ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS APLICADO AO PROBLEMA DE AGRUPAMENTO COM SOMA MÍNIMA

José André de Moura Brito
Escola Nacional de Ciências – ENCE/IBGE
Rua André Cavalcanti, 106, Bairro de Fatima – Centro – Rio de Janeiro- RJ
jambrito@gmail.com

Augusto César Fadel
Instituto Brasileiro de Geografia e Estatística – IBGE
Avenida República do Chile, 500 – Centro – Rio de Janeiro-RJ
augustofadel@gmail.com

Gustavo Silva Semaan
Instituto do Noroeste Fluminense de Educação Superior(INFES)da Universidade Federal Fluminense(UFF)
Av. João Jasbick, s/nº - Aeroporto - Santo Antônio de Pádua -RJ
gustavosemaan@gmail.com

Luciana Roque Brito
Centro Universitário Anhanguera de Niterói - UNIAN
Rua Visconde do Rio Branco, 137 – Centro – Niterói - RJ
lubritoroque@gmail.com

RESUMO

Este artigo propõe um algoritmo heurístico para resolução do problema de agrupamento com soma mínima de distâncias. Neste problema, deve-se alocar os *n* objetos de uma base de dados em *k* grupos, de forma que o somatório das distâncias entre todos os pares de objetos, dentro de cada um dos grupos, seja mínimo. Assim como em outros problemas de agrupamento, este problema é de alta complexidade computacional. Destarte, com o objetivo de produzir soluções de boa qualidade em um tempo computacional viável, foi desenvolvido um algoritmo baseado na metaheurística BRKGA (Biased Random-Key Genetic Algorithms). De forma a avaliar a eficácia deste algoritmo, frente a três algoritmos propostos na literatura e a uma formulação de programação inteira, apresenta-se, no final deste trabalho, um conjunto de resultados computacionais obtidos a partir da aplicação dos quatro algoritmos em 32 bases dados da literatura e artificiais.

PALAVRAS CHAVE. Agrupamento, Soma Mínima, Metaheurísticas.

Tópicos MH, PM e OC

ABSTRACT

This paper presents a heuristic algorithm to solve the minimum sum of distances clustering problem. In this problem, one must allocate *n* objects of a dataset in *k* groups, so that the sum of distances between all pairs of objects within each group is minimal. As with other clustering problems, this problem is computationally expensive. Thus, in order to produce good quality solutions in feasible time, an algorithm based on the BRKGA (Biased Random-Key Genetic Algorithms) metaheuristic was developed. In order to evaluate the effectiveness of this algorithm, three algorithms proposed in the literature and an integer programming formulation was considered. Besides, a set of computational results obtained from the application of the four algorithms in 32 datasets is presented.

KEYWORDS. Clustering, Minimum Sum, Metaheuristics.
Topics MH, MP and OC

1. Introdução

Atualmente, a análise de agrupamentos vem sendo amplamente utilizada nas mais diversas áreas do conhecimento [Hair et al. 2009], [Kaufman and Rousseeuw 1989] como, por exemplo, na Medicina, Inteligência Artificial, Biologia, Estatística, Economia, Sociologia etc. É uma técnica de análise multivariada que agrega um conjunto de métodos que têm, por objetivo, agrupar os objetos (registros) de uma base dados em grupos, de forma que estes grupos possuam, internamente, alto grau de homogeneidade e, entre si, baixo grau de homogeneidade.

Esta homogeneidade, por sua vez, é função das q variáveis (características ou atributos) associadas aos n objetos, bem como da métrica (distância) e da função objetivo adotadas. Em geral, ao definir a função objetivo, fica definido, consequentemente, o problema de agrupamento a ser abordado. Ainda neste sentido, em [Hansen and Jaumard 1997] e [Cruz 2010] são apresentadas várias funções objetivo adotadas em problemas de agrupamento.

Cabe observar que, independentemente da métrica e da função objetivo consideradas, os problemas de agrupamento são, em geral, de difícil solução computacional [Hansen and Jaumard 1997] [Cruz 2010] [Naldi 2011] [Semaan 2013]. Mais especificamente, à medida que o número de objetos da base de dados aumenta, mais difícil torna-se a obtenção do ótimo global para esses problemas, seja através de métodos exatos ou através de algoritmo heurísticos.

Em particular, neste trabalho será abordado o problema de agrupamento com soma mínima de distâncias [Friedman and Meulman 2004], [Brito e Brito 2008], [Nascimento, Toledo e de Carvalho 2010], [Serpa 2011]. Dada uma base de dados composta por n objetos (registros), com q atributos, e definida uma métrica, deve-se distribuir os n objetos em k grupos, de forma que a soma total das distâncias entre os objetos, tomados dois a dois, dentro de cada um dos grupos, seja mínima. Doravante, denotaremos este problema por PASMD.

Em [Freitas et al. 2007] e [Brito, Montenegro e Freitas 2011] encontra-se uma importante aplicação deste problema, associada ao uso de amostragem probabilística [Bolfarine e Bussab 2005]. Mais especificamente, os autores propõem um plano amostral que, tem por base, aplicação de uma amostragem estratificada. Isso implica, por sua vez, a definição de estratos (grupos) constituídos por setores censitários (objetos). A homogeneidade dos estratos é avaliada a partir da função objetivo considerada no PASMD.

Posto isso, de forma a resolver este problema, o presente trabalho traz a proposta de um algoritmo heurístico que foi desenvolvido a partir do estudo da metaheurística algoritmos genéticos de chaves aleatórias viciadas (do inglês, *Biased Random-key Genetic Algorithm* - BRKGA) [Gonçalves and Resende 2011], [Resende 2013a]. Além da introdução, este trabalho está dividido em mais cinco seções. Na seção dois são apresentados os principais conceitos da análise de agrupamentos. A seção três traz uma descrição do problema de agrupamento com soma mínima de distâncias (PASMD), sendo apresentada, inclusive, uma formulação de programação inteira mista para este problema. A seção quatro apresenta a metaheurística BRKGA e a seção cinco traz uma descrição detalhada do algoritmo proposto. Finalmente, na seção seis, são apresentadas informações das bases de dados utilizadas e os resultados e as análises realizadas a partir dos experimentos computacionais, considerando a aplicação no novo algoritmo, de três algoritmos da literatura e de uma formulação de programação inteira mista, também utilizada em [Brito e Brito 2008], [Nascimento, Toledo e de Carvalho 2010].

2. Análise de Agrupamentos

A análise de agrupamentos é uma técnica de análise multivariada [Hair et al. 2009], [Kaufman and Rousseeuw 1989] que agrega um conjunto de métodos que têm, por objetivo, agrupar os n objetos de uma base dados em k grupos.

Segundo [Hair et al. 2009]: “A análise de agrupamentos classifica objetos de modo que cada objeto é semelhante aos outros no agrupamento com base em um conjunto de características escolhidas. Os agrupamentos resultantes de objetos devem exibir elevada homogeneidade interna (dentro dos agrupamentos) e elevada heterogeneidade externa (entre agrupamentos). ”

Os métodos associados à análise de agrupamentos são aplicados nas mais variadas áreas do conhecimento [Hair et al. 2009] como, por exemplo: Estatística, Inteligência Artificial, Economia, Marketing, Sociologia, Medicina etc.

As inúmeras aplicações desta técnica têm motivado, nos últimos tempos, o estudo e desenvolvimento de novos métodos, como pode ser visto, por exemplo, em: [Friedman and Meulman, 2004], [Cruz, 2010], [Naldi, 2011], [Semaan, 2013], [Carrizosa, Mladenovicb and Todosijevicc, 2013], [Santia, Aloise and Blanchardc, 2016]. Não obstante, os estudos e propostas de novos métodos de análise de agrupamentos convergem, em geral, para as limitações computacionais quanto à qualidade das soluções produzidas e ao tempo computacional necessário para a obtenção de tais soluções.

Em linhas gerais, estes métodos são concebidos com vistas à resolução do problema de agrupamento clássico: Dado um conjunto X formado por n objetos, $X = \{x_1, x_2,..., x_n\}$ com q atributos, tal que $x_i=(x_{i1},x_{i2},...,x_{iq})$, e sendo o número de grupos igual a um inteiro k, deve-se construir k grupos $C_1, C_2,..., C_k$, ou seja, uma partição Π de X (solução de agrupamento) tal que $\Pi = \{C_1,C_2,...,C_k\}$ e sejam observadas as três restrições a seguir:

$$C_1 \cup C_2 \cup ... \cup C_k = X \quad (1)$$

$$C_i \cap C_j = \varnothing \ , \ i,j=1,...,k \ \ (i<j) \quad (2)$$

$$|C_i| \geq 1 \ , \ i=1,...,k \quad (3)$$

A restrição (1) indica que a união dos grupos corresponde ao conjunto X, a restrição (2) indica que um objeto pertence a exatamente um grupo e a restrição (3) garante que cada grupo tem pelo menos um objeto.

A definição da alocação dos objetos aos grupos depende da métrica utilizada (medida de distância), da função objetivo $f(.)$ considerada para a avaliação da homogeneidade dos grupos, caracterizando, consequentemente, a solução ótima. Ainda neste sentido, define-se como solução ótima (agrupamento ótimo) aquela que tem o menor valor de função objetivo associado.

Não obstante, a determinação do agrupamento ótimo, ou seja, da partição Π que produz os grupos mais homogêneos, segundo a função objetivo definida a priori, consiste em uma tarefa árdua do ponto vista computacional. Esta afirmação é explicada, tendo em vista que o número de partições, ou seja, soluções possíveis para este problema é impactado, diretamente, pelo número de objetos da base de dados associada à aplicação. Mais especificamente, o número de soluções possíveis para o problema de agrupamento definido acima pode ser calculado a partir do número de *Stirling* de segundo tipo [Johnson and Wichern 2002], dado pela equação abaixo:

$$\frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n \quad (4)$$

Considerando, por exemplo, que n=25 objetos serão alocados em dois grupos (k=2), o número de soluções a serem consideradas é de 16.777.215. Mas, mantendo o mesmo número de grupos, e apenas dobrando o número de objetos, há mais de meio trilhão de soluções possíveis. Ao considerar uma quantidade n maior de objetos, esses valores crescem exponencialmente. Este fato torna inviável a aplicação de um método enumeração exaustiva que avalie todas soluções deste problema, e determine a solução ótima. Ainda em relação a esta questão, em alguns problemas de agrupamento como os descritos em [Vinod 1969], [Rao 1971], [Hansen and Jaumard 1997], [Cruz 2010], [Brito, Semaan e Brito 2015] é possível utilizar formulações de programação matemática.

Não obstante, em geral, a utilização dessas formulações só é viável para problemas de pequeno porte (n<100), tendo em vista que o número de variáveis e/ou restrições cresce substancialmente à medida que o número de objetos aumenta. Por sua vez, isto impacta diretamente no tempo de processamento necessário para resolver a formulação e produzir o ótimo, mediante a utilização de algum *solver* de otimização.

Destarte, considerando a complexidade intrínseca a estes problemas, busca-se, através de algoritmos heurísticos, a produção de soluções viáveis, no que diz respeito ao tempo

computacional, e de boa qualidade, em relação à coesão dos objetos e separação dos grupos, quando comparado ao tempo consumido por uma formulação de programação matemática.

Dependendo do problema de agrupamento abordado, mais especificamente, da função objetivo considerada, essas soluções podem ser produzidas mediante a aplicação de um dos métodos clássicos de agrupamento disponíveis na literatura, quais sejam: hierárquico ou não hierárquico [Mingoti 2007], [Hair et al. 2009].

Sem incorporar a aplicação de procedimentos de busca local sofisticados, esses métodos produzem soluções (grupos) de qualidade razoável, sem examinar todas as soluções possíveis (partições). Os métodos hierárquicos dividem-se em aglomerativos e divisivos. Os métodos não hierárquicos procuram encontrar uma partição viável dos n objetos sem a necessidade de associações hierárquicas. Dentre os métodos não hierárquicos disponíveis na literatura, os mais utilizados são o k-means [Mingoti 2007] e o dos k-medoids [Kaufman and Rousseeuw 1989].

Há vários problemas de agrupamento que têm funções objetivo e/ou restrições específicas, como no caso PASMD. No que concerne a esta classe de problemas, os métodos supracitados não são aplicáveis. Nestes casos, há a necessidade do desenvolvimento de algoritmos heurísticos que, em geral, são concebidos à luz do estudo das metaheurísticas como GRASP, VNS, Algoritmos Genéticos etc. Em [Ahmadia and Osmanb 2005], [Brito e Brito 2008], [Nascimento, Toledo e de Carvalho 2010], [Cruz 2010], [Carrizosa, Mladenovicb and Todosijevicc 2013], [Santia, Aloise and Blanchardc 2016], [Oliveira, Chaves e Lorena, 2016] e [Oliveira, Chaves e Lorena, 2017] são apresentados alguns exemplos desses problemas.

3. Problema de Agrupamento com Soma Mínima de Distâncias (PASMD)

Considere o conjunto X constituído por n objetos $X=\{x_1,x_2,...,x_i,...,x_n\}$, com q variáveis, tal que $x_i=(x_{i1},x_{i2},...,x_{iq})$ e que a distância d_{ij} entre dois objetos x_i e x_j quaisquer seja definida, por exemplo, como a distância euclidiana, conforme equação (5) abaixo. Suponha, ainda, que o número de grupos k seja fixado a priori.

$$d_{ij} = \sqrt{\sum_{r=1}^{q} (x_{ir} - x_{jr})^2} \tag{5}$$

Tendo em vista as definições acima, no PASMD, deve-se alocar os n objetos em k grupos, denotados por $C_1,C_2,...,C_k$, de forma que a soma total das distâncias d_{ij} entre todos os objetos, tomados dois a dois, dentro de cada um dos grupos, seja mínima. Ou seja, busca-se minimizar a seguinte função objetivo:

$$\min f = \sum_{g=1}^{k} \sum_{\forall x_i, x_j \in C_g} d_{ij} \tag{6}$$

O PASMD pode ser exemplificado utilizando um grafo completo $G=(V,A)$. Neste grafo, cada vértice $v_i \in V$ $(i=1,...,n)$ corresponde a um objeto e as arestas $(v_i,v_j) \in A$ têm pesos correspondentes às distâncias d_{ij} entre os todos vértices de G. Assim, resolver o PASMD é equivalente a determinar, a partir de G, k subgrafos completos (cliques) cuja soma dos pesos das arestas seja mínima. A Figura 1 ilustra uma possível solução para o PASMD, supondo um conjunto X com 7 objetos (n=7) e k=2 e que C_1 está associado a uma clique de tamanho 4 e C_2 está associado a uma clique de tamanho 3.

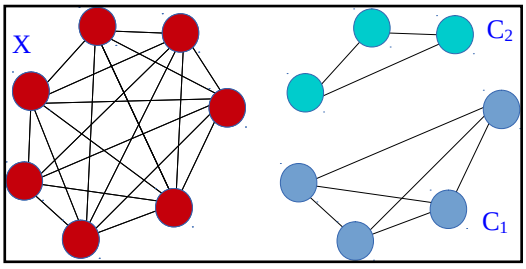

Figura 1- Grafo e subgrafos completos (cliques)

O PASMD também pode ser resolvido considerando a aplicação da formulação de programação inteira mista proposta nos trabalhos de [Brito e Brito, 2008] e [Nascimento, Toledo e de Carvalho 2010].

$$Minimizar \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_{ij} \quad (7)$$

Sujeito a

$$\sum_{g=1}^{k} y_i^g = 1,\ i = 1,..,n \quad (8)$$

$$\sum_{i=1}^{n} y_i^g \geq 1,\ g = 1,...,k \quad (9)$$

$$y_i^g + y_j^g - 1 \leq x_{ij},\ g = 1,...,k, i = 1,...,n\text{-}1, j = i+1,...,n \quad (10)$$

$$x_{ij} \geq 0\ ,\ i = 1,...,n\text{-}1, j = i+1,...,n \quad (11)$$

$$y_i^g \in \{0,1\}\ ,\ i = 1,...,n, g = 1,...,k \quad (12)$$

A função minimizada em (7) corresponde à função objetivo definida na equação (6). A variável binária y_i^g assume valor 1 quando o objeto *i* é alocado ao grupo C_g e zero caso contrário. A restrição (8) garante que cada objeto *i* deve ser alocado a exatamente um dos grupos C_g *(g=1,...,k)*, a restrição (9) certifica que cada grupo C_g tem, pelo menos, um objeto. As restrições (10) e (11) garantem que x_{ij} assume valor um se, simultaneamente, y_i^g e y_j^g assumem valor 1, ou seja, se os objetos *i* e *j* estão alocados a um mesmo grupo.

A referida formulação de programação inteira tem um total de $(n^2\text{-}n)/2+n.k$ variáveis e $[(n+k)+n.k+(k+1).(n\text{-}1).n/2]$ restrições. O substancial número de variáveis e restrições desta formulação limita a sua aplicação a instâncias de pequeno porte, ou seja, bases de dados com poucos objetos (da ordem centenas). Ainda assim, mesmo considerando este tipo de instância, e um tempo de processamento razoável (da ordem de horas), o máximo que se obtém para o PASMD, a partir da aplicação formulação, são soluções viáveis. Por sua vez, em muitos casos, estas soluções não são de boa qualidade, no que diz respeito à coesão dos objetos similares e separação dos grupos, quando comparadas às soluções produzidas por algoritmos heurísticos.

Esta observação é ratificada nos trabalhos de [Brito e Brito 2008], [Nascimento, Toledo e de Carvalho 2010] e [Serpa 2011]. Além da formulação acima, os três trabalhos trazem propostas de utilização de algoritmos baseados em metaheurísticas para a resolução do PASMD. Em [Brito e Brito, 2008]são propostos um algoritmo genético e um algoritmo VNS, o trabalho de [Nascimento, Toledo e de Carvalho 2010] traz a proposta de um algoritmo baseado na metaheurística GRASP e o trabalho de [Serpa 2011] traz a proposta de três algoritmos heurísticos associados, respectivamente, às metaheurísticas VNS, ILS e CS (*Clustering Search*). Em particular, as soluções produzidas pelos algoritmos propostos em [Brito e Brito 2008] e [Nascimento, Toledo e de Carvalho 2010] foram superiores à da formulação.

4. Algoritmo Genético de Chaves Aleatórias Viciadas

O algoritmo genético de chaves aleatórias viciadas (do inglês, *Biased Random-Key Genetic Algorithm* - BRKGA), proposto por [Gonçalves and Resende 2011], corresponde a uma variação do algoritmo genético de chaves aleatórias (RKGA) que foi introduzido por [Bean 1994]. Em um BRKGA os cromossomos (soluções) são representados por uma string ou um vetor cujos valores são obtidos a partir de valores reais selecionados aleatoriamente, segundo uma distribuição uniforme [0,1]. Após cada geração, aplica-se em cada um desses cromossomos um procedimento denominado decodificador. O decodificador, por sua vez, tem por finalidade estabelecer um mapeamento entre os valores dos cromossomos e as possíveis soluções viáveis para o problema de otimização para o qual a função objetivo deve ser computada. Destarte, o procedimento decodificador é específico para cada problema de otimização ao qual é aplicado o BRKGA.

Um BRKGA evolui a partir de uma população inicial de vetores de chaves aleatórias e evolui para novas populações, ou seja, gerando novos cromossomos durante certo número de gerações. A população inicial é constituída por p vetores de chaves aleatórias gerados conforme descrito anteriormente. Após a aplicação do decodificador e o cálculo da função objetivo, a população é particionada em dois conjuntos, a saber: um pequeno conjunto formado por (p_e) soluções elite, ou seja, formado pelas melhores soluções segundo o valor da função objetivo, e um conjunto formado pelas $(p\text{-}p_e)$ soluções restantes ou não elite, sendo $p_e<p\text{-}p_e$. Para atualizar a população, uma nova geração de cromossomos deve ser produzida. O BRKGA usa uma estratégia de elitismo, tendo em vista que todos os p_e cromossomos pertencentes ao conjunto elite na geração m são copiados para a população da geração $m+1$. A adoção dessa estratégia tende a produzir soluções viáveis cada vez melhores durante as gerações do algoritmo.

Em um BRKGA a mutação é implementada mediante a introdução na população de cromossomos chamados de mutantes, que nada mais são do que vetores de chaves aleatórias gerados de forma equivalente aos vetores da população inicial. Em cada geração um pequeno quantitativo (p_m) de soluções mutantes são introduzidas na população, e assim como as demais soluções, essas podem ser decodificadas em soluções viáveis para o problema.

No que diz respeito ao cruzamento, os $p\text{-}p_e\text{-}p_m$ cromossomos filhos gerados a partir da aplicação do cruzamento uniforme [Spears and De Jong 1991] são produzidos tomando uma solução do conjunto elite e uma solução do conjunto não elite. A Figura 2 ilustra a evolução de uma população de uma geração m para a geração $m+1$. Uma vez que $p_e<p\text{-}p_e$, a probabilidade de um cromossomo do conjunto elite ser selecionado para cruzamento é $1/p_e$, que é, maior que $1/(p\text{-}p_e)$. Isso possibilita que um elemento de conjunto elite tenha uma chance maior de passar as suas características para as gerações futuras do que um elemento do conjunto não elite.

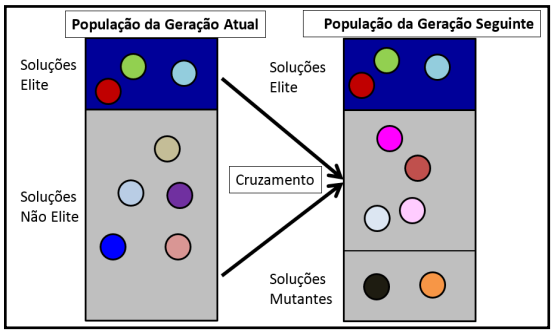

Figura 2 - Representação do BRKGA com a transição entre a geração m e a geração m+1.
(Baseada em figura utilizada na palestra de Maurício Resende [Resende 2013b]).

No cruzamento uniforme um cromossomo S_1 é selecionado do conjunto elite e um cromossomo S_2 é selecionado do conjunto não elite. Uma vez selecionados esses cromossomos, é gerado um vetor auxiliar $(\boldsymbol{v_a})$ com valores reais selecionados aletoriamente do intervalo $[0,1]$ e de igual tamanho a S_1 e S_2. Também é definido o valor de $\rho_e>0.5$ correspondentes à probabilidade de um gene de S_1 compor o cromossomo filho (S_f). Cada valor de v_a é comparado com ρ_e tal que, se o valor na i-ésima posição de v_a for menor ou igual a ρ_e (neste exemplo $\rho_e= 0.7$), temos que a i-ésima posição do cromossomo filho herda o valor da i-ésima posição de S_1. Em caso contrário, a i-ésima posição do cromossomo filho herda o valor da i-ésima posição de S_2.

Uma vez construída a população da geração seguinte, isto é, quando há p cromossomos, distribuídos em cromossomos elite, mutantes e filhos, aplica-se o decodificador e calcula-se os valores da função objetivo para todos os novos vetores mutantes e filhos. E, dando continuidade ao processo, a população é novamente particionada em um conjunto elite e não elite para iniciar uma nova geração. A Figura 3 ilustra todos os passos considerados à aplicação do BRKGA.

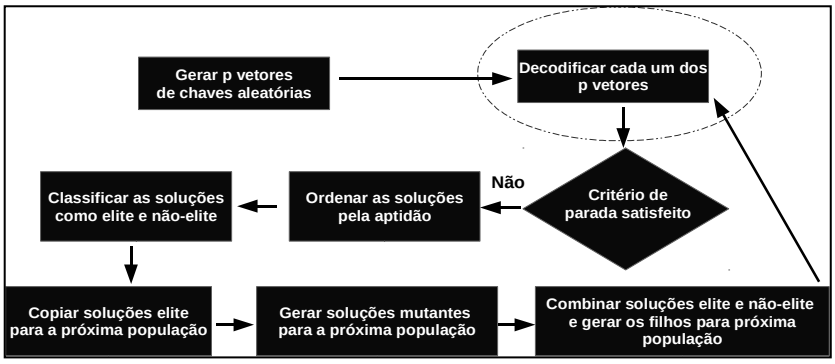

Figura 3 – Passos do BRKGA

(Baseada em figura do trabalho de [Gonçalves and Resende 2011])

Pela figura 3, observa-se que todos os passos do BRKGA, a menos da decodificação, independem do problema de otimização abordado. Mais especificamente, deve-se definir qual será a representação dos cromossomos e o procedimento decodificador. Neste sentido, são apresentados no trabalho de Gonçalves e Resende [2011] vários exemplos de problemas de otimização, comentando a representação adotada para os cromossomos, bem como o procedimento decodificador que foi implementado no algoritmo BRKGA proposto para resolver o problema, e os resultados dos experimentos com o algoritmo. Em particular, os trabalhos de [Oliveira, Chaves e Lorena, 2016] e [Oliveira, Chaves e Lorena, 2017] apresentam algoritmos heurísticos, baseados no BRKGA, que são aplicados a dois problemas de agrupamento.

5. Algoritmo BRKGA Proposto – Representação, Decodificação, Cruzamento e Cálculo da Função Objetivo

Em função do que foi exposto na Seção 4, a implementação do BRKGA para um problema de otimização requer, basicamente, a especificação da representação dos cromossomos e do decodificador que produz as soluções viáveis.

Assim sendo, no que diz respeito ao PASMD, cada cromossomo foi definido como um vetor $\boldsymbol{v}$ com n posições (número de objetos) preenchidas com valores reais gerados uniformemente no intervalo [0,1], sendo a i-ésima posição de $\boldsymbol{v}$ correspondente ao número do objeto x_i. O decodificador definido neste trabalho atribui a um vetor $\boldsymbol{w}$, os valores de $\boldsymbol{v}$ ordenados crescentemente. Em um passo seguinte, os valores de $\boldsymbol{w}$ são pesquisados em $\boldsymbol{v}$, retornando-se as posições que esses valores ocupam em $\boldsymbol{v}$ a um terceiro vetor $\boldsymbol{z}$.

As k primeiras posições de $\boldsymbol{z}$ contêm os objetos que serão utilizados para formação dos k grupos iniciais $(C_1, C_2,..., C_k)$, ou seja, o objeto na 1ª posição de $\boldsymbol{z}$ (objeto para o qual foi atribuído o menor valor de $\boldsymbol{v}$) será alocado ao grupo C_1, o objeto na 2ª posição de $\boldsymbol{z}$ será alocado ao grupo C_2, e assim sucessivamente, com a alocação do objeto na k-ésima posição de $\boldsymbol{z}$ ao grupo C_k.

Além disso, a ordem que os demais $(n-k)$ objetos ocupam entre posições de $(k+1)$ até n em $\boldsymbol{z}$ determinará a sua sequência de avaliação e alocação a um grupo. Mais especificamente, o objeto na posição $(k+1)$ de $\boldsymbol{z}$ será alocado ao grupo C_g $(g=1,...,k)$ mais próximo, ou seja, cuja soma das suas distâncias aos demais objetos desse grupo seja mínima. Seguindo procedimento análogo, os demais objetos nas posições $(k+2),(k+3),..., (n-1), n$ serão alocados ao grupo mais próximo.

A Figura 4 abaixo exemplifica a representação e a decodificação definidas no algoritmo BRKGA desenvolvido para o PASMD, considerando que n=10 e k=3. Após a ordenação de $\boldsymbol{v}$, e a definição de $\boldsymbol{z}$, observa-se que os objetos 4 (x_4), 6 (x_6) e 2 (x_2) serão alocados, respectivamente, aos grupos C_1, C_2, C_3. Em função da ordem que os demais valores aparecem na linha com vetor $\boldsymbol{z}$, o objeto 7 (x_7) será o 4º elemento a ser alocado a um dos grupos (considerando o grupo mais próximo), o objeto 1 (x_1) será o 5º elemento a ser alocado a um dos grupos, e assim

sucessivamente, até a alocação do objeto 8 (x_8) a um dos grupos. A cada alocação dos objetos entre as posições ($k+1$) e n, atualiza-se o valor da função objetivo. Considerando a matriz de distâncias hipotética da Figura 5 e o procedimento de alocação proposto acima, a Figura 6 traz o passo a passo da alocação e da atualização dos grupos e da função objetivo, onde cada linha apresenta o processo de alocação dos objetos de ***z*** entre as posições ($k+1$) e n.

Vetores \Posições	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
v	0,283	0,185	0,894	0,061	0,602	0,150	0,282	0,982	0,930	0,939
w	0,061	0,150	0,185	0,282	0,283	0,602	0,894	0,930	0,939	0,982
z* (objetos)	4	6	2	7	1	5	3	9	10	8
Grupos Iniciais	C_1={4}	C_2={6}	C_3={2}							
Sequência de Alocação dos objetos				**7 (4°)**	**1 (5°)**	**5 (6°)**	**3 (7°)**	**9 (8°)**	**10 (9°)**	**8 (10°)**

Figura 4 – Representação e decodificação de uma solução

(*Posição que o valor na i-ésima posição de w ocupa em v)

	1	2	3	4	5	6	7	8	9	10
1	0,00	0,29	0,06	0,46	0,13	0,38	0,28	0,60	0,21	0,06
2	0,29	0,00	0,23	0,17	0,42	0,08	0,01	0,31	0,08	0,23
3	0,06	0,23	0,00	0,40	0,20	0,31	0,22	0,54	0,15	0,01
4	0,46	0,17	0,40	0,00	0,60	0,09	0,18	0,14	0,25	0,41
5	0,13	0,42	0,20	0,60	0,00	0,51	0,42	0,74	0,35	0,19
6	0,38	0,08	0,31	0,09	0,51	0,00	0,10	0,23	0,16	0,32
7	0,28	0,01	0,22	0,18	0,42	0,10	0,00	0,32	0,07	0,22
8	0,60	0,31	0,54	0,14	0,74	0,23	0,32	0,00	0,39	0,55
9	0,21	0,08	0,15	0,25	0,35	0,16	0,07	0,39	0,00	0,16
10	0,06	0,23	0,01	0,41	0,19	0,32	0,22	0,55	0,16	0,00

Figura 5 – Distâncias entre os objetos

Passo	**Objeto**	**Distâncias**	**Soma Mínima**	**FOBJ**	**Grupos após cada passo**
1	7	d_{74},d_{76},d_{72}	0,01 (d_{72})	0,01	C_1={4}, C_2={6}, C_3={2,7}
2	1	d_{14}, d_{16}, d_{12}+d_{17}	0,38 (d_{16})	0,39	C_1={4}, C_2={6,1}, C_3={2,7}
3	5	d_{54}, d_{56}+d_{51}, d_{52}+d_{57}	0,60 (d_{54})	0,99	C_1={4,5}, C_2={6,1}, C_3={2,7}
4	3	d_{34}+d_{35}, d_{36}+d_{31}, d_{32}+d_{37}	0,37 (d_{36}+d_{31})	1,36	C_1={4,5}, C_2={6,1,3}, C_3={2,7}
5	9	d_{94}+d_{95}, d_{96}+d_{91}+d_{93}, d_{92}+d_{97}	0,15 (d_{92}+d_{97})	1,51	C_1={4,5}, C_2={6,1,3}, C_3={2,7,9}
6	10	d_{104}+d_{105}, d_{106}+d_{101}+d_{103}, d_{102}+d_{107}+d_{109}	0,39 (d_{106}+d_{101}+d_{103})	1,90	C_1={4,5}, C_2={6,1,3,10}, C_3={2,7,9}
7	8	d_{84}+d_{85}, d_{86}+d_{81}+d_{83}+d_{810}, d_{82}+d_{87}+d_{89}	0,88 (d_{84}+d_{85})	2,78	C_1={4,5,8}, C_2={6,1,3,10}, C_3={2,7,9}

Figura 6 – Alocação aos grupos e atualização da função objetivo

Concluindo a exposição do algoritmo BRKGA proposto para o PASMD, ressalta-se que o cruzamento propicia a troca dos k primeiros objetos que definirão inicialmente os grupos e, além disso, altera a ordem de alocação dos $(n-k)$ objetos aos grupos C_1,C_2,...,C_k.

6. Resultados Computacionais

Nesta seção são apresentados os resultados computacionais concernentes à aplicação do algoritmo BRKGA e dos três algoritmos propostos em [Brito e Brito, 2008] e [Nascimento, Toledo e de Carvalho 2010], além da formulação apresentada na Seção três, que foi implementada utilizando o software LINGO (versão 14.0). O algoritmo genético (AG) e o algoritmo VNS (cedido pelos autores) foram implementados em linguagem Delphi (versão 7.0), e os algoritmos GRASP e BRKGA foram implementados em linguagem R. Todos os experimentos foram efetuados em um computador com 16GB de memória RAM e dotado de seis processadores AMD FX-6300 de 3.5 GHz e sistema operacional Windows 7 (64 bits).

Considerando os recursos da arquitetura *multicore* do computador, e o fato de o software *R* ter um pacote (*snowfall* - https://cran.r-project.org/web/packages/snowfall/index.html) com funções de paralelismo, o algoritmo BRKGA foi paralelizado, mais especificamente, os procedimentos associados à alocação dos objetos aos grupos e ao cálculo da função objetivo.

De forma a avaliar a performance do algoritmo BRKGA frente aos demais algoritmos, no que diz respeito à qualidade das soluções produzidas, foi realizado um conjunto de experimentos computacionais com 32 instâncias (bases de dados). Quanto à origem, essas instâncias são classificadas em seis grupos, a saber: (1) da literatura, sendo citadas e utilizadas, por exemplo, em [Semaan 2013] e [Cruz 2010]; (2) nos sites do IBGE *(www.ibge.gov.br)* e da universidade da Califórnia (*archive.ics.uci.edu/ml/*); (3) geradas artificialmente no *software* R; (4) disponível no pacote MASS no R; (5) utilizadas em [Nascimento, Toledo e de Carvalho 2010] e (6) disponível em https://cs.joensuu.fi/sipu/datasets/. A Tabela 1 a seguir traz o nome da instância, os seus números de objetos (n) e atributos (q) e a sua origem. E Tabela 2 traz os parâmetros considerados na execução dos quatro algoritmos. Em particular, nos quatro algoritmos, o critério de parada foi o número máximo de gerações ou número máximo de iterações. No que diz respeito ao BRKGA, os parâmetros utilizados em todos os experimentos foram definidos a partir de experimentos preliminares. Mais especificamente, foram selecionadas, dentre as 32 instâncias, cinco instâncias sobre as quais foi aplicado o algoritmo BRKGA (50 vezes sobre cada instância), considerando todas as combinações (243) dos parâmetros: p= 75, 100 e 200; p_e=0.10, 0.15, 0.20; p_m=p_e ; ρ_e=0.7,0.8 e 0.9 e m=250, 500 e 1000.

Tabela 1 – Informações sobre as Bases de Dados

Base	*n*	*q*	Origem	Base	*n*	*q*	Origem
200DATA	200	2	1	idh2013	187	1	2
2face	200	2	1	indian	583	9	2
400p3c	400	2	1	iris	150	4	1
Aggregation	788	2	6	maronna	200	2	1
BreastB	49	1213	5	moreshapes	489	2	1
broken-ring	800	2	1	Normal300	300	2	3
Chart	600	60	1	numbers2	540	2	1
Concrete_Data	1030	9	2	Parkinsons	195	23	2
cpus	209	8	4	pib100	100	1	2
DBLCA	141	661	5	ruspini	75	2	1
DBLCB	180	661	5	spherical_4d3c	400	3	1
ecoli	336	7	5	UFS	27	1	2
face	296	2	1	Uniform_400	400	2	3
Gamma_400	500	3	3	vowel2	528	2	1
gauss9	900	2	1	wine	178	13	1
haberman	306	4	2	yeast	1484	7	1

Tabela 2 – Parâmetros e valores considerados para os algoritmos

Algoritmo	**Parâmetros**
BRKGA	p=75, m=1000, ρ_e=0,80, p_e=0,20, p_m=0,25
AG	Tamanho população=100, Maximo_Gerações=1000, Probabilidade_Cruzamento=0,35, Probabilidade_Mutação=0,05, Periodicidade_Path_Relinking=20
VNS	Maximo_Iterações=500, Maximo_Vizinhanças=4, Maximo_Trocas=100
GRASP	Máximo_Iterações=200, $\alpha \in [0,1]$ (vide [Nascimento, Toledo e de Carvalho 2010])

Tendo em vista que todas as instâncias têm, apenas, variáveis quantitativas, sendo algumas destas de magnitudes diferentes, foi realizada a padronização dos dados (escore z). Em relação à função objetivo da equação (6), foi adotada como métrica a distância euclidiana.

Em função de os algoritmos estarem implementados em diferentes linguagens de programação, optou-se por uma análise baseada nas soluções produzidas, e não em relação ao tempo de processamento demandado para produzir estas soluções. Assim, foi realizado um primeiro experimento em que as 32 instâncias foram submetidas aos quatro algoritmos, considerando o número de grupos (k) variando entre dois e quatro. Isso implicou obtenção de 96 soluções (nº de instâncias x nº de grupos). Em um segundo experimento aplicou-se a formulação de programação inteira mista apresentada na Seção 3, considerando um subconjunto de sete instâncias, escolhidas dentre as 32 apresentadas na Tabela 1, novamente com o número de grupos variando entre 2 e 4.

A Tabela 3 traz os resultados associados ao experimento I. A primeira coluna desta tabela tem o nome da instância e as colunas subsequentes têm, por grupo, os gaps entre o valor da função objetivo associado à solução produzida por cada um dos algoritmos (gap_{alg}) e a melhor solução produzida (f_{Best}) por, pelo menos, um dos algoritmos ($gap_{alg}=100(f_{alg}-f_{Best})/f_{Best}$). Doravante, denotaremos a melhor solução produzida por solução vencedora. A partir desta tabela é possível observar que, no que concerne ao quantitativo de soluções vencedoras, o algoritmo BRKGA teve maior eficácia em relação aos demais algoritmos. O segundo melhor algoritmo foi o algoritmo genético, seguido pelo GRASP. Entre os quatro algoritmos o VNS foi que teve pior performance.

Tabela 3 – Resultados dos algoritmos BRKGA, Genético, VNS e GRASP

	k=2					k=3					k=4				
Base	gap_{BRKGA}	gap_{AG}	gap_{VNS}	gap_{GRASP}	f_{Best}	gap_{BRKGA}	gap_{AG}	gap_{VNS}	gap_{GRASP}	f_{Best}	gap_{BRKGA}	gap_{AG}	gap_{VNS}	gap_{GRASP}	f_{Best}
200DATA	0,00%	0,00%	23,04%	0,00%	8.666,2	0,00%	0,00%	16,60%	0,49%	4.432,9	0,00%	0,00%	27,14%	0,17%	2.148,1
2face	0,00%	0,01%	2,34%	0,01%	12.524,8	0,00%	0,05%	11,65%	0,02%	6.404,6	0,04%	0,00%	6,30%	0,31%	4.010,3
400p3c	0,00%	0,00%	10,02%	0,01%	35.314,6	0,00%	1,19%	82,80%	0,00%	11.611,1	0,00%	9,04%	22,72%	0,44%	6.970,9
Aggregation	0,00%	0,02%	7,14%	0,39%	201.160,2	0,00%	11,67%	12,76%	0,18%	96.183,6	0,00%	29,94%	4,48%	0,12%	60.573,4
BreastB	0,10%	0,15%	0,00%	65,59%	26.948,8	0,35%	2,90%	0,00%	47,29%	17.210,3	0,55%	3,31%	0,00%	26,17%	12.465,1
broken-ring	0,00%	0,06%	4,17%	0,00%	217.866,0	0,00%	4,18%	7,49%	2,65%	115.608,1	0,00%	20,10%	25,88%	0,01%	65.984,4
Chart	0,01%	0,00%	0,36%	0,37%	914.852,0	0,00%	0,19%	0,86%	4,21%	591.497,1	0,00%	1,01%	0,23%	15,21%	436.414,9
Concrete_Data	0,00%	1,38%	0,99%	0,41%	955.562,6	0,00%	5,56%	2,84%	8,36%	580.860,6	0,00%	10,87%	4,08%	6,97%	402.830,8
cpus	0,00%	0,03%	4,11%	22,62%	24.333,7	0,00%	0,01%	9,26%	18,50%	14.260,4	0,00%	0,00%	10,80%	18,78%	9.303,6
DBLCA	0,00%	0,33%	0,16%	0,04%	168.375,4	0,10%	0,00%	0,18%	4,16%	106.878,0	0,17%	0,00%	0,03%	5,83%	79.112,0
DBLCB	0,00%	0,00%	0,15%	0,18%	270.583,0	0,24%	0,18%	0,00%	2,52%	173.675,2	0,31%	0,00%	0,73%	7,99%	126.731,2
ecoli	0,00%	0,00%	0,57%	1,45%	73.745,8	0,00%	0,00%	7,01%	3,64%	41.388,3	0,00%	0,85%	3,37%	24,39%	29.343,9
face	0,00%	0,15%	1,14%	0,17%	29.181,4	0,00%	0,00%	6,29%	0,49%	13.709,7	0,00%	0,15%	5,24%	0,51%	9.117,5
Gamma_400	0,00%	0,00%	5,18%	5,22%	106.536,7	0,00%	0,47%	6,81%	4,29%	61.216,8	0,00%	4,77%	11,54%	25,33%	40.138,5
gauss9	0,00%	0,59%	1,73%	0,46%	280.975,6	0,00%	18,95%	1,58%	0,62%	148.860,5	0,00%	24,22%	2,56%	0,33%	91.942,3
haberman	0,00%	0,00%	5,45%	1,79%	50.674,6	0,00%	8,61%	3,66%	8,92%	28.445,9	0,00%	1,23%	4,73%	12,65%	19.687,1
idh2013	0,00%	0,00%	13,36%	0,00%	5.189,0	0,00%	0,00%	3,54%	0,00%	2.260,0	0,00%	0,00%	15,32%	0,00%	1.282,0
indian	0,01%	0,00%	3,92%	1,61%	260.435,0	0,00%	0,65%	6,67%	11,49%	152.915,7	0,00%	2,58%	3,48%	32,65%	106.845,2
iris	0,00%	0,00%	4,46%	0,00%	9.135,7	0,00%	0,00%	14,78%	0,08%	4.498,5	0,00%	0,00%	6,49%	0,29%	3.160,6
maronna	0,00%	9,13%	7,11%	0,00%	12.229,5	0,00%	0,06%	3,27%	0,36%	6.688,5	0,00%	0,00%	14,17%	0,00%	3.232,1
moreshapes	0,00%	1,96%	3,58%	2,29%	72.780,7	0,00%	13,46%	21,92%	6,92%	35.505,8	0,00%	7,78%	18,34%	0,00%	18.013,0
Normal300	0,00%	0,00%	3,03%	0,04%	31.725,5	0,00%	0,84%	3,82%	1,84%	17.436,2	0,00%	1,96%	4,02%	3,35%	11.440,0
numbers2	0,00%	0,00%	2,89%	0,05%	93.523,1	0,00%	0,74%	3,45%	1,30%	51.013,4	0,00%	12,41%	5,93%	1,32%	30.182,1
Parkinsons	0,00%	0,00%	0,93%	0,00%	46.851,6	0,00%	0,00%	1,62%	3,98%	28.698,5	0,00%	1,32%	3,22%	7,07%	19.765,8
pib100	0,00%	0,00%	3,77%	0,09%	1.211,0	0,00%	0,00%	22,23%	0,73%	492,1	0,00%	0,00%	9,65%	1,07%	262,2
ruspini	0,00%	0,00%	4,06%	0,00%	1.631,4	0,00%	0,01%	1,63%	0,35%	822,7	0,00%	0,00%	0,83%	0,00%	316,5
spherical_4d3c	0,00%	0,00%	7,42%	0,00%	40.339,3	0,00%	0,07%	2,27%	2,42%	21.180,9	0,00%	3,12%	31,56%	0,00%	8.965,6
UFS	0,00%	0,01%	0,01%	0,00%	105,3	0,00%	0,00%	0,00%	0,00%	36,4	0,00%	11,34%	0,00%	0,00%	22,8
Uniform_400	0,00%	0,12%	1,66%	0,02%	55.415,0	0,00%	0,56%	3,72%	1,04%	28.882,1	0,00%	2,26%	3,73%	0,17%	17.738,1
vowel2	0,00%	1,65%	2,91%	0,12%	95.996,6	0,00%	0,49%	6,05%	1,43%	51.502,0	0,00%	6,54%	6,72%	3,51%	33.391,8
wine	0,00%	0,00%	2,16%	0,01%	32.415,2	0,00%	0,02%	3,30%	2,05%	18.948,2	0,00%	0,77%	1,90%	7,79%	13.563,4
yeast	0,00%	0,98%	2,16%	8,24%	1.543.363,6	0,00%	7,64%	1,57%	23,63%	957.978,5	0,00%	12,29%	4,25%	33,77%	673.856,1

Em complemento à Tabela 3, a Figura 7 traz o gráfico associado aos percentuais de soluções vencedoras dos quatro algoritmos. Estes percentuais foram calculados com base em 96 soluções produzidas. A partir deste gráfico, observa-se que o algoritmo BRKGA produziu a solução vencedora em 90% dos casos, seguido pelos algoritmos AG, GRASP e VNS com percentuais de, respectivamente, 35%, 18% e 6%.

Concluindo as análises, a Tabela 4 traz os resultados do segundo experimento realizado com sete instâncias, com número de objetos variando de 27 até 336. A primeira coluna desta tabela tem o nome da instância e as colunas subsequentes têm, por grupo, o gap entre a solução da formulação (f_{LINGO}) e a solução produzida pelo BRKGA (f_{BRKGA}) e o valor da função objetivo correspondente à solução produzida pela formulação (f_{LINGO}), ou seja: $gap_{BRKGA}=100.(f_{LINGO}-f_{BRKGA}) / f_{LINGO}$. Em cada um dos 27 casos (7 instâncias x nº de grupos) a formulação foi executada considerado um tempo máximo de processamento de 9 horas. Dentro deste limite, a formulação só produziu o ótimo global para a instância UFS, em um tempo inferior a cinco minutos. Para as demais instâncias, a solução apresentada corresponde, a apenas, uma solução viável. Além disso, o valor da solução (função objetivo) produzida pelo BRKGA foi inferior ao valor da solução formulação. Ainda neste sentido, pode-se observar que, em geral, à medida que o número de objetos aumenta, maior é o gap entre a solução do BRKGA e a solução produzida pela formulação, indicando que, quanto maior o porte da instância, mais vantajosa é a utilização do BRKGA, que demanda um tempo de processamento máximo da ordem de minutos.

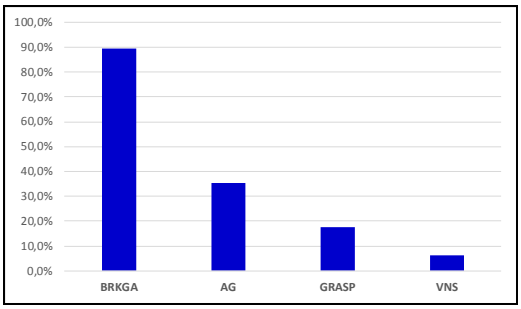

Figura 7 – Percentual de Soluções "Vencedoras" por Algoritmo

Tabela 4 – Comparação entre BRKGA e a formulação

		k=2		*k=3*		*k=4*	
Base	***n***	gap_{BRKGA}	f_{LINGO}	gap_{BRKGA}	f_{LINGO}	gap_{BRKGA}	f_{LINGO}
UFS	27	0,0%	**105,3**	0,1%	**36,4**	0,0%	**22,8**
BreastB	49	1,6%	27.424,8	2,0%	17.614,5	1,9%	12.770,4
ruspini	75	2,9%	1.680,5	21,3%	1.044,7	2,9%	325,9
iris	150	29,5%	12.962,3	28,0%	6.247,5	33,4%	4.743,1
DBLCB	180	5,0%	284.891,5	28,3%	242.955,8	12,6%	145.506,8
2face	200	23,7%	16.422,9	39,0%	10.500,9	37,2%	6.389,0
ecoli	336	36,0%	115.153,0	40,6%	69.652,2	32,3%	43.349,5

Os bons resultados apresentados para instâncias de porte variado indicam que o BRKGA pode ser uma boa alternativa à resolução do problema de agrupamento com soma mínima de distâncias. Não obstante, em trabalhos futuros, serão implementadas e avaliadas novas decodificações e novos tipos de cruzamento como, por exemplo, os cruzamentos de um e dois pontos. Serão realizados, também, novos experimentos, considerando a comparação do BRKGA com os algoritmos propostos no trabalho de [Serpa 2011].

Referências

Ahmadia,S. and Osmanb, I.H (2005). Greedy Random Adaptive Memory Programming Search for the Capacitated Clustering Problem. *European Journal of Operational Research*, 162, 1:30–44.

Bean, J.C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6:154-160.

Bolfarine, H. e Bussab, W.O. (2005). Elementos de Amostragem. ABE – Projeto Fisher. Editora Edgard Blücher.

Brito, J.A.M e Brito, L.R. (2008). Algoritmos VNS e Genéticos Aplicados ao Problema de Agrupamento com Soma Mínima de Distâncias. Simpósio Brasileiro de Pesquisa Operacional, 2008, João Pessoa - PB.

Brito, J.A.M., Montenegro, F. M. T. ; Freitas, M.P.S. (2011). Algoritmos de Otimização Aplicados à Estratificação da Amostra Mestra. Escola de Amostragem e Metodologia de Pesquisa, 2011, Juiz de Fora. Anais da III ESAMP.

Brito, J.A.M, Semaanm G.S. e Brito, L.R. (2015). Resolução do Problema dos k-medoids Via Algoritmo Genético de Chaves Aleatórias Viciadas. *Revista Pesquisa Naval*, 27:126-142.

Carrizosa, E., Mladenovicb, N., Todosijevicc R. (2013).Variable neighborhood search for minimum sum-of-squares clustering on networks. *European Journal of Operational Research*, 230, 2:356-363.

Cruz, M.D. (2010). O Problema de Clusterização Automática. (Tese de Doutorado). Coppe/UFRJ.

Freitas, M.P.S; Lila, M.F.; Azevedo, R.V.; Antonaci, G.A. (2007). Amostra Mestra Para o Sistema Integrado de Pesquisas Domiciliares, Textos para Discussão (23), Diretoria de Pesquisas.

Friedman, J. H. and Meulman J.J. (2004). Clustering objects on subsets of attributes. *Journal Royal Statistics Society* B, Part 4,66:815-849.

Gonçalves, J.R. e Resende, M.G.C. (2011). Biased random-key genetic algorithms for combinatorial optimization, *Journal of Heuristics*, 17: 487-525.

Hair, J.F, Black, W.C, Babin, B.J., Anderson, R.E. and Tatham, R.L. (2009). Análise Multivariada de Dados, Bookman, Sexta Edição.

Hansen, P. and Jaumard, B. (1997). Cluster Analysis and Mathematical Programming. *Mathematical Programming*, 79:191-215.

Johnson A.R. and Wichern D.W. (2002). *Applied Multivariate Statistical Analysis*. Prentice Hall. Fifth Edition.

Kaufman L. e Rousseeuw P.J. (1989). *Finding Groups in Data – An Introduction to Cluster Analysis*. Wiley-Interscience Publication.

Mingoti, S.A. (2007) *Análise de Dados Através de Métodos de Estatística Multivariada* – Uma Abordagem Aplicada. Belo Horizonte. Editora UFMG.

Naldi, C.N. (2011). *Técnicas de Combinação para Agrupamento Centralizado e Distribuído de Dados*. Tese de Doutorado – Universidade de São Paulo.

Nascimento, M.C.V., Toledo, F.M.B. e de Carvalho, A.C.P.L.F (2010). Investigation of a new GRASP- based clustering algorithm applied to biological data. Computers & Operations Research, 37:1381-1388.

Oliveira, R.M., Chaves, A.A. e Lorena, L.A.N. (2016). Aplicação da metaheurística BRKGA com heurística de busca local para o problema de agrupamento com restrições. Anais do XLVIII SBPO Simpósio Brasileiro de Pesquisa Operacional Vitória, ES.

Oliveira, R.M., Chaves, A.A. e Lorena, L.A.N. (2017). A comparison of two hybrid methods for constrained clustering problems. Applied Soft Computing, 54:256–266.

Rao, M.R. (1971). Cluster Analysis and Mathematical Programming. *Journal of American Statistical Association*, 66:622-626.

Resende, M.G.C. (2013a). Introdução aos Algoritmos Genéticos de Chaves Aleatórias Viciadas. Anais do XLVSBPO, p. 3680-3691, Natal/RN.

Resende, M.G.C (2013b). Biased random-key genetic algorithms. Palestra apresentada no XLV Simpósio Brasileiro de Pesquisa Operacional, Natal, RN, Brazil.

Santia,E, Aloise, D., Blanchardc, S.J (2016). A model for clustering data from heterogeneous dissimilarities. *European Journal of Operational Research*, 253, 3:659-672.

Semaan, G.S. (2013). Algoritmos para o Problema de Agrupamento Automático. (Tese de Doutorado) – Instituto de Computação, Universidade Federal Fluminense, Rio de Janeiro, RJ.

Serpa, D.R. (2011). Abordagens Heurísticas para Problemas de Agrupamento. Dissertação de Mestrado. INPE/São José dos Campos, SP.

Spears, W.M., Dejong, K.A. (1991). On the virtues of parameterized uniform crossover. *In*: *Proceedings* of the Fourth International Conference on Genetic Algorithms, p. 230-236.

Vinod, H. (1969). Integer Programming and Theory of Grouping. *Journal of American Statistical Association*, 64: 506-517.