Um Algoritmo GRASP Aplicado ao Problema dos *k*-Medoids

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
Universidade Federal Fluminense – Instituto de Computação – UFF/IC
Caixa Postal 68511 , CEP: 21941-972, Rio de Janeiro, RJ, Brasil
e-mail: gsemaan@ic.uff.br

RESUMO

O presente trabalho traz uma nova proposta para a resolução de um problema de agrupamento conhecido na literatura como o problema dos *k*-medoids. Neste problema, dado um conjunto de ***n*** objetos com ***p*** atributos, e fixado o número de grupos, deve-se selecionar, dentre os ***n*** objetos, ***k*** objetos denominados medoids (ou objetos representativos). Os demais objetos serão alocados ao medoid correspondente mais próximo, segundo uma medida distância. Desta forma, busca-se a determinação dos ***k*** medoids de forma que a soma das distâncias dos demais objetos ao seu respectivo medoid seja a menor possível. Assim como em outros problemas de agrupamento, este problema é NP-difícil. De forma a resolver este problema, propõe-se no presente trabalho um algoritmo que utiliza os conceitos da metaheurística GRASP. A última seção traz alguns resultados computacionais para um conjunto de instâncias da literatura, considerando a aplicação do GRASP, de três algoritmos da literatura e de uma formulação exata.

PALAVRAS CHAVE: Medoids, Análise de Agrupamentos, Metaheurística GRASP. MH.

ABSTRACT

This paper presents a new proposal to solve a clustering problem known in the literature as the *k*-medoids clustering problem. Given a set of objects (each one has *p* attributes) and given a specific number of groups (*k*), this problem aims to select k objects, called medoids *k* objects which minimize a fitness function. The remaining objects are allocated to the corresponding medoid closest, according to a distance measure. Thus, it seeks to determine the k medoids which minimize the sum of distances from other objects to their respective medoids. This problem is NP-hard and in order to solve it, this paper proposes a heuristic algorithm based on GRASP metaheuristic. The computational results are presented in the last section, and a set of instances from the literature are considered. Furthermore, the results of proposed GRASP were compared with others three algorithms of the literature and one exact formulation.

KEYWORDS: Medoids, Cluster Analysis, GRASP, metaheuristics.

1. Introdução

Segundo Tan et al. (2009) a mineração de dados combina técnicas tradicionais de análise de dados e algoritmos sofisticados para processar grandes bases de dados. A partir da aplicação destas técnicas e algoritmos é possível produzir informações úteis e, consequentemente, análises importantes no que concerne às bases de dados. Em particular, tem sido crescente nos últimos anos o número de aplicações estudadas em mineração de dados que podem ser mapeadas em um problema de agrupamento. Neste sentido, a análise de agrupamentos é uma tarefa de mineração de dados que tem sido muito explorada e utilizada em diversos domínios como a Biologia, Estatística, Seguros, Medicina, dentre outros.

Basicamente, a análise de agrupamentos é uma tarefa da etapa de mineração de dados no processo de descoberta de conhecimento em bases de dados (do inglês *Knowledge Discovery in Databases* (KDD)) que tem por objetivo a construção de grupos (ou *clusters*) a partir de uma base de dados composta por $\boldsymbol{n}$ objetos (registros) com $\boldsymbol{p}$ 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 a euclidiana.

Em face das inúmeras aplicações de análise de agrupamentos e da complexidade de resolução deste problema, encontram-se na literatura diversos métodos de agrupamento que podem ser classificados, basicamente, como não hierárquicos e hierárquicos. Em particular, propõe-se neste trabalho um novo algoritmo de agrupamento não hierárquico para a resolução do problema dos $\boldsymbol{k}$-medoids. Este algoritmo utilizou os conceitos da metaheurística GRASP.

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 comenta os principais trabalhos da literatura. Na seção quatro é feita uma descrição da metaheurística GRASP e do novo algoritmo proposto. E finalmente, a seção cinco traz um conjunto de resultados computacionais obtidos a partir da aplicação deste algoritmo, da formulação e de três algoritmos da literatura em instâncias da literatura e artificiais.

2. Análise de Agrupamentos

Atualmente, a análise de agrupamentos 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.

Em linhas gerais, 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 clusters (ou grupos).

Estes clusters, por sua vez, são definidos de forma que os objetos alocados a um mesmo cluster sejam similares entre si (considerando alguma métrica) e que os objetos pertencentes a grupos diferentes sejam dissimilares. Neste caso, a métrica está associada a uma medida de distância que considera os p atributos (variáveis) disponíveis para cada um dos n objetos. Dentre estas medidas, destacam-se: a distância euclidiana, a distância de Manhattan e a distância de Minkowski (Kaufman e Rousseeuw, 1989).

As medidas associadas à similaridade ou à dissimilaridade entre os n objetos (tomados dois a dois) são obtidas por meio de transformações a partir de dados quantitativos ou qualitativos (Kaufman e Rousseeuw, 1989). A figura 1 traz um exemplo de agrupamento baseado no posicionamento de oito objetos com dois ($\boldsymbol{p}$=2) atributos.

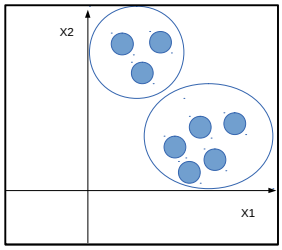


Figura 1- Exemplo de um Agrupamento

Se o número de grupos é um dos parâmetros definidos a priori, temos um problema de agrupamento clássico. Caso contrário, define-se um problema de agrupamento automático (Cruz, 2010 e Naldi, 2011). Formalmente, dado um conjunto ***X*** formado por ***n*** objetos, ***X***= $\{x_1, x_2,..., x_n\}$ com ***p*** atributos e, definido um inteiro ***k*** correspondente ao número de clusters, devem ser construídos ***k*** clusters $C_1, C_2,..., C_k$, considerando as três restrições a seguir:

(i) $C_1 \cup C_2 \cup ... \cup C_k = \boldsymbol{X}$
(ii) $C_i \cap C_j = \varnothing$, i,j=1,...,***k*** (i<j)
(iii) $|C_i| \geq 1$, i=1,...,***k***

A restrição (i) indica que a união dos clusters corresponde ao conjunto ***X***. A restrição (ii) indica que um objeto pertence a exatamente um cluster e a restrição (iii) garante que não existem clusters vazios.

Em geral, a determinação do agrupamento ótimo para muitas aplicações reais é impraticável. Entende-se por agrupamento ótimo a partição de ***X*** que produz os clusters mais homogêneos, segundo algum critério de similaridade (medida de distância). Este fato é decorrente de que o número de partições (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 (Johnson e Wichern, 2002), dado pela equação abaixo:

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

Supondo, por exemplo, que n = 20 objetos serão alocados em dois clusters, o número de soluções a serem consideradas é de 524.287. Mas, mantendo o mesmo número de clusters e apenas dobrando o número de objetos, temos 549.755.813.887 soluções possíveis. Ao considerar um número n maior de objetos, estes valores crescem exponencialmente. Este fato torna proibitiva a resolução deste problema mediante a aplicação de um método de enumeração exaustiva.

Não obstante, abrindo-se mão do ótimo global, é possível produzir soluções viáveis de qualidade razoável a expensas de um tempo computacional factível. Essas soluções podem ser produzidas mediante a aplicação de um método hierárquico ou de um método não hierárquico (Hair *et al.*, 2009, Mingoti, 2007). Estes métodos encontram soluções (*clusters*) de boa qualidade (razoáveis) sem examinar todas as soluções possíveis.

Os métodos hierárquicos, por sua vez, dividem-se em aglomerativos e divisivos. Em um método aglomerativo, inicialmente há ***n*** grupos de um objeto cada, sendo efetuadas uma série de uniões até obter ***k*** grupos. Já no método divisivo, inicialmente há um único grupo formado por ***n*** objetos, sendo efetuadas sucessivas divisões dos grupos até que sejam atingidos ***k*** grupos.

Existem vários métodos hierárquicos que diferem somente na escolha do critério de partição. Uma desvantagem desses métodos é a possibilidade de tornarem-se impraticáveis para grandes conjuntos de dados, devido à alta complexidade computacional do problema (Michaud, 1997). Os métodos não hierárquicos procuram encontrar uma partição viável dos n objetos sem a necessidade de associações hierárquicas. Primeiramente, uma partição inicial com um determinado número k de clusters deve ser considerada. A seguir, seleciona-se uma partição dos n objetos em k grupos, otimizada segundo algum critério. Dentre os métodos não hierárquicos disponíveis na literatura, os mais utilizados são o *k*-means (Mingoti, 2007) e dos *k*-medoids (Kaufman e Rousseeuw, 1989). Em particular, este último método está associado ao problema que foi o objeto de estudo deste trabalho.

3. O Problema dos k-medoids

Dado um conjunto ***X*** constituído por *n* objetos X= $\{x_1,x_2,...,x_n\}$ com *p* atributos (quantitativos e/ou qualitativos), deve-se selecionar, dentre os ***n*** objetos de ***X,*** um subconjunto de ***k*** objetos que definem o conjunto de medoids ***M***=$\{x_{m1},x_{m2},...,x_{mk}\}$, também chamados de objetos representativos (Kaufman e Rousseeuw**,** 1989).

Uma vez definidos os medoids, os ***(n-k)*** objetos restantes serão alocados ao grupo G_i (i=1,..., ***k***) cujo medoid esteja mais próximo, segundo alguma medida de distância, como, por exemplo, a distância euclidiana. Sendo assim, o conjunto ***M*** é definido de forma que minimizar a função objetivo definida pela média (por grupo) das distâncias de todos os objetos aos seus respectivos medoids:

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

A figura abaixo ilustra um exemplo deste problema com sete objetos com dois atributos e dois medoids. Neste exemplo, temos que os objetos sete e quatro são alocados ao grupo 1, cujo medoid corresponde ao objeto dois. E, de igual forma, os objetos um, três e cinco são alocados ao grupo 2, cujo medoid corresponde ao objeto seis. Considerando estes grupos, temos que ***f*** = $d_{24}+d_{27}+d_{61}+d_{63}+d_{65}$

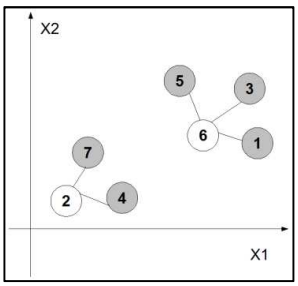


Figura 2 - Agrupamento com Dois Medoids

Conforme comentado na seção anterior, a obtenção do ótimo global para problemas de agrupamento, em particular o problema dos k-medoids, mediante a aplicação de um método de enumeração exaustiva é impraticável, tendo em vista a substancial quantidade de soluções que devem ser avaliadas. Alternativamente, este ótimo pode ser obtido através da resolução da formulação de programação matemática (Kaufman e Rousseeuw**,** 1989) apresentada a seguir.

3.1 Formulação e Algoritmos da Literartura

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

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

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

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

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

Nesta formulação, y_i é uma variável binária que assume valor um se o objeto x_i é definido como medoid, e zero em caso contrário, e x_{ij} é uma variável binária que assume valor um se o objeto x_j é alocado ao grupo definido pelo medoid x_i. A restrição (4) garante que cada objeto x_j será alocado a exatamente um grupo. A restrição (5) garante que um objeto x_j será associado a um objeto x_i somente se este objeto for um medoid e a restrição (6) garante que serão escolhidos exatamente k objetos como medoids. Finalmente, a função objetivo (3) minimiza a soma das distâncias d_{ij} (dissimilaridades) entre os objetos x_j e os seus respectivos medoids.

Analisando x_{ij} e y_i, pode-se observar que o número de variáveis binárias desta formulação é de ordem quadrática, mais especificamente, a formulação acima tem n^2+n variáveis binárias e n^2+n+1 restrições. Apenas para que se tenha uma ideia do porte dessa formulação, se forem considerados $n = 1000$ objetos, o número de variáveis binárias será da ordem de 1.000.000.

Em função desta última observação, a aplicação dessa formulação é pertinente quando o número de objetos é pequeno, ou seja, no máximo da ordem de centenas. E mesmo assim, a sua resolução pode levar ao consumo expressivo de tempo computacional produzindo, em muitos casos, apenas uma solução subótima. Não obstante, abrindo-se mão do ótimo global, é possível produzir soluções viáveis mediante a aplicação de alguns algoritmos heurísticos disponíveis na literatura. Neste sentido, um dos algoritmos mais conhecidos e utilizados para este problema é o algoritmo PAM (*Partitioning Around Medoids*), proposto por Kaufman e Rousseeuw (1989). Em linhas gerais, esta 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 que atua sobre a solução produzida pelo procedimento *Build*.

Kaufman e Rousseeuw (1989) também propuseram uma versão modificada da heurística PAM, denominada CLARA (*Clustering Large Applications*). Esta 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, esta 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).

Em um trabalho mais recente, Han and 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, a seção seguinte traz a proposta de um novo algoritmo para o problema dos *k*-medoids, que teve por base, as ideias da metaheurística GRASP. Inicialmente, com objetivo de propiciar um bom entendimento dessa

proposta, são apresentados os conceitos da metaheurística GRASP, fazendo-se em seguida uma descrição do novo algoritmo.

4. Metodologia Proposta

4.1 Metaheurística GRASP

A metaheurística *GRASP* (*Greedy Randomized Adaptive Search Procedure*) (Glover and Kochenberger, 2002) é um procedimento iterativo utilizado para resolver problemas de otimização combinatória. Cada iteração desta metaheurística consiste de duas fases, sejam elas: construção e busca local. A fase de construção é caracterizada pela produção de uma solução viável S_o cuja vizinhança é investigada na fase busca local. A melhor solução obtida após todas as iterações será a solução do problema.

Para construção da solução S_o é considerada uma lista de candidatos *(LC)* formada por todos os elementos que se incorporados em S_o não a tornam inviável. Uma vez definida a ***LC***, deve-se avaliar todos os seus elementos mediante a aplicação de uma função gulosa ***g(.)***, que representa o custo de se adicionar um novo elemento $t \in LC$ na solução S_o. Com o objetivo de possibilitar uma maior variabilidade nas soluções produzidas nesta fase, define-se uma lista de candidatos restrita ***(LCR)***, formada pelos melhores elementos avaliados na *LC* através de uma função ***g.*** Ou seja, aqueles que se incorporados à solução em construção produzem um acréscimo mínimo (problema de minimização). Este processo representa o aspecto *guloso* do *GRASP*, pois se considera sistematicamente os melhores elementos para serem inseridos na solução.

O trabalho de Feo e Resende (1995) descreve dois possíveis esquemas para a construção da ***LCR***, sejam eles: (i) um inteiro ***q*** é fixado e os ***q*** melhores candidatos, ordenados na ***LC*** segundo algum critério, são selecionados para compor a ***LCR*** e (ii) em cada iteração da fase de construção, denota-se respectivamente por g_{min} e g_{max} os menores e maiores acréscimos provocados pela inserção de um elemento $t \in LC$ na solução, segundo a avaliação de um função gulosa ***g(.)*** . A partir da aplicação dessa função e da utilização dos valores g_{min} e g_{max}, pode-se definir:

$$LCR=\{ t \in LC \mid g(t) \leq g_{min} + \alpha(g_{max} - g_{min}), \alpha \in [0,1]\}.$$

Quando α=0, o procedimento de construção torna-se *guloso* e quando α=1, produz-se uma solução aleatória, tendo em vista que a ***LCR*** terá todos os elementos da ***LC***. Tomando-se um valor intermediário para α o procedimento torna-se semi-guloso.

4.2 O Algoritmo GRASP para o Problema dos k-Medoids

4.2.1 Representação das Soluções

Assim como em outros problemas de otimização, o primeiro passo para a aplicação do algoritmo consiste da definição de uma estrutura que represente a solução. Neste sentido, cada solução produzida pelo GRASP proposto corresponde a um vetor ***v*** com ***k*** posições, sendo cada posição ocupada pelo código do objeto que foi escolhido como medoid. Supondo, por exemplo, n=50 e k=3, temos que ***v*** ={4,17,32} é uma solução onde os medoids são os objetos 4, 17 e 32.

Uma vez definidos os medoids, e por consequência os clusters, alocação de cada um dos ***(n-k)*** objetos restantes é feita conforme descrito na seção três, ou seja, alocando cada objeto ao medoid cuja distância seja mínima.

4.2.2 Procedimento de construção:

Inicialmente, define-se uma lista de candidatos LC cujos elementos correspondem aos ***n*** objetos da base de dados e em seguida, calcula-se para cada um dos ***n*** objetos, a soma das distâncias deste objeto aos demais (equação 8):

$$d_i = \sum_{j=1|\, j<>i}^{n} d_{ij}, i = 1,...,n \qquad (8)$$

A partir destas distâncias, define-se para cada um dos objetos x_i, uma probabilidade de seleção $\boldsymbol{p_i}$ tal que $\boldsymbol{p_i}=1/\boldsymbol{d_i}$. Dessa forma, quanto menor a soma associada à $\boldsymbol{d_i}$, maior será a probabilidade de seleção desse objeto como primeiro medoid a compor a solução ***So.*** Uma vez selecionado o primeiro objeto, atualiza-se a ***LC*** e cada um ***(k-1)*** medoids restantes serão definidos considerando os passos listados a seguir:

(1) Selecionar da ***LC*** uma amostra aleatória de tamanho ***l*** correspondente a 15% dos objetos restantes.

(2) Combinar cada objeto da amostra com os ***s*** (***s***<k) medoids adicionados até o momento em $\boldsymbol{S_o}$, ou seja, há ***l*** vetores correspondentes às ***l*** soluções parciais (medoids).

(3) Para cada um dos vetores, alocar os (***n-s***-1) objetos restantes ao seu medoid mais próximo.

(4) Calcular o valor da função objetivo apresentada na equação dois, que corresponderá ao valor da função gulosa $\boldsymbol{g_i}$.

(5) Definir a lista de candidatos restrita (LCR) segundo a seguinte equação:
$\boldsymbol{LCR}=\{\, \boldsymbol{i} \in \boldsymbol{LC} \mid \boldsymbol{g_i} \leq \boldsymbol{g_{min}}+0.2(\boldsymbol{g_{max}}-\boldsymbol{g_{min}})\}$.

(6) Selecionar da ***LCR*** um elemento ***i*** que corresponde ao objeto x_i que foi combinado com os medoids que compõem a solução $\boldsymbol{S_o}$ até o momento.

(7) Adicionar $\boldsymbol{x_i}$ a $\boldsymbol{S_o}$ e remover $\boldsymbol{x_i}$ da ***LC***.

Na construção acima, $\boldsymbol{g_{max}}$ e $\boldsymbol{g_{min}}$ correspondem, respectivamente, à pior e a melhor solução parcial (conjunto de objetos associados aos medoids) considerando a função objetivo da equação (2). Os passos acima são repetidos até que a solução ***So*** tenha ***k*** objetos selecionados da ***LC*** correspondentes ao medoids. O exemplo a seguir ilustra a aplicação do procedimento de construção, supondo que ***n***=20 e ***k***=4, e que o primeiro objeto selecionado (considerando as probabilidades de seleção $\boldsymbol{p_i}$) seja o objeto 10 , temos que ***So***={10}.

Quadro 1- Exemplo de uma solução considerando o procedimento de construção do GRASP

Solução até o momento (parcial)	Objetos selecionados da *LC*	Combinações Avaliadas (medoids candidatos)	Objetos da *LCR*	Objeto Selecionado da *LCR*
***So*={10}**	**3, 8, 9, 12, 19**	**{10,3},{10,8},{10,9}, {10,12}, {10,19}**	**3, 12, 19**	**3**
***So*={10,3}**	**1, 4, 12,17, 18**	**{10,3,1},{10,3,4}, {10,3,12},{10,3,17} {10,3,18}**	**4, 12, 17, 18**	**17**
***So*={10,3,17}**	**2, 5,14,16**	**{10,3,17,2},{10,3,17,5}, {10,3,17,14},{10,3,17,16}**	**2, 14**	**14**

Após inserção do elemento 14, temos uma solução inicial ***So*** para o problema dos ***k***-medoids formada por {10, 3, 17, 14} que correspondem aos medoids iniciais.

4.2.2 Procedimento de Busca Local:

Uma vez aplicado procedimento de construção, a busca local produz novas soluções explorando uma estrutura de vizinhanças associada aos medoids de $\boldsymbol{S_b}$ (melhor solução obtida até o momento, que inicialmente é a solução $\boldsymbol{S_o}$). Mais especificamente, seleciona-se o cluster mais heterogêneo (pior), ou seja, com maior valor de função objetivo, e aplica-se uma ordenação às distâncias dos objetos deste cluster ao seu medoid. Em seguida, efetua-se o cálculo dos decis considerando a distribuição destas distâncias. Os valores dos decis serão utilizados para definir

quatro vizinhanças, quais sejam: **vizinhança 1 (V_1)** = 1º decil e 9º decil; **vizinhança 2 (V_2)**= 2º decil e 8º decil; **vizinhança (V_3)** = 3º decil e 7º decil e **vizinhança (V_4)** = 4º decil e 6º decil.

Ao aplicar a vizinhança um, são selecionados do cluster pior todos os objetos cujas distâncias ao respectivo medoid sejam menores ou iguais ao 1º decil ou cujas distâncias sejam maiores ou iguais ao 9º decil, sendo feita uma construção análoga para as demais vizinhanças. Todos os objetos selecionados em cada uma das vizinhanças são utilizados como medoids que são combinados com os medoids dos demais clusters, com a vizinhança 1 tomada como ponto de partida. São listados a seguir os passos da busca local:

(1) Verificar qual o cluster mais heterogêneo em S_b.

(2) Calcular os decis relacionados às distâncias desse cluster.

(3) Definir as vizinhanças e o seus respectivos objetos.

(4) Tomar cada um dos objetos da *q*-ésima vizinhança e combinar com os medoids dos demais clusters.

(5) Alocar os ***(n-k)*** objetos restantes ao seu medoid mais próximo.

(6) Recalcular a função objetivo.

(7) Se houver redução na função, atualize S_b, faça $q=1$ (vizinhança 1) e retorne ao passo (1). Caso contrário, incremente a vizinhança ($q=q+1$) e vá para o passo 4.

A busca acima será finalizada quando as quatro vizinhanças tiverem sido exploradas e nenhuma melhoria for alcançada na última vizinhança. Uma vez concluída a busca local, verifica-se, dentro de cada um dos clusters, qual o objeto tem a menor a soma de suas distâncias aos demais objetos, sendo este objeto definido como novo medoid.

Considere, por exemplo, que $k=3$, $n=100$, So ={13, 25,36}, $n_1=30$, $n_2=25$, $n_3=45$ (nº de objetos alocados a cada cluster) e que o primeiro cluster fosse o mais heterogêneo. Neste caso, os decis seriam calculados a partir das 29 distâncias entre cada um dos objetos e o medoid 13, implicando na construção de 29 soluções compostas por cada um destes objetos e os objetos 25 e 36. Por sua vez, considerando cada uma destas soluções, efetua-se alocação dos 97 objetos restantes ao medoid mais próximo e calcula-se o valor da função objetivo. Se houver uma redução no valor da função, atualiza-se a solução S_b com os novos medoids e retorna-se ao passo inicial da busca local, qual seja, a identificação do pior cluster e o recálculo dos decis. Caso a melhoria não ocorra, procede-se à busca na vizinhança dois, considerando a combinação entre cada um dos objetos associados a esta vizinhança e os objetos 25 e 36, sendo este procedimento aplicado nas quatro vizinhanças.

5. Resultados Computacionais

A presente seção traz um conjunto de resultados computacionais obtidos dos algoritmos GRASP, Park e Jun, PAM e CLARA. Os dois primeiros algoritmos foram implementados em linguagem ***R*** (versão 2.15.2) e os algoritmos PAM e CLARA estão disponíveis na library **cluster** do software ***R.*** Além desses algoritmos, programou-se no software LINGO (versão 12.0) a formulação descrita na seção 3.1. Todos os experimentos foram efetuados em um computador com 24GB de memória RAM e dotado de oito processadores de 3.40 GHz (*I7*). Em face desta última observação e do software R disponibilizar uma *library* com funções de paralelismo, o procedimento de construção do GRASP foi paralelizado, com objetivo de reduzir o tempo computacional do algoritmo. Mais especificamente, ao invés de efetuar ***m*** execuções combinando os procedimentos de construção e de busca local, são realizadas, em um primeiro momento, apenas as ***m*** execuções associadas ao procedimento de construção, sendo essas execuções distribuídas dentre os oito processadores. Posteriormente, são realizadas ***m*** execuções do algoritmo submetendo cada solução construída à busca local e, escolhendo-se no final, a melhor dentre as ***m*** soluções.

De forma a avaliar a eficiência do algoritmo GRASP no que concerne à qualidade das soluções produzidas, foi realizado um conjunto de experimentos computacionais com 30 instâncias. Quanto à origem, estas instâncias são classificadas em três grupos, quais sejam: (1) da literatura, sendo citadas e utilizadas, por exemplo, no trabalho de Semaan *et al.* (2012) (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 (***p***) e a sua origem.

Tabela 1 – Informações sobre as Instâncias

Instância	n	p	Origem	Instância	n	p	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, as 30 instâncias foram submetidas aos quatro algoritmos e à formulação, considerando o número de grupos (***k***) variando entre 3 e 7, ou seja, um total de 150 soluções (30 x número de grupos). Tendo em vista que todas as instâncias têm apenas atributos quantitativos, utilizou-se a distância euclidiana para o cálculo das distâncias dos objetos aos seus respectivos medoids. E, no que concerne à aplicação do algoritmo GRASP, em todos os experimentos realizados, o número ***m*** de iterações foi fixado em 25 e o parâmetro α foi fixado em 0.20. Esses dois parâmetros foram ajustados experimentalmente a *priori*, mediante a execução do GRASP para um subconjunto de seis instâncias dentre as instâncias listadas na tabela 2. Mais especificamente, o algoritmo foi aplicado em cada uma dessas instâncias, considerando a seguinte combinação de parâmetros: ***m***= {25, 50, 100, 200} e α={0.10, 0.15, 0.20, 0.25}. Em seguida, as 92 soluções produzidas (6 . |***m***| . |α|) foram avaliadas no que concerne ao valor da função objetivo, sendo tomada a combinação de ***m*** e α correspondente ao maior quantitativo de soluções ótimas.

As tabelas de dois até quatro trazem os resultados produzidos pelos quatro 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 solução viável produzida por um ou mais dos algoritmos. Uma análise dessas tabelas mostra que o GRASP teve um desempenho superior aos demais algoritmos, no que concerne ao quantitativo de soluções ótimas.

Acrescenta-se, ainda, que das 150 soluções produzidas, a formulação produziu o ótimo global em **132** casos, uma solução viável em **11** casos e não produziu nenhuma solução em **7** casos. O valor de 132 foi inicialmente utilizado como base para o cálculo do percentual de ótimos globais produzidos por cada um dos algoritmos (vide gráfico 1). Neste sentido, em 94% dos casos o algoritmo GRASP produziu o ótimo global, seguido pelos algoritmos PAM, de ParkJUN e CLARA, com percentuais de, respectivamente, 74, 23 e 12.

Ao considerar a melhor solução obtida para todos os casos (150 = 30 instâncias x grupos) e por grupo (30 instâncias), observou-se, novamente, a superioridade do GRASP em relação à formulação e aos demais algoritmos. A partir de uma análise do gráfico dois, observa-se que o GRASP obteve um percentual de soluções melhores da ordem de 91%, seguido pela formulação e pelos algoritmos PAM, ParkJun e CLARA, com respectivos percentuais da ordem de 87%, 65%, 23% e 11%. Ao analisar este gráfico por grupos, novamente observa-se um desempenho superior do GRASP excetuando-se, apenas, o caso de seis grupos, onde o percentual de melhores soluções produzidas pela formulação foi da ordem de 87% e o pelo GRASP foi de 83%.

A tabela cinco traz, por estrato, a mediana e a média dos tempos de processamento dos quatro algoritmos e da formulação. Uma análise dessa tabela mostra que os algoritmos de ParkJUN, PAM e CLARA tiveram um performance superior ao GRASP e à formulação. Particularmente, em relação ao GRASP, esse fato era esperado, uma vez que estes algoritmos são basicamente heurísticas dissociadas de procedimentos de construção e de busca local mais intensivos como caso do GRASP.

Ainda que o algoritmo GRASP não seja tão eficiente quanto os demais algoritmos, no que concerne aos seus tempos de execução, o mesmo é mais eficaz no que diz respeito ao quantitativo de soluções ótimas. Neste sentido, vale destacar que a diferença entre os percentuais associados aos quantitativos de ótimos globais do GRASP e do 2º melhor algoritmo (PAM) foi de 20%, sendo este percentual bem significativo. Acrescenta-se, que ainda, os seus tempos de processamento médio e mediano ficaram abaixo dos 30 segundos.

Os bons resultados apresentados para instâncias de dimensão variada indicam que o GRASP pode ser uma alternativa para resolução do problema dos ***k***-medoids. Não obstante, em trabalhos futuros, serão implementadas novas buscas locais baseadas, por exemplo, no *Path Relinking* e no método VNS (Glover and Kochenberger, 2002). Estas novas versões do GRASP serão comparadas com os algoritmos considerados neste trabalho e com os algoritmos de Sheng e Liu (2004) e Chu *et al.* (2008). Também será efetuada uma modificação no procedimento de busca local descrito neste trabalho. Mais especificamente, de forma a inserir maior variabilidade nesse procedimento, será permitida a escolha de qualquer cluster no passo (1) da busca, ao invés de sempre escolher o pior cluster.

Tabela 2 – Resultados dos Algoritmos e da Formulação (3 e 4 grupos)

Instância	n	k	PAM	CLARA	ParkJUN	GRASP	Formulação	k	PAM	CLARA	ParkJUN	GRASP	Formulação
DS1-200DATA	200	3	0,371755	0,371804	0,371755	0,371755	0,371755	4	0,265603	0,265652	0,265603	0,265603	0,265603
DS1-broken-ring	800	3	0,806440	0,804171	0,803858	0,803858	0,803858	4	0,608921	0,608921	0,608921	0,608921	0,608921
DS1-chart	600	3	7,708550	7,708550	7,866560	7,708550	7,708550	4	7,598375	7,600272	7,905568	7,598375	7,598375
DS1-gauss9	900	3	0,815595	0,817382	0,812599	**0,812599**	1,35209*	4	**0,673955**	0,676209	0,675503	**0,673955**	1,76724*
DS1-iris	150	3	0,875705	0,874722	0,868645	0,868645	0,868645	4	0,776507	0,785464	0,797810	0,776507	0,776507
DS1-maronna	200	3	0,695663	0,703237	0,695663	0,695663	0,695663	4	0,472809	0,473640	0,472809	0,472809	0,472809
DS1-ruspini	75	3	0,598321	0,602516	0,598321	0,598321	0,598321	4	0,318706	0,323189	0,318706	0,318706	0,318706
DS1-spherical_4d3c	400	3	0,562392	0,560452	0,569530	0,560452	0,560452	4	0,333503	0,333503	0,333503	0,333503	0,333503
DS1-vowel2	528	3	0,803448	0,804264	0,803457	0,803448	0,803448	4	0,699827	0,703172	0,699559	0,696164	0,696164
DS1-wine	178	3	2,808597	2,808597	2,808597	2,808597	2,808597	4	2,688419	2,676843	2,711840	2,676843	2,676843
DS1-yeast	1484	3	**1,829516**	1,843443	1,829968	**1,829516**	-	4	1,738251	1,738876	1,748183	**1,732445**	-
DS2-400p3c	400	3	0,280307	0,280370	0,280307	0,280307	0,280307	4	0,239133	0,242666	0,239133	0,239133	0,239133
DS3-2face	200	3	0,677785	0,678331	0,677785	0,677785	0,677785	4	0,607911	0,597670	0,594638	0,592568	0,592568
DS3-face	296	3	0,667511	0,667736	0,667511	0,667511	0,667511	4	0,582965	0,586626	0,582973	0,582965	0,582965
DS3-moreshapes	489	3	0,634762	0,636355	0,634762	0,634762	0,634762	4	0,407425	0,407786	0,407425	0,407425	0,407425
DS3-numbers2	540	3	0,791152	0,782119	0,780678	0,779784	0,779784	4	0,601610	0,603293	0,601738	0,601610	0,601610
DS3-outliers	150	3	0,318933	0,318933	0,318933	0,318933	0,318933	4	0,232125	0,116662	0,242317	0,232125	0,232125
DS4-ecoli	336	3	1,561043	1,561043	1,561043	1,561043	1,561043	4	1,438577	1,454887	1,490302	1,438577	1,438577
DS4-Gamma400	500	3	1,073824	1,074031	1,072392	1,070977	1,070977	4	0,932886	0,935574	0,934164	0,932886	0,932886
DS4-haberman	306	3	1,315699	1,329278	1,316286	1,315699	1,315699	4	1,191291	1,198144	1,191291	1,191291	1,191291
DS4-idh	187	3	0,286936	0,287343	0,287061	0,286936	0,286936	4	0,218036	0,219884	0,218193	0,218036	0,218036
DS4-indian	583	3	2,005019	2,009863	2,040977	2,005019	2,005019	4	1,882190	1,882190	1,958390	1,880477	1,880477
DS4-Normal300	300	3	0,819265	0,821541	0,821177	0,819265	0,819265	4	0,716381	0,729088	0,716932	0,716381	0,716381
DS4-pib100	100	3	0,254521	0,254309	0,254249	0,254249	0,254249	4	0,177186	0,177917	0,177186	0,177186	0,177186
DS4-synthetic_control	600	3	7,105057	7,105057	7,303236	7,105057	7,105057	4	7,005453	7,005453	7,238325	7,005453	7,005453
DS4-Uniform400	400	3	0,797123	0,797963	0,796979	0,796168	0,796168	4	0,655958	0,662245	0,655958	0,655958	0,655958
DS4-Uniform700	700	3	0,782995	0,784634	0,783143	0,782995	0,782995	4	0,657854	0,660538	0,659118	0,657854	0,657854
DS4-wpbc	198	3	4,507711	4,422156	4,407410	4,407410	4,407410	4	4,186800	4,284158	4,324780	4,186800	4,186800
DS4-SP_LAVOURA	566	3	0,183472	0,183472	0,202767	0,183472	0,183472	4	0,138629	0,139351	0,175308	0,138629	0,138629
DS4-PIB_MINAS	853	3	**0,090097**	0,090104	**0,090097**	**0,090097**	0,926303*	4	**0,073264**	0,073287	0,075114	**0,073264**	0,0951102*

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

Tabela 3 – Resultados dos Algoritmos e da Formulação (5 e 6 grupos)

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

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

Tabela 4 – Resultados dos Algoritmos e da Formulação (7 grupos)

Instância	n	k	PAM	CLARA	ParkJUN	GRASP	Formulação
DS1-200DATA	200	7	0,193651	0,198902	0,198607	0,193651	0,193651
DS1-broken-ring	800	7	0,429272	0,425769	0,440152	0,422321	0,422321
DS1-chart	600	7	7,392426	7,406084	7,516040	7,392426	7,392426
DS1-gauss9	900	7	0,460879	0,462992	**0,460386**	0,460783	-
DS1-iris	150	7	0,614015	0,623579	0,622253	0,614015	0,614015
DS1-maronna	200	7	0,354853	0,368733	0,354860	0,352798	0,352798
DS1-ruspini	75	7	0,228690	0,235842	0,243987	0,228690	0,228690
DS1-spherical_4d3c	400	7	0,290165	0,288277	0,293400	0,283790	0,283790
DS1-vowel2	528	7	0,478582	0,480750	0,490702	0,478582	0,478582
DS1-wine	178	7	2,425616	2,477642	2,510904	2,425616	2,425616
DS1-yeast	1484	7	1,511777	1,499264	1,547214	**1,494475**	-
DS2-400p3c	400	7	0,179591	0,183281	0,181518	0,179591	0,179591
DS3-2face	200	7	0,399592	0,410220	0,402025	0,399592	0,399592
DS3-face	296	7	0,424080	0,431072	0,423199	0,421153	0,421153
DS3-moreshapes	489	7	0,211730	0,212762	0,224731	0,211730	0,211730
DS3-numbers2	540	7	0,403541	0,405498	0,414222	0,402857	0,402857
DS3-outliers	150	7	**0,139803**	0,139803	0,155270	**0,139083**	0,138964
DS4-ecoli	336	7	1,225282	1,223632	1,316771	1,218431	1,218431
DS4-Gamma400	500	7	0,770729	0,786522	0,773925	0,770729	0,770729
DS4-haberman	306	7	0,986895	0,995173	0,986587	0,957531	0,957531
DS4-idh	187	7	0,112244	0,115062	0,132851	0,112244	0,112244
DS4-indian	583	7	1,654579	1,683874	1,704389	1,662766	1,654579
DS4-Normal300	300	7	0,564535	0,575617	0,566987	**0,563748**	0,563331*
DS4-pib100	100	7	0,081258	0,081510	0,090590	0,081258	0,081258
DS4-synthetic_control	600	7	6,811775	6,823547	6,940608	**6,808662**	6,808662*
DS4-Uniform400	400	7	0,511534	0,507715	0,521375	**0,502537**	0,499476
DS4-Uniform700	700	7	0,506557	0,506373	0,501006	0,499956	0,499956
DS4-wpbc	198	7	3,807740	3,889597	3,850878	3,807740	3,807740
DS4-SP_LAVOURA	566	7	0,076197	0,076773	0,115351	0,076197	0,076197
DS4-PIB_MINAS	853	7	**0,037310**	0,043943	0,049848	**0,037310**	0,045120*

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

Tabela 5 – Mediana e Média (por grupo) dos Tempos de Processamento dos Algoritmos e da Formulação

	k=3		k=4		k=5		k=6		k=7	
	Mediana	Média	Mediana	Média	Mediana	Média	Mediana	Média	Mediana	Média
PAM	0,02	0,04	0,03	0,06	0,03	0,07	0,05	0,08	0,06	0,12
CLARA	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
PARKJUN	0,81	1,49	0,82	1,53	0,89	1,49	0,81	1,49	0,88	1,49
GRASP	5,69	14,53	6,13	16,53	9,94	16,70	10,70	22,00	18,53	25,76
FORMU	101,50	1270,5	102,5	1237,53	112,5	1728,67	120	1752,8	157	2003,03

Gráfico 1

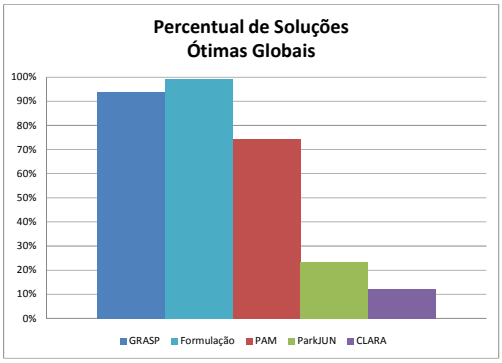
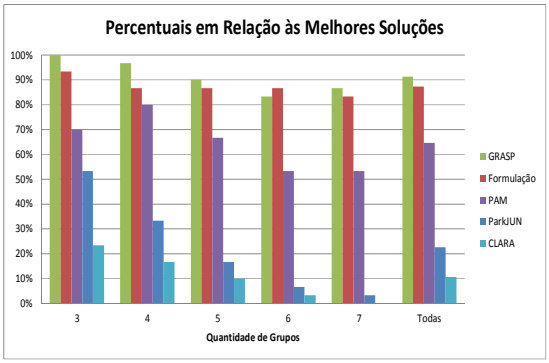


Gráfico 2



Bibliografia

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

Chu, S.C., Roddick J.F and Pan J.S.. Improved Search Strategies and Extensions to k-medoids-based Clustering Algorithms. International Journal of Business Intelligence and Data Mining, 3, 2, 212-231, 2008.

Cruz, M. D., O problema de clusterização automática, Tese de Doutorado, COPPE/UFRJ, 2010.

Feo, T.A. and Resende, M.G.C., Greedy randomized adaptive search procedures, *Journal of Global Optimization*, **6**, pp. 109-133, 1995.

Glover, F. and Kochenberger, G. A. , "*Handbook of Metaheuristic*", First Edition Norwell: K1uwer Academic Publishers, 2002.

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

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

Johnson A.R. e Wichern D.W.,. *Applied Multivariate Statistical Analysis*. Prentice Hall. Fifth Edition, 2002.

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

Linguagem R (versão 2.15.2). (www.**r**-project.org/).

Michaud, P.. Clustering techniques. *Future Generation Computer Systems*, 14, 135-147, 1997.

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

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

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

Semaan, G.S., Cruz, M.D., Brito, J.A.M. e Ochi, L.S.. Proposta de um Método de Classificação Baseado em Densidade para a Determinação do Número Ideal de Grupos em Problemas de Clusterização. Learning and NonLinear Models, 10, p. 242-262, 2012.

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

Tan, P. N., Steinbach, M. e Kumar V.. Introdução ao Data Mining – Mineração. Ciência Moderna, 2009.

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

Agradecimentos: À FAPERJ (projeto E-26/111.947/2012) e ao CNPQ (projeto 475245/2012-1) pelo financiamento parcial deste estudo.