10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

UM ALGORITMO GENÉTICO PARA O PROBLEMA DOS K-MEDOIDS

Wagner Marques de Brito
Centro Universitário Plínio Leite (UNIPLI)
wagnerbrito@gmail.com

Gustavo Silva Semaan
Instituto de Computação da Universidade Federal Fluminense (IC-UFF)
gsemaan@ic.uff.br

José André de Moura Brito
Escola Nacional de Ciências Estatísticas (ENCE-IBGE)
jose.m.brito@ibge.gov.br

Abstract – *This paper aims to present a new algorithm for a classical clustering problem, known as the k-medoids problem. In this problem, given a set of n objects with p attributes, you must allocate them to k distinct groups, so that each group is as homogeneous as possible, given an objective function that measures the distance (Euclidean) of each group object to a particular object (also in the group) called medoid. The high complexity associated with this problem led the study and implementation of an algorithm based on genetic algorithms metaheuristic. This metaheuristic is applied in many optimization problems, including various clustering problems. In order to assess the efficiency of this algorithm, several computational experiments were performed considering an artificial data repository and the data set of Brazilian HDI (Human Development Index).*

Keywords – Clustering, K-Medoids, Genetic Algorithms.

1 Introdução

De um modo geral, para organizar objetos em grupos, deve-se levar em conta as suas características (atributos) de forma a poder estabelecer algum tipo de semelhança entre os mesmos. A área de pesquisa que trata desta questão agrega um conjunto de métodos conhecidos como métodos de análise de agrupamentos (clusterização). Na análise de agrupamentos cada grupo é denominado um cluster. O número de clusters pode ser conhecido a priori ou não. Quando o número de clusters é conhecido a priori, define-se um *Problema de K-Clusterização* ou simplesmente um *Problema de Clusterização* (PC). Caso contrário, o problema é denominado *Problema de Clusterização Automática* (PCA). Ambos os problemas são difíceis, mas o fato do valor de *k* não ser previamente conhecido torna o problema ainda mais complexo, aumentando substancialmente o número de soluções possíveis.

Em função de suas variadas aplicações e do progresso da computação, diversos métodos de análise de agrupamentos têm sido desenvolvidos e estudados nas últimas décadas, tendo contribuições de diversas áreas. As limitações computacionais concernentes à automatização dos processos de agrupamento, à qualidade das soluções produzidas e ao tempo necessário para a obtenção de tais soluções são questões que têm motivado novas pesquisas nesta área.

Os métodos de análise de agrupamento são utilizados para diferentes fins, tais como: detecção de enfermidades, classificação de células sanguíneas, tratamento de imagens médicas (ultrassonografia e tomografia, por exemplo), identificação de padrões de assinatura, reconhecimento facial, classificação de minerais em rochas, classificação de componentes químicos, reconhecimento de imagens de satélites, estudo de padrões de consumo, logística, estudo de tráfego, marketing, análises comportamentais, etc. [3]. Diante desta vasta aplicabilidade e das limitações supracitadas, quando são considerados os algoritmos clássicos da literatura, foi proposto para este trabalho um algoritmo baseado na metaheurística algoritmos genéticos, amplamente utilizada em problemas de otimização combinatória [11].

O presente artigo está estruturado em quatro seções, incluindo a introdução. A segunda seção apresenta uma descrição detalhada da análise de agrupamentos, as medidas de distância utilizadas, métodos de análise de agrupamento e o problema dos k-medoids, alvo dos estudos desse trabalho. A terceira seção traz os principais conceitos de algoritmos genéticos e descreve o algoritmo proposto neste trabalho. Já a seção quatro apresenta a ferramenta proposta, que possui uma implementação do algoritmo genético para a resolução do problema dos k-medoids. Esta seção ainda contém detalhes sobre as instâncias utilizadas e os resultados obtidos nos experimentos computacionais realizados.

2 Análise de Agrupamentos

A análise de agrupamentos agrega uma série de métodos que incorporam critérios para agrupar objetos associados com uma base dados [5]. Os métodos disponíveis nessa técnica são de cunho estatístico multivariado, com conotação exploratória. Diante disso, dado um conjunto de *n* objetos, onde cada objeto é portador de *p* características, ou atributos, busca-se um padrão de formação que permita distribuir cada um destes objetos em *k* grupos similares.

É importante salientar que esse padrão é uma função de um conjunto de características previamente definidas para os objetos. O processo de classificação, por outro lado, pode se utilizar dos grupos definidos pela análise de agrupamentos, para

classificar novos objetos de acordo com suas características comuns. Ou seja, atribuir um novo objeto da base de dados ao grupo mais homogêneo [5].

Em geral, os atributos associados aos objetos são incorporados em uma função objetivo que permite avaliar o grau de similaridade (ou dissimilaridade) entre os objetos e os seus respectivos grupos, de forma a construir grupos mais homogêneos. Os tipos de atributos dos objetos são fundamentais para a determinação da medida de similaridade que será utilizada. Uma medida ou coeficiente que associe esses objetos pode ser obtida através de transformações nesses atributos, de acordo com o seu tipo (quantitativo, qualitativo, etc.). O tipo do atributo define a escala de mensuração que será utilizada para se avaliar a similaridade (homogeneidade) entre os objetos [5].

Ao invés de considerar a similaridade, pode-se utilizar uma medida de dissimilaridade para verificar e quantificar o grau de homogeneidade entre os objetos (tomados dois a dois), e consequentemente, construir grupos mais homogêneos. Essa medida avalia o quanto diferem os objetos em função dos seus atributos. Ou seja, quanto menor o valor da dissimilaridade, mais similares são os objetos em relação aos seus atributos.

Normalmente, encontrar um agrupamento ótimo (ótimo global) [10] é uma tarefa que tende a se tornar muito difícil à medida que o número de objetos aumenta, independentemente do método utilizado. O aspecto combinatório desse problema está associado com uma expressiva quantidade de configurações de agrupamentos, ou seja, das possíveis formas de distribuir (alocar) os objetos nos grupos. Em caso de aplicação de um procedimento de busca exaustiva (força bruta) para garantir a obtenção do ótimo global, deverão ser enumeradas todas as soluções, ou seja, todas as possibilidades de combinação dos n objetos em k grupos. O total de possibilidades, neste caso, está associado ao número de *Stirling* de segundo tipo [9]. Por exemplo, caso $n = 18$ objetos sejam alocados em quatro grupos, o número de soluções a serem consideradas é de $2{,}60537 \times 10^{13}$. Aumentando o número de grupos para cinco (uma unidade), este número cresce para $1{,}07886 \times 10^{16}$ soluções possíveis. Ou seja, temos um crescimento do tipo exponencial, no que concerne ao número de soluções possíveis para o problema em relação a n. Diante da dificuldade de encontrar agrupamentos ótimos (global), diversos métodos de análise de agrupamento foram desenvolvidos nas últimas décadas. Em geral, tais métodos diferenciam-se: (i) pela forma de construir os grupos, (ii) pela função objetivo adotada, (iii) tempo de processamento e (iv) qualidade das soluções produzidas, ou seja, pelos ótimos locais produzidos.

Ao iniciar o processo de agrupamento, deve-se primeiro analisar as características (atributos) que estão associadas aos objetos (indivíduos) e em seguida aplicar uma transformação nessas características, com o objetivo de trabalhar com uma única medida de dissimilaridade.

Supondo que existam n objetos que serão distribuídos em k grupos, duas estruturas matriciais podem ser consideradas de forma a possibilitar a aplicação dos algoritmos de agrupamento. A primeira representa os objetos ou indivíduos por meio de p medidas ou características, tais como altura, peso, sexo, cor, etc., através de uma matriz de ordem $n \times p$, onde as linhas correspondem aos objetos e as colunas aos valores de seus respectivos atributos (características). A segunda matriz, de ordem n^2, contém as distâncias entre todos os objetos. Essas distâncias representam o grau de dissimilaridade entre os objetos. Em particular, no presente trabalho, optou-se pela distância euclidiana.

2.1 Métodos Hierárquicos e não Hierárquicos

Existem diversos métodos de análise de agrupamentos disponíveis na literatura [7]. Esses métodos têm bases teóricas ou empíricas diferentes, e por isso, produzem estruturas de agrupamento diferentes. Além disso, cada método de agrupamento pode possuir mais de um algoritmo de agrupamento distinto ou até mesmo formas distintas de implementação. A escolha do método influenciará diretamente nos agrupamentos resultantes. Os métodos utilizados em análise de agrupamentos são classificados em: hierárquicos ou não hierárquicos.

Os métodos hierárquicos consistem em uma série de sucessivos agrupamentos ou sucessivas divisões de elementos, onde os elementos são agregados ou desagregados. Os grupos obtidos através da aplicação de um método hierárquico são geralmente representados (ou visualizados) por um diagrama bidimensional denominado dendograma ou diagrama de árvore. Nesse diagrama é possível observar os relacionamentos entre os grupos, e consequentemente a ordem na qual os grupos podem ser unidos ou divididos. Além disso, cada ramo representa um elemento, enquanto a raiz representa o agrupamento de todos os elementos. Através do dendograma e do conhecimento prévio sobre a estrutura dos dados, deve-se determinar uma distância de corte para definir quais serão os grupos formados. Essa decisão é subjetiva, e deve ser feita de acordo o objetivo da análise e do número de grupos desejados. Os métodos hierárquicos subdividem-se em dois conjuntos de métodos para os quais existem algoritmos específicos, quais sejam: os métodos Aglomerativos e os métodos Divisivos.

Os métodos não hierárquicos, ou de particionamento, foram desenvolvidos para agrupar $\boldsymbol{n}$ objetos em k de grupos, sendo $\boldsymbol{k}$ definido previamente. Uma vez que nem todos os valores de k produzem grupos satisfatórios, aplica-se o mesmo método várias vezes para diferentes valores de k, escolhendo os resultados que possibilitem uma melhor interpretação dos grupos, ou melhor representação gráfica, ou um melhor valor, considerando algum índice que mede a qualidade dos grupos formados [4].

A ideia central da maioria dos métodos não hierárquicos é a de escolher uma partição inicial dos objetos (grupos) e, em seguida, trocar os objetos entre os grupos de forma a obter uma melhor partição [1]. Ou seja, reduzir a dissimilaridade dentro de cada grupo e aumentar a dissimilaridade entre os grupos.

Quando comparado com um método hierárquico, um método por particionamento é mais rápido, porque durante o processamento não são efetivamente avaliadas todas as partições possíveis. Em geral, esses métodos diferem entre si pela maneira que constituem a melhor partição. Os métodos por particionamento mais conhecidos são o método *k-means* (k-médias) e o método *k-medoids* (k-medóides).

2.1.2 Método dos K-Medoids

Diferentemente do método k-means, o método dos k-medoids não utiliza a média como centro do grupo, e sim, considera um problema onde um objeto é o centro do próprio grupo, chamado de objeto representativo ou medoid. A estratégia básica para definir ***k*** grupos para alocar os ***n*** elementos está diretamente associada à determinação do medoid para cada grupo. Inicialmente, selecionam-se de forma aleatória ***k*** objetos, medoids iniciais em cada um dos grupos. Uma vez definidos os medoids iniciais, cada um dos *(n-k)* objetos remanescentes é alocado ao grupo cujo medoid está mais próximo. A estratégia é então fazer várias tentativas de troca de medoids por não medoids e reavaliar qualidade dos novos agrupamentos resultantes, ou seja, reduzir o valor da função objetivo (equação 1), a soma das dissimilaridades.

$$f = Minimizar \sum_{i=1}^{k} \sum_{\forall Oj \in med_i} d_{ij} \quad (1)$$

Sendo: $O_j \in$ ao conjunto X, k quantidade de medoids e $M = \{ med_1, med_2, ..., med_k \}$ objetos medoids que definem os grupos.

O problema dos *k-medoids* também pode ser interpretado em termos de análise de agrupamentos, como "localizar *n* objetos e então substituí-los por *k* pontos (ou objetos representativos) no espaço *n*-dimensional" (HANSEN e JAUMARD, 1997). De acordo com [2] "A solução exata (ótimo global) desse problema, bem como de suas variantes, pode ser obtida através de uma formulação de programação matemática [10]. Todavia, mesmo para uma quantidade de objetos apenas moderada (da ordem de centenas), a resolução dessa formulação pode levar a um consumo expressivo de tempo computacional (horas, dias, meses, anos,...) impossibilitando a obtenção de um ótimo global", ou até mesmo de um ótimo local. Ou seja, a execução da formulação pode terminar sem produzir nem ao menos uma solução viável para o problema.

Uma primeira alternativa para contornar esse consumo expressivo de tempo, e produzir um ótimo local, consiste em utilizar um dos algoritmos clássicos da literatura que foram propostos para esse problema. Um dos mais populares é o algoritmo proposto por [10], denominado PAM *(Partitioning Around Medoids*). Visando resolver o problema dos *k*-medoids para grandes bases de dados, [10] desenvolveram uma versão modificada do algoritmo PAM, denominada CLARA (*Clustering Large Applications*). Esse algoritmo pode ser aplicado em bases de dados de dimensão elevada, considerando a combinação de uma técnica de amostragem e do algoritmo PAM. Ao invés de determinar os *k*–medoids considerando toda a base de dados, o algoritmo CLARA seleciona *s* amostras compostas por *m* objetos da base de dados ($m < n$), Em seguida, após a seleção dessas amostras, aplica-se o algoritmo PAM em cada uma delas. Uma vez definidos os medoids e formados os grupos a partir dos *m* objetos, cada objeto não pertencente à amostra é alocado ao grupo cujo medoid é mais próximo. Esses dois algoritmos tendem a convergir para pontos de ótimo local que podem estar muito distantes do ótimo global.

EM [14] foi apresentado um algoritmo híbrido que combina um procedimento de busca local com a metaheurística algoritmos genéticos [11], de forma a produzir melhores soluções. A busca local, aplicada na fase de cruzamento do AG pode ser resumida da seguinte forma: depois de definidos os *k*-medoids, são selecionados os seus *p* pontos mais próximos, também definidos como vizinhos. Para cada um desses vizinhos é avaliada a sua troca com o atual medoid. Ocorrendo uma melhoria, o medoid é atualizado e novamente selecionam-se seus *p* vizinhos, aplicando-se o mesmo procedimento. Esse procedimento é repetido até que não haja mais uma mudança nos medoids, ou seja, até que não ocorram mais mudanças no valor da função objetivo. Neste algoritmo genético, a população é composta por *m* cromossomos, sendo que cada cromossomo corresponde a um conjunto de medoids. Em seguida, calcula-se o valor da função objetivo para cada solução. A partir desses valores, considerando-se o procedimento de reprodução, as melhores soluções são promovidas para a geração seguinte com a utilização do método do torneio. Definindo-se uma probabilidade de cruzamento e de mutação para as soluções, são aplicados os procedimentos de cruzamento e mutação. Nesses procedimentos, basicamente, selecionam-se dois cromossomos e trocam-se os valores associados aos seus medoids, atribuindo-se novos medoids a essas posições. Esse tipo de troca definirá duas novas soluções, isto é, um conjunto de medoids, aplicando-se o procedimento de busca local. Os procedimentos de reprodução, cruzamento e mutação são aplicados por certo número de iterações ou até que não haja melhoria da solução após *q* iterações.

3 Algoritmos Genéticos

Os Algoritmos Genéticos (AG) pertencem à classe dos algoritmos evolutivos que utilizam modelos computacionais dos processos naturais de evolução, objetivando a resolução de diversos problemas da área de pesquisa operacional e de áreas correlatas. Em geral, as diferentes variedades de algoritmos evolutivos têm em comum o conceito de evolução das espécies. Ou seja, os processos de seleção, mutação e de reprodução.

Os Algoritmos Genéticos, introduzidos por [8], baseiam-se na teoria de Darwin da evolução natural das espécies e nos princípios de herança genética de Gregor Mendel. Termos como: *cromossomos, genes, alelos, genótipo e fenótipo*, comuns nos sistemas biológicos, têm os seus termos correspondentes no modelo computacional especialmente proposto para simular os processos evolutivos. Um algoritmo genético funciona de forma similar à teoria biológica dos sistemas naturais, cujos indivíduos mais aptos sobrevivem e geram novos descendentes com suas características hereditárias. Estes descendentes, que comporão as novas gerações, tendem a ter a mesma aparência, ou *fenótipo*, que seus antecessores.

3.1 Passos Considerados para a Aplicação de um AG

- **Representação do problema:** Primeiramente, deve-se definir uma forma de representação do problema (cromossomo) no qual o AG será aplicado. É comum a utilização de vetores ou *strings* como forma de representação das soluções do problema, sendo considerada uma representação binária ou o *group number* [15].

- **Geração da população inicial:** Aleatoriamente, ***m*** soluções (*S1, S2,..., Sm*) devem ser geradas, onde "*m*" é o tamanho da população, que corresponde a um pequeno subconjunto contido no conjunto associado com todas as possíveis soluções viáveis para o problema.
- **Avaliação da função objetivo:** Para cada solução S_i (cromossomo) da população, calcular o valor função para essa solução: fi=f(Si), ∀ i=1,...,m.
- **Aplicação dos operadores genéticos na seguinte ordem: (i) Reprodução dos melhores indivíduos (seleção)-** Com base nos valores de f_i, as melhores soluções, ou seja, aquelas com menor valor de f_i **(minimização)** são selecionadas e copiadas para nova população de forma a substituir as piores soluções da população atual. Em geral, para realizar esta operação, são utilizados os métodos da roleta viciada e do torneio [11]. A roleta é um mecanismo que possibilita uma seleção aleatória que é realizada *m* vezes (uma vez para cada elemento da nova população). A cada vez que a roleta é acionada, a probabilidade de que uma dada solução seja copiada para a nova população é inversamente proporcional ao valor de sua correspondente função objetivo (problema de minimização). Desta forma, quanto menor for o valor da função objetivo relacionada a uma solução, maior deve ser o número de cópias esperadas dessa solução para a nova população. O torneio também é um mecanismo de seleção aleatório que é acionado *m* vezes (uma vez para cada elemento da nova população), sendo escolhidos ***t*** cromossomos da população atual em cada vez. Uma vez selecionados os ***t*** cromossomos, escolhe-se aquele com o menor valor de função objetivo. O número ***t*** é definido como um parâmetro do algoritmo genético. ***(ii) Cruzamento:*** É a operação que possibilita a recombinação das estruturas genéticas da população, ou seja, a troca de partes das soluções, objetivando a produção de novas soluções de melhor qualidade. Permite uma diversificação da população no espaço de soluções ao gerar soluções diferentes. Basicamente, o operador de cruzamento escolhe aleatoriamente duas soluções S_i e S_j e troca alguns de seus elementos. A frequência de execução deste procedimento também está condicionada a uma probabilidade, em que um valor real (entre 0 e 1) gerado aleatoriamente deve ser inferior à probabilidade submetida como parâmetro ao algoritmo. Os principais tipos [11] de cruzamento utilizados na maioria dos AGs são o **Cruzamento de um ponto,** o **Cruzamento de dois pontos** e o **Cruzamento Uniforme** [11]. ***(iii) Mutação:*** O procedimento de mutação corresponde a uma perturbação realizada em uma solução, de forma a evitar que a mesma fique estagnada em ótimos locais de baixa qualidade e regenerando possíveis soluções que tenham sido eliminadas pelos outros procedimentos. A execução desse procedimento, assim como para o procedimento de cruzamento, está condicionada a uma probabilidade. Nesse caso, o procedimento realiza trocas de genes entre soluções de forma aleatória. Para que uma determinada população não sofra muitas mutações, e consequentemente não perca suas características, esta operação é processada para um pequeno percentual ***PM*** de seus elementos (em torno de 1% de todos os genes). Após a aplicação de todos estes procedimentos, repete-se iterativamente os passos *3* (avaliação) e *4* (reprodução, cruzamento e mutação), que correspondem a uma nova geração ou iteração do algoritmo. Um possível critério de parada para o algoritmo é fixar a repetição desses passos por um determinado número de vezes (*MAXGEN*), definindo o que chamamos de número de gerações. A saída do algoritmo pode ser representada pela última população gerada.

Como o objetivo é normalmente obter a solução (cromossomo) de um problema de otimização, pode-se armazenar em uma estrutura de dados extra a "melhor" solução S* que foi obtida considerando todas as gerações do algoritmo. Entenda-se por melhor, aquela solução S* que tem o menor (maior) valor da função objetivo no problema de minimização (maximização). Além do número máximo de gerações (MAXGEN), alguns autores consideram também o tempo máximo de execução do algoritmo em substituição ou em conjunto a este parâmetro.

3.2 Algoritmo Genético para o Problema dos k-medoids

Assim como em outros problemas de agrupamento, o problema dos ***k***-medoids tem uma elevada complexidade computacional de resolução. Tal característica é uma natural motivação para a utilização de uma abordagem baseada em uma metaheurística. Inicialmente, para a aplicação do AG neste problema, deve-se escolher a forma de representação dos cromossomos (soluções). No caso desse problema, os cromossomos foram representados considerando o *group number*, conforme descrito a seguir: dado um conjunto de *n* objetos e definido um número *k* de grupos, cada solução para o problema de agrupamento dos ***k***-medoids corresponderá a um vetor com *n* posições e, para cada posição, será sorteado um valor entre 1 e *k*. Por exemplo, se $k = 4$ e $n = 10$, pode-se ter a seguinte solução viável:

1	2	4	1	4	3	3	1	2	2

Em seguida, são calculadas as distâncias euclidianas entre todos os *n* objetos (tomados dois a dois) que serão agrupados pelo AG. Essas distâncias ficam armazenadas em uma matriz $D_{n \times n}$. Considera-se que a partir desta representação, são gerados *m* vetores (soluções) de acordo com o tamanho da população inicial para sorteio (*POP_INI_SORTEIO*). Em seguida, são sorteadas *x* soluções para compor a população inicial (P_0), onde *x* é definido pelo tamanho da população do AG (*TAM_POP_AG*). Em outras palavras, dentre as *m* soluções, sorteiam-se as *x* melhores soluções (*S1 , S2 ,..., Sx*) que correspondem aos agrupamentos dos objetos em relação aos seus respectivos medoids. Uma vez gerada a população, calcula-se para cada solução *Si* da população o valor da função objetivo *f i* considerando a equação 2:

$$f = \sum_{l=1}^{k} \sum_{\forall Oj \in med_l} d_{lj} \qquad (2)$$

Ou seja, o valor da função objetivo *fi*, foi calculado como sendo a soma das funções objetivo de cada grupo que compõe a solução *Si* (cromossomo) ($f_i = \sum_{l=1}^{k} fg_l$). Sendo: k = número de grupos e fg_l = função objetivo do grupo g_l.

Para se calcular o valor da função objetivo por grupo (fg_l; $l=\{1,...,k\}$), deve-se definir o objeto do grupo que corresponde ao medoid, ou seja, aquele cuja soma das distâncias aos demais objetos do grupo é a menor possível. Esse procedimento é repetido em cada um dos grupos. Após o cálculo da função objetivo, são aplicados operadores genéticos na seguinte ordem: **(i) Reprodução das melhores soluções (seleção)** – Para realizar esta operação, foram utilizados os métodos da "roleta viciada" e do "torneio" [11]. O método é escolhido de acordo com o valor do parâmetro tipo de seleção (*TIPO_SELECAO*) informado pelo usuário antes de iniciar a execução do AG. Além destes procedimentos de seleção, foi implementado um procedimento de Elitismo, com o objetivo de garantir que a melhor solução obtida até o momento fosse armazenada e inserida na nova população (P'). O Elitismo é um método que seleciona o cromossomo com a menor f_i (melhor cromossomo) da população atual (P) e substitui um cromossomo sorteado aleatoriamente em P'. Com isso, garante-se que a melhor solução encontrada até o momento não seja perdida. **(ii) Cruzamento** – Neste algoritmo, foi considerada uma probabilidade de cruzamento variável. Mais especificamente, foram definidos quatro parâmetros para a aplicação desse operador, quais sejam: uma probabilidade inicial de cruzamento (PROB_INI_CRUZ), uma probabilidade final de cruzamento (PROB_FIM_CRUZ), o tipo de cruzamento (TIPO_CRUZ) e a probabilidade de cruzamento variável (PROB_CRUZ_VAR). Abaixo temos a definição de cada parâmetro:

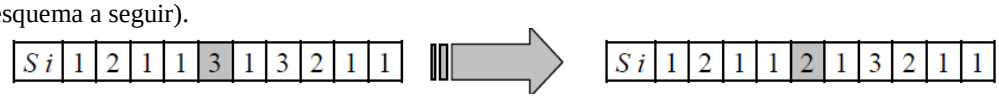
TIPO_CRUZ: Variável inteira que define se o cruzamento será de um ou dois pontos;
PROB_INI_CRUZ: Variável real que define qual será a probabilidade inicial de cruzamento utilizada pelo AG.
PROB_FIM_CRUZ: Variável real que define qual será a probabilidade final de cruzamento utilizada pelo AG.
PROB_CRUZ_VAR: Variável booleana que define se será utilizada a probabilidade variável de cruzamento. Caso o valor desta variável seja VERDADEIRO, então a probabilidade de cruzamento será dada pela equação (4) (descrita em [11]), senão a PROB_INI_CRUZ é utilizada.

$$PROB\_CRUZ = GERACAO_{ATUAL} * \left(\frac{PROB\_FIM\_CRUZ - PROB\_INI\_CRUZ}{MAXGEN}\right) + PROB\_INI\_CRUZ \quad (4)$$

Sendo: *GERACAO_ATUAL* corresponde à iteração atual do AG; *MAXGEN* o número máximo de gerações (iterações) que serão executadas pelo AG e *PROB_CRUZ* a probabilidade de cruzamento calculada pela equação (1);
Abaixo é apresentado um exemplo dos dois tipos de cruzamentos (um ponto e dois pontos, respectivamente) usados no AG:

Com aplicação do cruzamento, alguns dos objetos (dentre os n) são realocados novos grupos. Isso pode implicar na definição de novos medoids, e consequentemente, na redução do valor da função objetivo.

(iii) Mutação – Correspondeu a uma pequena perturbação realizada em algumas soluções *Si*. Neste problema, a mutação consistiu na alteração dos valores de algumas posições de um cromossomo por um valor entre 1 e k (excluindo o valor atual daquela posição). Ou seja, algumas possíveis alocações de objetos (para outros grupos) que porventura tivessem sido perdidas na reprodução ou no cruzamento. Tal operação foi efetuada em um pequeno percentual dos elementos que compõem uma solução *Si* (ver esquema a seguir).



Assim como no cruzamento (vide esquema acima), cada vez que se aplica o operador de mutação um objeto é realocado a um novo grupo.

Dois tipos de mutação foram definidos para este operador: Por cromossomo, onde um cromossomo é sorteado aleatoriamente da população P, e em seguida um gen desse cromossomo é sorteado e alterado pelo operador; "Por população" .

Assim como para o operador de cruzamento, o operador de mutação foi definido de forma a trabalhar com uma probabilidade variável. Desta forma, foram definidos quatro parâmetros para este operador, quais sejam: uma probabilidade inicial de mutação (PROB_INI_MUT), uma probabilidade final de mutação (PROB_FIM_MUT), o tipo de mutação (TIPO_MUT) e a probabilidade de mutação variável (PROB_MUT_VAR). Abaixo temos a definição de cada parâmetro:

TIPO_MUT: Variável inteira que define se o tipo de mutação é por cromossomo ou por população;
PROB_INI_MUT: Variável real que define qual será a probabilidade inicial de mutação utilizada pelo AG;
PROB_FIM_MUT: Variável real que define qual será a probabilidade final de mutação utilizada pelo AG;

PROB_MUT_VAR: Variável booleana que define se será utilizada a probabilidade variável de mutação. Caso o valor desta variável seja VERDADEIRO, então a probabilidade de mutação será dada pela equação (5) [11], senão a PROB_INI_MUT é utilizada.

$$PROB\_MUT = GERACAO_{ATUAL} * (\frac{PROB\_FIM\_MUT - PROB\_INI\_MUT}{MAXGEN}) + PROB\_INI\_MUT \quad (5)$$

Sendo: *GERACAO_ATUAL* corresponde à iteração atual do AG; *MAXGEN* é o número máximo de gerações (iterações) que serão executadas pelo AG e *PROB_MUT* é a probabilidade de mutação calculada pela equação (2);

Após todos estes procedimentos terem sidos executados, foram repetidos, iterativamente, os passos **(i)**, **(ii)** e **(iii)**, que correspondem a uma geração ou iteração. Os critérios de parada considerados neste algoritmo foram o número máximo de gerações (MAXGEN), o número de iterações sem encontrar uma solução melhor do que a atual (NMAX) e o tempo máximo de processamento (TEMMAX).

Considerando todas as possíveis combinações dos métodos de seleção, cruzamento e mutação, e as probabilidades variáveis ou não, foram desenvolvidas vinte e quatro versões deste algoritmo. Em seguida, de forma a ajustar os parâmetros do AG e escolher as melhores versões para a realização dos experimentos computacionais, as 24 versões foram aplicadas em um subconjunto de quatro instâncias de tamanho variado que foram previamente selecionadas dentre todas que foram consideradas nos experimentos. Posteriormente, foram escolhidas oito versões deste algoritmo para este trabalho. A **Tabela 1** apresenta as oito versões (selecionadas) e os operadores genéticos envolvidos em cada uma delas.

Tabela 1 - Oito versões selecionadas do algoritmo proposto

	Algoritmo							
Configuração	1	2	3	4	5	6	7	8
Seleção por Torneio	x	x	x	x				
Seleção por Roleta					x	x	x	x
Mutação por Cromossomo	x	x			x	x		
Mutação por População			x	x			x	x
Cruzamento 1 ponto	x	x			x	x		
Cruzamento 2 ponto			x	x			x	x
Probabilidade Variável		x		x		x		x

4 Resultados Computacionais

O presente capítulo traz uma descrição da ferramenta desenvolvida em JAVA e que incorpora o novo algoritmo genético que foi aplicado ao problema dos *k*-medoids. Além disso, descrevem-se os dados utilizados para este trabalho e os resultados computacionais obtidos a partir da aplicação das oito versões do algoritmo genético neste conjunto de dados. A ferramenta que incorpora o algoritmo genético foi implementada em linguagem Java, mais especificamente, na versão Java SE 1.6. Essa ferramenta tem por finalidade auxiliar na construção e na análise de agrupamentos de dados multidimensionais mediante a execução de um algoritmo genético que resolve o problema dos k-medoids. O algoritmo implementado nesta ferramenta trabalha apenas com dados quantitativos. Para facilitar a utilização da ferramenta foram desenvolvidos quatro módulos básicos, quais sejam :**i) O módulo de arquivo de objetos -** Neste módulo é informado o arquivo de dados que será utilizado pelo algoritmo. Ou seja, é possível utilizar arquivos já disponíveis (retirados de alguma base dados) ou gerar arquivos de dados artificiais (valores dos atributos no intervalo [0, 1]) para serem agrupados; **ii) O módulo de dados do algoritmo genético** - O usuário deve informar todos os parâmetros que serão utilizados pelo AG para realizar o agrupamento dos dados associados com uma particular instância (arquivo). **ii) O módulo de resultados** – Neste módulo são apresentadas algumas informações relativas à solução formada (objetos escolhidos para medoids, função objetivo, etc) e sobre a execução do AG (tempo de processamento, total de gerações, etc); **iv) O módulo gráfico** – Permite que os grupos formados e os seus respectivos medoids (bolas com contorno em vermelho) sejam visualizados graficamente. Esta visualização só é possível para instâncias cujos objetos tenham apenas dois atributos.

4.1 Dados utilizados

Neste trabalho foi utilizado um conjunto de quinze arquivos de dados, sendo que quatorze deles têm dois atributos e foram gerados artificialmente pela ferramenta. O outro arquivo (com um atributo) é arquivo do Índice de Desenvolvimento Humano (IDH) [13] dos municípios região norte do Brasil. No que concerne às instâncias artificias, todas com dois atributos, cada um destes atributos teve o seu valor gerado aleatoriamente, sorteando-se um número real no intervalo [0, 1]. Nestas instâncias o número de objetos variou de 50 até 1000. Mais especificamente, 2 instâncias para os seguintes números de objetos: 50, 75,100,150 e uma instância para o seguinte número de objetos: 200, 250, 300, 400, 500,1000. No caso do IDH o arquivo da região norte tem 449 objetos.

4.2 Resultados Computacionais e Análises

Todos os experimentos computacionais foram realizados em um computador com processador Intel Core 2 Duo P8600 2.40GHz com 2 Gb de memória RAM e Sistema Operacional Windows XP 32 bits.

Cada uma das trinta e cinco instâncias foram submetidas às oito versões (algoritmos). O tamanho da população foi de 100, o tempos máximo de processamento foi de duas horas, o número máximo de gerações foi de 5000, as probabilidades inicial e final de cruzamento e mutação foram de respectivamente 0.8, 0.2, 0.005, 0.2. Outros detalhes das configurações podem ser observados na Tabela 1.

Para cada instância foram geradas soluções com 4 e 5 grupos em cada uma das oito versões do AG. A escolha em trabalhar com o número de grupos nesta faixa foi decorrente da observação de que os maiores ganhos (redução) na função objetivo foram obtidos com estes valores. Ou seja, com um número maior de grupos não produziu ganhos substanciais.

Além do AG proposto, cada instância também foi processada (considerando os mesmos números de grupos) pelo Solver LINGO [12], utilizando a formulação exata de programação inteira descrita em [2]. Os resultados provenientes do algoritmo e da formulação são sumarizados e comparados na Tabela 3, que contém as seguintes informações: **Instância** – Arquivo de dados; **Grupos** – Número de grupos; **Modelo FOBJ** – Valor da função objetivo (k-medoids) obtido a partir da formulação; **GAP modelo** – que representa o percentual de diferença entre o valor da solução produzida pela formulação (Solver) e a melhor solução produzida, considerando as oito versões do AG; **Tempo modelo** (Solver), que representa o tempo de execução em segundos; **Melhor Fobj** – que corresponde ao menor valor da função objetivo (k-medoids) dentre as oito versões do algoritmo executados para cada instância; **GAP(i)**(para i de 1 até 8)representa o percentual de diferença entre a melhor solução obtida e solução de cada versão (i) do algoritmo (i); **T(i)** (para i de 1 até 8) – representa o tempo de execução de cada versão (i) do AG em segundos;

Tabela 2 – GAPs obtidos a partir dos resultados da Formulação x AG (versão 2)

Médio	Mediano	Mínimo	1º Quartil	3º Quartil	Máximo
0,43	0,00	0,00	0,00	0,32	3,28

Tabela 3 – Resultados da execução do AG proposto

Instância	Grupos	Modelo FOBJ	Modelo-T	GAP Modelo	Melhor Fobj	GAP(1)	GAP(2)	GAP(3)	GAP(4)	GAP(5)	GAP(6)	GAP(7)	GAP(8)	T(1)	T(2)	T(3)	T(4)	T(5)	T(6)	T(7)	T(8)
f1_50	4	9,04	1	0,00	9,04	2,91	18,37	0,00	0,00	7,36	1,23	1,34	0,00	5	3	3	3	13	8	11	8
	5	7,74	1	0,00	7,74	3,13	0,00	3,13	3,13	11,55	4,77	4,47	3,23	5	3	4	3	19	8	14	12
f2_50	4	7,82	1	0,00	7,82	1,35	0,00	0,85	0,00	3,29	0,00	6,38	0,00	5	3	3	3	11	11	12	10
	5	6,79	1	0,00	6,79	5,54	5,26	5,54	0,00	10,00	4,49	5,54	0,00	8	3	5	3	13	11	8	11
f3_75	4	13,59	3	0,00	13,59	1,35	0,00	1,85	0,94	11,45	3,02	3,64	0,46	11	7	8	8	41	33	28	29
	5	11,80	2	0,04	11,81	10,50	2,32	0,05	0,05	12,29	0,05	9,13	0,00	17	6	10	8	36	33	38	29
f4_75	4	13,70	3	0,00	13,70	10,88	0,00	11,06	0,00	5,81	12,22	19,34	8,22	10	7	9	9	39	31	34	26
	5	11,84	2	0,05	11,85	0,15	1,10	0,59	3,67	7,99	0,00	6,03	14,38	23	8	8	6	35	25	37	28
f5_100	4	17,64	3	0,00	17,64	8,54	0,00	8,62	0,55	26,88	2,05	8,58	1,87	22	12	14	13	65	45	63	52
	5	15,45	3	0,00	15,45	14,02	5,29	1,42	0,00	21,53	3,89	20,45	3,03	21	17	27	15	59	50	61	53
f6_100	4	18,57	3	0,00	18,57	0,23	0,00	0,05	8,06	19,68	1,73	13,61	0,55	22	11	33	12	64	61	67	46
	5	16,14	3	0,00	16,14	7,32	0,00	5,82	0,00	31,43	4,43	15,33	5,00	32	13	16	16	53	63	59	62
f7_150	4	28,09	9	0,00	28,09	0,98	0,00	0,00	0,12	39,59	11,85	25,11	1,26	91	33	61	32	132	136	133	134
	5	24,83	9	2,61	25,48	2,85	1,18	3,81	0,00	47,61	11,99	32,72	10,87	82	34	46	31	119	122	122	125
f8_150	4	27,83	8	0,00	27,83	2,92	0,00	0,00	0,00	34,10	6,93	19,19	15,13	83	34	78	30	124	135	131	133
	5	25,23	9	2,71	25,91	0,92	1,20	0,23	0,00	53,78	4,45	32,27	12,70	81	32	60	37	117	121	120	122
f9_200	4	38,78	35	0,08	38,81	7,33	4,51	7,62	0,00	44,98	8,67	29,62	14,13	121	62	100	59	211	217	222	216
	5	34,83	107	1,06	35,20	4,41	6,47	0,00	1,17	61,02	21,32	53,48	25,02	130	47	135	54	194	198	203	196
f10_250	4	47,37	31	0,00	47,37	6,89	0,00	2,27	0,03	60,76	35,21	52,03	40,13	228	104	204	99	330	322	334	330
	5	42,01	35	0,00	42,01	24,41	0,00	6,13	0,21	84,14	33,86	65,60	37,36	200	91	196	110	283	293	298	292
f11_300	4	56,23	62	0,30	56,40	4,25	12,99	1,57	0,00	59,05	34,41	50,44	33,20	321	141	302	169	452	460	465	454
	5	49,91	62	3,28	51,55	14,71	0,21	4,33	0,00	85,90	41,40	70,72	50,46	275	168	273	161	401	408	417	400
f12_400	4	73,91	156	0,00	73,91	12,62	0,00	5,23	0,00	78,26	42,53	70,26	53,70	566	386	575	395	763	789	794	785
	5	67,48	7200*	0,14	67,58	22,45	0,00	7,39	1,20	87,82	51,23	85,59	55,42	475	395	474	300	690	694	718	694
f13_500	4	94,24	265	0,27	94,50	28,71	0,00	3,65	9,57	77,17	60,74	78,72	63,45	867	735	825	524	1155	1189	1220	1170
	5	84,14	258	0,38	84,46	39,27	0,00	11,13	4,66	102,49	76,71	93,30	80,28	729	638	736	640	1047	1051	1083	1047
f14_1000	4	246,55	7200*	-20,46	196,11	45,89	0,45	33,32	0,00	87,76	73,25	84,50	77,98	5827	5760	6128	6466	7200*	7200*	7200*	7200*
	5	196,21	7200*	-8,24	180,04	61,68	0,00	33,98	0,97	104,44	89,25	102,06	91,36	5801	5124	5890	5624	7200*	6954	7200*	7200*
f15_centro	4	5,16	224	0,74	5,20	27,11	0,00	15,70	0,00	124,36	69,14	100,85	75,11	697	515	696	425	1144	1195	1206	1172
	5	4,26	217	0,42	4,28	52,57	1,01	16,74	0,00	172,24	113,75	146,40	120,00	587	334	592	383	1041	1078	1087	1074

Analisando os resultados da Tabela 3 (a seguir) e considerando apenas os 30 casos (instância x número de grupos), verifica-se que em 50% destes casos a melhor solução produzida pelo AG (Melhor Fobj) corresponde ao ótimo global da formulação exata. Além disso, para os casos em que a formulação produziu soluções melhores do que o AG (43%), a pior solução do AG ficou somente a 3,3% da solução produzida pela formulação (GAP) (instância f11_300) e destes 43% em 69% dos casos, o gap foi inferior a 1%. Ainda com base na Tabela 3, verifica-se através do GAPs (versões de um até oito do AG) que a versão dois do AG foi a que produziu o maior número de melhores soluções (GAP2 = 0,00%). E finalmente, em 7% dos casos o AG produziu soluções melhores do que as da formulação.

Com base nas informações dos GAPs entre a melhor versão do AG e da formulação (casos em que o AG produziu soluções iguais ou piores do que as da formulação), pode-se chegar à Tabela 2, onde se verifica a eficiência do AG. Cinquenta por cento das soluções (1º quartil e mediana) produzidas pelo AG correspondem ao ótimo global produzido pela formulação. A análise do 3º quartil indica que 75% das soluções produzidas pelo AG têm um GAP inferior a 0,5% em relação à formulação. Além disso, a pior solução do AG está no máximo a 3,28% da solução da formulação.

Após a análise e a comparação apresentada no capítulo 4, pode-se observar que o algoritmo genético constitui-se como uma boa alternativa para se trabalhar em problema de agrupamento. Definindo-se uma representação adequada para o problema, e com os ajustes corretos dos parâmetros do AG, bons resultados podem ser obtidos para diversos problemas de agrupamento. Como futuros desdobramentos deste trabalho se pode: Ajustar o algoritmo proposto para trabalhar com outros tipos de atributos que não sejam somente quantitativos reais; Devido às características de paralelismo inerentes aos algoritmos genéticos e aos problemas de agrupamento, a utilização de uma versão para processamento paralelo de populações de indivíduos pode trazer benefícios principalmente no tempo total de execução e na qualidade das soluções obtidas**;** Desenvolvimento de algoritmos baseados em outras metaheurísticas, tais como: ILS (*Iterated Local Search*), VNS (*Variable Neighborhood Search*) ou uma combinação destas metaheurísticas [6]; É possível desenvolver uma abordagem para o problema de clusterização automática, onde o número de grupos não é fixado previamente.

5 Referências

[1] Anderberg, M. R. *Cluster Analysis for Aplications,in: Probability and Mathematical Statistics*, vol. 19, Editora Academic Press, 1973.

[2] Brito, J. A. M.; Ochi, L. S.; Brito, R. L.; Montenegro, F. M. T. *Um Algoritmo para o Agrupamento Baseado em K-Medoids*. Artigo a ser publicado na Revista Brasileira de Estatística (volume de 2010).

[3] Coelho, G. A. A. *Cl@ssAuto – Uma ferramenta WEB de ensino em classificação automática de dados*. Dissertação de Mestrado. Universidade Estadual do Ceará/Centro Federal de Educação Tecnológica do Ceará – CE, 2003.

[4] Faceli , K.; DE Carvalho, A. C. P. L. F.; Souto, M. C. P. *Validação de Algoritmos de Agrupamento*. Relatórios Técnicos do ICMC, São Carlos, 2005.

[5] Frei, F. *Introdução à análise de agrupamentos: teoria e prática*. São Paulo: Editora UNESP, 2006.

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

[7] Hair, J. F. Jr.; Anderson, R. E.; Tatham, R. L.; BLACK, W. C.; tradução. Adonai Schlup Sant'Anna e Anselmo Chaves Neto. *Análise Multivariada de Dados* – 5. Ed. – Porto Alegre: Bookman, 2005.

[8] Holland, J. H., Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[9] Johnson, A. R.; Wichern D. W. *Applied Multivariate Statistical Analysis*. Prentice Hall. Fifth Edition, 2002.

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

[11] Linden, R. *Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional*, Brasport, 2ª Edição, 2008.

[12] Lingo. The Modeling Language and Optimize. LINDO Systems (www.lindo.com), 2010.

[13] Relatório de Desenvolvimento Humano - Síntese. *A Verdadeira Riqueza das Nações: Vias para o Desenvolvimento Humano*. Publicado para o Programa das Nações Unidas para o Desenvolvimento (PNUD), 2010. disponível em: <http://hdr.undp.org/en/reports/global/hdr2010/chapters/pt/>.

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

[15] Trindade, A. R.; Ochi, L. S. *Um algoritmo evolutivo híbrido para a formação de células de manufatura em sistemas de produção*. Abstracts in Operational Research / Statistical Theory and Methods Abstracts) vol. 26(2), 2006.