RESOLUÇÃO DO PROBLEMA DOS *K*-MEDOIDS VIA ALGORITMO GENÉTICO DE CHAVES ALEATÓRIAS VICIADAS

José André de Moura Brito
Escola Nacional de Ciências Estatística – ENCE/IBGE
Rua André Cavalcanti, 106, sala 403, Centro – Rio de Janeiro – RJ.
e-mail: jose.m.brito@ibge.gov.br

Gustavo da Silva Semaan
Instituto do Noroeste Fluminense de Educação Superior
Universidade Federal Fluminense - INFES/UFF
Av. João Jasbick, s/n, Bairro Aeroporto - Santo Antônio de Pádua - RJ - CEP 28470-000
e-mail: gustavosemaan@gmail.com

Luciana Roque Brito
Centro Universitário Anhanguera de Niterói - UNIAN
Rua Visconde do Rio Branco, 137, Centro Niterói, Rio de Janeiro
e-mail: lubritoroque@gmail.com

Resumo

O presente trabalho propõe um novo algoritmo de otimização para a resolução do problema dos *k*-medoids. Um problema de agrupamento, onde dado um conjunto de ***n*** objetos com ***f*** atributos, e fixado o número de grupos, deve-se selecionar, dentre os ***n*** objetos, ***k*** objetos denominados medoids. Os demais objetos são alocados ao medoid correspondente mais próximo, segundo uma medida distância. Especificamente, o conjunto dos ***k*** medoids é definido de forma que a soma das distâncias dos demais objetos ao seu respectivo medoid seja a menor possível. De forma a resolver este problema, propõe-se no presente trabalho um algoritmo que utiliza os conceitos da metaheurística algoritmo genético de chaves aleatórias viciadas *(biased random-key genetic algorithm* - BRKGA) e um procedimento de reconexão por caminhos. A última seção traz alguns resultados computacionais para um conjunto de instâncias da literatura, reais e geradas artificialmente, considerando a aplicação do algoritmo proposto, de quatro algoritmos da literatura e de uma formulação exata.
Palavras-Chaves: Algoritmos Genéticos; Chaves Aleatórias; Medoids; Análise de Agrupamentos; Processamento Paralelo

Abstract

This paper proposes a new Optimization Algorithm for the *k*-medoid Clustering Problem. In this problem, given a dataset *X* with *n* objects and *f* attributes and a fixed number of clusters (*k*), it is necessary to select *k* objects called medoids. Each medoid creates a new cluster and the remaining (*n*–*k*) objects should be placed into nearest of these clusters, according a distance measure. The goal is minimize the sum of distances between each object and the medoid of its group. This work presents a new algorithm that considers concepts of Biased Random-key Genetic Algorithm. Besides, an approach of path-relinking procedure is related. The computational experiments results are presented in the last section. It was used thirty instances, among well-known datasets of the literature and new datasets artificially constructed. The proposed heuristics are compared with five approaches (four algorithms and one exact method) and the presented algorithms are a alternative and effective way to solve the problem.

Keywords: Genetic Algorithms; Random-key; Medoids; Cluster Analysis; Parallel Processing

1. INTRODUÇÃO

A análise de agrupamentos teve o seu desenvolvimento na década de 30, mas a descrição inicial foi formulada por Tryon em 1939. O maior estímulo para o seu desenvolvimento foi o livro Princípios de Taxonomia Numérica, dos Biólogos Sokal e Sneath. Acrescenta-se a esse estímulo o desenvolvimento de processadores com uma capacidade de efetuar cálculos e operações matemáticas cada vez maiores. Em particular, tem sido crescente nos últimos anos o número de aplicações reais que podem ser mapeadas em um problema de agrupamento. Neste sentido, a análise de agrupamentos tem sido muito utilizada em diversos domínios tais como: na Biologia, na Estatística, na Atuária, na Medicina, entre outros.

Basicamente, a análise de agrupamentos é uma técnica de análise multivariada que tem por objetivo a construção de grupos (ou *clusters*) a partir de uma base de dados composta por $\boldsymbol{n}$ objetos (registros) com $\boldsymbol{f}$ atributos. Estes grupos, por sua vez, são construídos de forma que tenham um alto grau de homogeneidade internamente e um baixo grau de homogeneidade entre si. Para realizar a alocação dos objetos aos grupos e, consequentemente, avaliar a homogeneidade dos mesmos, utiliza-se uma função objetivo que está associada a algum tipo de distância, como por exemplo, a euclidiana. Em decorrência das inúmeras aplicações de análise de agrupamentos, e da complexidade de resolução dessas aplicações, no que concerne à obtenção de soluções de boa qualidade, encontram-se na literatura diversos métodos de agrupamento. Tais métodos podem ser classificados, basicamente, como não hierárquicos e hierárquicos. Em particular, propõe-se neste trabalho um novo método de agrupamento não hierárquico para a resolução do problema dos $\boldsymbol{k}$-medoids. Este método utilizou os conceitos da metaheurística algoritmo genético de chaves aleatórias viciadas *(biased random-key genetic algorithm* – BRKGA) e incorpora, também, um procedimento de reconexão por caminhos (*path relinking*).

O presente trabalho está dividido da seguinte forma: A seção dois traz uma descrição dos principais conceitos de análise de agrupamentos; a seção três apresenta o problema dos *k*-medoids, a sua formulação exata e faz uma revisão dos principais trabalhos da literatura. Na seção quatro é feita uma descrição da metaheurística algoritmo genético de chaves aleatórias viciadas e do reconexão por caminhos, além da descrição do novo algoritmo proposto. E finalmente, a seção cinco traz um conjunto de resultados computacionais obtidos a partir da aplicação desse algoritmo, da formulação e de quatro algoritmos da literatura em instâncias da literatura e artificiais.

2. ANÁLISE DE AGRUPAMENTOS

A análise de agrupamentos é uma técnica de análise multivariada que vem sendo utilizada com sucesso em diversas áreas (Hair et al., 2009, Mingoti, 2007), tais como aquelas concernentes ao Reconhecimento de Padrões, à Biologia, aos Seguros, à Busca na Web e à Estatística, dentre outras.

Basicamente, ao se aplicar uma análise de agrupamentos, o objetivo é resolver o problema de particionar um conjunto de $\boldsymbol{n}$ objetos (registros) de uma base de dados em subconjuntos disjuntos denominados grupos (ou clusters).

Estes grupos são definidos de forma que os objetos alocados a um mesmo grupo sejam similares (homogêneos) entre si, considerando alguma métrica, e que os objetos pertencentes aos grupos diferentes sejam dissimilares. Neste caso, a métrica está associada a uma medida de distância que considera os f atributos quantitativos e/ou qualitativos disponíveis para cada um dos n objetos. A figura 1 traz um exemplo de uma base de dados (instância) constituída por mais de cinco mil objetos com três atributos ($\boldsymbol{f}$=3), e que foram separados em três grupos. Uma vez que todas as instâncias utilizadas no presente trabalho têm atributos apenas quantitativos, optou-se pelo uso da distância euclidiana para avaliar a similaridade entre os objetos. Quando o número de grupos é definido a priori, temos um problema de agrupamento clássico. Caso contrário, define-se um problema de agrupamento automático (Cruz, 2010, Naldi, 2011 e Semaan, 2013).

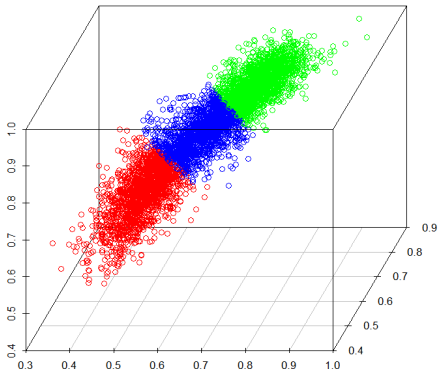


Figura 1- Exemplo de um Agrupamento

Formalmente, dado um conjunto ***X*** formado por ***n*** objetos, ***X***= $\{x_1, x_2,..., x_n\}$ com ***f*** atributos, e sendo o número de grupos igual a um inteiro ***k***, devem ser construídos ***k*** grupos $G_1, G_2,..., G_k$, considerando as três restrições a seguir:

(i) $G_1 \cup G_2 \cup ... \cup G_k = \boldsymbol{X}$

(ii) $G_i \cap G_j = \varnothing$, i,j=1,...,***k*** (i<j)

(iii) $|G_i| \geq 1$, i=1,...,***k***

A restrição (i) indica que a união dos grupos corresponde ao conjunto ***X***. A restrição (ii) indica que um objeto pertence a exatamente um grupo e a restrição (iii) garante que cada grupo tem pelo menos um objeto.

Para muitas aplicações reais, a determinação do agrupamento ótimo, ou seja, a partição de ***X*** que produz os grupos mais homogêneos, segundo algum critério de similaridade é uma tarefa inviável do ponto vista computacional. Esse fato é decorrente de 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 clássico está associado ao número de *Stirling* de segundo tipo (ver Johnson e Wichern, 2002), dado pela equação abaixo:

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

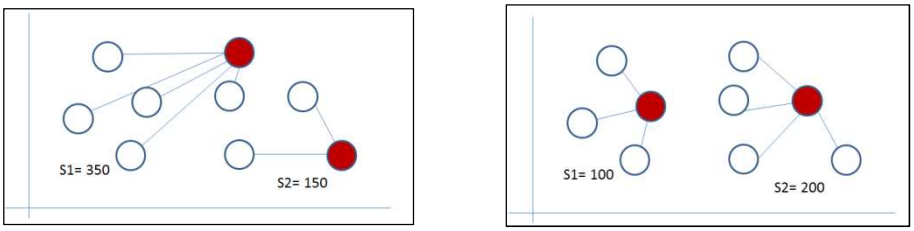
Considerando, por exemplo, que *n* = 30 objetos serão alocados em dois clusters, o número de soluções a serem consideradas é de 536.870.911. Mas, mantendo o mesmo número de clusters, e apenas dobrando o número de objetos, temos mais de meio quatrilhão de soluções possíveis. Ao considerar um número *n* maior de objetos, esses valores crescem exponencialmente. Este fato torna inviável a aplicação de um método de enumeração exaustiva para a resolução deste problema. Mas, abrindo-se mão do ótimo global, é possível produzir soluções viáveis de qualidade razoável consumindo um tempo computacional factível. Normalmente, essas soluções podem ser produzidas mediante a aplicação de um dos métodos clássicos de agrupamento disponíveis na literatura, quais sejam: hierárquico ou não hierárquico (Hair *et al.*, 2009, Mingoti, 2007). Sem incorporar a aplicação de buscas locais sofisticadas, esses métodos produzem soluções (grupos) de qualidade razoável, sem examinar todas as soluções possíveis. Os métodos hierárquicos, por sua vez, dividem-se em aglomerativos e divisivos. Os métodos não hierárquicos procuram encontrar uma partição (*clusters*) 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 e Rousseeuw, 1989). Em particular, esse último método está associado ao problema que foi o objeto de estudo deste trabalho.

3. O PROBLEMA DOS *K*-MEDOIDS

Considere um conjunto ***X*** formado por *n* objetos ***X***= {x_1, x_2,...,x_n} com *f* atributos. A partir de ***X***, deve-se selecionar ***k*** objetos que definem um subconjunto ***Y*** (Y⊂X), tal que ***Y***={y_1,...,y_k}. Os objetos de ***Y*** são denotados por medoids ou objetos representativos (Kaufman e Rousseeuw**,** 1989). Uma vez selecionados os ***k*** medoids, os ***(n-k)*** objetos de ***X*** são alocados ao grupo G_i (i=1,..., ***k***) cujo medoid esteja mais próximo, considerando alguma métrica (distância). Mais especificamente, os elementos de ***Y*** devem ser escolhidos de forma a minimizar a função objetivo definida pela média (por grupo) das distâncias de todos os objetos aos seus respectivos medoids:

$$f = \sum_{i=1}^{k} \sum_{\forall x_j \in G_i} \frac{d_{y_i x_j}}{|G_i|} \qquad (2)$$

As figuras 2.a e 2.b ilustram um exemplo desse problema, considerando a alocação de nove objetos em dois grupos, ou seja, a determinação de dois medoids. Em particular, neste exemplo são apresentadas duas soluções diferentes. Os dois objetos coloridos na figura correspondem aos medoids, os segmentos de reta correspondem às distâncias, e os valores de S1 e S2 correspondem às somas das distâncias dos objetos aos seus respectivos medoids. Neste caso, a melhor solução seria a representada na figura 2.b (soma total igual a 300).



(a) (b)

Figuras 3 - Agrupamento com Dois Medoids

3.1 FORMULAÇÃO PARA O PROBLEMA DOS k-MEDOIDS

$$\text{Minimizar} \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} x_{ij} \qquad (3)$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1,...,n \qquad (4)$$

$$x_{ij} \le y_i,\ i = 1,...,n,\ j = 1,...,n \qquad (5)$$

$$\sum_{i=1}^{n} y_i = k \qquad (6)$$

$$y_i, x_{ij} \in \{0,1\},\ i, j = 1,...,n \qquad (7)$$

A presente formulação de programação inteira pode ser encontrada no livro de Kaufman e Rousseeuw (Kaufman L. e Rousseeuw, 1989). A ideia de utilizar essa formulação neste problema de análise de agrupamentos foi introduzida por Vinod (1969) e mais tarde também foi discutida por Rao (1971), Church (1978) e Mulvey e Cowder(1979). É uma formulação definida por três restrições e dois tipos de variáveis binárias. Mais especificamente, y_i é uma variável binária que assume valor um se o objeto ***i*** (i=1,...,n) é definido como medoid, e zero em caso contrário, e x_{ij} é uma variável binária que assume valor um se o objeto ***j*** é alocado ao grupo definido pelo medoid ***i***. A restrição (4) garante que cada objeto ***j*** será alocado a exatamente um medoid. A restrição (5) garante que um objeto ***j*** pode somente ser associado a um objeto ***i*** se este último for escolhido como um medoid. E a restrição (6) garante que serão escolhidos exatamente ***k*** objetos como medoids. A função objetivo a ser minimizada na equação (3) (equivalente à função da equação 2) corresponde à soma das distâncias d_{ij} (dissimilaridades) entre os objetos ***j*** e os seus respectivos medoids.

A formulação acima tem n^2+n variáveis binárias e n^2+n+1 restrições. Em função do número de variáveis binárias, a aplicação desta formulação é viável quando o problema é de pequeno porte, ou seja, se o número de variáveis é no máximo da ordem de centenas

(Kaufman e Rousseeuw, 1989). E mesmo assim, a sua resolução pode levar ao consumo expressivo de tempo computacional produzindo, em muitos casos, apenas um ótimo local. Todavia, abrindo-se mão do ótimo global, é possível produzir soluções viáveis mediante a aplicação de alguns algoritmos heurísticos disponíveis na literatura.

3.2 ALGORITMOS DA LITERATURA

Um dos algoritmos heurísticos mais conhecidos da literatura, e utilizado para este problema, é o algoritmo PAM (*Partitioning Around Medoids*), proposto por Kaufman e Rousseeuw (1989). Basicamente, essa heurística efetua a construção dos agrupamentos mediante a aplicação de dois procedimentos determinísticos, sejam eles: *Build* e *Swap*. O primeiro procedimento é responsável pela construção de uma solução viável, e o segundo procedimento tem caráter similar ao de uma busca local, atuando sobre a solução produzida pelo procedimento *Build*. Kaufman e Rousseeuw (1989) também propuseram uma versão modificada da heurística PAM, denominada CLARA (*Clustering Large Applications*). Essa heurística pode ser aplicada em bases de dados de dimensão elevada (muitos objetos), considerando a combinação de uma amostragem aleatória simples e do algoritmo PAM. Isto é, em vez de determinar os *k*–medoids considerando os *n* objetos, essa heurística seleciona *m* amostras aleatória simples compostas por n' objetos da base de dados ($n' < n$), aplicando, em seguida, a heurística PAM em cada uma dessas amostras. Neste caso, a solução do problema dos *k*-medoids corresponderá à melhor solução (menor valor da função objetivo) encontrada dentre as ***m*** soluções (amostras).

Além dessas duas heurísticas são encontrados na literatura alguns trabalhos mais recentes. Han e Ng (2002) propuseram uma nova versão dessa heurística denominada CLARANS, que utiliza uma técnica de computação mais intensiva para determinação dos medoids. Em Park e Jun (2009) foi apresentado um algoritmo para os *k*-medoids que utiliza algumas ideias do algoritmo *k*-means. Somam-se a essas abordagens os trabalhos de Sheng e Liu (2004), Zhang e Couloigner (2005), Chu *et al.* (2008) e Brito *et al.* (2010). Considerando um enfoque baseado em uma estratégia de busca local mais eficiente, Brito e Semaan (2013) desenvolveram um novo algoritmo para o problema dos *k*-medoids, que teve por base as ideias da metaheurística GRASP (Feo e Resende, 1995).

Inicialmente, com objetivo de propiciar um bom entendimento dessa proposta, são apresentados os conceitos básicos da metaheurística algoritmo genético de chaves aleatórias viciadas, efetuando-se em seguida uma descrição do novo algoritmo.

4. METODOLOGIA PROPOSTA

4.1 ALGORITMOS GENÉTICOS DE CHAVES ALEATÓRIAS VICIADAS (BRKGA)

Segundo Gonçalves e Resende (2011), o algoritmo genético de chaves aleatórias (random-key genetic algorithm – RKGA) foi introduzido por Bean (1994) para resolver um problema de otimização combinatória denominado problema de sequenciamento.

Em um RKGA, os cromossomos (soluções) são representados por uma string ou um vetor cujos valores (genes) são obtidos a partir de valores reais selecionados aleatoriamente, segundo uma distribuição uniforme [0,1]. Após a geração, aplica-se em cada um desses cromossomos um procedimento denominado decodificador. O decodificador tem por finalidade estabelecer um mapeamento entre os valores dos cromossomos e as possíveis soluções viáveis para o problema de otimização combinatória para o qual a função objetivo deve ser computada. Dessa forma, o procedimento decodificador é específico para cada problema de otimização combinatória ao qual é utilizado um RKGA. Um RKGA 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, aplicando um princípio darwinista. A população inicial é constituída por *p* vetores de chaves aleatórias geradas conforme descrito acima. Após a aplicação do decodificador e o cálculo da função objetivo, a população é particionada em dois conjuntos, quais sejam: 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-p_e soluções restantes (não elite), sendo p_e <p-p_e. Para atualizar a população, uma nova geração de cromossomos deve ser produzida. Um RKGA usa uma estratégia de elitismo, tendo em vista que todos os p_e cromossomos pertencentes ao conjunto elite na geração g são copiados para a população da geração g+1. A adoção dessa estratégia tende a produzir soluções viáveis cada vez melhores durante as gerações do algoritmo. Em um RKGA a mutação é implementada mediante a introdução na população de cromossomos chamados de mutantes.

Um mutante é um vetor de chaves aleatórias gerado da mesma forma que os vetores da população inicial são gerados. Em cada geração um pequeno número (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 RKGA, Bean(1994) considera que cada uma das p-p_e-p_m soluções são produzidas para a geração seguinte através do cruzamento de duas soluções (pais) selecionadas da população inteira na g-ésima geração. As soluções (filhos) são produzidas mediante aplicação do cruzamento uniforme (Spears and Dejong, 1991).

O algoritmo genético de chaves aleatórias viciadas (*biased random-key genetic algorithm* – BRKGA) difere do RKGA no que concerne ao cruzamento aplicado às soluções. Mais especificamente, as p-p_e-p_m soluções (filhos) geradas por cruzamento são produzidas tomando uma solução do conjunto elite e uma solução do conjunto não elite. A figura 4 ilustra a evolução de uma população de uma geração g para a geração $g+1$. Uma vez que p_e<p-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 grande chance de passar as suas características para as gerações futuras.

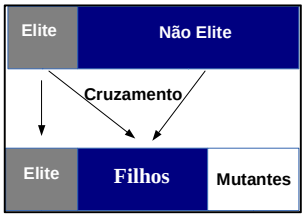


Figura 4 – Soluções produzidas pelo BRKGA entre duas gerações seguidas

Esta pequena diferença entre as duas metaheurísticas possibilita, em geral, que o BRKGA tenha um desempenho superior (Gonçalves et al., 2012) ao RKGA no que concerne às soluções produzidas. A figura cinco ilustra a aplicação do procedimento de cruzamento para dois cromossomos com 10 genes cada. Nessa figura o cromossomo C_1 foi selecionado do conjunto elite e o cromossomo C_2 foi selecionado do conjunto não elite. Uma vez selecionados estes cromossomos, é gerado um vetor auxiliar (v_a) com valores reais selecionados aletoriamente do intervalo [0,1], sendo esse vetor de igual tamanho a C_1 e C_2. Também é definido o valor de ρe>0.5, sendo esse valor correspondente à probabilidade de um gene de C_1 compor o cromossomo filho (Cf) . 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 ρ_e= 0.7), temos que a i-ésima posição do cromossomo filho herda o valor da i-ésima posição de C_1. Em caso contrário, a i-ésima posição do cromossomo filho herda o valor da i-ésima posição de C_2.

C1	0,31	0,77	0,81	0,49	0,32	0,97	0,72	0,15	0,56	0,92
C2	0,26	0,15	0,36	0,41	0,93	0,11	0,28	0,56	0,34	0,87
va	0,58	0,89	0,11	0,41	0,75	0,99	0,43	0,71	0,88	0,58
Cf	0,31	0,15	0,81	0,49	0,93	0,11	0,72	0,56	0,34	0,92

Figura 5 – Cruzamento Uniforme (ρ_e=0.7)

Quando a próxima população está completa, isto é, quando há *p* cromossomos, considerando os cromossomos elite, mutantes e filhos, são computados os valores da função objetivo para todos os novos vetores mutantes e filhos, e a população é novamente particionada em um conjunto elite e não-elite para iniciar uma nova geração. Em linhas gerais, um BRKGA efetua buscas no espaço de soluções viáveis ***S*** de um problema de otimização combinatória indiretamente, considerando a busca em um hipercubo unitário no R^n, usando um decodificador para mapear as soluções do hipercubo na soluções de ***S*** onde a função objetivo é avaliada. A figura seis ilustra todos os passos considerados à aplicação do BRKGA.

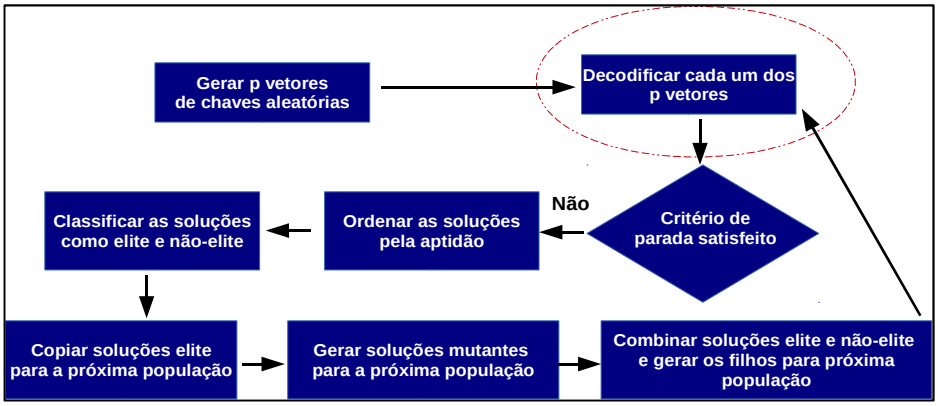


Figura 6 – Passos do BRKGA[1]

Pela descrição feita sobre o BRKGA, chega-se à conclusão que todos os passos ilustrados na figura acima, exceto a decodificação, independem do problema do problema de otimização combinatória 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) várias 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.

4.2 O ALGORITMO BRKGA PARA O PROBLEMA DOS K-MEDOIDS – REPRESENTAÇÃO, DECODIFICAÇÃO E CRUZAMENTO

Conforme comentado na seção anterior, a implementação do BRKGA para um problema de otimização requer, basicamente, a especificação da representação dos cromossomos (soluções) e do decodificador. No que concerne ao problema dos *k*-medoids, cada cromossomo foi definido como um vetor ***v*** com *n* posições (número de objetos) preenchidas com valores reais gerados uniformemente no intervalo [0,1], sendo cada posição i de ***v*** correspondente ao objeto x_i. Uma vez gerado o cromossomo, o decodificador atribui os elementos de *v* ordenados crescentemente a outro vetor *w*. Em seguida, os *k*-primeiros valores de *w* são pesquisados em *v*, retornando-se as posições que esses valores ocupam em *v*. Essas posições corresponderão aos objetos escolhidos para medoids. A figura abaixo exemplifica a representação e a decodificação definidas no BRKGA desenvolvido para o problema dos *k*-medoids, considerando que *n*=10 e *k*=3.

Posições de v	1	2	3	4	5	6	7	8	9	10
v	0,918	0,114	0,219	0,397	0,981	0,245	0,483	0,546	0,504	0,898
w	0,114	0,219	0,245	0,397	0,483	0,504	0,546	0,898	0,918	0,981
Medoids	2	3	6							

Figura 7 – Representação e Decodificação de uma Solução

[1] Figura baseada em figura apresentada no trabalho de Gonçalves e Resende (2011).

Considerando a população atual, aplica-se m ($m=p-p_e-p_m$) vezes o procedimento de cruzamento uniforme (vide figura 5), sendo utilizado em cada cruzamento um vetor do conjunto elite e outro do conjunto não elite. No que diz respeito ao problema dos *k*-medoids, o cruzamento propicia a troca dos medoids entre duas soluções, produzindo vetores filhos que correspondem às novas soluções.

4.3 O PROCEDIMENTO DE RECONEXÃO POR CAMINHOS (PATH-RELINKING)

O procedimento de reconexão por caminhos está associado a uma estratégia de intensificação proposta inicialmente por Glover (Glover e Kochenberger, 2002), para explorar "caminhos" entre as soluções classificadas como elite, sendo essas soluções obtidas a partir da aplicação de uma *busca tabu* ou do *scatter search*. Neste procedimento, dado um conjunto de soluções de melhor qualidade, denominadas soluções elite, devem ser gerados "caminhos" entre duas delas, sendo tais caminhos correspondentes às soluções intermediárias. Basicamente, a reconexão por caminhos funciona como uma fase de intensificação sobre as soluções obtidas na fase de busca local, procurando produzir uma solução de qualidade superior à melhor solução obtida na busca local.

Em particular, o procedimento de reconexão por caminhos (PRC) adotado neste trabalho foi aplicado às soluções elite obtidas após a última geração do BRKGA. Mas especificamente, dentre as p_e soluções, são tomadas as q melhores (valor da função objetivo), definindo um conjunto $C_e=\{m_1,...,m_i,...,m_q\}$, sendo cada $m_i \in C_e$ correspondente a um vetor que contém os *k*-medoids utilizados para a definição dos grupos. Em seguida, toma-se cada dupla de soluções de C_e (considerando todas as combinações das soluções de C_e tomadas a duas a duas), sendo cada dupla associada a uma solução s_1 e uma solução s_2. Ou seja, são definidos dois vetores s_1 e s_2 com *k*-posições e que contêm os objetos correspondentes aos *k*-medoids.

Para cada um dos medoids de s_1 e s_2, são tomados os seus $n_p=3$ objetos mais próximos, segundo a distância euclidiana, sendo escolhidos um total de $2.n_p.k$ objetos que serão utilizados para compor as soluções intermediárias (novos medoids). As tabelas 1 e 2 a seguir exemplificam a construção das soluções intermediárias, considerando duas soluções s_1 e s_2 . Mais especificamente, a tabela 1 traz para cada um dos medoids de s_1 e s_2 os seus 3 vizinhos (objetos) mais próximos e a tabela 2 mostra as soluções intermediárias que são produzidas mudando o 1º, o 2º e 3º medoids de s_1 e de s_2 , ou seja, substituindo pelos respectivos vizinhos de s_1 e de s_2. Para cada solução intermediária s_i efetua-se a alocação dos *(n-k)* objetos restantes ao seu medoid mais próximo e calcula-se o valor da função objetivo. Se o valor função for inferior ao valor da melhor solução produzida pelo BRKGA, atualiza-se a solução (os medoids).

Tabela 1 – Duas soluções do conjunto elite e seus três objetos mais próximos

	medoids		
s1	**3**	**17**	**24**
1o ponto mais próximo	7	37	49
2o ponto mais próximo	44	20	19
3o ponto mais próximo	11	28	33
3o ponto mais próximo	43	27	11
2o ponto mais próximo	16	7	38
1o ponto mais próximo	1	22	50
s2	**8**	**12**	**21**

Tabela 2 – Soluções Intermediárias com base nos três objetos mais próximos

m1	m2	m3
7	17	24
44	17	24
11	17	24
43	17	24
16	17	24
1	17	24
m1	**m2**	**m3**
1	12	21
16	12	21
43	12	21
11	12	21
44	12	21
7	12	21

m1	m2	m3
3	**37**	24
3	**20**	24
3	**28**	24
3	**27**	24
3	**7**	24
3	**22**	24
m1	**m2**	**m3**
8	**22**	21
8	**7**	21
8	**27**	21
8	**28**	21
8	**30**	21
8	**37**	21

m1	m2	m3
3	17	**49**
3	17	**19**
3	17	**33**
3	17	**11**
3	17	**38**
3	17	**50**
m1	**m2**	**m3**
8	12	**50**
8	12	**38**
8	12	**11**
8	12	**33**
8	12	**19**
8	12	**49**

5. Resultados Computacionais

Nesta seção são apresentados resultados computacionais obtidos dos algoritmos BRKGA-RC, GRASP, de ParkJun, PAM e CLARA. Os três primeiros algoritmos foram desenvolvidos em linguagem ***R***[2] e os algoritmos PAM e CLARA estão disponíveis na library *cluster* do software ***R.*** Além desses algoritmos, a formulação descrita na seção 3.1 foi programada no software LINGO (versão 12.0), sendo todos os experimentos realizados em um computador com 24GB de memória RAM e dotado de oito processadores de 3.40 GHz (*I7*). Aproveitando os recursos da arquitetura *multicore* do computador, e o fato de o software *R* disponibilizar uma library (*snowfall*) com funções de paralelismo, o algoritmo BRKGA foi paralelizado. Ou seja, considerando cada uma das instâncias, foram efetuadas oito chamadas ao BRKGA, sendo essas chamadas distribuídas dentre os oito processadores. Ao final do processamento, foi tomada a melhor solução produzida dentre as oito.

De forma a avaliar a eficiência do algoritmo BRKGA no que diz respeito à qualidade das soluções produzidas, foi realizado um conjunto de experimentos computacionais com 30 instâncias. Quanto à origem, essas instâncias são classificadas em três grupos, quais sejam: (1) da literatura, sendo citadas e utilizadas, por exemplo, na tese de Semaan (2013) e Cruz (2010); (2) nos sites do IBGE *(www.ibge.gov.br)* e da universidade da Califórnia (*archive.ics.uci.edu/ml/*) e (3) geradas artificialmente no software ***R***. A tabela a seguir traz o nome da instância, o seu número de objetos (***n***) e atributos (***f***) e a sua origem.

Tabela 3 – Informações sobre as Instâncias

Instância	n	f	Origem	Instância	n	f	Origem
DS1-200DATA	200	2	1	DS3-numbers2	540	2	1
DS1-broken-ring	800	2	1	DS3-outliers	150	2	1
DS1-chart	600	60	1	DS4-ecoli	336	7	2
DS1-gauss9	900	2	1	DS4-haberman	306	4	2
DS1-iris	150	4	1	DS4-idh	187	4	2
DS1-maronna	200	2	1	DS4-indian	583	9	2
DS1-ruspini	75	2	1	DS4-pib100	100	1	2
DS1-spherical_4d3c	400	3	1	DS4-synthetic_control	600	51	2
DS1-vowel2	528	2	1	DS4-wpbc	198	30	2
DS1-wine	178	13	1	DS4-SP_LAVOURA	566	1	2
DS1-yeast	1484	7	1	DS4-PIB_MINAS	853	1	2
DS2-400p3c	400	2	1	DS4-Gamma400	500	3	3
DS3-2face	200	2	1	DS4-Normal300	300	2	3
DS3-face	296	2	1	DS4-Uniform400	400	2	3
DS3-moreshapes	489	2	1	DS4-Uniform700	700	2	3

Nestes experimentos, todas as instâncias foram submetidas aos cinco algoritmos e à formulação, considerando o número de grupos (***k***) variando entre três e seis. Ou seja, foi produzido um total de 120 soluções (nº de instâncias x nº de grupos). A função objetivo, ou seja, a soma das distâncias dos objetos aos seus respectivos medoids foi calculada considerando a distância euclidiana. Esses parâmetros foram definidos a partir das recomendações feitas nos trabalhos de Gonçalves e Resende (2011) e Resende (2013), e partir de alguns experimentos computacionais realizados a *priori*. Mais especificamente, foram selecionadas dentre as 30 instâncias, cinco instâncias sobre as quais foi aplicado o algoritmo BRKGA (50 vezes sobre cada instância), considerando todas as combinações (1152) dos parâmetros: p=50, 75, 100 e 200; p_e=0.10, 0.15, 0.20 e 0.25; p_m=p_e ; ρ_e=0.7,0.8 e 0.9 e m=250, 300, 350, 400, 500 e 1000. A partir da realização deste procedimento, foram definidos os seguintes valores para os parâmetros acima: p=100, p_e=0.20p (10), p_m=0.15p, ρ_e =0.7 e m=400. Esses valores foram considerados nos experimentos computacionais realizados com as 30 instâncias. As tabelas 4 e 5 trazem os resultados produzidos pelos cinco algoritmos e pela formulação. Nessas tabelas, os valores destacados em cinza indicam os casos em que os algoritmos produziram o ótimo global, e os valores em negrito correspondem à melhor

[2] Linguagem R (versão 3.0.2 de 64 bits). (www.r-project.org/)

solução viável produzida por pelo menos um dos algoritmos. Das 120 soluções produzidas, a formulação produziu o ótimo global em **106** casos, uma solução viável em **9** casos e não produziu nenhuma solução em apenas 5 casos. Esse valor (106) foi utilizado como base de comparação para o cálculo do percentual de ótimos globais produzidos por cada um dos algoritmos (vide gráfico 1). Assim, em 96% dos casos o algoritmo BRKGA-RC produziu o ótimo global, seguido pelo algoritmo GRASP com 93% e pelos algoritmos PAM, de ParkJUN e CLARA, com respectivamente, 70%, 29% e 15%. Independente da instância considerada, e do número de grupos, o tempo consumido pelo PRC é bem que pequeno quando comparado ao tempo do BRKGA, ficando a inferior a 4% do tempo total. Além disso, em relação à melhor solução do BRKGA, o PRC produziu um ganho em cerca de 20% dos casos, sendo este ganho, em média, de 3%. No que se refere ao BRKGA, o tempo total de execução para cada instância corresponde à soma dos tempos consumidos nas gerações do BRKGA (1ª fase) e pelo procedimento de reconexão por caminhos (PRC) (2ª fase).

A tabela 6 traz, por grupo, média dos tempos de processamento dos cinco algoritmos e da formulação. Uma análise dessa tabela indica que os algoritmos de ParkJUN, PAM e CLARA tiveram um performance superior ao BRKGA-RC, ao GRASP e à formulação. Particularmente, em relação ao BRKGA-RC e ao GRASP, esse fato não é surpreendente, uma vez que esses dois algoritmos têm procedimentos de construção e de buscas locais mais sofisticados e intensivos, quando comparados com os demais algoritmos. Ainda que o algoritmo BRKGA-RC não seja tão eficiente quanto os demais algoritmos, no que concerne aos seus tempos de execução, o mesmo é bem eficaz no que diz respeito ao quantitativo de soluções ótimas. Neste sentido, vale destacar que há uma diferença significativa entre os percentuais associados aos quantitativos de ótimos globais do BRKGA-RC e dos algoritmos PAM, de ParkJUN e CLARA. Uma vez que os algoritmos BRKGA-RC e GRASP tiveram um desempenho superior em relação aos demais algoritmos e à formulação, foi efetuada uma comparação entre esses dois algoritmos no que se refere a todas as soluções produzidas. Ou seja, os ótimos locais e globais, com o algoritmo BRKGA apresentando um percentual de 93,3% e o GRASP 92,5%.

Os bons resultados apresentados para instâncias de dimensão variada indicam que o BRKGA-RC pode ser uma boa alternativa para resolução do problema dos ***k***-medoids. Não obstante, em trabalhos futuros, serão implementados novas decodificações e novos tipos de cruzamento, como por exemplo, os cruzamentos de um e dois pontos (Glover and Kochenberger, 2002).

Gráfico1

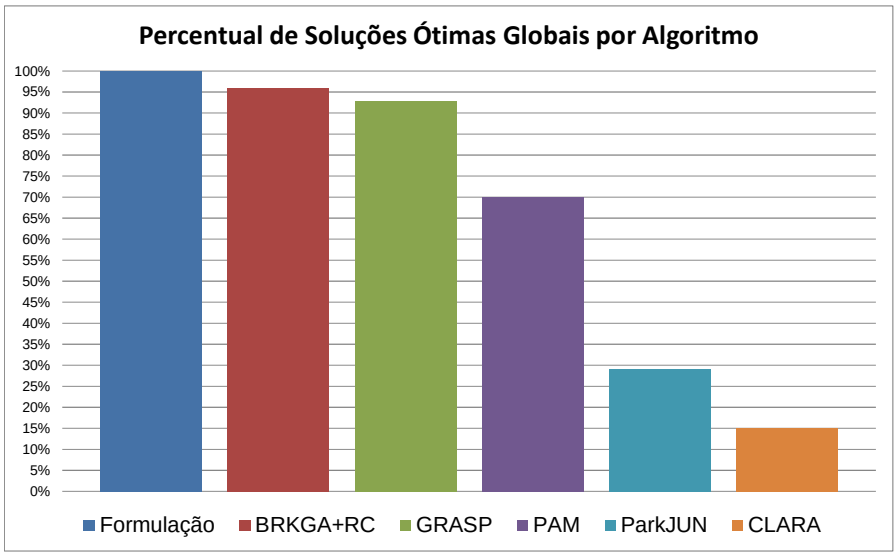


Tabela 4– Resultados dos Algoritmos e da Formulação (3 e 4 grupos)

Instância	n	k	PAM	CLARA	ParkJUN	GRASP	BRKGA-RC	Formulação	k	PAM	CLARA	ParkJUN	GRASP	BRKGA-RC	Formulaçã
DS1-200DATA	200	3	0,371755	0,371804	0,371755	0,371755	0,371755	0,371755	4	0,265603	0,265652	0,265603	0,265603	0,265603	0,265603
DS1-broken-ring	800	3	0,806440	0,804171	0,803858	0,803858	0,803858	0,803858	4	0,608921	0,608921	0,608921	0,608921	0,608921	0,608921
DS1-chart	600	3	7,708550	7,708550	7,866560	7,708550	7,708550	7,708550	4	7,598375	7,600272	7,905568	7,598375	7,598375	7,598375
DS1-gauss9	900	3	0,815595	0,817382	**0,812599**	**0,812599**	**0,812599**	1,35209*	4	**0,673955**	0,676209	0,675503	**0,673955**	**0,673955**	1,76724*
DS1-iris	150	3	0,875705	0,874722	0,868645	0,868645	0,868645	0,868645	4	0,776507	0,785464	0,797810	0,776507	0,776507	0,776507
DS1-maronna	200	3	0,695663	0,703237	0,695663	0,695663	0,695663	0,695663	4	0,472809	0,473640	0,472809	0,472809	0,472809	0,472809
DS1-ruspini	75	3	0,598321	0,602516	0,598321	0,598321	0,598321	0,598321	4	0,318706	0,323189	0,318706	0,318706	0,318706	0,318706
DS1-spherical_4d3c	400	3	0,562392	0,560452	0,569530	0,560452	0,560452	0,560452	4	0,333503	0,333503	0,333503	0,333503	0,333503	0,333503
DS1-vowel2	528	3	0,803448	0,804264	0,803457	0,803448	0,803448	0,803448	4	0,699827	0,703172	0,699559	0,696164	0,696164	0,696164
DS1-wine	178	3	2,808597	2,808597	2,808597	2,808597	2,808597	2,808597	4	2,688419	2,676843	2,711840	2,676843	2,676843	2,676843
DS1-yeast	1484	3	**1,829516**	1,843443	1,829968	**1,829516**	**1,829516**	-	4	1,738251	1,738876	1,748183	**1,732445**	1,734773	-
DS2-400p3c	400	3	0,280307	0,280370	0,280307	0,280307	0,280307	0,280307	4	0,239133	0,242666	0,239133	0,239133	0,239133	0,239133
DS3-2face	200	3	0,677785	0,678331	0,677785	0,677785	0,677785	0,677785	4	0,607911	0,597670	0,594638	0,592568	0,592568	0,592568
DS3-face	296	3	0,667511	0,667736	0,667511	0,667511	0,667511	0,667511	4	0,582965	0,586626	0,582973	0,582965	0,582965	0,582965
DS3-moreshapes	489	3	0,634762	0,636355	0,634762	0,634762	0,634762	0,634762	4	0,407425	0,407786	0,407425	0,407425	0,407425	0,407425
DS3-numbers2	540	3	0,791152	0,782119	0,780678	0,779784	0,779784	0,779784	4	0,601610	0,603293	0,601738	0,601610	0,601610	0,601610
DS3-outliers	150	3	0,318933	0,318933	0,318933	0,318933	0,318933	0,318933	4	0,232125	0,242325	0,242317	0,232125	0,232125	0,232125
DS4-ecoli	336	3	1,561043	1,561043	1,561043	1,561043	1,561043	1,561043	4	1,438577	1,454887	1,490302	1,438577	1,438577	1,438577
DS4-Gamma400	500	3	1,073824	1,074031	1,072392	1,070977	1,070977	1,070977	4	0,932886	0,935574	0,934164	0,932886	0,932886	0,932886
DS4-haberman	306	3	1,315699	1,329278	1,316286	1,315699	1,315699	1,315699	4	1,191291	1,198144	1,191291	1,191291	1,191291	1,191291
DS4-idh	187	3	0,286936	0,287343	0,287061	0,286936	0,286936	0,286936	4	0,218036	0,219884	0,218193	0,218036	0,218036	0,218036
DS4-indian	583	3	2,005019	2,009863	2,040977	2,005019	2,005019	2,005019	4	1,882190	1,882190	1,958390	1,880477	1,880477	1,880477
DS4-Normal300	300	3	0,819265	0,821541	0,821177	0,819265	0,819265	0,819265	4	0,716381	0,729088	0,716932	0,716381	0,716381	0,716381
DS4-PIB_MINAS	853	3	**0,090097**	0,090104	**0,090097**	**0,090097**	**0,090097**	0,926303*	4	**0,073264**	0,073287	0,075114	**0,073264**	**0,073264**	0,0951102*
DS4-pib100	100	3	0,254521	0,254309	0,254249	0,254249	0,254249	0,254249	4	0,177186	0,177917	0,177186	0,177186	0,177186	0,177186
DS4-SP_LAVOURA	566	3	0,183472	0,183472	0,202767	0,183472	0,183472	0,183472	4	0,138629	0,139351	0,175308	0,138629	0,138629	0,138629
DS4-synthetic_control	600	3	7,105057	7,105057	7,303236	7,105057	7,105057	7,105057	4	7,005453	7,005453	7,238325	7,005453	7,005453	7,005453
DS4-Uniform400	400	3	0,797123	0,797963	0,796979	0,796168	0,796168	0,796168	4	0,655958	0,662245	0,655958	0,655958	0,655958	0,655958
DS4-Uniform700	700	3	0,782995	0,784634	0,783143	0,782995	0,782995	0,782995	4	0,657854	0,660538	0,659118	0,657854	0,657854	0,657854
DS4-wpbc	198	3	4,507711	4,422156	4,407410	4,407410	4,407410	4,407410	4	4,186800	4,284158	4,324780	4,186800	4,186800	4,186800

(*) Melhor solução viável produzida pela formulação em 3 horas; "-" Nenhuma solução viável produzida em 3 horas

Tabela 5 – Resultados dos Algoritmos e da Formulação (5 e 6 grupos)

Instância	n	k	PAM	CLARA	ParkJUN	GRASP	BRKGA-RC	Formulação	k	PAM	CLARA	ParkJUN	GRASP	BRKGA-RC	Formulaçã
DS1-200DATA	200	5	0,236501	0,237509	0,233511	0,233494	0,233494	0,233494	6	0,213090	0,217450	0,217690	0,213090	0,213090	0,213090
DS1-broken-ring	800	5	0,515006	0,515067	0,515006	0,515006	0,515006	0,515006	6	0,468672	0,469272	0,468750	0,468326	0,468474	0,468326
DS1-chart	600	5	7,522857	7,522857	7,643665	7,522857	7,522857	7,522857	6	7,451490	7,451490	7,610846	7,451490	7,451490	7,451490
DS1-gauss9	900	5	**0,588689**	0,592890	0,595372	0,588759	**0,588689**	0,646526*	6	0,524183	0,522784	0,521012	0,520916	**0,520738**	-
DS1-iris	150	5	0,707165	0,698771	0,709166	0,697853	0,697853	0,697853	6	0,654021	0,667408	0,667386	0,654021	0,654021	0,654021
DS1-maronna	200	5	0,434895	0,436112	0,429668	0,429668	0,429668	0,429668	6	0,397353	0,399424	0,390348	0,390071	0,390071	0,390071
DS1-ruspini	75	5	0,287457	0,296047	0,287841	0,287457	0,287457	0,287457	6	0,256593	0,268778	0,259555	0,256593	0,256593	0,256593
DS1-spherical_4d3c	400	5	0,317516	0,318649	0,316503	0,315834	0,315834	0,315834	6	0,307834	0,301160	0,300516	0,299847	0,299777	0,299777
DS1-vowel2	528	5	0,613589	0,615378	0,604448	0,603675	0,603938	0,603675	6	0,533956	0,536652	0,533483	0,534082	0,533619	0,533483
DS1-wine	178	5	2,574040	2,599080	2,628861	2,574040	2,574040	2,574040	6	2,502357	2,504289	2,520055	2,494848	2,494848	2,494848
DS1-yeast	1484	5	1,662371	1,651881	1,658370	**1,643712**	1,644052	-	6	1,588827	1,577949	1,612636	**1,567688**	**1,567688**	-
DS2-400p3c	400	5	0,213192	0,215115	0,213193	0,212919	0,212919	0,212919	6	0,193235	0,194070	0,193627	0,193213	0,193213	0,193213
DS3-2face	200	5	0,513669	0,525588	0,515618	0,513669	0,513669	0,513669	6	0,455188	0,461922	0,455196	0,455188	0,455188	0,455188
DS3-face	296	5	0,521134	0,532389	0,521134	0,521134	0,521134	0,521134	6	0,465789	0,475005	0,467891	0,465789	0,465789	0,465789
DS3-moreshapes	489	5	0,293581	0,294589	0,351932	0,293581	0,293581	0,293581	6	0,238088	0,239014	0,238088	0,238088	0,238088	0,238088
DS3-numbers2	540	5	0,525319	0,528666	0,525319	0,525488	0,525319	0,525319	6	0,448407	0,448517	0,478524	0,448407	0,448407	0,448407
DS3-outliers	150	5	0,188842	0,188842	0,219296	0,188842	0,188842	0,188842	6	0,166780	0,166780	0,177211	0,158387	0,158387	0,158387
DS4-ecoli	336	5	1,360999	1,374475	1,419336	1,360999	1,360999	1,360999	6	1,297382	1,316910	1,303797	1,288955	1,288955	1,288955
DS4-Gamma400	500	5	0,869367	0,883886	0,880912	0,869367	0,869367	0,869367	6	0,815423	0,831357	0,827600	0,817467	0,815423	0,815423
DS4-haberman	306	5	1,107550	1,114506	1,128823	1,107550	1,107550	1,107550	6	1,029185	1,058116	1,043185	1,029185	1,029185	1,029185
DS4-idh	187	5	0,179139	0,177730	0,175005	0,174943	0,174943	0,174943	6	0,136609	0,138332	0,162822	0,136609	0,136609	0,136609
DS4-indian	583	5	1,802988	1,806585	1,823389	1,800296	1,798827	1,798827	6	1,730287	1,739832	1,747471	1,725689	1,726056	1,725689
DS4-Normal300	300	5	0,656631	0,662713	0,658334	0,656631	0,656631	0,656631	6	0,605240	0,610877	0,611184	0,605250	0,604765	0,604765
DS4-PIB_MINAS	853	5	**0,057538**	0,057903	0,060659	**0,057538**	**0,057538**	0,0584092*	6	0,043609	0,050588	0,067939	0,043609	0,043609	0,052954*
DS4-pib100	100	5	0,125105	0,125225	0,125106	0,125105	0,125105	0,125105	6	0,102070	0,104412	0,102832	0,102070	0,102070	0,102070
DS4-SP_LAVOURA	566	5	0,115764	0,110316	0,147455	0,109731	0,109731	0,109731	6	0,087705	0,087819	0,134655	0,087705	0,087705	0,087705
DS4-synthetic_control	600	5	6,925285	6,925285	7,137230	6,925285	6,925285	6,925285	6	6,860733	6,871276	6,976996	6,860733	6,860733	6,860733
DS4-Uniform400	400	5	0,597190	0,609862	0,603585	0,597190	0,597190	0,597190	6	0,548147	0,555578	0,554524	0,544417	0,543508	0,543466
DS4-Uniform700	700	5	**0,599060**	0,601602	0,600530	**0,599060**	**0,599060**	0,599060*	6	0,552360	0,549457	0,551564	**0,546404**	0,546439	0,546350*
DS4-wpbc	198	5	4,026464	4,092351	4,129980	4,026464	4,026464	4,026464	6	3,912867	3,925242	4,057367	3,912867	3,912867	3,912867

(*) Melhor solução viável produzida pela formulação em 3 horas; "-" Nenhuma solução viável produzida em 3 horas

Tabela 6 – Média dos Tempos de Processamento dos Algoritmos e da Formulação

Algoritmo	k=3	k=4	k=5	k=6
PAM	0,04	0,06	0,07	0,08
CLARA	0,00	0,00	0,00	0,00
PARKJUN	1,49	1,53	1,49	1,49
GRASP	14,53	16,53	16,70	22,00
BRKGA-RC	98,92	99,30	98,09	99,36
FORMULAÇAO	1270,50	1237,53	1728,67	1752,80

***Tempos de Processamento em segundos**

5. REFERÊNCIAS BIBLIOGRAFICAS

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

[2] Brito, J.A.M., Ochi L.S., Brito L.R. e Montenegro, F.M.T (2010). Um algoritmo para o agrupamento baseado em K-Medoids. *Revista Brasileira de Estatística*, **71**, p. 75-99.

[3] Brito, J.A.M. e Semaan, G.S. (2013). Um Algoritmo Grasp Aplicado ao Problema dos *k*-Medoids. XLV Simpósio Brasileiro de Pesquisa Operacional, Natal. Anais do XLV Simpósio Brasileiro de Pesquisa Operacional.

[4] Chu, S.C., Roddick J.F and Pan J.S. (2008). Improved Search Strategies and Extensions to k-medoids-based Clustering Algorithms. *International Journal of Business Intelligence and Data Min*ing, **3**, 2, 212-231.

[5] Church, R. (1978). Contrasts Between Facility Location Approaches and NonHierarquical Cluster Analysis. Paper presented at ORSA/TIMS Joint National Meeting, Los Langeles, California, nov. 1978.

[6] Cruz, M. D. (2010). O problema de clusterização automática, Tese de Doutorado, COPPE/UFRJ.

[7] Feo, T.A. e Resende, M.G.C. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6, pp. 109-133, 1995.

[8] Glover, F. e Kochenberger, G. A. (2002). *"Handbook of Metaheuristic"*, First Edition Norwell: K1uwer Academic Publishers, 2002.

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

[10] Han, J. and Ng, R. (2002). "CLARANS:A Method for Clustering Objects for Spatial DataMining. IEEE *Transactions Knowledge of Data Enginnering* 14(5), pp. 1003-1016.

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

[12] Johnson A.R. e Wichern D.W. (2002). *Applied Multivariate Statistical Analysis*. Prentice Hall. Fifth Edition.

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

[14] Mingoti, S.A. (2007). Análise de Dados Através de Métodos de Estatística Multivariada – Uma Abordagem Aplicada, UFMG, 1ª Edição.

[15] Mulvey, J. and Cowder, H. (1979). Cluster Analysis: An Application of Lagrangian Relaxation. *Management Science*, **25**, 329-340.

[16] Naldi, C. N. (2011). *Técnicas de Combinação para Agrupamento Centralizado e Distribuído de Dados*. Tese de Doutorado, USP - São Carlos.

[17] Park H.S. and Jun C.H. (2009). A Simple and Fast Algorithm for K-medoids Clustering. *Expert Systems with Applications*, **36**, 2, 3336-3341.

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

[19] Semaan, G.S. (2013). *Algoritmos para o Problema de Agrupamento Automático*. Tese de Doutorado, UFF/IC.

[20] Sheng W. and Liu X. (2004). A Hybrid Algorithm for K-medoid Clustering of Large Data Sets. Congress on Evolutionary Computation, 1, 77-82.

[21] Spears, W.M e DeJong, K.A. (1991). On the virtues of parameterized uniform crossover. Em Proceedings of the Fourth International Conference on Genetic Algorithms, 230-236.

[22] Tryon, R. (1939). Cluster Analysis. Oxford, England: Edward Bros.

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

[24] Zhang, Q. and Couloigner, I. (2005). A New Efficient k-medoid Algorithm for Spatial Clustering. *Lecture Notes in Computer Science*, v3482, 181-189.

Agradecimentos: Ao CNPQ (projeto 475245/2012-1) pelo financiamento parcial deste estudo.