	897642 (3)	884033 (1)
Y_3	745855 (1)	777853 (2)	782705 (4)	807860 (6)	782617 (3)	786749 (5)
Y_4	928706 (6)	812106 (1)	914016 (5)	819088 (3.5)	817912 (2)	819088 (3.5)
Y_5	988056 (6)	970598 (5)	917237 (4)	896682 (1.5)	915284 (3)	896682 (1.5)
Y_6	1437470 (6)	1047600 (4)	1053350 (5)	958309 (2)	958309 (2)	958309 (2)
Y_7	840278 (6)	824257 (5)	808217 (4)	802717 (3)	802416 (1.5)	802416 (1.5)
Y_8	816433 (3)	838344 (6)	784697 (1)	821466 (4)	822686 (5)	812399 (2)
Y_9	1423550 (6)	1351440 (3)	1385890 (4)	1417960 (5)	1122020 (1)	1350230 (2)
Y_{10}	1229890 (6)	1206290 (5)	1146940 (2)	1127420 (1)	1164490 (3.5)	1164490 (3.5)
p.m.	5.20	4.10	3.70	2.90	2.70	2.40

K-means++ ($4.20 - 1.60 = 2.60 > 2.38$).

Para $K = 7$, o teste indica as seguintes diferenças: melhoria do algoritmo *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório ($4.30 - 1.85 = 2.45 > 2.38$), ao *K-means* com *PCA_Part* ($4.80 - 1.85 = 2.95 > 2.38$) e ao *K-means++* ($4.50 - 1.85 = 2.65 > 2.38$); melhoria do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* inicializado com o mesmo método ($4.80 - 2.40 = 2.40 > 2.38$).

Quando, $K = 8$, o teste apresenta as seguintes diferenças: melhoria do *Simulated Annealing* inicializado com *PCA_Part* em relação ao *K-means* aleatório ($5.20 - 2.70 = 2.50 > 2.38$); melhoria do *Simulated Annealing* com inicialização usada pelo *K-means++* em relação ao *K-means* aleatório ($5.20 - 2.40 = 2.80 > 2.38$).

As Tabelas A.16, A.17 e A.18 mostram os resultados obtidos com a base *Spam*, quando K = 2, 3 e 4, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

Para essa base, o teste de *Teste de Friedman* rejeita a hipótese nula apenas quando $K = 4$, reportando probabilidade de 16.57%, 10.06% e 3.69%, para $K = 2$, $K = 3$ e $K = 4$, respectivamente. Quando $K = 4$, o teste reporta a seguinte diferença: o *Simulated Annealing* inicializado com o método do *K-means++* apresenta melhores resultados do que o *K-means* aleatório ($5 - 2.55 = 2.45 > 2.38$).

As Tabelas A.19, A.20 e A.21 mostram os resultados obtidos com a base *Pen digit*, quando K = 9, 10 e 11, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

Para a base *Pen digit*, *Teste de Friedman* rejeita a hipótese nula apenas quando $K = 11$, reportando probabilidade de 15.68%, 25.53% e 0.014%, para $K = 9$, $K = 10$ e $K = 11$, re-

Tabela A.16: Comparação estatística para a base *Spam* quando K = 2.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	4.9e+07(3)	4.9e+07 (3)	5.0e+07 (6)	4.9e+07 (3)	4.9e+07 (3)	4.9e+07 (3)
Y_2	9.8e+07 (2.5)	9.8e+07 (2.5)	1.3e+08 (6)	9.8e+07 (2.5)	9.8e+07 (2.5)	1.0e+08 (5)
Y_3	4.1e+07 (2.5)	4.1e+07 (2.5)	4.8e+07 (6)	4.1e+07 (2.5)	4.1e+07 (2.5)	4.3e+07 (5)
Y_4	8.8e+07 (2.5)	8.8e+07 (2.5)	1.0e+08 (5.5)	8.8e+07 (2.5)	8.8e+07 (2.5)	1.0e+08 (5.5)
Y_5	9.3e+07 (3.5)	9.3e+07 (3.5)	9.3e+07 (3.5)	9.3e+07 (3.5)	9.3e+07 (3.5)	9.3e+07 (3.5)
Y_6	5.6e+07 (3.5)	5.6e+07 (3.5)	5.7e+07 (5.5)	5.5e+07 (1.5)	5.5e+07 (1.5)	5.7e+07 (5.5)
Y_7	3.7e+07 (5)	3.7e+07 (5)	3.7e+07 (5)	3.6e+07 (2)	3.6e+07 (2)	3.6e+07 (2)
Y_8	2.1e+08 (4.5)	2.1e+08 (4.5)	8.6e+07 (1.5)	2.1e+08 (4.5)	2.1e+08 (4.5)	8.6e+07 (1.5)
Y_9	1.8e+07 (5)	1.8e+07 (5)	1.8e+07 (5)	9.2+07 (2)	9.2e+07 (2)	9.2e+07 (2)
Y_{10}	4.6e+07 (3.5)	4.6e+07 (3.5)	4.6e+07 (3.5)	4.6e+07 (3.5)	4.6e+07 (3.5)	4.6e+07 (3.5)
p.m.	3.55	3.55	4.75	2.75	2.75	3.65

Tabela A.17: Comparação estatística para a base *Spam* quando K = 3.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	1.8e+07 (3.5)	1.8e+07 (3.5)	1.8e+07 (3.5)	1.8e+07 (3.5)	1.8e+07 (3.5)	1.8e+07 (3.5)
Y_2	4.0e+07 (3.5)	4.0e+07 (3.5)	4.0e+07 (3.5)	4.0e+07 (3.5)	4.0e+07 (3.5)	4.0e+07 (3.5)
Y_3	1.5e+08 (5.5)	1.5e+08 (5.5)	4.9e+07 (1.5)	1.4e+08 (3.5)	1.4e+08 (3.5)	4.9e+07 (1.5)
Y_4	2.4e+07 (5.5)	2.4e+07 (5.5)	2.2e+07 (1.5)	2.3e+07 (3.5)	2.3e+07 (3.5)	2.2e+07 (1.5)
Y_5	1.7e+08 (5.5)	1.7e+08 (5.5)	4.8e+07 (2.5)	4.8e+07 (2.5)	4.8e+07 (2.5)	4.8e+07 (2.5)
Y_6	2.3e+07 (4.5)	2.3e+07 (4.5)	2.0e+07 (1.5)	2.3e+07 (4.5)	2.3e+07 (4.5)	2.0e+07 (1.5)
Y_7	2.2e+07 (4.5)	2.2e+07 (4.5)	2.3e+07 (6)	2.1e+07 (2)	2.1e+07 (2)	2.1e+07 (2)
Y_8	2.4e+07 (3.5)	2.4e+07 (3.5)	2.4e+07 (3.5)	2.4e+07 (3.5)	2.4e+07 (3.5)	2.4e+07 (3.5)
Y_9	1.7e+07 (3.5)	1.7e+07 (3.5)	1.7e+07 (3.5)	1.7e+07 (3.5)	1.7e+07 (3.5)	1.7e+07 (3.5)
Y_{10}	4.9e+07 (4.5)	4.9e+07 (4.5)	4.8e+07 (1.5)	4.9e+07 (4.5)	4.9e+07 (4.5)	4.8e+07 (1.5)
p.m.	4.4	4.4	2.85	3.45	3.45	2.45

Tabela A.18: Comparação estatística para a base *Spam* quando K = 4.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	2.3e+07 (5.5)	1.8e+07 (2)	2.1e+07 (4)	2.3e+07 (5.5)	1.7e+07 (1)	2.0e+07 (3)
Y_2	3.9e+07 (4.5)	3.9e+07 (4.5)	1.7e+07 (1.5)	3.9e+07 (4.5)	3.9e+07 (4.5)	1.7e+07 (1.5)
Y_3	1.4e+07 (6)	1.3e+07 (5)	1.2e+07 (2.5)	1.2e+07 (2.5)	1.2e+07 (2.5)	1.2e+07 (2.5)
Y_4	6.8e+07 (5.5)	6.8e+07 (5.5)	5.2e+07 (2.5)	5.2e+07 (2.5)	5.2e+07 (2.5)	5.2e+07 (2.5)
Y_5	2.7e+07 (5.5)	2.7e+07 (5.5)	2.3+07 (2)	2.6e+07 (3.5)	2.6e+07 (3.5)	2.2e+07 (1)
Y_6	2.2e+07 (5.5)	2.2e+07 (5.5)	1.8e+07 (2)	2.1e+07 (3.5)	2.1e+07 (3.5)	1.6e+07 (1)
Y_7	1.5e+07 (3.5)	1.5e+07 (3.5)	1.5e+07 (3.5)	1.5e+07 (3.5)	1.5e+07 (3.5)	1.5e+07 (3.5)
Y_8	1.5e+07 (4)	1.4e+07 (2)	1.7e+07 (6)	1.4e+07 (2)	1.4e+07 (2)	1.6e+07 (5)
Y_9	1.1e+07 (5)	1.1e+07 (5)	1.1e+07 (5)	1.0e+07 (2)	1.0e+07 (2)	1.0e+07 (2)
Y_{10}	1.7e+07 (5)	1.4e+07 (1)	1.5e+07 (2)	1.8e+07 (6)	1.6e+07 (3.5)	1.6e+07 (3.5)
p.m.	5	3.95	3.1	3.55	2.85	2.55

Tabela A.19: Comparação estatística para a base *Pen digit* quando K = 9.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	6024850 (6)	5453850 (4)	5461460 (5)	5449290 (2)	5452000 (3)	5369970 (1)
Y_2	5724460 (6)	5622550 (5)	5599820 (4)	5376060 (3)	5375850 (2)	5315180 (1)
Y_3	5128080 (1)	5162480 (5)	5369790 (6)	5161150 (2.5)	5161500 (4)	5161150 (2.5)
Y_4	4985740 (1)	5051580 (4)	5239100 (6)	5050010 (3)	5049410 (2)	5225240 (5)
Y_5	5633780 (6)	5294890 (4)	5410390 (5)	5289750 (1)	5292840 (2)	5294030 (3)
Y_6	5185930 (3)	5145310 (2)	5076850 (1)	5265510 (4.5)	5265510 (4.5)	5620610 (6)
Y_7	5387770 (5)	5254040 (1)	5482930 (6)	5345650 (4)	5328090 (3)	5327990 (2)
Y_8	5888050 (5)	5274090 (3)	5307830 (4)	5960520 (6)	5205120 (1.5)	5205120 (1.5)
Y_9	5515310 (5)	5540580 (6)	5337060 (1)	5368540 (2)	5442940 (3.5)	5442940 (3.5)
Y_{10}	5405650 (5)	5371010 (3)	5531060 (6)	5280200 (2)	5257750 (1)	5376990 (4)
p.m.	4.3	3.7	4.4	3	2.65	2.95

Tabela A.20: Comparação estatística para a base *Pen digit* quando K = 10.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	4931000 (2)	4994200 (4)	5158090 (6)	4897460 (1)	4989930 (3)	5155010 (5)
Y_2	4785710 (2)	4825940 (4)	4974310 (6)	4785670 (1)	4825880 (3)	4938460 (5)
Y_3	5258560 (4)	5304860 (6)	4926370 (2)	5000320 (3)	5302500 (5)	4925590 (1)
Y_4	5078700 (5)	4958160 (2)	5006510 (4)	5093620 (6)	4948760 (1)	5002380 (3)
Y_5	5225640 (6)	4971480 (4)	4971570 (5)	4968820 (2)	4968820 (2)	4968820 (2)
Y_6	5122520 (4)	5187530 (6)	5054280 (3)	5052610 (2)	5185200 (5)	5044410 (1)
Y_7	5022450 (5)	4993670 (3)	4912480 (1)	5000680 (4)	4992840 (2)	5126200 (6)
Y_8	5040010 (4)	5051450 (5)	4955280 (2)	4800050 (1)	5081230 (6)	4961950 (3)
Y_9	5029800 (3)	5031970 (5)	5030590 (4)	5027420 (1.5)	5211990 (6)	5027420 (1.5)
Y_{10}	4757850 (1)	4861870 (5)	4897620 (6)	4861070 (3.5)	4861070 (3.5)	4761210 (2)
p.m.	3.6	4.4	3.9	2.5	3.65	2.95

Tabela A.21: Comparação estatística para a base *Pen digit* quando K = 11.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	5011270 (6)	4721210 (3)	4773720 (4)	4647290 (2)	4533710 (1)	4818420 (5)
Y_2	4806420 (5)	4814260 (6)	4769400 (2)	4682040 (1)	4804330 (4)	4783070 (3)
Y_3	4911350 (5)	4702910 (4)	4953150 (6)	4693890 (1)	4701170 (3)	4698530 (2)
Y_4	5249060 (6)	4814660 (2)	5061520 (5)	4795650 (1)	4930500 (3)	4965900 (4)
Y_5	4805740 (5)	4613520 (2)	4725880 (4)	4973930 (6)	4610070 (1)	4723410 (3)
Y_6	4701920 (6)	4617520 (5)	4587370 (3)	4612830 (4)	4554570 (1.5)	4554570 (1.5)
Y_7	4872920 (5)	4898980 (6)	4720030 (1)	4747310 (2.5)	4747310 (2.5)	4855950 (4)
Y_8	4803510 (4)	4814890 (5)	4962300 (6)	4549690 (1)	4793490 (3)	4718240 (2)
Y_9	5128270 (6)	4525780 (3)	4758480 (5)	4523240 (1.5)	4523240 (1.5)	4714110 (4)
Y_{10}	4844870 (5)	4742440 (4)	4863230 (6)	4733880 (2.5)	4733880 (2.5)	4670700 (1)
p.m.	5.3	4	4.2	2.25	2.3	2.95

spectivamente. Quando $K = 11$, o teste reporta as seguintes diferenças: o *Simulated Annealing* aleatório apresenta melhores resultados do que o *K-means* aleatório ($5.3 - 2.25 = 3.05 > 2.38$); o *Simulated Annealing* inicializado com o método *PCA_Part* apresenta melhores resultados do que o *K-means* aleatório ($5.3 - 2.3 = 3 > 2.38$).

As Tabelas A.22, A.23 e A.24 mostram os resultados obtidos com a base *Letter*, quando K = 25, 26 e 27, respectivamente. Essas tabelas seguem o mesmo modelo das tabelas anteriores.

Tabela A.22: Comparação estatística para a base *Letter* quando K = 25.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	61626.1 (2)	61168.1 (1)	62944 (4)	69467.5 (6)	65130.6 (5)	62186.2 (3)
Y_2	60638.1 (4)	60765.9 (5)	62463 (6)	60510.8 (2)	60506.9 (1)	60608.9 (3)
Y_3	62039.5 (3)	61619.6 (1)	62411.4 (5)	62206.2 (4)	66859.6 (6)	61648.8 (2)
Y_4	62225.7 (2)	61332.1 (1)	62406.5 (3)	74070.2 (6)	66305.4 (4)	70335.7 (5)
Y_5	61791.9 (3)	62741.1 (6)	62501 (5)	61132.6 (1)	62065.4 (4)	61615.7 (2)
Y_6	62084.8 (4)	62214 (6)	62111.7 (5)	61009.9 (1)	61297.1 (2)	61718.3 (3)
Y_7	63066 (5)	61799.1 (2)	63385.5 (6)	61688.3 (1)	61994.4 (4)	61866.6 (3)
Y_8	62643.6 (3)	62311.8 (2)	62302.9 (1)	73341 (6)	66200.3 (4)	69850.2 (5)
Y_9	61893.1 (2)	62175.4 (4)	62366.9 (5)	61873.2 (1)	62607.1 (6)	61961.4 (3)
Y_{10}	61790.2 (3)	62739.6 (6)	62500.2 (5)	61133.1 (1)	62064.6 (4)	61616.2 (2)
p.m.	3.1	3.4	4.5	2.9	4	3.1

Tabela A.23: Comparação estatística para a base *Letter* quando K = 26.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	62443.3 (4)	61563.6 (2)	61319.9 (1)	61611 (3)	65434 (5)	69652 (6)
Y_2	61466.5 (2)	60408.1 (1)	61831.1 (3)	72186.5 (5)	65246.4 (4)	72780.8 (6)
Y_3	62512.4 (4)	61837.4 (1)	62314 (3)	62223.3 (2)	65286.5 (5)	72912.6 (6)
Y_4	59367.2 (1)	60074.3 (4)	59651.7 (2)	59834.6 (3)	63265.3 (5)	68992.2 (6)
Y_5	62699 (4)	61972.2 (1)	62951.8 (5)	62613.5 (3)	66168.7 (6)	62542 (2)
Y_6	59971.8 (1)	60838.2 (2)	60953.3 (3)	70298.2 (5)	64461.3 (4)	70584.3 (6)
Y_7	60255 (3)	59943.6 (1)	62391.3 (5)	60011.9 (2)	65233.2 (6)	60399.4 (4)
Y_8	60211.6 (1)	60408.1 (3)	60801.1 (4)	60313.2 (2)	63993.9 (5)	69260.3 (6)
Y_9	61588.5 (6)	60984.7 (3)	60537.4 (1)	60650.5 (2)	61250 (4)	61359.7 (5)
Y_{10}	62956.3 (3)	61761.6 (1)	62446.1 (2)	62998.6 (4)	66877 (5)	70284.2 (6)
p.m.	2.90	1.90	2.90	3.10	4.90	5.30

Para essa base, o teste de *Teste de Friedman* rejeita a hipótese nula apenas quando $K = 26$, reportando probabilidade de 36.24%, de 7.2e-04% e de 31.21%, para $K = 25$, $K = 26$ e $K = 27$, respectivamente. Quando $K = 26$, o teste reporta as seguintes diferenças: o *K-means* aleatório apresenta melhores resultados do que o *Simulated Annealing* inicializado com o método do *K-means++* ($5.30 - 2.90 = 2.40 > 2.38$); o *K-means* inicializado com *PCA_Part* apresenta um melhor desempenho em relação ao *Simulated Annealing* inicializado com *PCA_Part* ($4.90 -$

Tabela A.24: Comparação estatística para a base *Letter* quando K = 27.

	K-means			*S.A.*		
	Aleat.	*PCA_Part*	*Kmeans++*	*Aleat.*	*PCA_Part*	*Kmeans++*
Y_1	60005.2 (6)	59208.2 (2)	58952.5 (1)	59241.8 (3)	59998.9 (5)	59967.8 (4)
Y_2	61582.4 (3)	60179.2 (1)	60279.2 (2)	74000.9 (6)	65229.2 (5)	61833.6 (4)
Y_3	60347.7 (6)	59710.3 (4)	59279.3 (3)	59011.4 (1)	59238.4 (2)	59820.7 (5)
Y_4	59706 (1)	59908 (2)	61179.5 (4)	60039 (3)	64391.5 (5)	68679.3 (6)
Y_5	58903.9 (1)	58915.5 (2)	60042.3 (5)	59138 (3)	62609 (6)	59674.2 (4)
Y_6	60472.3 (3)	60483.8 (4)	61824.5 (6)	59523.8 (1)	61414.6 (5)	60378.8 (2)
Y_7	60353.3 (4)	61253.8 (6)	59508.8 (1)	60188.1 (3)	60521.5 (5)	59955.7 (2)
Y_8	60422.2 (3)	60298.7 (2)	59890.6 (1)	72259.6 (6)	64468.8 (5)	61769 (4)
Y_9	62183.4 (6)	60578.7 (4)	61809 (5)	60484.6 (3)	60237.7 (2)	60149.8 (1)
Y_{10}	61119.7 (4)	59993.8 (1)	61099.2 (3)	60396.8 (2)	63622.5 (6)	61321.5 (5)
p.m.	3.70	2.80	3.10	3.10	4.60	3.70

$1.90 = 3.00 > 2.38$) e com o método de inicialização do *K-means++* ($5.30 - 1.90 = 3.40 > 2.38$) ; o algoritmo *K-means++* apresenta melhores resultados do que o *Simulated Annealing* com inicialização usada pelo *K-means++* ($5.30 - 2.90 = 2.40 > 2.38$).

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo