Artigo apresentado e publicado nos Proc. of the XII CLAIO 2004 – XII Latin-Ibero-American Congress on Operations Research

Um Algoritmo Evolutivo com Reconexão de Caminhos para o Problema de Clusterização Automática

Stênio Sã Rosário Furtado Soares, Luiz Satoru Ochi*
Instituto de Computação – Universidade Federal Fluminense, Niterói, Rio de Janeiro, Brasil.
e-mail: {(*) satoru,ssoares}@ic.uff.br

Abstract

Clustering is the process by which discrete objects can be assigned to groups or clusters which have similar characteristics. In clustering algorithms, it is usually assumed that the number of clusters is known or given. In the absence of such an information, a procedure is needed to find the optimal number of clusters. This problem is known as Automatic Clustering Problem (ACP). In this work we present an Evolutionary Algorithm with Path-Relinking for this class of the problem. The high potencial of our algorithm is shown through the comparison with a Genetic Algorithm that has presented the best results for this problem so far.

keywords: metaheuristics, automatic clustering methods, evolutionary algorithms

1. Introdução

O Problema de Clusterização (PC) consite em agrupar os elementos de uma base de dados em subconjuntos disjuntos denominados clusters, de forma que os pontos pertencentes a um mesmo cluster sejam similares entre si e, ao mesmo tempo, os pontos pertencentes a clusters diferentes apresentem alta dissimilaridade. Existe uma gama de aplicações do PC, como em Reconhecimento de Padrões, Formação de Células de Manufatura, Roteamento de Tráfego em Redes etc. Os métodos de clusterização podem ser classificados como hierárquicos ou de particionamento. O PC pode ser descrito da seguinte forma: dado uma base de dados $X = \{x_1,x_2,...,x_N\}$, encontrar a partição de X em k clusters disjuntos cuja similaridade entre os pontos de um mesmo cluster seja maximizada e a similaridade entre pontos de clusters diferentes seja minimizada, sujeito às seguintes condições:

$$C_i \neq \phi \qquad i = 1 \ldots k$$
$$C_i \cap C_j = \phi \qquad i, j = 1, \ldots k$$
$$C_1 \cup \ldots \cup C_k = X$$

A maioria das abordagens tratam o PC considerando o número de clusters *k* pré fixado. No entanto, na maioria das aplicações reais não se tem nenhum conhecimento prévio quanto ao número de clusters ideal; devendo o algoritmo resolver tanto o problema de agrupar os pontos, como também encontrar o número ótimo de clusters para um dado conjunto de entrada. Neste caso, o problema é conhecido como *Problema de Clusterização Automática* (PCA). Para um conjunto de entrada *X* contendo *n* pontos, o número *N(x)* de soluções viáveis, quando *k* é conhecido, é dado por:

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

No caso do valor de *k* não ser conhecido, o espaço de busca formado pelo número de soluções possíveis aumentaria drasticamente, o que nos é dado pela equação 2, definida por:

$$N(x) = \sum_{k=1}^{n} \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n \qquad (2)$$

O PCA é classificado como NP-Hard e o PC com k fixado é NP-completo para $k > 3$. Neste trabalho apresentamos um algoritmo evolutivo construtivo utilizando o módulo *Path Relinking* (reconexão por caminhos) denominado AEC-RC, que obtém o número de clusters ideal através do uso de um critério semelhante ao "*Average Silhouette Width*" apresentado em Kaufman (1989), além de uma solução aproximada de boa qualidade. O AEC-RC é dividido em três fases: uma fase de construção, através da qual se faz uma avaliação dos dados de entrada de uma instância buscando reduzir o tamanho do grafo associado

1 - Parcialmente financiado pelo CNPq.

e com isso, reduzir parâmetros do AE como o tamanho do *string* para representar um indivíduo, reduzir o espaço de busca e finalmente obter soluções iniciais semi otimizados para as próximas etapas do AE. A segunda etapa, é a fase evolutiva, que executa os operadores genéticos tradicionais e um operador de mutação proposto baseado na sobreposição de janelas no *string* solução; e, por último, uma fase de busca local através da técnica de reconexão por caminhos (*path-relinking*). Os resultados obtidos são comparados com um algoritmo genético da literatura que até então apresentava as melhores soluções aproximadas para um conjunto de instâncias da literatura.

2. Algoritmo Evolutivo Construtivo com Reconexão por Caminhos (AEC-RC)

O AEC-RC, como já mencionamos, é composto de três módulos: o módulo de construção, o módulo evolutivo e o módulo de busca local através de reconexão de caminhos. Em termos de aplicação de algoritmos de clusterização, o AEC-RC difere da maioria dos AG's, uma vez que o mesmo é capaz de resolver o problema de agrupamento e também resolver o problema de encontrar o número ideal de clusters. Uma vez que, em geral, as bases de dados das aplicações de PCA apresentam um número consideravelmente elevado de elementos, buscamos desenvolver uma heurística de construção eficaz para tratar de bases de dados de grande porte e com características tais como: regiões densas e esparsas, ruídos, características estas, usualmente presentes na maioria das aplicações reais.

2.1 A fase de construção do AEC-RC

A fase de construção do AEC-RC aqui proposta tem como meta, tentar reduzir o espaço de busca das soluções viáveis do PCA buscando com isso, tanto melhorar a qualidade das soluções geradas pelo algoritmo como também reduzir consideravelmente os tempos computacionais exigidos. De fato, sabe-se que o tamanho da *string* que codifica um indivíduo em um AG pode influir no tempo de processamento do mesmo. Assim, dependendo da codificação utilizada pelo AG para o PC, pode ser interessante substituir grupos de pontos cuja similaridade é considerada alta, por um único ponto. Podemos tomar cada objeto x_i de um conjunto de entrada como uma tupla (x_{i1}, x_{i2}, . . , x_{ip}), onde cada coordenada x_{ij} está relacionada com um atributo do objeto. Podemos, então, considerar um objeto como um ponto no espaço R^p. Desta forma, a substituição de um grupo de pontos similares por um único ponto que os represente pode ser feita tomando-se o ponto médio calculado segundo as coordenadas de todos os pontos deste grupo. Nossa proposta é apresentarmos uma abordagem hierárquica baseada em grafos para reduzir o tamanho da *string* que codifica o indivíduo no AE e ao mesmo tempo tirar proveito desta representação na ocasião da geração da população do AEC-RC. Considere os pontos de uma base de dados X como vértices de um grafo construído conforme o conceito de grafo de vizinhança relativa (GVR) (Toussaint (1980)). O GVR do conjunto *X*, denotado *GVR*(*X*), é o grafo associado à matriz de adjacência M dada por:

$$M[i,j] = \begin{cases} 1 \quad se \ d(i,j) \leq \max[d(x_i,x_k), d(x_j,x_k)], \ \forall \ x_k \in X, \ k \neq i, j \\ 0, \quad caso \quad contrário \end{cases} \qquad (3)$$

onde $d(i,j)$ denota a distância euclidiana entre os pontos x_i e x_j. A equação (3) diz que dois pontos x_i e x_j são ditos vizinhos relativos se não existir nenhum outro ponto x_k cuja distância em relação x_i e x_j seja inferior a $d(i,j)$. 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 todos os arcos e_{ij} cujo valor de M[i,j] é igual a 1. Como podemos observar, os componentes conexos de GV R(X) são agrupamentos que refletem uma visão de vizinhança mínima, ou seja, considerando *m* como sendo o número de componentes conexos obtidos de *GVR*(*X*), podemos esperar, como número máximo de clusters o valor *m* ($m < n$) e como número mínimo 2.

Para concluir a fase de construção do AEC-RC, tomamos os componentes conexos de *GVR*(*X*) e obtemos o conjunto $G = \{G_1, G_2, \ldots, G_m\}$, onde cada G_i é um componente conexo candidato a se tornar um cluster raiz, e V_i, para $i = 1, \ldots, m$ é o centróide do componente conexo G_i. Devemos, então, efetuar agrupamentos entre G_i's de forma a termos a clusterização final. Isto é feito no sentido de otimizar uma função objetivo, tarefa realizada pelo módulo evolutivo do AGS. Antes de explicarmos como se constrói a população inicial do AGS, é importante entendermos a modelagem utilizada para a codificação do indivíduo. Para tanto, considere uma *string* binária $r = (r_1, r_2, \ldots, r_m)$, representando uma solução (semente). Se $r_i = 1$, o cluster $G_i \subset G$ fará parte da solução como cluster pai (cluster raiz). Caso contrário ($r_i = 0$), 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, da 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 um componente conexo de *GVR*(X), podemos dizer que a solução apresentará três clusters, inicialmente G_2, G_4 e G_8 (clusters pais). Os demais clusters (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. Na geração do *string* que codifica uma solução, procuramos priorizar os componentes conexos com maior cardinalidade. Assim, a geração da população inicial do AEC-RC se dá de tal forma que a probabilidade de ocorrer o valor 1 em cada elemento r_i da *string* r é proporcional à cardinalidade do componente conexo $G_i \subset$ G. Apresentamos agora o processo de formação de um cluster, ou seja, a transformação de uma semente em solução propriamente dita. Após a geração da semente, considere $Y = \{G_1,G_2, \ldots, G_t\}$ o conjunto formado pelos componentes conexos G_i's associados aos elementos cujo valor é 1 no *string* indivíduo. O conjunto solução $C = \{C_1, \ldots, C_t\}$ é inicialmente formado pelos t clusters iniciais G_i's $\subset$ Y e os centróides iniciais H_i's dos clusters da solução são, inicialmente, os centróides V_i's destes clusters iniciais, onde $|C_i| = |G_i|$ para $i = 1, 2, \ldots, t$ e $|G_i|$ denota o número de pontos pertencentes a G_i. No processo de clusterização, os componentes G_i's em C\Y = $\{G_1,G_2, \ldots, G_m\}$ - Y são analisados um de cada vez de tal forma que o cluster $C_j \subset C$ irá anexar os pontos do componente conexo G_i se $\|V_i - H_j\| \leq \|V_i - H_q\|$ para $1 \leq q \leq t$ e $q \neq j$. Quando algum componente conexo G_i é associado a um cluster C_j, a nova cardinalidade de C_j , denotado por $|C_j|$, e o novo centróide de C_j , denotado por H'_j são atualizados pelas equações (4) e (5).

$$H'_j = \frac{H_j * |C_j| + V_i * |G_i|}{|C_j| + |G_i|} \qquad (4)$$

$$|C'_j| = |C_j| + |G_i| \qquad (5)$$

Após todos os componentes conexos cujos índices na string apresentam valor zero tiverem sido associados aos clusters pais indicados pelos índices cujo valor é igual a 1, a clusterização está concluída e podemos então avaliar a qualidade da mesma.

2.2 O Módulo Evolutivo do AEC-RC

O módulo evolutivo do AEC-RC compõe-se de um AG cuja população inicial é gerada segundo as densidades dos componentes conexos obtidos na fase de construção. Como podemos ver, o número de uns (1s) no indivíduo fornecerá o número de clusters do mesmo. Assim, podemos ver facilmente que dentro do intervalo [1,m], com m dado pelo número de componentes conexos, o número ideal de clusters a ser encontrado pelo algoritmo é bem menor quando comparado ao número de combinações possíveis dentro do intervalo original [1, n] dado pela cardinalidade da base de dados X. Para avaliar um indivíduo, utilizamos o critério "*Average Silhouette Width*" apresentado em Kaufman (1989). Antes de explicarmos os mecanismos dos operadores do AEC-RC, definimos a função de avaliação que resulta no *fitness* de cada indivíduo. Para tanto, seja i um ponto pertencente ao cluster C_w construído na clusterização, onde $|C_w| = M > 1$. Considere a dissimilaridade média de i em relação a todos os pontos $j \in C_w$ dada pela equação (6), onde d_{ij} é a distância euclidiana descrita pela equação entre os pontos i e j. Nos casos em que Cw possuir um único elemento, definimos a(i) = 0.

$$a(i) = \frac{1}{M - 1} \sum_{j=1}^{M} d_{ij} \quad \forall \quad j \neq i \qquad (6)$$

Considere, ainda, cada um dos clusters $C_k \subset$ C com $k \neq w$. A dissimilaridade média do ponto i em relação a todos os pontos de C_k é mostrada na equação (7), onde M é o número de pontos do cluster C_k.

$$d(i, C_k) = \frac{1}{|M|} \sum_{i=1}^{M} d_{ij} \quad \forall \quad j \in C_k \qquad (7)$$

Denota-se $b(i)$ o menor valor dentre todos os $d(i,C_k)$, o que é obtido pela equação (8). Note que $b(i)$ é obtido pelo cluster que seja o vizinho mais próximo do ponto i

$$b(i) = \min_{C_k \neq c_w} d(i, C) \qquad (8)$$

Definimos o valor da silhueta do ponto i pela equação (12), dada por:

$$s(i) = \frac{b(i) - a(i)}{\max\ (a(i), b(i))} \qquad (9)$$

Para avaliarmos a qualidade da solução, calculamos a média aritmética de todos os $s(i)$'s obtidos. Desta forma, a função de aptidão do indivíduo é dada por:

$$f = \frac{1}{N} \sum_{i=1}^{N} s(i) \qquad (10)$$
$$sujeito \quad a \quad i \in S$$

Para efetuar a seleção dos indivíduos reprodutores, o operador utilizado é o *Método da Roleta*. Entretanto, como a equação que define a aptidão do indivíduo admite valores dentro do intervalo [-1, 1], portanto podendo assumir valores negativos, aumentamos uma unidade no valor de *f* para cada indivíduo. Ao final do processamento, subtraímos uma unidade do valor de *f* do indivíduo campeão. O operador de *crossover* utilizado no AEC é o *crossover* de dois pontos. Foram desenvolvidos dois operadores de mutação para o AEC cujo emprego é alternado a cada cruzamento. O primeiro é o operador tradicional, no qual cada elemento da string pode sofrer mutação com probabilidade igual a $p(m)$, passando do valor zero para um, ou vice-versa. O segundo operador de mutação funciona através de sobreposição de janelas. Para exemplificar, considere o indivíduo F_1 gerado do *crossover*. Dentro do intervalo $[1,|F_1|]$, onde $|F_1|$ é o tamanho da *string* do indivíduo, são escolhidos aleatoriamente dois pontos de corte p e q, com $p < q$, e um ponto de inserção t escolhido aleatoriamente no intervalo (p, q). Os dois pontos de corte definem uma janela $w_{p,q}$, como no operador crossover. Definida a janela, um outro indivíduo é gerado a partir de F_1 da seguinte forma: considerando a string de F_1 como uma lista circular, a partir do ponto de inserção t, a janela w_{pq} será inserida na cadeia original de F_1, gerando um novo indivíduo $F_1(mut)$. Após a aplicação dos operadores de crossover e mutação, cada descendente será inserido na população caso o valor do seu *fitness* seja superior ao pior indivíduo da população atual. Além disso, a cada geração, a taxa de reposição é de 95%, ou seja, apenas 5% da população é mantida para a geração seguinte.

2.3 O Módulo de Reconexão de caminhos (RC)

A busca local aplicada em AEs tem encontrado bastante aceitação por parte dos pesquisadores desta área (Ochi et al. (2001)), fazendo crescer o interesse no desenvolvimento de trabalhos na área de Algoritmos Evolutivos. Neste contexto, a Reconexãp por Caminhos (RC) (*Path-Relinking*) apresenta-se como uma técnica de busca alternativa eficiente em relação às buscas tradicionais. O RC consiste em gerar todas as soluções intermediárias entre uma solução base (SB) e uma solução alvo (AS). Normalmente ambas as soluções são escolhidas dentre as melhores geradas até o momento pela heurística. Pode-se pesquisar no sentido SB para AS ou vice versa, ou em ambas as direções se isso for conveniente. O princípio básico do RC é o de que entre duas soluções de qualidade pode existir uma terceira melhor que os extremos.

3. RESULTADOS COMPUTACIONAIS E ANÁLISE QUALITATIVA

Inicialmente para avaliar o AEC-RC, implementamos este e também um AG da literatura que segundo os autores apresentavam os melhores resultados para o PCA até então. Para isso, antes de comentarmos detalhes das nossas simulações, vamos resumidamente descrever o AG da literatura CLUSTERING. O algoritmo CLUSTERING de Lin and Shiueng (2001) consiste de um Algoritmo Genético (AG) que, a exemplo do AEC-RC, também se propõe resolver o PCA. Este algoritmo faz uso de uma heurística baseada em busca binária para a obtenção da melhor solução através de várias chamadas do próprio AG com variações de um peso w que, presente na função de aptidão, prioriza uma clusterização com muitos ou poucos clusters conforme w cresça ou diminua. Antes de iniciar a execução do CLUSTERING, vamos explicar a etapa de construção que utiliza também conceitos de componentes conexos como no AEC-RC, mas utilizando fórmulas e critérios diferentes da nossa proposta. Salienta-se que nossa proposta surgiu da necessidade de melhorar o desempenho desta construção de Lin e Shiueng (2001) que como veremos apesar da idéia ser muito boa, a forma de definir os seus passos é extremamente instável. No CLUSTERING calcula-se, para cada ponto i, distância euclidiana entre este ponto e seu vizinho mais próximo, denotada $d_v(i)$ e em seguida toma-se a média destas distâncias, dada por $\ell = \sum_{i=1}^{N} d_v(i)$ para $i = 1 \ldots N$, onde N é o tamanho do conjunto de entrada. A visão de vizinhança do algoritmo é dependente de um parâmetro de entrada α, que será usado para a construção da matriz de adjacência M do conjunto de entrada. A matriz M é gerada de acordo com a equação (14).

$$M[i, j] = \begin{cases} 1 & se \quad d_v(i, j) \le \alpha . \ell \\ o & caso \quad contrário \end{cases} \qquad (11)$$

Da matriz de adjacência *M,* é construído o conjunto $X = \{C_1, ...C_m\}$, onde cada C_i é um componente conexo obtido do grafo associado a *M*, tal qual foi feito no AEC-RC. O algoritmo construirá sua população como *strings* binárias de tamanho *m*, nas quais a ocorrência de 1 numa dada posição *i* da cadeia indica que o i-ésimo componente conexo iniciará um clusters na solução final. O CLUSTERING tem a função de aptidão *f* dada segundo a relação entre o somatório das distancias entre cada ponto e o centróide do cluster ao qual pertence e a distância entre cada ponto e o centróide dos demais clusters, conforme a equação (12), onde podemos ver a existência de um parâmetro de entrada *w*.

$$f(s) = \sum_{i=1}^{k} D_{\text{int er}}(C_i) * w - D_{\text{int ra}}(C_i) \qquad (12)$$

O CLUSTERING recebe como parâmetros de entrada dois valores reais w_s e w_l que definem o intervalo $[w_s,w_l]$ em que o parâmetro *w* da equação (12) assumirá na busca da melhor clusterização, ou seja, aquela com o número ideal de clusters. Na heurística de busca do melhor valor para o parâmetro *w* da equação (12), o módulo genético é chamado inicialmente para $w = w_s$, depois para $w = w_l$ e posteriormente para $w = w_m = (w_s + w_l) / 2$. Podemos ver que o CLUSTERING não tem uma visão de vizinhança apurada. O que o algoritmo faz é calibrar seus parâmetros numa tentativa de achar a calibragem ideal, o que para algumas instâncias acaba sendo uma tarefa árdua e muitas vezes ineficaz.

Os parâmetros usados do CLUSTERING foram os mesmos sugeridos pelos autores, ou seja, tamanho da população igual a 50 indivíduos, número de gerações igual a 100, taxa de crossover igual a 85% e taxa de mutação igual a 5%. No caso do AEC e do AEC-RC, apenas o número de indivíduos foi diferente do CLUSTERING, tendo sido fixado em 10 indivíduos. Utilizamos como medida de qualidade da solução o valor apresentado pela solução de cada algoritmo expresso pela função de aptidão utilizada pelo AGS, já que a solução expressa por esta função está tão próxima da solução ideal quanto seu valor se aproximar de 1. Desta forma, uma vez que a função de aptidão usada pelo algoritmo da literatura (CLUSTERING) na mesma função usada pelo AGS, após o mesmo ter encontrado a solução, esta é avaliada para que seu valor seja expresso com base na equação (10). Os resultados apresentados na tabela (1) apresentam a média dos valores das 100 soluções obtidas por cada algoritmo.

Foram utilizadas 17 instâncias para verificar a eficiência do AGS, deste total, quatro instâncias são da literatura e 13 foram geradas artificialmente de forma a gerar clusters com formatos variados e com densidades variadas. Das quatro instâncias da literatura, listadas em negrito na tabela 1, duas são artificiais no espaço R^2 a saber: Ruspini e 200p Kaufman (1989). A base Ruspini é composta de 75 pontos distribuídos em quatro clusters. Já a base de dados 200p consiste de uma distribuição normal onde cada ponto p_i é um vetor (μ_x, μ_y) com desvio padrão ρ para *x* e *y*. A distribuição gera três grupos conforme segue.

Grupo 1: 120 pontos	$\mu_x = 0$	$\mu_y = 10$	ρ 1.7
Grupo 2: 60 pontos	$\mu_x = 20$	$\mu_y = 12$	ρ 0.7
Grupo 3: 20 pontos	$\mu_x = 10$	$\mu_y = 20$	ρ 1.0

As outras duas instâncias da literatura são bases reais. A instância Winconsin Breast Cancer Database (Mangasarian (1990)) é uma base de dados composta de 699 pontos formados por 9 atributos que caracterizam tumores no tratamento de câncer. Dos 699 pontos, 16 foram excluídos da base por apresentarem atributos ausentes. Desta forma, o conjunto de pontos efetivamente avaliado é composto de 683 elementos. A última instância da literatura é a Iris Plants Database (R. A. Fisher (1936)), composta de 150 pontos no espaço R^4 divididos em três grupos de 50 pontos. Cada grupo refere-se a uma classificação de tipo de iris de planta: íris Setosa, íris Versicolor e íris Virginica. Os atributos dos pontos referem-se às medidas de largura e comprimento da pétala e da sépala da planta. As demais instâncias da tabela 1, foram geradas aleatoriamente. Para efeito de comparação dos resultados, a tabela 1 apresenta desempenho médio do AEC e do AEC-RC além do desempenho do CLUSTERING. Nesta análise, cada algoritmo executou 100 vezes e a média foi calculada para gerar a tabela. Os valores em negrito indicam o melhor desempenho, já que a função objetivo é de maximização.

Instância	**AEC-RC**	**AEC**	**CLUSTERING**
200p	**0.823**	0.823	0.741
CancerData	**0.596**	0.592	0.374
IrisData	**0.686**	**0.686**	0.651
RuspiniData	**0.737**	**0.737**	0.683
1000p6c	**0.727**	0.708	0.367
157p	**0.667**	0.666	0.657
2000p11c	**0.611**	0.602	0.287
2face	**0.666**	**0.666**	0.513

350p5c	**0.758**	**0.758**	0.568
3dens	**0.762**	**0.762**	0.742
97p	**0.710**	**0.710**	0.706
Convdensity	**0.854**	**0.854**	0.818
Covexo	**0.667**	**0.667**	0.618
Face	**0.511**	**0.511**	0.402
Moreshapes	**0.725**	0.720	0.436
Numbers	**0.542**	0.540	0.417
Numbers2	**0.565**	0.562	0.527

Tabela 1: Desempenho médio dos algoritmos avaliados.

Na análise dos resultados obtidos, observamos a eficiência do AEC e principalmente do AEC-RC em termos de encontrar o número ideal de clusters, a sua robustez e a qualidade da solução gerada. Podemos ver que tanto o AEC como o AEC-RC apresentaram desempenho bem superior ao algoritmo da literatura. Além disso, podemos ver que a RC incorporado pelo AEC-RC implicou em melhora do desempenho em 8 das 17 instâncias testadas. Para efeito de verificação da robustez do algoritmo proposto, efetuamos também uma análise probabilística sugerida em Aiex et al. (2003). Esta análise se justifica por uma das limitações clássicas de heurísticas e algumas metaheurísticas que é a sua grande sensibilidade com os parâmetros de entrada do problema. Ou seja, o desempenho destes algoritmos pode variar muito de uma instancia a outra. Para verificarmos a robustez dos métodos aqui propostos, colocamos como critério de parada para todos os algoritmos, um valor alvo (valor de MQ), obtido de simulações preliminares destes métodos para cada instância. O alvo pode ser uma média dos valores obtidos, ou um dos melhores valores, se queremos alvos mais difíceis. Definido o alvo, cada algoritmo é executado m vezes para cada instância selecionada. A cada execução i, armazena-se o tempo de cpu ti, e uma probabilidade $pi = (i - ½).m$ (no nosso caso m = 100). Com isso geramos pontos no R^2 da forma zi = (ti,pi). Como forma de avaliar a eficiência do algoritmo proposto, a figura 1 mostra a análise probabilística do AEC, do AEC-RC e do CLUSTERING, além das versões AEC-SC e AEC-RCSC, que são o AEC e AEC-RC sem a heurística de construção.

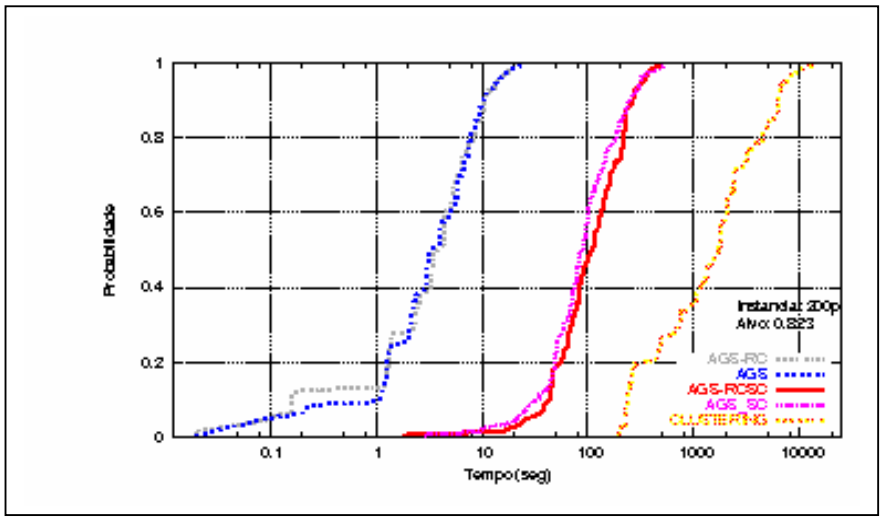

Figura 1: Análise probabilístico para instância 200p.

Podemos ver pela figura 1 que o AEC e o AEC-RC apresentam 100% de chance de chegar ao alvo antes dos primeiros 100 segundos de processamento (as duas curvas mais a esquerda). Já as versões sem a heurística de construção AEC-SC e AEC-RCSC para esta probabilidade precisam de mais de 500 segundos de processamento (duas curvas intermediárias). Ainda assim, as versões sem construção são melhores que o algoritmo da literatura, que para esta probabilidade precisa de mais de 10.000 segundos (curva mais a direita). Isto reflete a robustez dos algoritmos propostos.

4. Conclusões

Neste trabalho apresentamos um algoritmo de clusterização hierárquica para o PCA que, para todas as instâncias submetidas, apresentou resultados de boa qualidade em termos de otimização da função objetivo, do número ideal de clusters e tempo de processamento. Além disso, nossa abordagem enquanto reduz a

cardinalidade do conjunto de entrada, tira proveito desta redução para abreviar o processo de busca, fato este comprovado pela capacidade do algoritmo de encontrar a solução nas primeiras gerações mesmo numa população pequena de indivíduos.

5. Bibliografia.

Aiex, R., Binato, S., e Resende (2003), M. Parallel GRASP with path-relinking for job shop scheduling. Parallel Computing 29, 393 - 430.

R. A. Fisher (1936), The use of multiple measurements in taxonomic problems, Annual Eugenics, **7**, Part II, 179 - 188.

L. Kaufman and P. J. Rousseeuw (1989), Finding Groups in Data - An Introduction to Cluster Analyses, Wiley Series in Probability and Mathematical Statistics, John Wiley and Sons.

O. L. Mangasarian and W. H. Wolberg (1990), Cancer diagnosis via linear programming, SIAM News, **23**, Number 5, September, 1 - 18.

Y. T. Lin, Y. B. Shiueng (2001), A genetic approach to the automatic clustering problem, Pattern Recognition, **34**, 415-424.

Ochi, L. S., Drummond, L. M. A., Vianna, D. S.(2001), "An asynchronous parallel metaheuristic for the period vehicle routing problem", Future Generations on Comp. Systems, Elsevier, pp. 379-386, vol 17(4).

G.Toussaint (1980), The relative neighborhood graph of a finite planar set, *Patt. Recognition,* **12**, 261 - 268.