UM ALGORITMO EVOLUTIVO COM MEMÓRIA ADAPTATIVA E BUSCA LOCAL PARA O PROBLEMA DE CLUSTERIZAÇÃO

marcelo dib cruz (coppe-ufrj)
mdibcruz@bol.com.br
luiz satoru ochi (ic-uff)
satoru@ic.uff.br

Clusterização é o processo no quais elementos de um conjunto são alocados para grupos ou clusters de elementos similares. Nos algoritmos de clusterização, normalmente é assumido que o número de clusters é um dado de entrada. Estes problemass são conhecidos como Problemas de Clusterização (PC). Neste trabalho é apresentado um Algoritmo Evolutivo Híbrido para a solução do PC incluindo módulos de busca local e memória adaptativa. Resultados computacionais realizados para um conjunto de instâncias mostram a eficiência e a robustez da heurística proposta.

Palavras-chaves: Clusterização, Algoritmos Evolutivos, Memória Adaptativa, Metaheuristicas

1. Introdução

Clusterização é o termo genérico para um processo que une objetos similares em um mesmo grupo. Cada grupo é denominado um cluster. O número de clusters é conhecido a priori e denominamos este problema de *Problema de K-Clusterização* ou simplesmente *Problema de Clusterização* (PC).

Existem várias aplicações relacionadas à clusterização incluindo: Particionamento de Grafos, Problema de Manufatura Flexível, Formação de Anéis em Sistemas de Telecomunicações, Reconhecimento de padrões onde se destaca Processamento de Imagens, Biologia Computacional, Pesquisa de Mercado, Classificação de Documentos, Mineração de Dados, entre outros.

O objetivo deste trabalho é propor uma heurística denominada *Algoritmo Evolutivo com Memória Adaptativa (AEMA)* para o PC. Neste contexto, é efetuado inicialmente um pré-processamento para tentar reduzir as dimensões dos dados de entrada do problema através de um algoritmo construtivo. Após esta primeira fase, é utilizado um Algoritmo Genético (AG) com um módulo de Busca Local cuja função, é refinar uma parte das soluções encontradas pelo AG. O algoritmo proposto utiliza adicionalmente conceitos de memória adaptativa na estrutura de um conjunto de soluções elite, cuja função, é armazenar as melhores soluções distintas geradas até o momento pelo algoritmo.

Para avaliar a qualidade da clusterização utilizaremos a função definida por Rao, M. R.(1971). Dado um conjunto de clusters, a função minimiza a maior diagonal de todos os clusters.

A vantagem da utilização desta função é devido à existência de uma formulação matemática linear inteira deste problema. Esta função é conhecida como função Diagonal. Com isso, poderemos avaliar os resultados da heurística em relação às soluções ótimas.

Este trabalho está organizado da seguinte forma: Na seção 2, é descrito o PC acompanhado da formulação matemática do problema e uma revisão parcial dos trabalhos da literatura; na seção 3 é descrito o algoritmo proposto; na seção 4 são mostrados os resultados e análise dos testes computacionais realizados e finalmente, na seção 5 são apresentadas as conclusões.

2. O Problema de Clusterização (PC)

Formalmente podemos definir o problema de clusterização da seguinte forma: dado X um conjunto de n objetos $X=\{x_1, x_2, x_3....x_n\}$, onde cada objeto x_i é uma tupla $(x_{i1},x_{i2},..., x_{ip})$, e cada coordenada x_{ij} está relacionada com um atributo do objeto. Cada objeto pode ser considerado um ponto no espaço R^p. Como objetivo devemos encontrar o conjunto $C=\{C_1,C_2,...C_k\}$ de clusters, onde é k conhecido, onde a similaridade entre os objetos de um mesmo cluster seja maximizada e a similaridade entre objetos de diferentes clusters seja minimizada , sujeito as seguintes condições:

$C_i \neq \phi$, *para* $i = 1,..,k$

$C_i \cap C_j = \phi$, *para* $i, j = 1,..,k$ *e* $i \neq j$

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

Outra definição necessária é o conceito de centróide de um cluster C_i. A substituição de um conjunto C_i com t pontos similares por um único ponto $v_m = (v_{m1}, v_{m2}, ...v_{mp})$ que os represente pode ser feita considerando-se v o centróide de G, dada pela equação

$$v_{mi} = \frac{1}{t}\sum_{j=1}^{t} x_{mj} \quad , \quad i = 1,..,p$$

2.1. Formulação Matemática

Este modelo foi definido por Rao, M. R.(1971). O objetivo deste modelo é minimizar a maior distância interna de todos os clusters, ou ainda, minimizar a maior diagonal de todos os clusters.. Cada ponto só pode pertencer a um cluster e o número de clusters k é um dado de entrada.O modelo está descrito abaixo.

$$Min \quad Z = D_{\max}$$

$$S.A. \quad D_l \geq d_{ij}(x_{il} + x_{jl} - 1) \quad \forall i, j, l, \quad i = 1,\ldots,n, \quad j = 1,\ldots,n, \quad l = 1,\ldots,k$$

$$\sum_{l-1}^{k} x_{il} = 1 \quad \forall i, \quad i = 1,\ldots,n$$

$$D_{\max} \geq D_l \quad \forall l, \quad l = 1,\ldots,k$$

$$x_{il} \in \{0,1\} \quad \forall i, l, \quad i = 1,...,n, \quad l = 1,...,k$$

$$D_l \geq 0 \quad \forall l, \quad l = 1,...,k$$

$$onde\ x_{ij} = \begin{cases} 1, & se\ o\ objeto\ i\ pertence\ ao\ cluster\ j \\ 0, & caso\ contrário. \end{cases}$$

$d_{ij} = distância\ entre\ os\ objetos\ i\ e\ j.$

$e \quad D_l = diâmetro\ do\ cluster\ l.$

$e \quad D_{\max} = o\ maior\ diâmetro\ de\ todos\ os\ clusters.$

2.2. Trabalhos Relacionados

Existem vários modelos de classificação na literatura. O modelo escolhido foi definido por Garai, G., Chaudhuri, B. B.(2004) e possui vários métodos de clusterização, como mostra a figura 1.

Os métodos hierárquicos operam sobre os n pontos do conjunto de entrada X e procuram construir uma árvore de clusters denominada dendograma, na qual, cada nó é um cluster que pode ter outros clusters filhos, pais ou irmãos. Desta forma que cada ponto de X é associado a um único nó da arvore em um dos seus níveis. O modelo apresenta ainda uma divisão dos métodos hierárquicos em dois grupos de acordo como o dendograma é construído. Quando a arvore é construída da raiz para as folhas (estratégia *top-down*) dizemos que o

algoritmo é hierárquico por divisão. . Quando é construído das folhas para a raiz (estratégia *bottom-up*) o algoritmo é dito hierárquico por aglomeração. O hierárquico por divisão começa com um único cluster contendo todos os pontos e recursivamente divide os clusters ate um critério de parada. No hierárquico por aglomeração começa com cada ponto sendo um cluster e a cada momento eles são unidos para formar um novo cluster. Exemplos de algoritmos hierárquicos são CURE (S. GUHA, R. RASTOGI, AND K. SHIM, 1998) e CHAMALEON (G. KARYPIS, E.-H. HAN, AND V. KUMAR,1999)

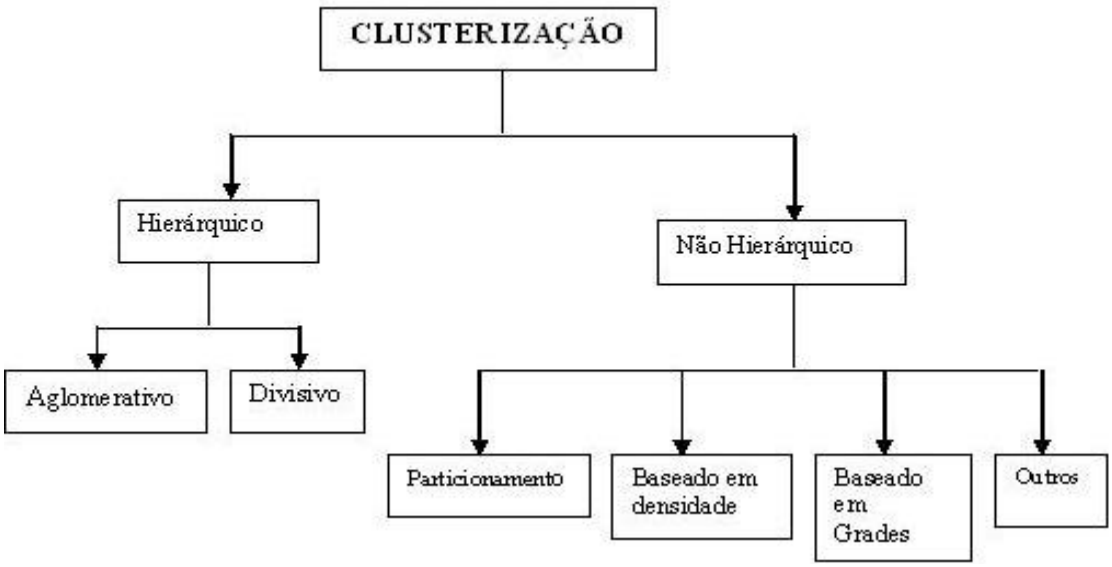

Figura 1: O modelo de clusterização

Os métodos de particionamento consistem em classificar os *n* pontos do conjunto de entrada *X* em *k* grupos de tal forma que cada grupo contenha pelo menos um ponto e que cada ponto pertença a pelo menos um grupo. É uma forma de otimização iterativa, que a cada iteração realoca os pontos entre os clusters. Diferente que os métodos hierárquicos onde os clusters não são revisitados depois de construídos, os métodos de particionamento gradualmente melhoram os clusters. Como exemplo destes algoritmos, temos o k-means (MACQUEEN, J. B.,1967), que é o método mais conhecido da literatura e que a cada iteração constrói o centróide do cluster através da media dos seus pontos e que avalia a clusterização através das distâncias entre os centróides. Temos também as variações do k-means como proposto por Redmond, S. J. , Henegghan , C., (2007).

Os métodos baseados em Densidade são baseados em funções que definem densidade e conectividade. Eles utilizam a idéia de pontos que formam uma região densa podem ser agrupados num mesmo cluster. Essas regiões podem ter qualquer forma e os pontos nessas regiões podem estar arbitrariamente distribuídos. Como exemplo, temos LDBSCAN (DERYA BIRANT AND ALP KUT, (2007).) e o St-DBSCAN (DUAN, L. XU, L., GUO, F., LEE, J., YAN, B. ,2007)

Os métodos Baseados em Grades consistem em dividir o espaço que contém os pontos em um determinado número de células. Células contendo um número relativamente grande de pontos são candidatos potenciais a clusters. O resultado do método depende do tamanho das células, que normalmente é um parâmetro de entrada. Como exemplo, temos o CLIQUE (AGRAWAL, R., GEHRKE, J, GUNOPULOS,D., RAGHAVAN, P. ,2005).

Em outras classes podemos citar o baseado em modelos. Um modelo de dados é utilizado para cada cluster e o objetivo é encontrar os pontos que se adequam melhor para cada modelo de dados. Como exemplo temos o COB-WEB (FISHER, L. ,1987). Temos ainda os métodos que pertencem a mais de um grupo como proposto por Soares, S. S. R., Ochi, L. Satoru, Drummond, L. M. (2006), Tseng, L. Y., Yang, S. B (2001) e Garai, G., Chaudhuri, B. B.(2004) que são algoritmos de duas fases, que na primeira utilizam os conceitos dos métodos baseados em densidade e na segunda fase, utilizam os conceitos dos métodos de particionamento.

3. O Algoritmo Evolutivo Proposto

O *Algoritmo Evolutivo com Memória Adaptativa (AEMA)* aqui proposto, é um método composto de duas fases. Na fase inicial existe uma etapa de pré-processamento cujo objetivo é tentar reduzir as dimensões dos dados de entrada do problema como para gerar soluções iniciais de boa qualidade ao contrário do que ocorre nos AGs tradicionais onde a população inicial é gerada de forma aleatória. Estas duas metas são obtidas através de um algoritmo construtivo baseado em conceitos de *componentes conexas*. Na segunda etapa do AEMA, existe um algoritmo genético com busca local e utiliza conceitos de memória adaptativa para a busca da melhor configuração de uma solução possível. As duas fases do AEMA são denotadas por: fase de construção e o módulo evolutivo

3.1. A Fase de Construção

A fase inicial de construção do AEMA é uma etapa de pré-processamento que tem por objetivos tentar reduzir a cardinalidade dos dados de entrada do problema, bem como facilitar a geração de soluções iniciais de boa qualidade pelo algoritmo genético. O procedimento é baseado no critério de densidade proposto por Tseng, L. Y, Yang, S. B (2001). A idéia é substituir grupos de objetos da base de dados cuja similaridade é considerada alta, por um único objeto *(cluster parcial*) que represente o grupo.

1. ***Procedimento Construtivo (X,u)***
2. ***PARA*** *i = 1* ***até*** *n* ***FAÇA***
3. $d_{\min}(x_i) = \min \| x_i - x_j \|, i \neq j \ , j = 1,...n$
4. ***FIM PARA***
5. $d_{medio} = \frac{1}{n} \sum_{i=1}^{n} d_{\min}(x_i)$
6. $r = u * d_{medio}$
7. ***PARA*** *i = 1* ***até*** *n* ***FAÇA***
8. $N_i = circulo\ (x_i, r)$
9. $T = T \bigcup N_i$
10. ***FIM PARA***
11. *ordenar T em ordem decrescente*
12. *i = 1*

13. ***ENQUANTO*** *(T≠Ø)* ***FAÇA***
14. C_i = *próximo*($N_j \in T$)
15. $T = T - \{N_j\}$
16. $i = i + 1$
17. ***FIM ENQUANTO***
18. *retornar* $C = \{ C_1, C_2, .. C_t \}$ *, os clusters parciais.*
19. ***FIM construtivo***

Figura 2: Descrição do procedimento construtivo

Neste trabalho usamos como medida de similaridade a métrica da distancia euclidiana. A redução do tamanho da entrada é realizada agrupando-se em um mesmo cluster os pontos pertencentes a uma região densa, como mostra a figura 2. Inicialmente, nas linhas 2,3 e 4, para cada ponto é definido a menor distância a outro ponto qualquer. Depois, na linha 5, é feito uma média destas distâncias, denominada d_{medio} . Então, cada ponto $x_i \in X$ é considerado o centro de um círculo cujo valor do raio é $r = u * d_{medio}$, onde u é um parâmetro de entrada. Logo após, na linha 8, é calculado o conjunto de pontos contidos no círculo de centro x_i e raio r ($N_i = circulo\ (x_i, r)$), exemplificado na figura 3. Estes valores são colocados em uma lista T que é ordenada em ordem decrescente. Os elemento de T são considerados os clusters parciais $C = \{C_1, C_2, .. C_t\}$. Para que os clusters não possuam elementos em comum, toda vez que um círculo é selecionado, todos os seus pontos não podem mais entrar em outro círculo. Com este procedimento as regiões mais densas são selecionadas.

Após este procedimento inicial, será realizado um refinamento que tem como objetivo central diminuir o número de clusters parciais. Assim, é efetuada uma agregação de clusters parciais de menor cardinalidade, que sejam adjacentes a algum outro cluster de maior cardinalidade. Isto é realizado verificando se este cluster menor está a uma distancia d_{adj} de outro cluster, onde $d_{adj} = 3 \times d_{medio}$ *(onde o valor 3 é um dado de entrada).*

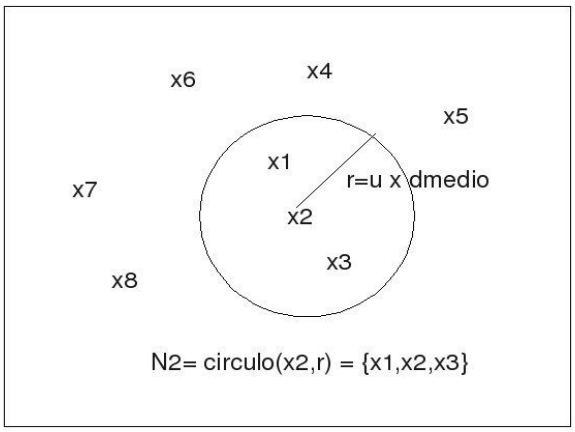

Figura 3: Cálculo dos pontos contidos no círculo de centro x_2 e raio r = u x d_{medio}

Os clusters gerados no procedimento construtivo são utilizados neste fase da seguinte forma, como mostra a figura 4. Para cada cluster parcial *pequeno*, verifique qual o cluster que está mais próximo em relação à distância entre os centróides (linhas 4, 5 e 6). Se este cluster possuir cardinalidade maior e possuir pelo menos um ponto a uma distancia de no máximo d_{adj} de qualquer ponto do outro cluster, então ele será incorporado ao outro, como mostram as linhas 7 e 8. Denominamos este procedimento de *Junção de Clusters Iniciais Adjacentes*

(JCIA) e tem por objetivo incorporar clusters pequenos a clusters maiores. Este procedimento vai reduzindo o número de clusters remanescentes da primeira fase.

1. ***Procedimento*** *JCIA* ***(X,t,d_{medio})***
2. *C = construtivo(X)*
3. *d_{adj}= 3 * d_{medio}*
4. ***PARA*** *i = 1* ***até*** *t* ***FAÇA***
5. ***SE*** *(cardinalidade (C_i)<=4)* ***ENTÃO***
6. *C_k = menor_distancia_centroide(C_i)*
7. ***SE*** *($\exists x_r \in C_i$ e $\exists x_s \in C_k$ tq $\| x_r - x_s \| < d_{adj}$)* ***ENTÃO***
8. *$C_k = C_k \cup C_i$*
9. ***FIM SE***
10. ***FIM SE***
11. ***FIM PARA***
12. *retornar C= {C_1,C_2,...,C_m} clusters iniciais, onde m <= t*
13. ***FIM JCIA***

Figura 4**:** Descrição do procedimento JCIA

Os clusters gerados na fase de construção, após o procedimento construtivo e o JCIA, são denominados clusters iniciais. Como a seqüência de análise dos clusters de menor cardinalidade são escolhidos de forma aleatória, ao final uma população de soluções iniciais distintas com diferentes números de clusters podem ser geradas por este procedimento.

3.2. O Módulo Evolutivo

O módulo evolutivo é composto de um Algoritmo Genético tradicional acrescido de duas buscas locais. A população inicial é construída a partir dos clusters iniciais gerados na fase de construção. Porém, o algoritmo genético na sua forma tradicional nem sempre consegue bons resultados. Então, é necessária a utilização de busca local para melhorar a qualidade das soluções. Por isso, foi proposto uma busca local denominada *Troca entre Pares* que será realizada nos melhores indivíduos da população, a cada X (cinco) iterações (X é dado entrada), e que trocará dois a dois, todos os elementos do individuo com valores diferentes , para investigar possíveis soluções diferentes com o mesmo número de clusters. A busca *Troca entre Pare* será explicada na seção 3.3.

Considere os clusters iniciais gerados na fase de construção como $C=\{C_1,...,C_m\}$ e seja v_i, $i = 1,2,...m$ o centróide do cluster C_i. Para representar uma solução utilizaremos uma cadeia binária de *m* posições. Seja $B= (B_1,B_2,...B_m)$ onde cada B_i tem valor *0* ou *1*. Se $B_i = 1$, o cluster inicial C_i fará parte da solução como cluster pai (cluster raiz). Se $B_i = 0$, C_i é considerado um cluster filho e será unido posteriormente a um dos clusters pais. Os clusters filhos serão unidos aos clusters pais utilizando o critério de menor distância entre os centróides. A cada união, o valor do centróide será recalculado. No final, todos os clusters filhos serão acoplados aos clusters pais para gerar uma solução. Portanto, o número de clusters pais gerados em cada indivíduo da população não será alterado. Os clusters gerados após esse processo são denominados clusters finais $C =\{C_1,C_2,...,C_k\}$.

Para mostrar melhor o processo, seja $B^o = \{B_1^o, B_2^o,...B_r^o\}$ um subconjunto de B onde seus elementos tem valor 0 e $B^1= \{ B_1^1,B_2^1,...,B_s^1\}$ um subconjunto de *B* onde seus elementos tem valor 1 e $m = r + s$. Inicialmente cada B_i^1 é candidato único a cada cluster C_i.. Então para cada $B_h^o \in B^o$, encontrar um $B_z^1 \in B^1$ que satisfaça a relação abaixo :

$B_z^1 = min \,||\, v_i^1 - v_h^o ||\,,\ i = 1,....,r.$

A cada busca, B_z^1 é atualizado incorporando B_h^o e o centróide é recalculado. O conjunto B^o é atualizado. O processo continua até $B^o = Ø$.

Para avaliar a solução gerada, o algoritmo utiliza a mesma função do modelo exato proposto. Para isso, é definido d_{ij} que é a distância euclidiana entre os objetos x_i e x_j. Então, a diagonal de um cluster C_j é definida como:

e a função

$$f(C) = Min\ d(C_j), \quad j = 1, \cdots, k$$

Para construir a população inicial, o algoritmo gera um número bem maior de indivíduos (dez vezes o tamanho da população) e escolhe aqueles com os maiores valores de aptidão. Este procedimento permite começar o algoritmo genético com uma população de melhor qualidade.

No algoritmo genético, são utilizados operadores clássicos como o operador de mutação e também novos operadores propostos neste trabalho, como o operador de seleção para o cruzamento. Para efetuar a seleção dos indivíduos para cruzamento, são utilizados dois operadores. Esses operadores se alternam. O primeiro operador escolhe aleatoriamente os dois indivíduos dentre os 30% com melhores valores de aptidão. E o segundo operador, escolhe um indivíduo aleatoriamente dentre os 30% com melhores valores de aptidão e o outro, entre os 70% restantes. O operador de cruzamento utilizado no AEMA é o cruzamento de dois pontos, que atua sobre dois indivíduos com genótipos diferentes. Este operador funciona da seguinte maneira: pares de pontos de cruzamento são obtidos de forma aleatória e os valores dos indivíduos localizados entre cada par de pontos de cruzamento são trocados. Os indivíduos são submetidos ao cruzamento com probabilidade p_c. Os dois pontos de corte definem os segmentos dos vetores que serão trocados entre os mesmos para gerar novos indivíduos. Quando, após o cruzamento e a mutação, o número de clusters pais de um indivíduo é diferente de *k*, o algoritmo seleciona aleatoriamente elementos deste para igualar a *k*.

O operador de mutação realiza trocas aleatórias de alguns valores dos indivíduos com probabilidade p_m, com o intuito de pesquisar novas áreas do espaço de busca, a partir de indivíduos selecionados.

Após a aplicação dos operadores de cruzamento e mutação, os descendentes que obtiverem valores de aptidão melhores que os valores da população atual são inseridos na nova população. A cada iteração a melhor solução será armazenada no conjunto Elite. No final, o conjunto elite passará pela busca local Troca entre Pares.

3.3. A Busca Local Troca Entre Pares

O terceiro procedimento do algoritmo AEMA é a realização de uma busca local denominada *Troca entre Pares.* Esta busca chamada também de busca intensiva, troca o status de dois elementos do indivíduo com valores diferentes. Por exemplo, suponha que o indivíduo é *{10111010}*. Primeiro é trocado o primeiro e o segundo elemento do individuo. Então o novo indivíduo é *{01111010}*. Se este novo indivíduo melhorar o valor de aptidão,

então ele será aceito e será a nova solução corrente. A próxima troca é feita entre o primeiro e o terceiro elemento sobre a solução corrente. A busca local termina quando todas as trocas entre dois genes com valores distintos são testadas.

Devido ao elevado tempo computacional exigido por este modulo, esta busca é utilizada 3 vezes durante as gerações e ao final do AG nos *3* melhores indivíduos do conjunto elite final (conjunto das *q* melhores soluções geradas pelo AG).

4. Resultados Computacionais

Para avaliar o desempenho do algoritmo AEMA aqui proposto, testes computacionais foram realizados. Todos os algoritmos foram implementados usando compilador C++ no ambiente Linux Ubuntu 7.5. Nos testes de contagem de tempo foram utilizados computadores com processadores AMD Athlon 64 X2 Dual core 4800 com 2G de memória RAM.

Foram avaliadas cinco (5) instâncias, duas são conhecidas na literatura como RuspiniDataSet e 200DATA (FISHER, R.,1936). Outras três instâncias foram construídas através de uma ferramenta gráfica denominada *Dots*, desenvolvida por Soares, S. S. R., Ochi, L. Satoru, Drummond, L. M. (2006), denominadas 100p3c, 200p4c e 300p3c com 100. 200 e 300 pontos respectivamente. A característica principal desta ferramenta é gerar instâncias onde a solução ótima pode ser visualizada. Cada instância possui o número de pontos e o número ideal de clusters. Por exemplo, a instância 200p4c possui 200 pontos e o número ideal de clusters é 4. Com isso, podemos avaliar o comportamento do algoritmo próximo ao seu número ideal de clusters. A tabela 1 mostra os resultados obtidos. Esta tabela mostra o nome da instância, a variação do *k*, o valor ótimo para cada *k* (os valores ótimos foram obtidos através da implementação da formulação matemática, descrito na seção 2.2, utilizando o *software* XPRESS), o tempo em segundos exigido pelo XPRESS, e o melhor valor encontrado pelo AEMA, o percentual do AEMA em relação ao ótimo (%) e na última coluna o tempo em segundos do AEMA.

Observamos que à medida que o valor de *k* aumenta, a dificuldade do método exato em encontrar o ótimo aumenta, pois o número de restrições aumenta muito. Portanto, só consideramos valores de *k* que puderam ser executados pelo XPRESS em no máximo *8* horas (ou *28800* segundos) de processamento.

Em relação aos parâmetros utilizados no algoritmo AEMA, estes foram definidos a partir de testes preliminares. A taxa de cruzamento dos indivíduos de cada geração foi fixada em *80%* do tamanho da população, e a taxa de mutação foi fixada em *10%* do tamanho da população. Foi utilizado o valor de u variou entre *0.5 e 2.* O tamanho da população escolhido foi de *1/3* do tamanho do indivíduo com um valor máximo de *20* (vinte) indivíduos. O tamanho do conjunto Elite foi fixado com *5* (cinco) elementos.

Instância	k	ótimo	t(s)	AEMA	%	t(s)
ruspini	K=7	32,984	14163s	32,984	**0.0%**	13s
	K=6	36,619	1345s	36,619	**0.0%**	11s
	K=5	40,249	658s	40,249	**0.0%**	9s
	K=4	47,634	60s	47,634	**0.0%**	7s
	K=3	88,588	18s	90,553	2,25%	7s
	K=2	102,078	9s	102,078	**0.0%**	6s

200DATA	K=4	9,257	3688s	9,257	**0.0%**	33s
	K=3	10,308	344s	10,308	**0.0%**	24s
	K=2	16,669	243s	16,669	**0.0%**	22s
100p3c	K=6	38,600	5763s	38,600	**0.0%**	17s
	K=5	39,217	355s	39,217	**0.0%**	15s
	K=4	57,982	449s	57,982	**0.0%**	14s
	K=3	66,069	24s	66,069	**0.0%**	12s
	K=2	129,096	10s	129,096	**0.0%**	9s
200p4c	K=4	69,641	23250s	69,641	**0.0%**	40s
	K=3	133,304	288s	133,304	**0.0%**	66s
	K=2	182,486	118s	182,486	**0.0%**	28s
300p3c	K=4	129,465	28563s	131,712	1,73%	88s
	K=3	140,132	926s	140,132	**0.0%**	72s
	K=2	235,767	650s	235,767	**0.0%**	56s

Tabela 1: Comparação da Heurística com o método Exato

Em todas as instâncias o que se observa é que o AEMA consegue resultados muito bons, atingindo o ótimo ou ficando sempre muito próximos ao ótimo. Acreditamos que isso se deve aos procedimentos aqui propostos: o primeiro é o algoritmo construtivo, que forma clusters iniciais já de forma semi-otimizada além deste procedimento reduzir as dimensões dos dados de entrada. O segundo, responsável pelo bom rendimento da heurística proposta, se deve a busca local, que possibilita verificar muitos ótimos locais, aumentando com isso a chance de se obter soluções de boa qualidade ao seu final. Outra vantagem no uso de heurísticas é a constatação de que à medida que o valor k aumenta, torna-se mais oneroso ou inviável o uso de métodos exatos como comprovado de forma empírica neste trabalho.

5. Conclusão

Este trabalho aborda a utilização de algoritmos evolutivos híbridos com memória adaptativa na resolução do PC. Constatamos mesmo que de forma empírica a importância de se usar heuristicas híbridas com regras de redução (pré-processamento) e buscas locais eficazes. Como trabalhos futuros, incluímos o uso dos módulos construtivo e busca local aqui propostos em outras metaheuristicas tais como o *Iterated Local Search* e *Tabu Search* na solução do Problema de Clusterização.

Agradecimentos

Os autores agradecem ao apoio parcial dos seguintes órgãos de fomento: CNPq: (CT-INFO & UNIVERSAL & Produtividade); CAPES:(Pró-Engenharia); FAPERJ: (PENSA RIO & Cientista do Estado).

Referências Bibliográficas

RAO, M. R. Cluster Analisys and Mathematical Programming. J. American Statistic Association . Vol 66. pp. 622-626. 1971

VINOD, H. D. Integer Programming and Theory of Groups; J. American Statistic Association . Vol 64. pp. 506-519. 1969.

GARAI, G. , CHAUDHURI, B. B. A novel genetic algorithm for automatic clustering. Pattern Recognition Letters, pp. 173-187.2004

S. GUHA, R. RASTOGI, AND K. SHIM. "CURE: An efficient clustering algorithm for large databases", *Proc.of ACM SIGMOD Conference*, pp. 73-84. 1998

G. KARYPIS, E.-H. HAN, AND V. KUMAR. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling",*IEEE Computer*, 32 (8), pp. 68-75. 1999

MACQUEEN, J. B. Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1, pp. 281-29.1967

REDMOND, S. J. , HENEGGHAN , C. A method for initialing the k-means clustering algorithm using kd-trees. Pattern Recognition Letters 28. 2007

DERYA BIRANT AND ALP KUT. ST-DBSCAN: An algorithm for clustering spatial–temporal data. Data & Knowledge Engineering 60 , pp. 208-221. 2007

DUAN, L. XU, L., GUO, F., LEE, J., YAN, B. A local-density based spatial clustering algorithm with noise, Information Systems 32;pp. 978-986. 2007

AGRAWAL, R., GEHRKE, J, GUNOPULOS,D., RAGHAVAN, P. Automatic Subspace Clustering of High Dimensional Data. Data Mining and Knowledge Discovery, 11, 5–33 . 2005

FISHER, L. ;Knowledge acquisition via incremental conceptual clustering, Machine Learning 2 (2) . 139–172. 1987

SOARES, S. S. R., OCHI, L. SATORU, DRUMMOND, L. M. Um algoritmo de construção e busca local para o Problema de Clusterização de Bases de Dados. Tendências de Matemática Aplicada e Computacional, Vol. 7(1), pp. 109-118. 2006

TSENG, L. Y., YANG, S. B. A genetic approach to the automatic clustering problem; Pattern Recognition 34, pp. 415- 424. 2001

GARAI, G., CHAUDHURI, B. B. A novel genetic algorithm for automatic clustering. Pattern Recognition Letters, pp. 173-187. 2004

FISHER, R. .The use of multiple measurements in taxonomic problems. Annual Eugenics 7,pp. 179-188.1936