Artigo aceito para o VII CBRN 2005 – Natal/RN

Proposta e Análise de um Algoritmo Evolutivo Construtivo para o Problema de Clusterização Automática

Stênio Sã Furtado Soares
Luiz Satoru Ochi

Instituto de Computação – Universidade Federal Fluminense (IC/UFF)
Niterói – Rio de Janeiro – Brasil
{ssoares, satoru}@ic.uff.br

Abstract

Clustering, also known as unsupervised classification, is a process by which a data set is partitioned into different clusters such that objects of the same cluster are as similar as possible and objects of different clusters are as dissimilar as possible. In clustering algorithms, it is usually assumed that the number of clusters is known or given. Unfortunately, the optimal number of clusters is unknown for many application of this problem. These problems are known as Automatic Clustering Problems (ACP). This paper presents an Efficient Constructive Evolutionary Algorithm to solve the ACP. Extensive computational experiments show that the proposed algorithm outperforms a genetic algorithm from the literature, improving both, the solution quality and the computational time required.

Palavras-chave: Algoritmos Evolutivos, Problemas de Clusterização Automática.

1. Introdução

O *Problema de Clusterização* (PC) consiste em agrupar os objetos de uma base de dados de modo que objetos mais similares estejam em um mesmo cluster e objetos menos similares sejam alocados para clusters distintos. Existem basicamente duas classes de Problemas de Clusterização. O caso mais estudado é quando o número de clusters já é previamente definido (também conhecido como o *Problema de K - Clusterização* ou simplesmente *Problema de Clusterização (PC)*). O outro caso é quando este número *K* não é conhecido previamente. Neste caso, o problema é denotado por *Problema de Clusterização Automática (PCA)* [4]. Este trabalho apresenta uma nova heurística evolutiva híbrida que utiliza conceitos de Algoritmos Genéticos (AG) com modificações em alguns passos de forma a torná-la mais eficiente na solução do PCA para bases de dados genéricos; considerando, por exemplo, as seguintes características: instâncias de grande porte (tanto no número de objetos como no número de atributos de cada objeto; presença de ruídos (*outliers)*; escalabilidade; procedimentos que detectem regiões densas; eficácia na determinação de clusters com formas não esféricas e, finalmente, detectar o número ideal de clusters a cada instância. Para tanto, são propostos: (i) uma fase de pré-processamento nos dados de entrada do PCA de forma a reduzir o trabalho de busca de soluções de boa qualidade bem como possibilitar a calibração do intervalo para a busca do número ideal de clusters numa solução; e (ii) uma heurística construtiva randomizada para a geração de uma população inicial semi-otimizada para o AG.

2. O Problema de Clusterização Automática (PCA)

De uma maneira mais formal, pode-se definir o Problema de Clusterização como: Dado um conjunto com *n* elementos $X = \{X_1, X_2, ..., X_n\}$, obtenha os *k clusters* $C = \{C_1, C_2, ..., C_k\}$, tal que os elementos contidos em um *cluster* C_i possuam uma maior similaridade entre si do que com os elementos de qualquer um dos demais *clusters* do conjunto *C*. O conjunto *C* é considerado uma *clusterização* com *k clusters* caso as seguintes condições sejam satisfeitas:

$$\bigcup_{i=1}^{k} C_i = X \quad (1)$$

$$C_i \neq \emptyset, \text{ para } 1 \leq i \leq k \quad (2)$$

$$C_i \cap C_j = \emptyset, \text{ para } 1 \leq i,j \leq k \text{ e } i \neq j \quad (3)$$

O PC e o PCA são classificados como NP-Difícil [13]. Dessa forma, métodos heurísticos ou aproximados têm sido propostos com freqüência para obter soluções sub-ótimas em tempos computacionais viáveis. Entretanto, devido à grande heterogeneidade das aplicações do problema, as heurísticas são normalmente desenvolvidas para atender apenas determinadas classes de problemas, ou seja, não existe uma heurística que seja genérica a tal ponto que possa obter bons resultados em todas as aplicações de clusterização. As heurísticas existentes para a solução de problemas de clusterização podem ser

classificadas, de forma geral, em *métodos hierárquicos* e *métodos de particionamento* [4].

Dentre os vários algoritmos hierárquicos existentes na literatura, destacam-se aqueles propostos para trabalhar com grandes bases de dados, como é o caso do algoritmo BIRCH [17]. Além deste, um outro algoritmo aglomerativo é o ROCK [10] que, além de trabalhar bem com grandes bases de dados, é um algoritmo eficiente para dados categóricos. Os mesmos autores de ROCK apresentaram o CURE [11], que representa um algoritmo aglomerativo que, além de manter o bom desempenho quando submetido a instâncias de grande porte, encontra clusters de diferentes formas e se mostra insensível a *outliers* (ruídos). Deve-se ressaltar, entretanto, que tanto o BIRCH como o ROCK e até mesmo o CURE são algoritmos desenvolvidos para o PC e não para o PCA, ou seja, não são capazes de determinar o melhor número de clusters. Berkhin [4] aponta como vantagens dos algoritmos de clusterização hierárquica a facilidade em lidar com qualquer medida de similaridade utilizada e a sua conseqüente aplicabilidade a qualquer tipo de atributo.

Os algoritmos de clusterização que utilizam métodos de particionamento são conhecidos como *k-clustering methods*. Isso se justifica pelo fato da quase totalidade dos métodos desta classe serem originalmente desenvolvidos para problemas de clusterização (PC) onde o número de clusters a serem formados (*k*) é previamente conhecido. Além dos métodos hierárquicos e de particionamento, é possível observar um crescimento significativo de propostas utilizando metaheurísticas aplicadas a problemas de clusterização [2], [3], [5], [6], [7], [8], [16].

3. Um Algoritmo Evolutivo Construtivo para o PCA

Os algoritmos evolutivos (AEs) e, em particular os algoritmos genéticos (AGs), têm sido usados com sucesso na solução de diferentes problemas de elevada complexidade computacional. Contudo, os AGs na sua forma original (população inicial gerada aleatoriamente, reprodução unicamente através dos operadores clássicos de mutação e *crossover*, substituição automática dos indivíduos pais pelos filhos) [12] não têm se mostrado competitivos em diversas aplicações da área de otimização combinatória onde já existem outras heurísticas ou metaheuristicas eficientes.

As principais limitações dos AGs dizem respeito à baixa qualidade da população inicial, à dificuldade em obter ótimos locais de boa qualidade e ao esforço computacional exigido, que normalmente é maior que o de outras heurísticas. A questão da qualidade da população inicial em muitos casos tem se resolvido através do uso de heurísticas de construção. Ainda assim, poucas contribuições têm sido propostas para simultaneamente melhorar a qualidade dos AGs e, ao mesmo tempo, reduzir ou não onerar ainda mais os tempos computacionais exigidos.

Este é o desafio colocado como meta neste trabalho, ou seja, obter um algoritmo evolutivo baseado em conceitos de AGs que produza soluções competitivas para o Problema de Clusterização Automática (PCA) em tempos computacionais também competitivos.

O algoritmo evolutivo híbrido aqui proposto, denominado *Algoritmo Evolutivo Construtivo* (AEC), é composto de dois módulos, a saber: o módulo de pré-processamento e de construção de uma população inicial; e o módulo evolutivo propriamente dito. Em termos de algoritmos de clusterização, o AEC difere da maioria dos AGs, por ser capaz de resolver o problema de agrupamento e do número ideal de clusters, sendo, portanto, aplicável ao PCA. Além disso, o algoritmo foi desenvolvido para lidar de forma eficaz com características de dados que surgem comumente em aplicações reais, como dimensões elevadas, ruídos, regiões de diferentes densidades e clusters de formatos diversos.

3.1 Pré-processamento e Construção de uma Solução Inicial do AEC

Este módulo inicial procura reduzir o espaço de busca das soluções viáveis do PCA e, com isso, melhorar a qualidade das soluções geradas pelo algoritmo e também reduzir os tempos computacionais exigidos.

De fato, sabe-se que o tamanho da *string* (vetor) que codifica um indivíduo em um AG pode influir de forma significativa no tempo de processamento final do mesmo. Assim, dependendo da codificação utilizada pelo AG para o Problema de Clusterização, pode ser interessante substituir grupos de elementos cuja similaridade é considerada alta, por um único elemento representativo para cada grupo. Desta forma, a substituição de um grupo de pontos similares por um único ponto que os represente pode ser feita tomando-se, por exemplo, o ponto médio calculado de acordo com as coordenadas de cada ponto deste grupo. Este trabalho propõe uma abordagem hierárquica baseada em grafos para reduzir o tamanho do vetor que codifica o indivíduo no AE e ao mesmo tempo tirar proveito desta representação na ocasião da geração da população inicial do AEC. Considere, portanto, os elementos de uma base de dados X como vértices de um grafo construído conforme o conceito de *grafo de vizinhança relativa* (GVR) [15]. O GVR do conjunto *X*, denotado *GVR*(*X*), é o grafo associado à matriz de adjacência M dada por:

$$M[i,j] = \begin{cases} 1, se\ d(i,j) \leq \min\ [d(i,k), d(j,k)], \\ \forall\ k \in X,\ k \neq i, j\quad e \\ 0, caso\ contrário\ . \end{cases} \quad (4)$$

onde $d(i,j)$ denota a distância (aqui usada a euclidiana) entre os pontos i e j. A equação (4) diz que dois pontos i e j são considerados *vizinhos relativos* se não existir nenhum outro ponto k cuja distância em relação a i e j seja inferior a $d(i,j)$, podendo até ser igual. O $GVR(X)$ é o grafo $G(V, E)$ onde o conjunto de vértices V é formado por todos os pontos do conjunto X e o conjunto de arestas E é formado por todas as conexões entre dois vértices (i, j) cujo valor de $M[i,j]$ é igual a 1.

Como pode ser visto, as componentes conexas de GVR(X) são agrupamentos que refletem uma visão de *vizinhança mínima*, ou seja, considerando m como sendo o número de componentes conexas obtidas de $GVR(X)$, pode-se esperar, como número máximo de clusters, o valor m $(onde\ m < n)$ e, como número mínimo de clusters, o valor 2. Este resultado é certamente uma contribuição importante no processo de solução do PCA. De fato, o valor m será usado como o novo limite superior para o número possível de clusters para uma solução, ao invés do limite original n. Para concluir a fase de construção do AEC, as componentes conexas G_i de $GVR(X)$ são vistas como elementos de um conjunto $G = \{G_1, G_2, \ldots, G_m\}$, onde cada G_i representa uma componente conexa candidata a se tornar um *cluster raiz*, e V_i, para $i = 1, \ldots, m$ representa o centróide da componente conexa G_i. Deve-se então, efetuar agrupamentos entre G_i's de forma a obter a clusterização final. Isto é feito no sentido de otimizar uma função objetivo, tarefa realizada pelo módulo evolutivo do AEC. A codificação utilizada para representação de um indivíduo no AE proposto considera uma *string* binária $r = (r_1, r_2, \ldots, r_m)$, representando uma *pré-solução (semente*). Se $r_i = 1$, o cluster associado $G_i \subset G$ fará parte da solução definitiva como *cluster pai* (cluster raiz). Caso contrário, se $r_i = 0$, G_i é considerado um *cluster filho* e será posteriormente associado a um dos clusters pais indicados pelos r_i's que constam na *string* como sendo iguais a 1. Como exemplo, considere um string $r = [0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ ]$ de uma entrada que gerou um conjunto $G = \{G_1, G_2, G_3, G_4, G_5, G_6, G_7, G_8\}$, onde cada G_i é uma componente conexa de $GVR(X)$. Pode-se dizer que a solução apresentará três clusters, compostos inicialmente pelas componentes conexas G_2, G_4 e G_8 (clusters pais). As demais componentes conexas (G_1, G_3, G_5, G_6 e G_7) são os clusters filhos, cujos pontos serão anexados a um dos clusters pais para formar com este um novo cluster, conforme o critério de similaridade adotado. Com isto, busca-se detectar os elementos mais similares que têm grande chance de, numa clusterização ótima, estarem localizados num mesmo cluster, o que atribui ao algoritmo a capacidade de identificar regiões densas presentes na base de dados da entrada. Esta fase de pré-processamento é, portanto, fundamental para que o algoritmo possa lidar com bases de dados de elevadas dimensões e detectar regiões densas através da construção de componentes conexas.

3.2 O Módulo Evolutivo do AEC

O módulo evolutivo do AEC compõe-se basicamente de um Algoritmo Genético a menos do uso da fase de pré-processamento envolvendo conceitos de componentes conexas descritos anteriormente. Esta etapa preliminar determina o formato de dois passos da fase evolutiva. O primeiro passo se refere à forma como a população inicial é obtida e o segundo passo se refere ao problema de como estimar o número ideal de clusters para cada indivíduo construído. Inicialmente, num PCA, é necessário determinar o número ideal de clusters numa solução ótima. Esta tarefa não é nada simples pelo fato de que a princípio este valor poder ser qualquer número entre 2 e $n-1$ (cardinalidade da base de dados menos 1). Ou seja, quanto maior for a base de dados (valor de n), maior será o intervalo de busca deste valor ideal. A fase de pré-processamento e construção de uma clusterização, descrita no item anterior, também contribui de maneira significativa ao efetuar uma calibração neste intervalo de busca, pois reduz o número máximo de clusters numa solução de n para m, onde m representa o número de componentes conexas obtidas. Dependendo da configuração dos dados de entrada, pode-se obter reduções drásticas no número máximo de clusters. No AEC aqui proposto, um indivíduo é representado como uma *string* de tamanho m. Na geração da população inicial, *95%* dos indivíduos são gerados de tal forma que o i-ésimo elemento da *sting* tem probabilidade de ser 1 proporcional a cardinalidade da componente conexa associada G_i. Assim, quanto maior for a cardinalidade da componente, maior será a chance do elemento da *string* associado à mesma receber valor 1. Para avaliar um indivíduo x, é utilizado o critério "*Average Silhouette Width*" (média da silhueta de cada ponto da base de dados) apresentado em Kaufman et al. [13]. Definida a função de aptidão, faz-se necessário definir a fase de reprodução do AEC, que utiliza operadores tradicionais de *mutação* e *crossover* de um ponto [8], [12].

4. Resultados Computacionais

Para avaliar o AEC proposto, foi implementado também o algoritmo genético da literatura conhecido como CLUSTERING, proposto em 2001 [16]. A escolha deste algoritmo para fazer parte dos testes comparativos com o AEC levou em conta o fato do algoritmo CLUSTERING apresentar algumas características interessantes, como por exemplo: resolver problemas de

clusterização automática (PCA), propriedade que a maioria das heurísticas da literatura não possui; apresentar também uma forma de reduzir as dimensões dos dados de entrada através de uma etapa de pré-processamento ou, em outras palavras, é direcionado para a solução de instâncias do PCA de elevada cardinalidade; apresentar, segundo seus autores, um bom desempenho para instâncias de PCA; utilizar uma função de avaliação comum a maioria dos algoritmos de PCA. Em relação aos parâmetros utilizados pelos algoritmos AEC e CLUSTERING, o critério de parada foi o número de gerações, fixado em *100* para os dois algoritmos. Entretanto, nos testes realizados observou-se que o AEC não necessitaria de um número tão elevado de gerações, pois em média a melhor solução encontrada por este algoritmo foi obtida até a geração *50*. Este fato não se deve necessariamente a limitação do AEC, mas à qualidade elevada dos indivíduos gerados inicialmente, fato que pode ser observado pelas soluções geradas, que na maioria dos casos já representa a melhor solução da literatura ou até mesmo uma solução ótima (quando esta pode ser visualizada graficamente ou é conhecida).

O tamanho da população foi fixado em *10* indivíduos para o AEC, enquanto a calibragem deste parâmetro no CLUSTERING seguiu a sugestão dada pelos autores, que é usar uma população de *50* indivíduos [16]. Os parâmetros referentes à taxa de cruzamento e mutação foram idênticos para os dois algoritmos; *80% crossover* e *5%* para a mutação. Os demais parâmetros do CLUSTERING foram fixados nos valores sugeridos pelos autores. A literatura apresenta diferentes conjuntos de instâncias para o problema de clusterização com número fixo de clusters (PC), mas de nosso conhecimento isso infelizmente não ocorre para problemas de clusterização automática. A literatura apresenta para o PCA um conjunto de 36 instâncias, mas aplicado ao problema de formação de células de manufatura (PFCM), cujo objetivo é diferente do PCA aqui tratado. Para o modelo aqui analisado, só foram detectadas quatro instâncias, disponibilizadas em bibliotecas públicas. Desta forma, outras instâncias artificiais foram geradas de forma a permitir uma análise visual no espaço R^2 que possibilite avaliar a qualidade da clusterização obtida. Neste trabalho, os testes computacionais são divididos em duas etapas. Na primeira, um conjunto composto de instâncias disponíveis na literatura, acrescido de outras de maior porte geradas artificialmente, são analisados. Na segunda bateria, parte das instâncias analisadas anteriormente são usadas para efetuar um estudo sobre a sua convergência empírica para soluções sub-ótimas utilizando os dois algoritmos.

4.1 Avaliação com Instâncias

Numa primeira bateria de testes, foi utilizado um conjunto de 17 instâncias descrito na tabela 1. Destas, quatro são instâncias disponíveis na literatura (*em negrito*). Destas quatro, a *RuspiniDataSet*, na tabela denotada por *RuspiniData*, e a instância *200Data* são artificiais e definidas no espaço R^2, estando disponíveis em [13].

Instância	Dimensão	Tamanho	# Grupos
200Data	R^2	200	3
CancerData	R^9	699	2
IrisData	R^4	150	3
RuspiniData	R^2	75	4
1000p6c	R^2	1.000	6
157p	R^2	157	?
2000p11c	R^2	2.000	11
2face	R^2	200	?
350p5c	R^2	350	5
3dens	R^2	128	3
97p	R^2	97	?
Convdensity	R^2	175	3
Convexo	R^2	199	3
Face	R^2	296	?
Moreshapes	R^2	489	?
Numbers	R^2	437	10
Numbers2	R^2	540	10

Tabela 1. Características das instâncias usadas

As outras duas instâncias da literatura representam bases reais. A primeira delas é a *Wisconsin Breast Câncer Database* [14] e é denotada na tabela 1 como *CancerData*. A outra base de dados real da literatura é a *Iris Plants Database* [9], denotada na tabela 1 como *IrisData*, A primeira coluna da tabela 1 identifica a instância; a dimensão de cada objeto é dada na coluna *2*, o número de objetos da base de dados é descrita na coluna *3* e a coluna *4* (# Grupos) indica o número ideal de clusters nos casos onde esta informação é conhecida ou quando é possível determinar através da imagem dos agrupamentos para instâncias no R^2 (o sinal de interrogação significa que esta informação não está disponível ou não é óbvia ao analisar a imagem). Ressalte-se, entretanto, que a informação do número de grupos não é repassada aos algoritmos. O objetivo é que cada algoritmo descubra, durante a execução, além da clusterização, o número de clusters ideal. Na tabela 2, cada uma das instâncias, com exceção do *2000p11c* e da *CancerData* foram executadas por cada algoritmo 100 vezes e a média destas execuções para cada algoritmo é mostrada nas colunas *2*, *3*, respectivamente. As duas exceções foram executadas *30* vezes e este procedimento é justificado pelo elevado tempo computacional exigido pelo CLUSTERING nestas duas instâncias. Note-se que na tabela, *quanto maior for o valor, melhor será a qualidade da solução*. Nas colunas *2, 3 e 4* aparecem respectivamente: os valores da melhor solução conhecida da literatura (*best*), onde o sinal de interrogação significa a inexistência ou o desconhecimento do *best*; valor da solução média do

AEC, e do algoritmo CLUSTERING [16]. Nas colunas *3* e *4* os valores em negrito indicam o valor da melhor média comparado com o melhor valor da literatura (quando for o caso) ou a melhor média entre os dois algoritmos. Os valores das soluções em todos os algoritmos, são determinados usando a função de aptidão proposta em [13].

Pelos resultados da tabela 2, observa-se que nas quatro instâncias da literatura, em três o algoritmo AEC alcançou sempre a melhor solução da literatura, o que dá ao algoritmo um ***desempenho médio*** muito alto, enquanto o desempenho apresentado pelo algoritmo CLUSTERING foi muito baixo, já que a média nunca foi igual ao melhor resultado da literatura.

Instância	Best	AEC	CLUSTERING
200Data	**0.823**	**0.823**	0.541
CancerData	**0.596**	0.592	0.374
IrisData	**0.686**	**0.686**	0.601
RuspiniData	**0.737**	**0.737**	0.550
1000p6c	**?**	**0.708**	0.367
157p	**?**	**0.666**	0.657
2000p11c	**?**	**0.602**	0.287
2face	**?**	**0.666**	0.513
350p5c	**?**	**0.758**	0.568
3dens	**?**	**0.762**	0.742
97p	**?**	**0.710**	0.706
Convdensity	**?**	**0.854**	0.818
Convexo	**?**	**0.667**	0.618
Face	**?**	**0.511**	0.402
Moreshapes	**?**	**0.720**	0.436
Numbers	**?**	**0.540**	0.417
Numbers2	**?**	**0.562**	0.513

Tabela 2: Desempenho médio dos algoritmos avaliados.

Isso mostra a confiabilidade do método e a regularidade nestes testes. Além disso, na instância *CancerData* em muitas das *30* execuções, o AEC obteve a melhor solução (*best*) da literatura, como pode ser visto na diferença muito pequena entre o seu rendimento médio e o *best*. No restante dos testes da tabela 2, nas *13* instâncias verifica-se a supremacia total do AEC, que sempre alcançou as melhores médias em relação ao CLUSTERING. A tabela 3 mostra os tempos médios em segundos dos algoritmos AEC e CLUSTERING. Os resultados mostram que os tempos médios exigidos pelo AEC são muito menores que os exigidos pelo CLUSTERING. Essa diferença se acentua ainda mais à medida que crescem as dimensões das instâncias. Isso mostra a viabilidade do algoritmo aqui proposto para tratar com base de dados de grande porte.

O diferencial nos resultados apresentados se deve possivelmente a diferença nos critérios usados para a obtenção do número ideal de clusters. No algoritmo CLUSTERING, isso é feito através de uma heurística que faz várias chamadas ao módulo evolutivo alterando a cada vez os parâmetros associados, numa tentativa de encontrar a melhor calibragem para obtenção do número de clusters adequado. Contudo, parte do mérito dos tempos reduzidos do AEC se deve a etapa de pré-processamento, que, como ilustrado anteriormente, é capaz de em muitos casos reduzir drasticamente a dimensão da base de dados original e, ao mesmo tempo, reduzir o intervalo de busca do número ideal de clusters.

Instância	AEC	CLUSTERING
200Data	**89,58**	508,73
CancerData	**3.648,02**	20.523,29
IrisData	**91,17**	583,04
RuspiniData	**16,26**	507,52
1000p6c	**1.309,89**	16.370,43
157p	**35,37**	739,01
2000p11c	**5.166,50**	28.294,57
2face	**79,27**	3.024,95
350p5c	**247,85**	5.319,80
3dens	**34,18**	2.069,69
97p	**20,17**	299,12
Convdensity	**61,96**	1.394,18
Convexo	**79,36**	1.043,82
Face	**172,09**	6.524,39
Moreshapes	**457,84**	11.530,23
Numbers	**416,47**	12.166,79
Numbers2	**603,89**	26.944,17

Tabela 3: Resultado quanto ao tempo médio em segundos: AEC, CLUSTERING.

4.2 Análise Probabilística Empírica

Um dos gargalos de alguns algoritmos heurísticos e metaheurísticas da literatura é quanto a sua instabilidade ou a falta de regularidade no desempenho. Em outras palavras, é comum concluir-se que uma dada heurística apresenta um bom desempenho para uma classe de problemas ou, mais especificamente, para algumas instâncias, mas para outras classes não obtém desempenho satisfatório. Desta forma, quando se propõe uma heurística evolutiva, é importante mostrar que a mesma possui uma boa regularidade, mesmo que empiricamente. Ou seja, deve-se mostrar que num conjunto de execuções de uma dada instância, sempre se obtenha soluções sub-ótimas de boa qualidade.

Para efeito de verificação da robustez dos algoritmos AEC e CLUSTERING, uma análise probabilística, empírica sugerida por Aiex et al. [1], é aqui apresentada. Nesta nova bateria de testes, o critério de parada para os dois algoritmos foi a obtenção de um valor alvo, valor este obtido de simulações preliminares de cada método para as instâncias testadas. O alvo pode ser uma média dos valores obtidos, ou um dos melhores valores, caso deseje-se alvos mais difíceis. Definido o alvo, cada algoritmo é executado *m* vezes para cada instância selecionada (neste trabalho foi usado *m = 100*). Cada vez que o algoritmo encontra uma solução maior ou igual ao alvo (neste caso, o problema é de maximização), o tempo de CPU *ti* que o mesmo levou para obter tal solução é armazenado. Após as *m*

execuções, os *m* tempos *ti* são dispostos em ordem crescente. Para cada execução *i*, é calculada uma probabilidade empírica $pi = (i - ½)/m$. Feito isso, os tempos ordenados e as probabilidades são tomados como pontos no R^2 da forma $zi = (ti,pi)$ para em seguida gerar um gráfico que reflete a probabilidade de cada algoritmo de atingir o alvo até um determinado tempo. A análise probabilística apresentada neste trabalho foi feita para todas as instâncias da tabela 1, usando para cada uma, três tipos de alvos: fácil, médio e difícil. A figura 1 ilustra o resultado para a instância *RuspiniData* com o alvo *0.737* considerado um alvo difícil. Nesta análise foram considerados os algoritmos CLUSTERING, AEC e o AEC sem o módulo construtivo da população inicial (AEC-SC), que difere do AEC pela população inicial ser gerada aleatoriamente. Percebe-se pela figura 1 que o AEC possui a convergência mais rápida para o alvo (*curva mais a esquerda*), seguido pelo AEC-SC e o pior desempenho foi do CLUSTERING (*curva mais a direita*). Este panorama foi similar nas outras instâncias e com outros valores alvos. Ou seja, sempre o melhor desempenho foi do AEC, seguido pelos AEC-SC e CLUSTERING, estes últimos com desempenho similar. Ainda na figura 1, a diferença significativa entre as versões AEC e AEC-SC, nos mostra a importância do módulo construtivo, que faz com que o AEC convirja muito mais rapidamente a valores sub-ótimos que os demais algoritmos.

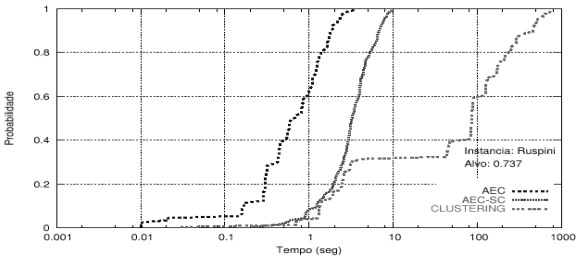

Figura 1: Análise empírica do AEC, AEC-SC, e CLUSTERING.

5. Conclusões

Este trabalho apresentou um novo algoritmo evolutivo construtivo para a solução aproximada do Problema de Clusterização Automática (PCA). As contribuições deste trabalho incluem: *Um módulo de pré-processamento com as seguintes funções:* reduzir o tamanho dos dados de entrada; fornecer uma população inicial semi-otimizada para o algoritmo evolutivo e calibração no intervalo de busca do número ideal de clusters numa dada solução. Nas poucas instâncias encontradas na literatura para o modelo de PCA aqui tratadas, o AEC sempre alcançou a melhor solução existente para três das quatro instâncias existentes e uma aproximação muito boa para a quarta instância. Para um conjunto de instâncias de R^2 geradas artificialmente, o AEC sempre obteve soluções de melhor qualidade que os obtidos por um AG da literatura. Estes resultados indicam um caminho promissor do método proposto na solução do PCA e incentiva sua adaptação na solução de outros problemas de elevada complexidade computacional.

6. Bibliografia

[1] Aiex, R., Resende, M. G. C. and Ribeiro, C. C. Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8, 343-373, 2003.

[2] Bandyopadhyan, S., and Maulik, U. An evolutionary technique based on k means algorithm for optimal clustering in R^n. *Inf. Sciences*, 146**,** 221-237, 2002.

[3] Battiti, R., Bertossi, A. & Cappelletti, A. Multilevel Reactive Tabu Search for Graph Partitioning. *Preprint UTM 554, Dip. Mat*., Univ. Trento, Itália, 1999.

[4] Berkhin, P. *Survey of Clustering Data Mining Techniques*. *Accrue Software Technical Report, 2002*.

[5] Chen, M., and Cao, D. Coordinating production planning in cellular manufacturing environment using Tabu Search. *Computers & Industrial Engineering*, 46, 571-588, 2004.

[6] Cheng, C. H., Lee, W. K., and Wong, K. F. A genetic algorithm based clustering approach for database partitioning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 32(3**)**, 215-230, 2002.

[7] Chiun, Y. and Lan, L. W. Genetic Clustering Algorithms. *EJOR*, 135(2**),** 413-427, 2001.

[8] Dias, C. R. and Ochi, L. S. Efficient Evolutionary Algorithms for the Clustering Problem in Directed Graph. *In Proc. of the 2003 IEEE Congress on Evolutionary Computation*), 983-990, 2003.

[9] Fisher, R. The use of multiple measurements in taxonomic problems. *Annual Eugenics* , 7, 179-188, 1936.

[10] Guha, S., Rastrogi, R., and Shim, K. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems* , 25, 345-366, 2000.

[11] Guha, S., Rastrogi, R., and Shim, K. CURE: A clustering alg. for large databases. *Inf. Systems*, 26, 35-58, 2001.

[12] Holland, J. H. *Adaptation in Natural and Artificial Systems*. *University of Michigan Press, Ann Arbor, 1975*.

[13] Kaufman, L. and Rousseuw , P. J. Finding Groups in Data - An Introduction to Cluster Analyses, *Wiley Series in Probability and Mathematical Statistics* , John Wiley, 1990.

[14] Mangasarian, O. L., and Wolberg, W. H. Cancer diagnosis via linear prog.. *SIAM News*, 23, 1-18, 1990.

[15] Toussaint, G. The relative neighborhood graph of a finite planar set, *Pattern Recog.*, 12, 261 – 268, 1980.

[16] Tseng, L. Y., and Yang, S. B. A genetic approach to the automatic clustering problem. *Pattern Recognition,* 34, 415-424, 2001.

[17] Zang, T., Ramakrishnan, R. E., Livny, M. BIRCH: an efficient data clustering method for very large databases. *Proc. of ACM SIGMOD Conf.*, Montreal, 103-114, 1996.