OTIMIZAÇÃO DO PROCESSO DE SELEÇÃO DE ATRIBUTOS PARA AGRUPAMENTO DE DADOS

José Cláudio Garcia Damaso

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

_____

Prof. Beatriz de Souza Leite Pires de Lima, D.Sc

_____

Prof. Nelson Francisco Favilla Ebecken, D.Sc

_____

Prof. Maria Inês Vasconcellos Furtado, D.Sc

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2008

DAMASO, JOSÉ CLÁUDIO GARCIA
Otimização do Processo de Seleção de Atributos para Agrupamento de Dados [Rio de Janeiro] 2008
VII, 106p. 29,7cm (COPPE/UFRJ, M.Sc., Engenharia Civil, 2008)
Dissertação - Universidade Federal do Rio de Janeiro, COPPE
1.Seleção de Atributos 2.Algoritmo Genético 3.Agrupamento de dados 4. Índices de Agrupamento
I.COPPE/UFRJ II.Título (série)

AGRADECIMENTOS

Primeiramente a Deus, por ter guiado meus passos, juntamente com seus enviados, para que eu pudesse atingir este grande objetivo.

À Professora Beatriz e Nelson, por toda ajuda, paciência e boa vontade para que eu completasse mais esta importante etapa de minha vida com êxito.

À minha esposa Hanriette, pela confiança, paciência, apoio e extrema colaboração durante todo o meu período de mestrado, tempo este que, quando estive nervoso, desanimado, sem esperança ou perdendo o objetivo, sempre ouvi suas palavras de carinho, conforto, consolo e animação.

Aos meus pais Arthur e Nolmira, minha avó materna Maria e todos os meus familiares e amigos, pela formação individual e compreensão pelas horas ausentes para que fosse possível finalizar com sucesso mais esta tarefa empreendida.

Um especial agradecimento a Lúcio, que lançou a idéia, muito me apoiou e esteve sempre atento ao andamento deste desafio, não me deixando estremecer em nenhum momento deste percurso.

A Anderson por realizar a revisão do texto desta dissertação, bem como a Rafael, Fernando, Inês e Frederico pela ajuda técnica e colaboração oferecida por todos durante minha fase de mestrado e dissertação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

OTIMIZAÇÃO DO PROCESSO DE SELEÇÃO DE ATRIBUTOS PARA AGRUPAMENTO DE DADOS

José Cláudio Garcia Damaso

Março/2008

Orientadora: Beatriz de Souza Leite Pires de Lima

Programa: Engenharia Civil

Este trabalho apresenta um programa computacional implementado na linguagem Java que seleciona atributos mais representativos de uma base de dados, agrupa dados utilizando a técnica de segmentação e calcula índice de agrupamentos.

A seleção de atributos utiliza a técnica probabilística de algoritmo genético através da correlação de variáveis para selecionar as mais correlatas entre si e menos correlatas entre as outras classes.

O agrupamento de dados utiliza a técnica de segmentação utilizando método particional K-Means e os resultados podem ser encontrados e avaliados através dos índices Calinski e Harabasz ou PBM.

Para validar a ferramenta desenvolvida foram empregadas 4 bases de dados e os resultados foram bastante satisfatórios.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ATRIBUTE SELECTION OTIMIZATION PROCESS FOR DATA CLUSTERING

José Cláudio Garcia Damaso

March/2008

Advisor: Beatriz de Souza Leite Pires de Lima

Department: Civil Engineering

This paper presents a computer program implemented in Java language that selects the most representative attributes from a data base, executes data clustering using segmentation technique and calculates clustering index.

The attribute selection uses the genetic code probabilistic algorithm technique through correlation of variables to choose the most correlated with each other and the less correlated among the other classes. The user chooses how many attributes will be selected as a result.

Data clustering uses the segmentation technician through the partitional K-Means method and its result can be viewed and evaluated either by the Calinski or Harabasz or PBM indexes.

The validation of the program developed in this work was done by using 4 data bases and the results found were very good.

SUMÁRIO

RESUMO iv
ABSTRACT v
1 INTRODUÇÃO 1
2 SELEÇÃO DE ATRIBUTOS 3
2.1 INTRODUÇÃO 3
2.2 TÉCNICA PARA SELEÇÃO DE ATRIBUTOS CFS 4
3 SELEÇÃO DE ATRIBUTOS POR ALGORITMO GENÉTICO 6
3.1 ALGORITMO GENÉTICO 6
3.2 CODIFICAÇÃO DAS VARIÁVEIS 10
3.3 CRIAÇÃO DA POPULAÇÃO INICIAL 12
3.4 APTIDÃO - AVALIAÇÃO DA RESPOSTA 13
3.5 CRUZAMENTO 13
3.6 MUTAÇÃO 14
3.7 DEFINIÇÃO DOS PARÂMETROS DOS OPERADORES GENÉTICOS 14
4 AGRUPAMENTO DE DADOS 17
4.1 INTRODUÇÃO 17
4.2 CLASSES DE ALGORITMOS DE AGRUPAMENTO 18
4.3 ALGORITMO K-means 18
4.4 ÍNDICES DE AGRUPAMENTO 20
4.4.1 CALINSKI E HARABASZ 20
4.4.2 PBM 22
5 APLICATIVO AJC 24
6 EXPERIMENTOS E RESULTADOS 29
6.1 AGRUPAMENTO DE DADOS SEM SELEÇÃO DE ATRIBUTOS 29
6.1.1 BASE DE DADOS ÍRIS 29
6.1.2 BASE DE DADOS GLASS 31
6.1.3 BASE DE DADOS HEART 34
6.1.4 BASE DE DADOS CANA DE AÇÚCAR 36
6.2 AGRUPAMENTO DE DADOS COM SELEÇÃO DE ATRIBUTOS 39
6.2.1 AVALIAÇÃO DA SELEÇÃO DE ATRIBUTOS 39
6.2.1.1 BASE DE DADOS ÍRIS 40
6.2.1.2 BASE DE DADOS GLASS 41

6.2.1.3 BASE DE DADOS HEART 43
6.2.1.4 BASE DE DADOS CANA DE AÇÚCAR 45
6.2.2 AVALIAÇÃO DO AGRUPAMENTO COM ATRIBUTOS SELECIONADOS 46
6.2.2.1 BASE DE DADOS ÍRIS 46
6.2.2.2 BASE DE DADOS GLASS 49
6.2.2.3 BASE DE DADOS HEART 52
6.2.2.4 BASE DE DADOS CANA DE AÇÚCAR 55
7 CONCLUSÃO E TRABALHOS FUTUROS 59
8 REFERÊNCIAS BIBLIOGRÁFICAS 61
9 APÊNDICE 64

1 INTRODUÇÃO

A motivação para este trabalho é o constante esforço aplicado na direção do estudo dos dados, que vem, ao longo dos anos, despertando grande interesse em vários segmentos, incluindo científicos e comerciais. Esse estudo pode ser exploratório ou confirmatório, baseado na disponibilidade do modelo apropriado para um determinado grupo de dados. Entretanto, o elemento chave para esses dois tipos de procedimentos é o agrupamento ou a classificação baseados na melhor aptidão para uma classificação ou um agrupamento natural revelado através da análise, onde essa análise de grupos é a organização de uma coleção de padrões, usualmente conhecida como um vetor de medidas, em grupos baseados em similaridades.

De uma maneira natural e lógica, podemos afirmar que os padrões dentro de um grupo são mais similares para cada um de seus elementos do que para os padrões pertencentes a outros grupos.

É sempre importante lembrarmos que existe uma grande diferença entre o agrupamento e a classificação de dados. Nesta última, tem-se uma coleção de dados, onde cada elemento possui um rótulo com seu padrão e, desta forma, podemos rotular um elemento novo com base nos padrões estabelecidos para o grupo. Assim podemos entender que os padrões já rotulados são usados para aprender as características das classes, que por sua vez, são usados para rotular um novo padrão. No caso do agrupamento, o problema reside em agruparmos uma determinada coleção de dados de padrões não estabelecidos em grupos significativos. A técnica de agrupamento é aplicada na análise de exploração de padrões, no agrupamento propriamente dito, na tomada de decisão, nos casos de auto-aprendizado (*machine-learning*), incluindo mineração de dados, mineração de textos, segmentação de imagens e classificação de padrões. Entretanto, em muitos desses casos, existe uma informação estatística anterior sobre os dados que o tomador de decisão deve aproveitar ao máximo. E é a partir dessa grande característica que a metodologia de agrupamento é apropriada na exploração do inter-relacionamento e posterior formação de uma estrutura de grupos.

O termo agrupamento é usado em muitas comunidades para descrever métodos que realizam operações em dados não supervisionados. Essas comunidades possuem diferentes terminologias e considerações sobre o processo de agrupamento e, também, sobre o contexto no qual o agrupamento é usado [1].

O objetivo principal deste trabalho é aproveitar os conceitos dos Algoritmos Genéticos, Agrupamento de Dados e Índices de Grupos, com opção de serem independentes umas das outras para criar uma ferramenta de fácil utilização, com uma interface gráfica amigável e intuitiva. Essa ferramenta, denominada AJC, utiliza um Algoritmo Genético para a seleção de variáveis - onde essa opção não se faz obrigatória - e um agrupamento dos dados em grupos distintos.

2 SELEÇÃO DE ATRIBUTOS

2.1 INTRODUÇÃO

Muitos fatores afetam o sucesso dos algoritmos de mineração de dados para uma determinada tarefa. A qualidade dos dados é um desses fatores. Se uma informação é irrelevante ou redundante, ou os dados possuem alguma forma de ruído ou são ilegíveis, então a descoberta de conhecimento durante a fase de treinamento é muito mais difícil.

No processo de seleção dos atributos, quanto maior o número de atributos em uma base de dados, maior o poder discriminatório do algoritmo de agrupamento e a facilidade de obter-se modelos de conhecimento. Devemos sempre lembrar que quando pensamos em essência, muita coisa funciona de maneira excelente, mas quando entramos no mundo real nos confrontamos com o volume dos dados, pois o tempo computacional pode aumentar agressivamente se a quantidade de atributos for grande.

A seleção de atributos é o processo de identificação e remoção, tanto quanto possível, das redundâncias ou informações irrelevantes. Muitos algoritmos de aprendizagem empregam diferentes soluções na seleção dos atributos. Em uma extremidade estão os algoritmos como os de aprendizagem dos vizinhos mais próximos, que classificam novos exemplos através da recuperação do vizinho mais próximo registrado, utilizando-se de todas as características disponíveis com base nas distâncias calculadas. Do outro lado estão os algoritmos que explicitamente aplicam prioridades para as características relevantes e ignoram as irrelevantes. As induções das árvores de decisão são exemplos de aproximações. Pelo teste dos valores de alguns atributos, muitos algoritmos tentam separar os dados de treinamento em subgrupos que contenham uma grande maioria de uma classe. Isso requer a seleção de um grupo pequeno das maiores características preditivas, para evitar que os dados sejam separados erroneamente. Sem querer levar em consideração a tentativa de aprendizado para seleção dos atributos ou ignorar a questão, a seleção de atributos antes do aprendizado pode ser benéfica. A redução da dimensionalidade dos dados reduz o tamanho do espaço de hipóteses e permite ao algoritmo operar mais rápido e eficientemente [2].

Blum e Langley [3] e Dash e Liu [4] apresentam bons trabalhos no campo de aprendizagem de máquina através da seleção de atributos. Particularmente, Dash e Liu [4] utilizam pequenos arquivos de dados para explorar os pontos fortes e fracos dos

diferentes métodos de seleção de atributos com relação à questões como ruído, diferentes tipos de atributos, dados com múltiplas classes e complexidade computacional.

As técnicas de seleção de atributos podem ser categorizadas de acordo com um número de critérios. Uma categorização popular nomeou termos *filter* e *wrapper* para descrever a natureza das métricas usadas para auxiliar a avaliação dos atributos [5]. *Wrappers* avaliam os atributos pelo uso da acurácia estimada pelo atual alvo do algoritmo de aprendizado. *Filter*, por outro lado, utiliza as características gerais dos dados para avaliar os atributos e operar independentemente de qualquer algoritmo de aprendizado. Outra taxonomia útil pode ser conquistada pela divisão de algoritmos entre aqueles que avaliam atributos individuais e os que avaliam os subgrupos dos atributos. Esse último grupo pode ser mais diferenciado nos princípios da técnica comum de pesquisa empregada em cada método para explorar o espaço de busca do subgrupo. Algumas técnicas de seleção de atributos podem suportar problemas de regressão, isto é, quando existem muitas classes e poucas variáveis.

Existem muitos outros métodos largamente utilizados para a seleção de atributos, entre eles a técnica de Correlação de Variáveis (CFS), Ganho de Informação na Contagem de Atributos, Componentes Principais, *Relief* e Avaliação dos Subgrupos Baseada na Consistência.

2.2 TÉCNICA PARA SELEÇÃO DE ATRIBUTOS CFS

Neste trabalho utiliza-se a seleção de atributos baseada na correlação das variáveis, também conhecida como CFS (*Correlation-based Feature Selection*) [6,7], que é o primeiro dos métodos que avaliam os atributos de cada subgrupo mais do que os atributos individualmente. O centro do algoritmo é a heurística da avaliação dos subgrupos, que leva em consideração a utilização das características individuais para predição de cada grupo, onde o nível de correlação é analisado. Essa heurística assinala maior pontuação para os subgrupos que possuem os atributos mais correlatos entre si e menos correlatos entre as outras classes. Pelo fato dos atributos serem tratados independentemente, o CFS não pode identificar fortes interações de características,

como em um problema de paridade. Entretanto, ele tem mostrado ser muito eficiente para identificar os atributos sob níveis moderados de interações.

O CFS aplica a heurística da busca estratégica para achar características de bons grupos, que é definido de acordo com os atributos selecionados e com uma considerável contribuição para o subgrupo calculado.

Para a sua aplicação é necessário calcular a dependência entre os atributos. Para isso, utilizam-se métodos estatísticos para se obter média entre os valores de cada atributo, seus desvios padrões, bem como a correlação entre cada atributo para o posterior cálculo da matriz de correlação. O algoritmo segue com a criação de uma população inicial gerada aleatoriamente. Em seguida, cada indivíduo da população inicial é avaliado acumulando-se o produto de cada atributo deste indivíduo com o valor do respectivo atributo da matriz de correlação. Um valor adicional arbitrário é acrescentado a essa aptidão calculada, que chamamos convergência. Esta é calculada em etapas. Primeiro, realizar o somatório dos valores da população inicial correspondentes a cada atributo (que só podem ser 0 ou 1). Segundo, subtrai-se o valor total de atributos pelo somatório dos valores do indivíduo, calculado na primeira etapa. Terceiro, calcula-se o produto entre a diferença calculada e 2, que é o peso atribuído no AJC. Por fim, acrescenta-se a convergência ao valor acumulado do indivíduo. Para cada indivíduo, valida-se se sua aptidão é a melhor aptidão calculada. Caso afirmativo, seu cromossomo é guardado. Este processamento é repetido até que se atinja o número de população estipulado.

Depois de ter sido calculada a aptidão da população, efetua-se o cruzamento selecionando-se como pai metade da população de melhor aptidão e, aleatoriamente, a outra metade como mãe para completar toda a população que sofrerá o cruzamento, considerando-se o parâmetro passado para o AJC.

Logo em seguida há o processo de mutação, que efetuará a troca de valores dos genes de cada cromossomo de todas as gerações de acordo com o parâmetro passado para o AJC. Como os valores só podem ser 0 ou 1, onde for 0 passará a ser 1 e onde for 1 passará a ser 0.

O passo seguinte é recalcular a aptidão da nova população criada.

No final do processamento será exibido cada gene do melhor cromossomo. O AJC também poderá utilizar este cromossomo como entrada para o seu agrupamento de dados.

3 SELEÇÃO DE ATRIBUTOS POR ALGORITMO GENÉTICO

3.1 ALGORITMO GENÉTICO

Os Algoritmos Genéticos (AG), integram a família dos algoritmos evolucionários, que também é composta por Estratégia de Evolução e Programação Evolutiva.

John H. Holland [8], um conhecido pesquisador da Universidade de Michigan, apresentou, nos anos 60, uma proposta para construção de um algoritmo estatístico-matemático para otimização aplicável em sistemas muito complexos. Este algoritmo ficou conhecido como Algoritmo Genético, que propôs uma simulação da evolução inspirada na natureza, com características e ganhos desse processo.

Os Algoritmos Genéticos (AG), propostos por Holland, são uma classe de algoritmos de busca probabilísticas que acumulam operadores genéticos naturais, e são capazes de localizar uma solução ótima ou próxima da ótima em uma larga escala de problemas.

Somente nos início dos anos 80, esta solução algorítmica ganhou grande incentivo prático, vindas de muitas áreas de aplicações científicas, que necessitavam da versatilidade e dos excelentes resultados que já haviam sido apresentados por esta forma de solução.

O Algoritmo Genético, então, passou a ser aplicado cada vez mais em várias áreas com excelentes resultados no processamento de imagem, modelagem e identificação de sistemas, filtros para minimização de ruídos, robótica e seleção de entradas em Redes Neurais [9].

Ultimamente este algoritmo ganhou um grande espaço em trabalhos nas mais diversas áreas, pois há uma crescente concentração de informações que podem ser trabalhadas e os problemas complexos necessitam de uma melhor otimização na escolha das variáveis, dando preferência àquelas mais significativas.

Os algoritmos genéticos são os mais conhecidos e empregados dentre os algoritmos evolucionários. São mecanismos de busca ou otimização, inspirados nos mecanismos de seleção natural e na genética natural. Eles combinam sobrevivência da estrutura mais adaptada com troca de informação aleatória, formando um algoritmo de busca com elementos que lembram o comportamento inovador humano. Em cada geração um novo conjunto de criaturas artificiais é gerado usando alguns dos indivíduos

mais ajustados da geração anterior. Alguns novos indivíduos podem ser incluídos ou alterados aleatoriamente [10].

Os dados são convertidos em um código que representa um gene ou cromossomo (indivíduo) e o processamento ocorre através dos operadores genéticos de seleção natural (indivíduos mais aptos), cruzamento e mutação. Estes operadores alteram a população inicial composta de soluções candidatas gerando indivíduos novos potencialmente melhores. A seleção é realizada através de um mecanismo probabilístico (geralmente através de um mecanismo conhecido como "roleta") garantindo a tendência de que os indivíduos com maior aptidão (mais próximos da solução) permanecem para a próxima geração. O cruzamento ocorre com a fusão de dois indivíduos, divididos em uma posição aleatória e re-combinados. Já a mutação consiste na alteração aleatória de parte do código de alguns indivíduos. Muitas vezes os cromossomos são binários, mas também podem ser números reais [10,11].

Por outro lado, segundo Mitchell [12], não existe uma definição aceita de algoritmos genéticos que permita diferenciá-lo totalmente dos outros métodos evolucionários. Porém, pode-se dizer que os métodos chamados genericamente de algoritmos genéticos têm ao menos os seguintes elementos em comum: populações de cromossomos (representando potenciais soluções), seleção de acordo com uma função-objetivo (aptidão), cruzamento para produzir uma nova descendência e mutação aleatória destes novos descendentes. A inversão, que consiste no quarto elemento dos algoritmos genéticos e foi introduzido por Holland, raramente é utilizada, pois suas vantagens ainda não são bem conhecidas [13].

Para Goldberg, os algoritmos genéticos são diferentes dos procedimentos comuns de otimização porque trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros. Eles buscam a solução a partir de uma população de pontos, e não de um ponto simples. Outro ponto importante é que utilizam funções-objetivos e não derivadas ou outro conhecimento auxiliar. Por fim, usam regras de transição probabilísticas e não determinísticas.

Estes quatro elementos também diferenciam os algoritmos genéticos dos outros mecanismos evolucionários como é discutido adiante. Mesmo contando com componentes trabalhados aleatoriamente, os algoritmos genéticos não são simples passeios aleatórios. Eles exploram eficientemente as informações históricas para inferir novos pontos de busca. O ponto central de pesquisa em algoritmos genéticos tem sido sua robustez, ou seja, o balanço entre eficiência e eficácia [10].

Embora a versão básica dos algoritmos genéticos seja simples, existem importantes aplicações, tais como a otimização, a programação automática, a aprendizagem de máquina, a economia, a ecologia (modelagem de fenômenos como co-evolução de hospedeiro/parasita, simbiose e proliferação de armas biológicas), a genética populacional, a evolução e a aprendizagem, os sistemas sociais (tais como modelagem de comportamento social de colônias de insetos ou esquemas de cooperação em sistemas multi-agentes), a otimização de recursos (redes de distribuição de água, cargas e dimensões em estruturas), e diversos outros casos [12]. Também são largamente empregados em sistemas híbridos, no ajuste de topologias neurais ou na geração de bases de regras difusas [13,14].

As razões para o crescente número de aplicações, segundo Goldberg [10], são claras: os algoritmos são poderosos, computacionalmente simples e não são limitados por condições restritivas com relação ao espaço de busca. Essas vantagens, paradoxalmente, podem ser limitações, como apontam, pois tornam a busca excessivamente ampla em alguns casos.

Portanto, quando se trata de projeto com um elevado grau de complexidade e otimização, este algoritmo vem sendo largamente aplicado em conjunto com outros métodos de otimizações, apresentando excelentes resultados. Além de não necessitar de informações sobre o gradiente da superfície de resposta, as eventuais descontinuidades da mesma não afetam a performance da otimização. Outra característica do Algoritmo Genético é a de que a presença de mínimos locais não atrapalha em sua performance, podendo, assim, ser aplicados em problemas de otimização de grande escala.

O Algoritmo Genético [8] foi elaborado e codificado de forma semelhante aos cromossomos, tornando-o muito parecido com o processo evolutivo do ser vivo. Alguns conceitos naturais que podemos estabelecer uma correlação com o Algoritmo Genético são:

- cromossomo (genótipo) - cadeia de bits que representa uma solução possível para o problema.
- gene - representação de cada parâmetro de acordo com o alfabeto utilizado (binário, inteiro ou real).
- fenótipo - cromossomo codificado
- população - conjunto de pontos (indivíduos) no Espaço de Busca
- geração - iteração completa do AG que gera uma nova população

- aptidão - saída gerada pela função objetivo para um indivíduo da população

Uma visão ligeiramente diferente é a da “hipótese dos blocos de construção”. Neste caso os blocos são esquemas ajustados de pequeno tamanho de definição [10]. Os operadores genéticos de cruzamento e mutação têm a habilidade de gerar, promover e justapor, lado a lado, blocos de construção, para formar as cadeias de caracteres ótimas. O cruzamento tende a conservar a informação genética presente nas cadeias de caracteres. Assim, quando essas cadeias são similares, sua capacidade de gerar novos blocos diminui. Por outro lado, a mutação é capaz de gera blocos radicalmente novos, pois não é um operador conservativo. A seleção dos elementos que devem permanecer para a próxima geração é um procedimento que tende a ser enviado para os blocos que possuem maior aptidão, garantindo sua representação de geração a geração. Esta hipótese sugere que o problema de codificação é crucial para a performance dos algoritmos genéticos, e que tal codificação deve satisfazer a idéia de blocos pequenos para ampliar a possibilidade de intercâmbio [13].

Contudo Mitchell [12] afirma que existem controvérsias sobre algumas abordagens, revelando que os algoritmos genéticos não têm uma teoria acabada, mas, ao contrário, ainda apresentam questões sem respostas. Segundo esta autora, embora os algoritmos genéticos sejam largamente empregados e simples de descrever e programar, seu comportamento pode ser complexo, não estando ainda completamente compreendido seu funcionamento.

Para Illich e Simovic [15] os algoritmos genéticos não podem ser aplicados a qualquer problema e sua eficiência varia de muito eficiente a ineficiente em função da complexidade e tamanho do problema; os algoritmos genéticos buscam soluções em todo o espaço, mesmo em regiões de soluções inviáveis, e o esforço de busca pode ser 99% nesta área (perdido); os algoritmos genéticos não levam em conta a forma e o gradiente da função objetivo, que poderiam aumentar as chances de se encontrar um ótimo global e, por fim, os algoritmos genéticos requerem sintonia dos parâmetros de busca.

Outro ponto interessante é a especificação de um determinado número de indivíduos que são selecionados deterministicamente, escolhendo-se os melhores (mais ajustados) em cada geração antes da seleção aleatória. Esta forma é conhecida como “elitismo”.

3.2 CODIFICAÇÃO DAS VARIÁVEIS

Nos primeiros algoritmos genéticos os cromossomos eram tipicamente binários. Neste formato cada gene em um cromossomo tem dois alelos possíveis: 0 e 1. Cada cromossomo pode ser visto como um ponto (de dimensão n) no espaço de busca de soluções. Os algoritmos genéticos processam populações de cromossomos substituindo progressivamente uma população por outra potencialmente mais adaptada. Para isto requerem uma função de aptidão que assinale uma pontuação a cada cromossomo na população corrente, dependendo de quão bem o cromossomo resolve o problema em questão [12].

Um algoritmo genético simples é composto basicamente de três operadores: reprodução (através de seleção dos indivíduos mais ajustados), cruzamento e mutação. A reprodução é um processo no qual os indivíduos são copiados de acordo com os valores das suas funções-objetivo f (funções de aptidão). A função f pode ser entendida como uma medida de lucro ou utilidade a ser maximizada. Copiar de acordo com f significa que os indivíduos com maior valor de f têm maior probabilidade de contribuir com um ou mais descendentes na próxima geração. Este operador pode ser implementado de várias formas. Provavelmente a forma mais simples é a roleta, com as casas de cada indivíduo proporcionais à sua aptidão. Cada elemento selecionado é copiado para uma população provisória [10]. A seleção é realizada sobre uma população provisória P(t) para gerar a nova população, P(t+1). A probabilidade de seleção dos indivíduos é dada pela proporção de sua aptidão em relação ao somatório de aptidões da população (seleção proporcional), como apontado na “Fórmula 1” indicada por Goldberg [10]:

$$p(a_i) = \frac{f(a_i)}{\sum_{j=1}^{n} f(a_i)}$$

Fórmula 1

Proporção da aptidão

Esta definição assume valores de aptidão positivos e uma tarefa de maximização. Os algoritmos genéticos geralmente mantêm população de tamanho constante, isto é, da ordem de 50 a 100 indivíduos. Normalmente esta população é inicializada aleatoriamente [16], mas o analista pode fazer uso do conhecimento anterior

na formulação destes indivíduos, o que pode ser apontado como uma das vantagens dos algoritmos genéticos [17].

Após a reprodução o cruzamento é realizado em duas etapas. Em primeiro lugar são escolhidos aleatoriamente alguns pares de indivíduos. Em seguida, para cada par, é escolhido um número inteiro k representando a posição de corte no intervalo [1, n], sendo n o tamanho do cromossomo (quantidade de genes). Caso não seja a primeira população, os novos indivíduos são gerados combinando ou re-combinando as parcelas [1, k] e [k+1, n] dos indivíduos pai e mãe [10].

Cada gene de um cromossomo [8,9,18] é responsável por uma determinada característica de um indivíduo. De forma similar podemos compor um cromossomo não natural e, por um processo artificial, podemos representar cada parâmetro que será otimizado.

Codificando-se os cromossomos em números binários, podemos obter muitas vantagens como: tornar as operações genéticas mais fáceis de serem executadas, facilitar a manipulação de variáveis que possuem valores contínuos, facilitar a variação da faixa de trabalho dos parâmetros e apresentar uma simplicidade na variação da precisão dos valores de cada parâmetro.

A codificação é outro ponto importante na especificação de uma aplicação em algoritmos genéticos. Originalmente os algoritmos genéticos eram codificados em cromossomos binários. Porém, Man, Tang e Kwong [13] afirmaram recentemente que a manipulação direta de cromossomos com valores reais receberam considerável interesse, especialmente para trabalhar com problemas que têm parâmetros reais. Para estes autores alguns trabalhos indicam que a representação em ponto flutuante pode ser mais rápida em termos de computação (por exemplo, evitando as conversões no início e no final do processamento) e também mais consistente. Contudo nem sempre é superior à codificação binária segundo estes autores. Tudo depende do caso em que será aplicado, pois nos casos onde existam variáveis com alta precisão seriam necessários muitos bits para representá-las. Segundo Cordón [17], no caso de codificações em números reais, o cruzamento é realizado através do cruzamento aritmético com a combinação linear dos genes correspondentes de um par de indivíduos.

No Algoritmo Genético proposto cada indivíduo representa um subconjunto de atributos selecionados. Cada cromossomo é composto por M genes, onde M é o número de atributos de uma base de dados. Cada gene pode assumir valores 0 ou 1, indicando respectivamente a ausência ou a presença do atributo no subconjunto criado [19].

Tendo uma base de dados composta por 8 atributos, podemos gerar um cromossomo com 8 genes, cada um representando um atributo. Depois de efetuado o processamento do Algoritmo Genético, podemos obter um resultado que nos mostre a ausência dos atributos 2, 5 e 6, como mostrado na “Figura 1” abaixo:

1	0	1	1	0	0	1	1

Figura 1: Codificação final de um cromossomo

Estes três atributos poderão ser descartados na execução do processo de Agrupamento dos dados executado em conjunto do algoritmo AJC.

3.3 CRIAÇÃO DA POPULAÇÃO INICIAL

Um gerador aleatório cria a população inicial de cromossomos, ou seja, para cada atributo (também chamado de variável) de cada registro (cromossomo), é atribuído aleatoriamente um 0 ou um 1. No modelo genético da “Figura 2” mostramos um conjunto de cromossomos para uma base composta por 8 genes, onde estes receberam seus alelos aleatoriamente.

[1] 1 0 0 0 1 0 0 1
[2] 0 0 0 1 0 0 1 0
[3] 1 0 0 0 0 1 0 0
[4] 0 0 0 1 1 0 0 0
[5] 0 1 0 0 0 1 0 0
[6] 0 0 1 0 0 0 0 1
[7] 0 0 0 0 0 1 0 0

Figura 2: População inicial de cromossomos

Criando uma população inicial aleatória garantimos que esta não sofrerá influência tendenciosa do meio externo, garantindo a legitimidade da pesquisa.

Entretanto, podemos inserir informações adicionais que nos permitam diminuir o tempo computacional para encontrar a melhor solução no algoritmo, ou seja, influência

tendenciosa na população inicial, já que podemos ter um conhecimento prévio e útil para se chegar ao melhor resultado.

3.4 APTIDÃO - AVALIAÇÃO DA RESPOSTA

Esta é uma das tarefas essenciais do AG, pois é onde se encontra o valor associado às características de cada cromossomo. Sua avaliação é assaz significativa para o AG.

A aptidão [8,9] é uma característica de cada indivíduo e é verificada pelas suas habilidades, ou seja, se ele consegue sobreviver a predadores e outros obstáculos, e seguir sua reprodução dando origem a uma outra geração. Em um algoritmo genético, esta função se verifica, muitas das vezes, avaliando-se a diferença entre um valor esperado e um previsto, chamando essa diferença de erro. Portanto, se acharmos o menor erro para um indivíduo, este provavelmente sobreviverá, caso contrário, não passará para a nova geração. Esta função determinará, probabilisticamente, com quem um determinado cromossomo irá cruzar, podendo, um indivíduo, realizar vários cruzamentos em uma mesma geração.

3.5 CRUZAMENTO

Nesta parte há o cruzamento [8,9] entre o material genético entre diferentes indivíduos. Uma nova população é criada a partir de um cruzamento aleatório entre cromossomos, ou seja, uma população de herdeiros, onde estes recebem informações de seus progenitores, através do material genético passado por estes.

$X_1\ X_2 \mid X_3\ X_4\ X_5\ X_6$

$Y_1\ Y_2 \mid Y_3\ Y_4\ Y_5\ Y_6$

Figura 3: Indivíduos antes do cruzamento

$X_1\ X_2 \mid Y_3\ Y_4\ Y_5\ Y_6$

$Y_1\ Y_2 \mid X_3\ X_4\ X_5\ X_6$

Figura 4: Indivíduos depois do cruzamento

Durante o cruzamento, há uma troca de material genético e haverá uma tendência da transmissão de características dominantes para as gerações seguintes, que podem ser mostradas por variáveis dominantes. Esse cruzamento é responsável direto pela convergência para a seleção desejada de atributos.

Bäck e Kursawe [16] lembram que o mecanismo de cruzamento simples pode facilmente ser estendido para cruzamento múltiplo (vários pontos de corte), e até para cruzamento uniforme, no qual a decisão aleatória sobre a troca de informação entre genes é realizada bit a bit.

3.6 MUTAÇÃO

A mutação consiste na alteração aleatória de uma parte do indivíduo. Este operador realiza uma função importante, trazendo inovação e diminuindo os riscos de convergência prematura [10]. A mutação é geralmente interpretada como sendo de importância marginal em algoritmos genéticos. Funciona invertendo aleatoriamente genes isolados, escolhendo o gene a ser modificado (gene m) com uma probabilidade pequena. Geralmente pm assume valores na faixa de 0,01 a 0,001. Investigações mais recentes, todavia, indicam que a mutação pode ser importante em alguns casos e recomendam que pm fique em torno de pm=1/n, sendo n o número de bits do cromossomo [16].

3.7 DEFINIÇÃO DOS PARÂMETROS DOS OPERADORES GENÉTICOS

Uma questão importante é a determinação dos parâmetros das operações. A probabilidade de cruzamento (pc) é geralmente da ordem de 0,6 [16]. Para Man, Tang e Kwong [13] é comum adotar-se uma taxa de cruzamento (pc) com valores entre 0,6 e 1,0, enquanto que a probabilidade de mutação é tipicamente menor do que pm=0,01. Dependendo do domínio, a escolha de pm e pc como parâmetros de controle pode ser um problema complexo de otimização não linear. Ademais, dependem criticamente da natureza da função objetivo. De Jong [20] desenvolveu grande estudo exploratório sobre os parâmetros, concluindo pelo uso de populações de 50 a 100 indivíduos, com taxas de cruzamento de 0,6 e de mutação de 0,001. Grefenstette [21] revisou o trabalho desse pesquisador, indicando populações de 30 indivíduos, cruzamento de 0,95, de mutação de 0,01 e escolha elitista de um indivíduo por geração como valores ótimos. Contudo, há resultados contraditórios. Para Sharif e Wardlaw [22], a aplicação de mais de uma

mutação por indivíduo degradou a performance. Este assunto ainda não está resolvido, embora existam algumas sugestões na literatura [13]:

a) populações grandes (100 ou mais indivíduos): pc=0,6 e pm=0,001;
b) populações pequenas (até 30 indivíduos): pc=0,9 e pm=0,01.

Por outro lado, alguns pesquisadores, como Spears [23,24] apontam para um relativo equilíbrio entre os operadores de cruzamento e mutação. Para esse autor, não há provas da superioridade de um ou de outro, defendendo a análise do desempenho de ambos, nos casos concretos. De qualquer forma, aparentemente os parâmetros são dependentes do problema a ser resolvido, ou seja, é necessário explorar os efeitos da variação das taxas, do tamanho da população e do tempo de treinamento (número de gerações).

Neste processo [9] há a troca de informações dos genes dentro de cada cromossomo, gerando uma mutação. Estas alterações promovem a modificação no código genético de uma pequena parte da população, que pode atuar de forma benigna. Se forem gerados indivíduos mais aptos, suas características serão transmitidas para as próximas gerações.

Este recurso coopera na solução do problema dos mínimos locais, pois fazem com que as pesquisas apontem para outros locais em uma superfície.

A “Figura 5” apresenta um gráfico com as fases de um Algoritmo Genético:

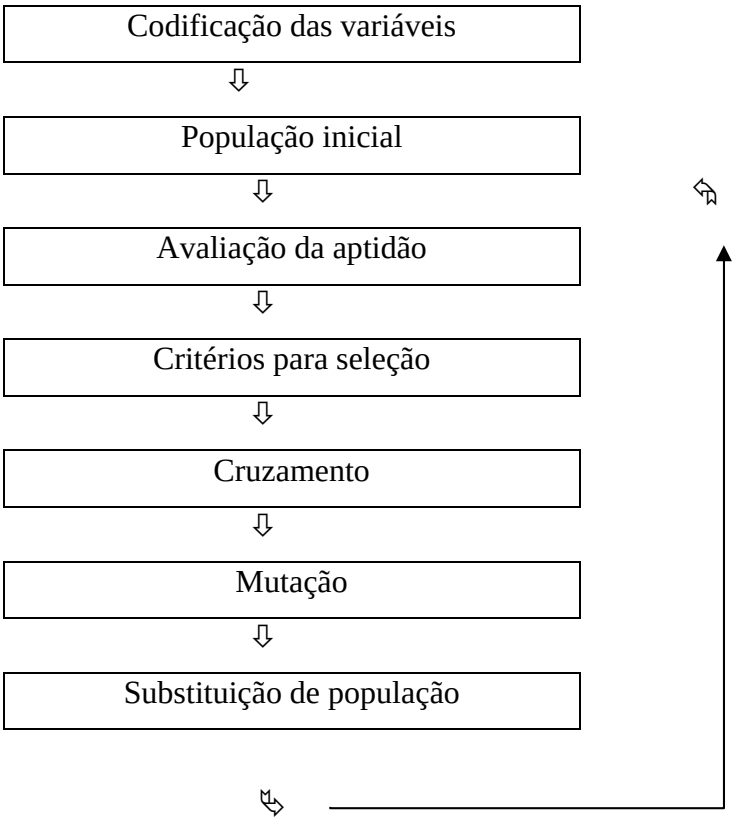

Figura 5: Principais fases do Algoritmo Genético

4 AGRUPAMENTO DE DADOS

4.1 INTRODUÇÃO

Uma das principais tarefas da análise de agrupamento é organizar os dados em um pequeno número de grupos (*clusters*), de forma que os elementos similares estejam alocados no mesmo grupo e os padrões muito distintos estejam em grupos diferentes [25].

Apesar de existirem diversos algoritmos para realizar esta tarefa, há dois aspectos fundamentais da natureza dos métodos de agrupamento [26]:

a) não existe efeito explícito de supervisão, isto é, os padrões são organizados de acordo com um critério de agrupamento considerando que os grupos não são conhecidos no início;

b) a noção de similaridade entre dois padrões é essencialmente realizada por suas distâncias, onde a função de distância é a quantificação da similaridade, ou seja, quanto menor a distância, maior a similaridade; se a distância é zero, os padrões são iguais.

O processo de descoberta de conhecimento demanda vários passos obrigatórios como seleção e preparação dos dados, busca do conhecimento dos padrões propriamente ditos, e por fim, a análise dos resultados encontrados. Esses resultados devem ser aplicados em novas informações e serem validados com um aceitável grau de certeza, pois essas informações poderão servir como base útil e confiável para uma tomada de decisões.

Na análise de agrupamento não existe pré-classificação dos dados nem distinção entre variáveis dependentes ou independentes. Trata-se de um tipo de aprendizagem não supervisionada, no qual o método lida diretamente com os dados, visando determinar a estrutura dos padrões [25].

Em alguns casos de *data mining*, uma das dificuldades a ser enfrentada é o excesso de informações. Torna-se necessário utilizar técnicas que permitam particionar ou reduzir os dados, facilitando o entendimento dos padrões. Freqüentemente não há informação inicial para apoiar as tarefas de segmentação ou classificação dos dados.

Para Kaski [26] um dos maiores problemas com esta técnica é a interpretação dos grupos obtidos, que pode ser difícil. Assim, as grandes metas são compactar o arquivo de dados e realizar inferência sobre a estrutura dos grupos resultantes, o que nem sempre pode gerar bons resultados.

Por este motivo, a análise de agrupamento raramente é utilizada isoladamente. Detectados os grupos, outros métodos são aplicados para descobrir o que significam. Esta técnica tem aplicação em uma larga variedade de campos, incluindo descoberta de conhecimento, análise estatística, compressão de dados, quantização vetorial, reconhecimento e classificação de padrões. Uma das aplicações comuns é a construção automatizada de categorias ou taxonomias [27].

4.2 CLASSES DE ALGORITMOS DE AGRUPAMENTO

Existem duas classes básicas de algoritmos: agrupamento hierárquico ou particional.

O agrupamento hierárquico busca reunir sucessivamente grupos menores, formando grupos maiores, ou dividir grupos grandes em outros de maior similaridade interna. Os métodos diferem pela regra adotada para decidir quais grupos devem ser reunidos ou divididos. O resultado do algoritmo é um gráfico tipo árvore chamado de "dendograma" que mostra como os grupos são inter-relacionados [25,26].

Já o agrupamento particional busca dividir o conjunto de dados em um conjunto de grupos distintos entre si, maximizando as dissimilaridades dos diferentes grupos [28]. As técnicas que seguem o agrupamento particional, incluindo *K-means* e diversas outras, geralmente são baseadas na otimização de uma função de custo, que envolve, por exemplo, a minimização do erro quadrático, e são de natureza combinatorial.

Em função do tamanho dos arquivos, o tempo de processamento é muito grande. Por isto, técnicas de otimização tais como *simulated annealing* e algoritmos genéticos são empregadas para acelerar o processamento [25,26].

4.3 ALGORITMO K-means

Em função de ser o método de implementação mais comum em softwares dentre todos os métodos, K-means é uma referência, contra o qual os outros algoritmos são comparados.

Muitos algoritmos para agrupamentos têm sido propostos, especialmente para agrupamento espacial, alguns dos quais serão apresentados a seguir. Por ser uma das técnicas mais importantes na busca de informações implícitas existentes nos dados, há muito esforço de aperfeiçoamento [29].

K-means é o algoritmo de agrupamento particional proposto por MacQueen em 1967, mais conhecido e aplicado para agrupar dados, embora existam muitas variantes. Este algoritmo requer que os dados sejam compostos de variáveis numéricas, pois uma parte do processo é baseada no cálculo das médias [28].

Em termos mais precisos, agrupar pelo método K-means pode ser descrito como: "dado um conjunto de *n* pontos no espaço real d-dimensional R^d e um número inteiro *k*, definir os *k* conjuntos de pontos em R^d que minimizem a distância média quadrada de cada ponto ao centróide do conjunto mais próximo" [27]. O processo de cálculo consiste basicamente das seguintes etapas:

a) selecionar *k* pontos;
b) determinar as coordenadas destes pontos como sendo os centróides dos grupos;
c) calcular a distância do próximo ponto aos *k* centróides (geralmente empregando a distância euclidiana);
d) incorporar o ponto ao grupo mais próximo;
e) recalcular o centróide deste grupo;
f) passar ao próximo ponto – se terminarem os pontos, recomeçar do primeiro ponto, revendo seu posicionamento;
g) encerrar o processo se não houver possibilidade dos dados mudarem de grupo ou retornar à etapa (c).

Embora simples e razoavelmente eficiente, o algoritmo K-means tem algumas desvantagens. Um dos problemas apontados é o tempo de processamento em função da aplicação em espaços de muitas dimensões (muitas variáveis) e da necessidade de muitas iterações até a condição de final [27]. Outro problema é a escolha das condições iniciais.

O número de grupos *k* e a inicialização dos centróides (escolha dos primeiros *k* pontos) pode influir decisivamente nos resultados. A própria limitação a um determinado número de grupos pode não ser adequada quando a análise é exploratória. Por fim, o algoritmo freqüentemente encerra o processo com ótimos locais, desprezando

valores mais globais. Não obstante, ainda é o algoritmo mais utilizado na prática, em função de ser a variante implementada nos pacotes estatísticos comerciais [26].

4.4 ÍNDICES DE AGRUPAMENTO

Foi implementado no sistema AJC dois tipos de índices para avaliaçào do agrupamento: o índice PBM e o indice Calinski e Harabasz. Uma vez aplicado um desses índices para uma determinada base de dados, será exibido um resultado que expressa o número ótimo de grupos do método para esta base.

4.4.1 CALINSKI E HARABASZ

Este índice foi desenvolvido por Calinski e Harabasz [30]. É um método estatístico para encontrar a melhor distribuição de pontos para um conjunto de grupos. Este procedimento primeiramente recebe a quantidade de grupos (k). Posteriormente divide a base de dados em k grupos e realiza, aleatoriamente, uma distribuição eqüitativa dos pontos em cada grupo formado. Após esta divisão em k grupos, o método verifica a que grupo cada ponto pertence através da medida de distância entre o ponto e o centro de cada grupo. O grupo que apresentar a menor distância recebe este ponto. Sempre que acontece para um ponto uma troca de grupos, o método calcula novamente seu centro. Este procedimento acontece até que acabe a troca de pontos entre os grupos. Sua função de distribuição pode ser vista como:

$$G(k) = (n-k) * B \Big/ (k-1) * W$$

Fórmula 2
Função de distribuição

Onde n é o número de pontos existentes e k é o número de grupos em que foi dividida a base de dados.

Para se chegar aos valores de B e W, temos as seguintes fórmulas:

$$W = \sum_{i=1}^{K} \sum_{j=1}^{n_i} (X_{ij} - \overline{X}_i)^2$$

Fórmula 3

Somatório dos quadrados das distâncias

$$T = \sum_{i=1}^{K} \sum_{j=1}^{n_i} (X_{ij} - \overline{X})^2$$

Fórmula 4

Somatório dos quadrados das diferenças

$$B = T - W = \sum_{i=1}^{k} n_i (\overline{X_i} - \overline{X})^2$$

Fórmula 5

Diferença entre T e W

O valor de W é o somatório dos quadrados das distâncias dos pontos para o centro do grupo a que pertence, onde X_{ij} é o j-ésimo ponto do grupo i, Xbarrai é o centro do grupo, que é a média dos pontos ao centro do grupo, e n_i é a quantidade de pontos que estão no grupo i.

T é o somatório dos quadrados das diferenças de cada ponto de toda a base de dados e o centro de toda a base, representado por Xbarra.

Para completar a fórmula geral de distribuição, o valor de B pode ser obtido pela diferença entre T e W, que é o somatório dos produtos entre o número de pontos de toda a base e os quadrados das diferenças entre o centro de toda a base e o centro de cada grupo.

Portanto, G(k) é o resultado da função. O resultado efetivo e ideal do índice Calinski e Harabasz será aquele que maximizar o resultado desta função, ou seja, o maior valor para G entre todas as quantidades mínima e máxima informadas pelo usuário.

4.4.2 PBM

O Índice PBM, desenvolvido por PAKHIRA, BANDYOPADHYAY e MAULIK, pode ser utilizado para avaliar a qualidade de segmentação de bases de dados em diferentes partições, podendo ser aplicado tanto em segmentações clássicas quanto difusas. Este índice é obtido pela composição de três fatores, conforme apresentado na equação, e a sua maximização dá-se em torno de formações com pequeno número de grupos compactos e com uma grande separação entre dois deles, ao menos [31].

Este índice apresenta a seguinte fórmula:

$$PBM(K) = \left(\frac{1}{K} * \frac{E_1}{E_K} * D_K \right)^2$$

Fórmula 6

Função geral de distribuição

A fórmula nos mostra 1/k, que descreve o índice PBM como inversamente proporcional ao número de grupos encontrados. Em outras palavras, este índice tende a apontar como resultado final para uma solução com a menor quantidade de grupos.

Em seguida a fórmula E_1 / E_k que diminui com o aumento de k. O resultado do índice PBM aumenta ou diminui na medida desta proporção.

$$E_K = \sum_{k=1}^{K} E_k$$

Fórmula 7

Densidade dos grupos

E_k mede a densidade dos grupos formados e é sempre desejável que o seu valor seja o mais alto possível. É calculado pelo somatório das distâncias de todos os pontos ao centro da base como se fosse um único grupo.

Para finalizar a fórmula geral do índice PBM, tem-se D_k, que representa a medida da maior distância encontrada entre os grupos formados, tomados dois a dois. O valor de D_k pode ser aumentado de acordo com o valor de k e puxado para cima em função da separação máxima entre dois pontos da base.

O que se pretende com este índice, além de sua maximização, é manter o menor número de grupos possível enquanto a densidade interna e a separação externa dos grupos aumentam tanto quanto possível [31].

5 APLICATIVO AJC

O programa AJC para agrupamento foi construído utilizando-se do ambiente JAVA, pois com o byte code (arquivo.class) gerado em sua compilação, podemos executar o programa a partir de qualquer plataforma operacional, com possibilidade de posterior distribuição pública utilizando-se os recursos da rede mundial, internet.

AJC possui efetivamente cinco módulos bem definidos para o agrupamento inteligente de dados. Para que se tenha acesso aos módulos, existe a interface gráfica para a interação com o usuário, onde podemos efetivamente passar para o código principal todas as informações obrigatórias e/ou parametrizáveis, essenciais na execução do algoritmo. Um módulo chamado *Arquivo* recebe o nome da base de dados e a sua quantidade atributos (quantidade de variáveis); outro módulo denominado *Seleção* recebe a taxa de cruzamento, taxa de mutação, número de gerações, número de indivíduos e a convergência, que é a quantidade desejada de variáveis; um terceiro módulo chamado *Agrupamento* recebe os parâmetros para o agrupamento de dados propriamente dito, quantidade mínima de grupos, quantidade máxima e a escolha entre os índices PBM ou Calinski e Harabasz; outro módulo implementado chamado de *Índices* efetua o cálculo do índice de grupos desejado, PBM ou Calinski e Harabasz; por fim, um módulo *Ajuda*, que serve de apoio operacional na utilização do AJC.

O algoritmo implementado, como qualquer outro aplicativo desenvolvido neste segmento, possui argumentos externos necessários na execução do AG. Esses parâmetros são fornecidos pelo usuário quando se inicia o processo. Cada um desses argumentos atua na formação e identificação do melhor cromossomo dentre as gerações e suas populações. Esses parâmetros são: nome do arquivo que se deseja processar, quantidade de atributos (campos do arquivo), convergência, mutação, geração e população.

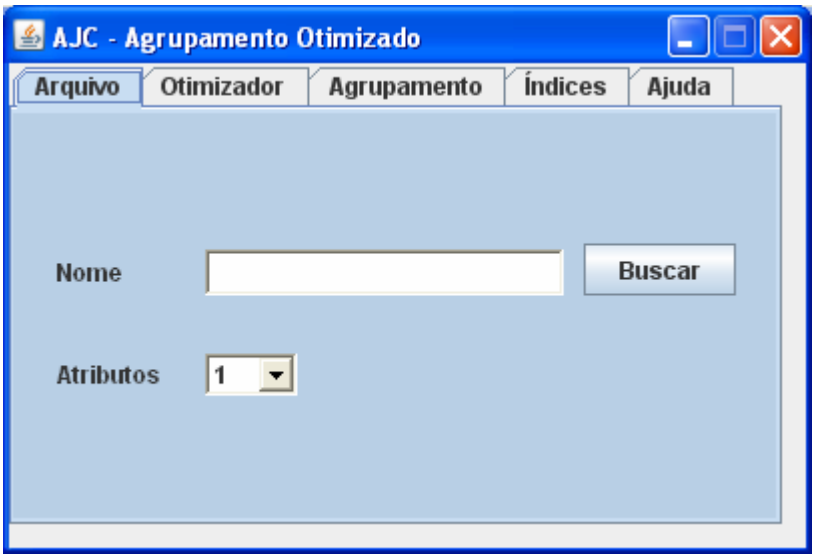

Figura 6: Tela AJC - Arquivo

Nome do arquivo é o repositório ou depósito de dados que servirá de entrada para a busca do melhor indivíduo.

Quantidade de atributos representa cada gene de um cromossomo, ou seja, cada campo constitutivo do registro. Essa quantidade deve ser inteira e precisa ser exatamente o somatório dos genes do cromossomo, caso contrário o AJC não conseguirá interpretar cada atributo de forma correta, podendo interromper sua execução e exeibir uma mensagem de erro.

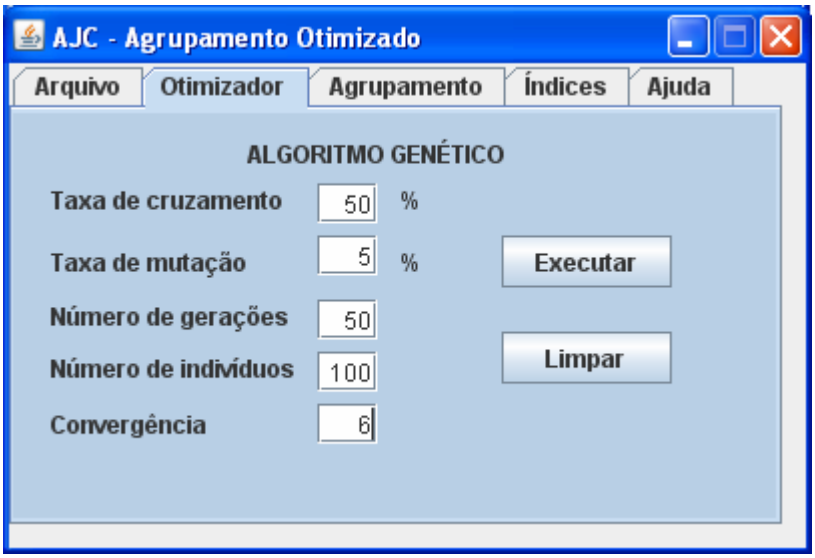

Figura 7: Tela AJC - Otimizador

Convergência representa a quantidade efetiva dos genes que queremos levar em consideração. Melhor explicando, se possuímos um arquivo com muitos atributos nem todos podem ser úteis e, por esse motivo, queremos realizar uma otimização dos genes que serão utilizados sem, contudo, afetar o resultado final.

Mutação é um número inteiro que representa o percentual de uma população que sofrerá uma alteração em suas características, ou seja, terão seus genes alterados. Isto cria uma situação onde são incluídos elementos potencialmente novos e que poderão ajudar na busca do melhor indivíduo.

Geração é um argumento inteiro que indica quantas existências poderemos ter para a execução do algoritmo, criando como se fosse uma árvore genealógica dos indivíduos, onde os mais fortes ou aptos sobrevivem.

População é um conjunto dos cromossomos existentes em uma determinada geração, que possui seu próprio conjunto de indivíduos.

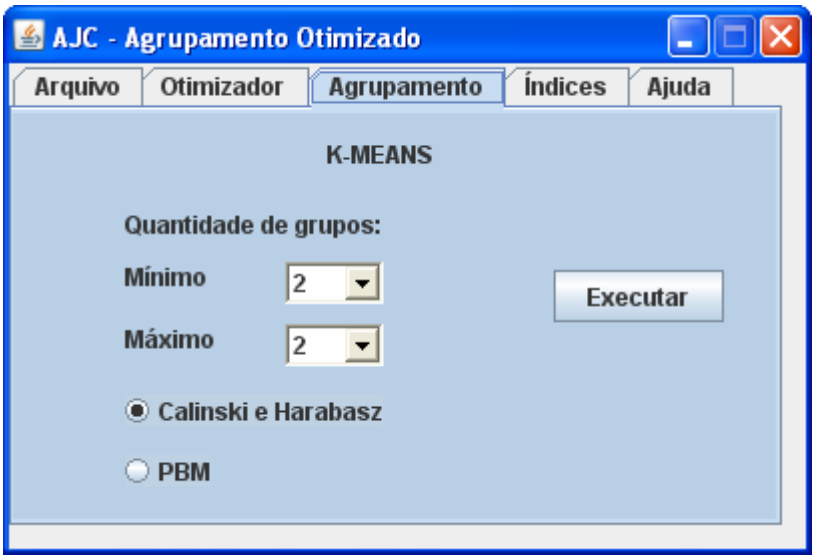

Figura 8: Tela AJC - Agrupamento

No módulo Agrupamento encontramos a quantidade de grupos mínima e máxima que se deseja para o processamento, bem como se os resultados serão calculados com base na técnica Caliski e Harabasz ou PBM.

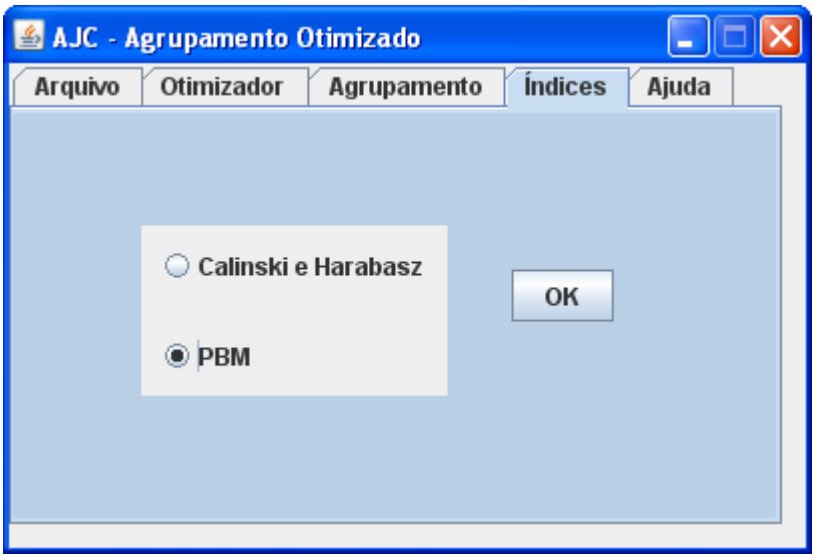

Figura 9: Tela AJC - Índices

Visando um melhor aproveitamento computacional, com relação ao tempo despendido no processamento da busca da melhor quantidade de grupos de um arquivo, bem como o esforço computacional, o AG é utilizado para encontrar um conjunto de variáveis de entrada que minimizam o erro calculado entre os valores dos atributos de um arquivo proposto para estudo.

No modelo implementado na etapa de pré-processamento realizada pelo AG, é feita a seleção das variáveis de acordo com os resultados obtidos pela função objetivo, levando-se em consideração a convergência informada como argumento externo e cada gene de um cromossomo.

Em uma primeira etapa são definidas as áreas para que se possam armazenar todas as informações lidas do arquivo. Portanto, em um vetor com o nome *dados* estão contidos os valores de cada cromossomo lido. Após essa leitura e carga do arquivo saberemos a quantidade de linhas deste, que se torna um importante valor para o algoritmo.

Depois de carregado o vetor *dados*, calcula-se a média de cada variável, que é armazenada em um vetor chamado *medBase* e, em um passo seguinte, calculamos o desvio padrão do arquivo e atribuímos seu resultado em desvPad. Com base nesses dois resultados armazenados nos últimos vetores, *medBase* e *desvPad*, encontramos a variância de cada componente dos atributos e, utilizando método quadrático, calculamos as covariâncias destes componentes, gerando, assim, sua matriz quadrada de elementos para as distâncias entre os valores de um atributo para os outros.

Os códigos referentes aos algoritmos de processamento do AJC e sua interface gráfica estão divididos em quatro principais arquivos. *CalcGer.java*, onde podemos encontrar alguns métodos como o cálculo geral entre as diferenças das distâncias entre cada elemento do grupo, a montagem do grupo inicial, a realocação dos objetos entre os grupos, a exibição dos resultados e a gravação do arquivo de saída, onde encontramos a classificação final de cada elemento em seus respectivos grupos; *LerArquivo.java*, onde encontramos a leitura efetiva de todo o arquivo e a contagem de linhas em tempo de execução; *Matriz.java*, que calcula a média, desvio padrão, variância e posterior matriz de covariância a partir dos dados do arquivo de entrada e, por fim, *AJC.java*, onde encontramos o código efetivo de toda a parte gráfica, a leitura dos argumentos de entrada e posterior execução do algoritmo desejado, bem como a crítica e validação dos dados externos informados.

6 EXPERIMENTOS E RESULTADOS

Para avaliarmos os resultados do agrupamento de dados do AJC utilizamos o algoritmo SearchCluster [31], tanto para os dados otimizados, ou seja, com os atributos selecionados, ou sem otimização.

Para avaliarmos a seleção de atributos do AJC utilizamos o programa Weka, que possui algoritmos de seleção de atributos.

Todos os dados apresentados no plano cartesiano estão considerando as duas melhores componentes principais.

6.1 AGRUPAMENTO DE DADOS SEM SELEÇÃO DE ATRIBUTOS

6.1.1 BASE DE DADOS ÍRIS

A base de dados Íris é bem conhecida no meio acadêmico. Ela registra informações de 3 espécies diferentes de flores e possui 150 registros e 4 atributos: largura da pétala, largura da sépala, altura da pétala e altura da sépala. Esta base forma 3 grupos distintos, cada um com 50 registros.

Tabela 1

Agrupamento sem otimização de atributos – base Íris

Programa	Classe 1	Classe 2	Classe 3	Registros
AJC	**50**	**49**	**51**	**150**
AJC - PBM	**55**	**95**		**150**
SearchCluster	**50**	**50**	**50**	**150**

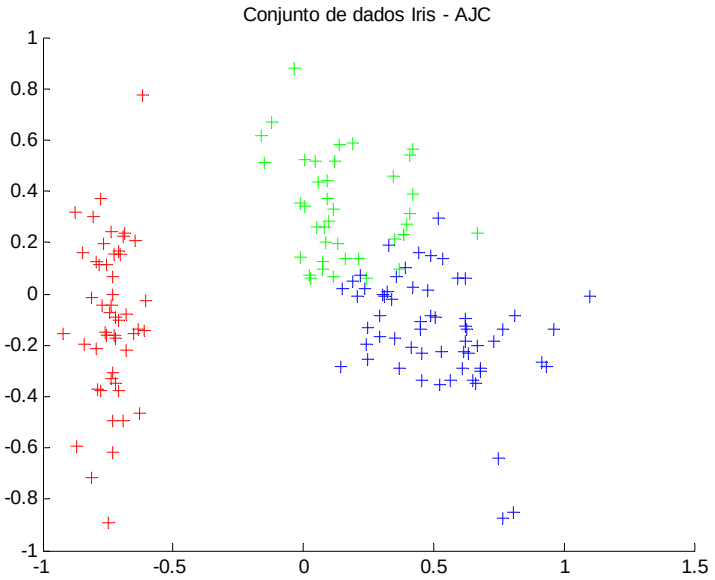

Figura 10: Distribuição dos pontos – base Íris AJC - Calinski e Harabasz

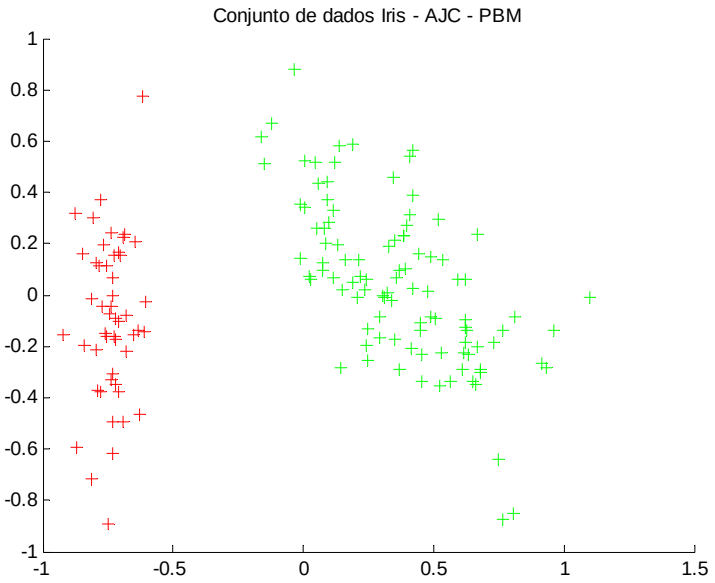

Figura 11: Distribuição dos pontos – base Íris AJC - PBM

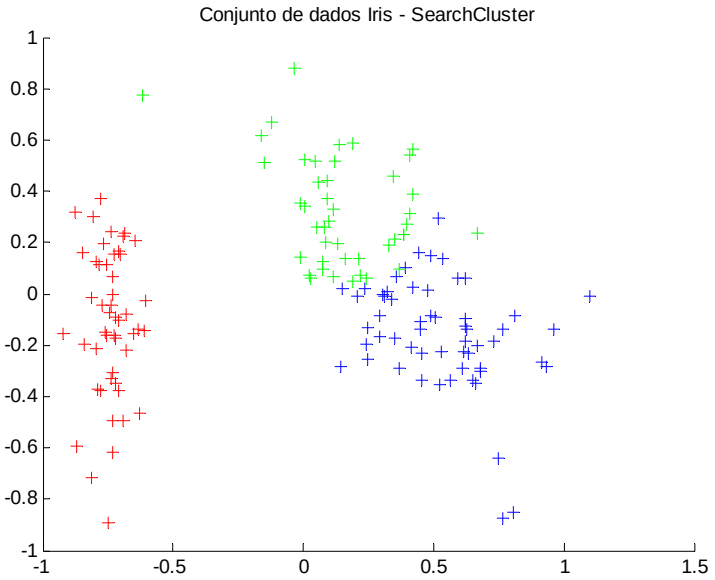

Figura 12: Distribuição dos pontos – base Íris SearchCluster

Analisando o gráfico, podemos notar o único elemento divergente entre os grupos calculados pelos dos programas AJC e o SearchCluster. Entre o grupo vermelho e o verde podemos notar a presença de um elemento que não foi agrupado da mesma maneira pelos dois programas. Para o AJC esse elemento pertence a um grupo diferente do grupo indicado pelo SearchCluster. Comparando-se os resultados da execução para estes dois programas, a primeira classe foi agrupada exatamente igual. Entre as duas outras classes tivemos a diferença no agrupamento de um registro, ou seja, uma diferença entre os programas de apenas 0.67% e um resultado total acima de 99% de similaridade.

Na execução do AJC, índice PBM, não houve uma boa separação de classes. Seu processamento nos mostrou apenas 2 grupos. Visualmente podemos ver na “Figura 11” as duas classes calculadas. Portanto, o AJC, utilizando o índice PBM, pode não ser considerado bom para bases de dados com o mesmo comportamento da base Íris.

6.1.2 BASE DE DADOS GLASS

Os dados desta base foram disponibilizados ao público em geral pelo Estabelecimento Geral de Pesquisa da Alemanha. Ela foi gerada através de um estudo de classificação de tipos de vidros para atender ao setor de investigação criminal, pois

para eles os estilhaços de vidros deixados em uma cena de crime podem servir de evidências, se puderem ser corretamente identificados.

A base de dados glass possui 214 registros e 9 atributos relevantes. O usuário proprietário das informações indicou 2 grandes grupos, vidros de janelas e vidros que não são de janelas, sendo que, em laboratório, tem-se 3 subgrupos para os vidros de janelas e 3 de vidros que não são de janelas, perfazendo-se um total de 6 subgrupos.

Tabela 2

Agrupamento sem otimização de atributos – base Glass

Programa	Classe 1	Classe 2	Registros
AJC	**152**	**62**	**214**
AJC - PBM	**152**	**62**	**214**
SearchCluster	**152**	**62**	**214**

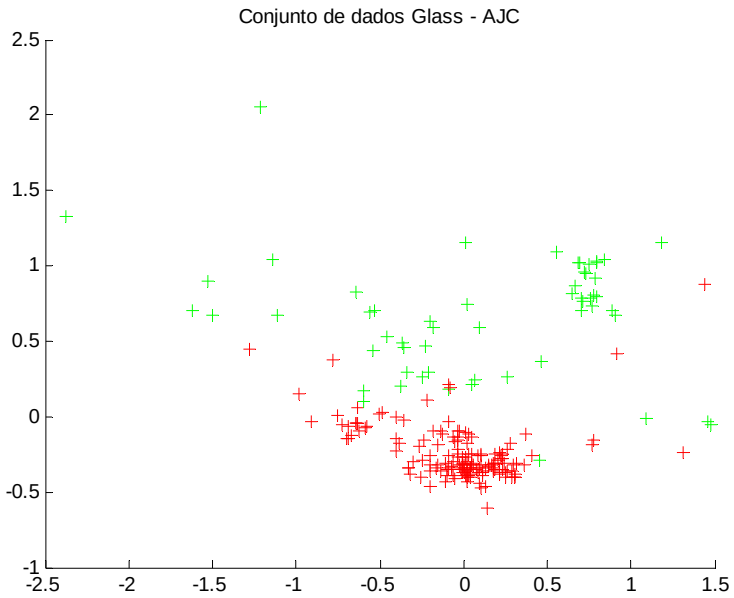

Figura 13: Distribuição dos pontos – base Glass AJC - Calinski e Harabasz

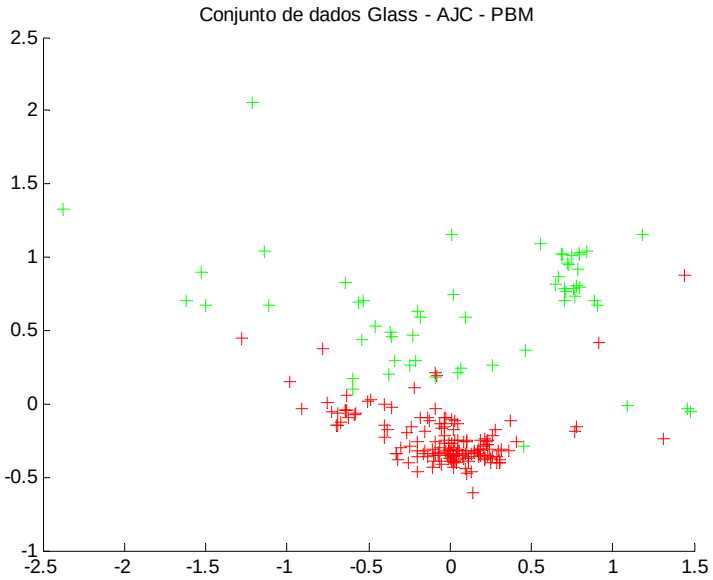

Figura 14: Distribuição dos pontos – base Glass AJC - PBM

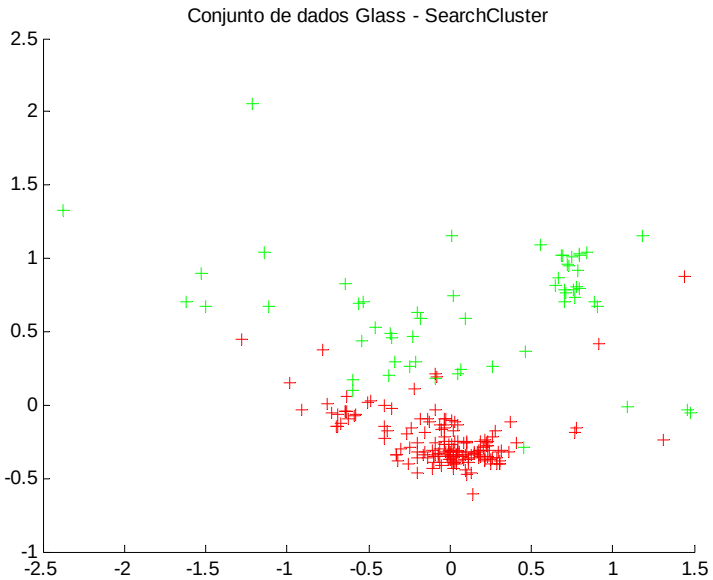

Figura 15: Distribuição dos pontos – base Glass SearchCluster

Podemos concluir, com o trabalho realizado com AJC, que a base de dados *glass* é de difícil agrupamento, visto que o AJC, índice Calinski e Harabasz, AJC, índice PBM e o SearchCluster, obtiveram os mesmos resultados finais. Os agrupamentos criados pelos programas nos aproximaram em mais de 93% dos dois grandes grupos de

estilhaços de vidros de janela e estilhaços de vidros que não são de janelas. Isso nos permite afirmar que o AJC nos oferece resultados excelentes para a base glass, que é considerada pelos especialistas como uma base com um elevado grau de complexidade.

6.1.3 BASE DE DADOS HEART

A base de dados *heart-cleveland* contém informações de diagnósticos de doenças do coração, coletados na Fundação Clínica de Cleveland, Estados Unidos. Inicialmente esta base continha 76 atributos, mas foram disponibilizados apenas os 13 mais relevantes.

Os responsáveis pelos dados apontam 5 grupos distintos, onde um grupo indica diagnóstico negativo e, as outras 4 indicam faixas diferentes de diagnósticos da doença. Para eles, dos 303 objetos, 164 são negativos.

Na etapa de pré-processamento 6 objetos com diagnósticos positivos apresentaram dados faltantes em 1 atributo. Todos foram completados com médias dos respectivos atributos.

Tabela 3

Agrupamento sem otimização de atributos – base Heart

Programa	Classe 1	Classe 2	Registros
AJC	**155**	**148**	**303**
AJC - PBM	**155**	**148**	**303**
SearchCluster	**155**	**148**	**303**

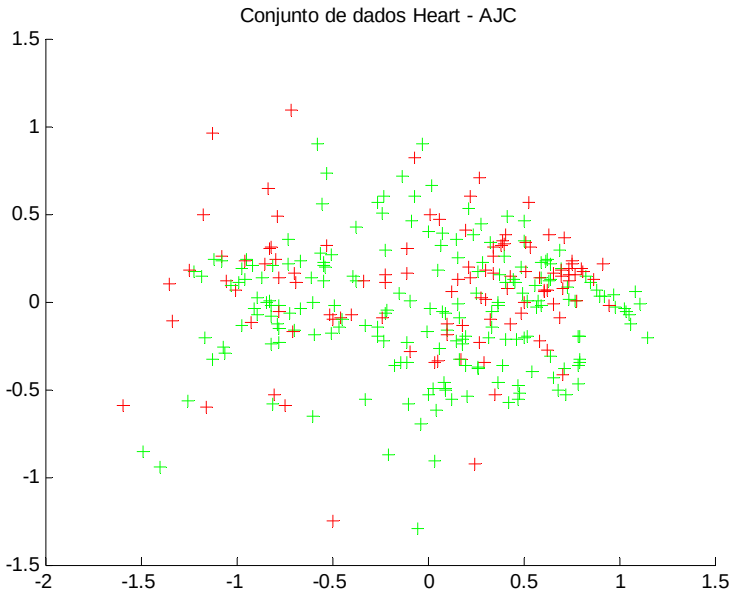

Figura 16: Distribuição dos pontos – base Heart AJC - Calinski e Harabasz

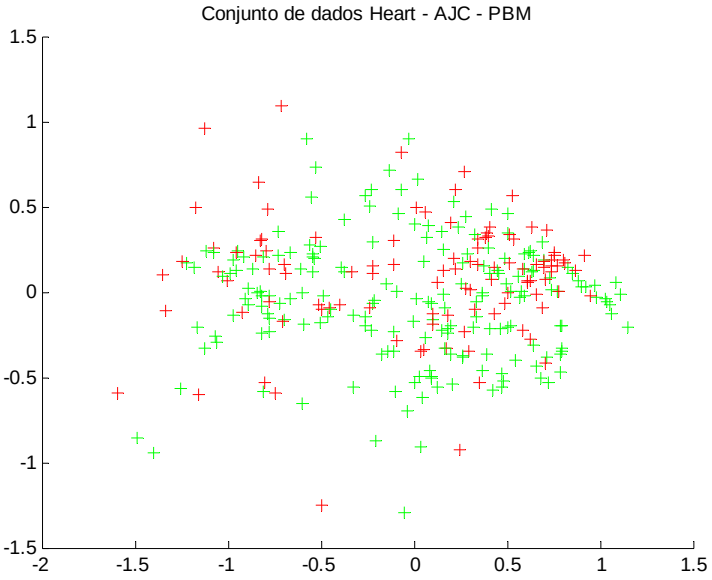

Figura 17: Distribuição dos pontos – base Heart AJC - PBM

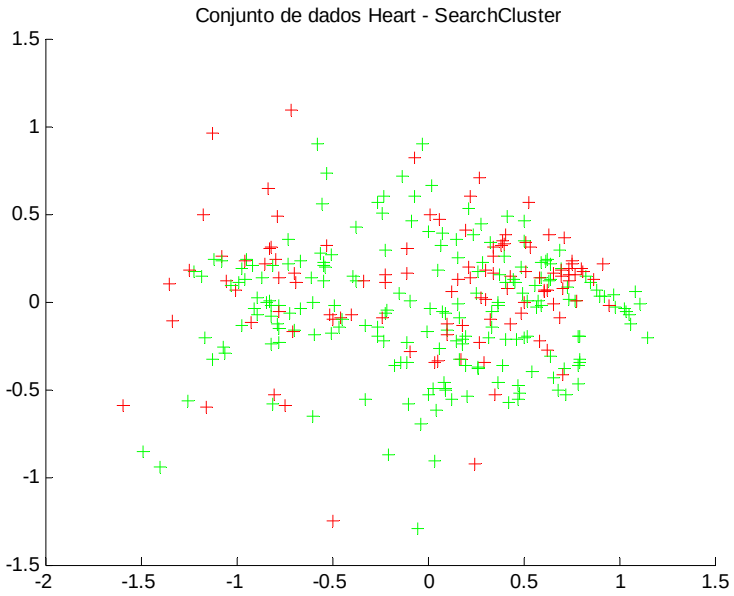

Figura 18: Distribuição dos pontos – base Heart SeachCluster

Ao executarmos o agrupamento AJC simples, pelo índice Calinski e Harabasz, sem a seleção de atributos por algoritmo genético, encontramos um resultado de 2 grupos, um com 155 diagnósticos negativos, e 148 positivos.

O mesmo aconteceu com a execução do AJC, índice PBM, sem a seleção de atributos e o SearchCluster, que apresentou os mesmos 2 grupos, com a mesma quantidade de positivos e negativos.

Os resultados apresentados pelas execuções dos programas para a base heart-cleveland demonstram bastante consistência. Portanto, com as informações disponibilizadas pelo proprietário dos dados e os resultados das execuções, podemos afirmar que foram bem selecionados aproximadamente 95% de acerto para os diagnósticos negativos. Apenas 9 diagnósticos de falsos positivos foram encontrados.

6.1.4 BASE DE DADOS CANA DE AÇÚCAR

A base de dados da cana de açúcar é o resultado de um estudo de genética realizado em conjunto por diversos laboratórios e grupos de minerações de dados na

produção de bibliotecas de seqüências de DNA [31]. Esta base possui 193 registros e 19 atributos. Os especialistas apontam 20 diferentes grupos em sua composição.

Tabela 4

Agrupamento sem otimização de atributos – base Cana de açúcar

	AJC	AJC – PBM	SearchCluster
Classe 1	**11**	**82**	**11**
Classe 2	**5**	**111**	**5**
Classe 3	**12**		**12**
Classe 4	**6**		**6**
Classe 5	**17**		**17**
Classe 6	**8**		**8**
Classe 7	**16**		**16**
Classe 8	**11**		**11**
Classe 9	**5**		**5**
Classe 10	**14**		**14**
Classe 11	**9**		**9**
Classe 12	**4**		**4**
Classe 13	**9**		**9**
Classe 14	**5**		**5**
Classe 15	**7**		**7**
Classe 16	**17**		**17**
Classe 17	**4**		**4**
Classe 18	**33**		**33**
Registros	**193**	**193**	**193**

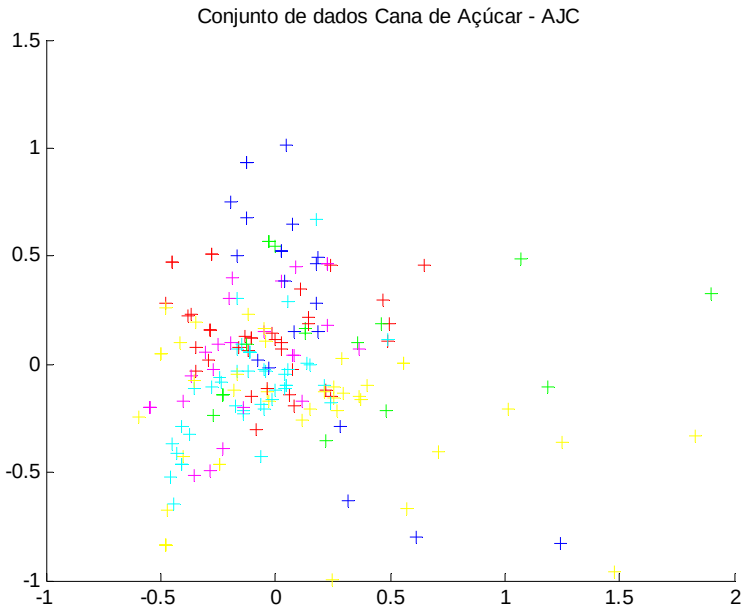

Figura 19: Distribuição dos pontos – base Cana de açúcar AJC - Calinski e Harabasz

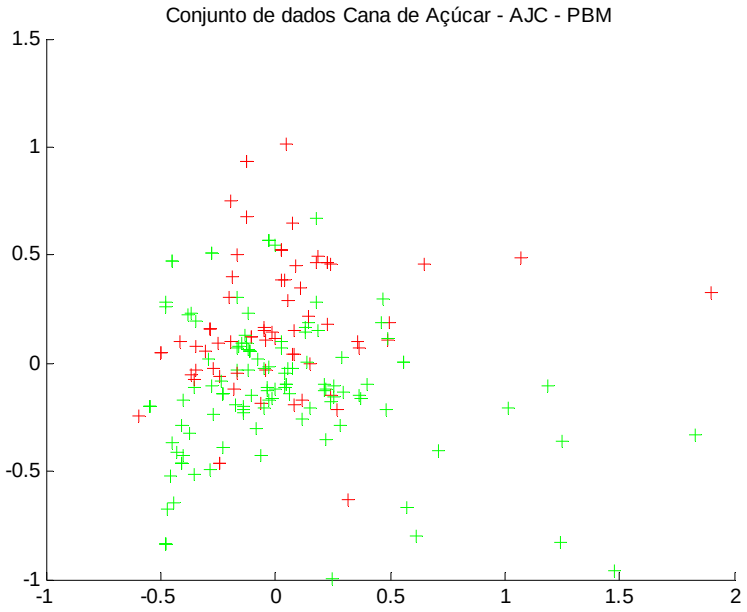

Figura 20: Distribuição dos pontos – base Cana de açúcar AJC - PBM

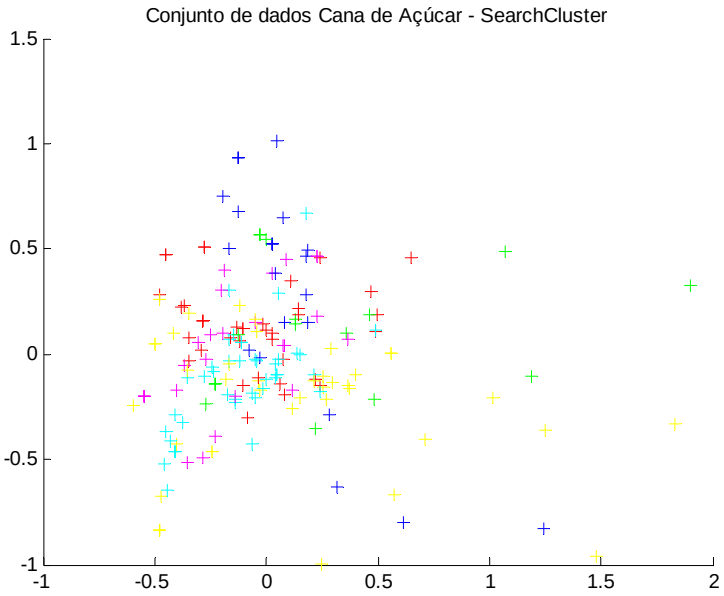

Figura 21: Distribuição dos pontos – base Cana de açúcar SeachCluster

Ao executarmos o agrupamento AJC simples, pelo índice Calinski e Harabasz, sem a seleção de atributos por algoritmo genético, encontramos um excelente resultado de 18 grupos, visto que os especialistas apontam para 20 agrupamentos distintos.

O mesmo não ocorreu com a execução do AJC, índice PBM, sem a seleção de atributos, que apresentou como resultado a formação de apenas 2 grupos, muito diferente do que dizem os especialistas.

Os programas SearchCluster e AJC utilizam o índice Calinski e Harabasz e calculam as distâncias dos objetos através da distância euclidiana, o que nos direcionam para a exatidão e similaridade dos resultados, pois os grupos formados são exatamente iguais no caso da cana de açúcar.

6.2 AGRUPAMENTO DE DADOS COM SELEÇÃO DE ATRIBUTOS

6.2.1 AVALIAÇÃO DA SELEÇÃO DE ATRIBUTOS

Para a validação de seleção de atributos do AJC utilizamos a ferramenta Weka. A seguir mostraremos os resultados das seleções de atributos para cada uma das bases de dados em estudo.

6.2.1.1 BASE DE DADOS ÍRIS

Base de dados inicialmente formada por 4 atributos e 150 registros.

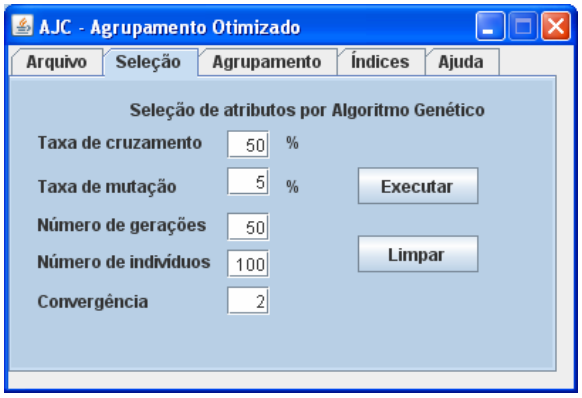

Figura 22a: Seleção de atributos AJC

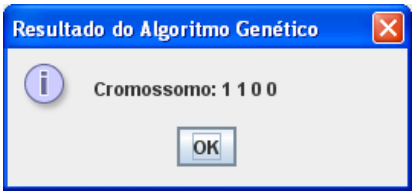

Figura 22b: Atributos selecionados

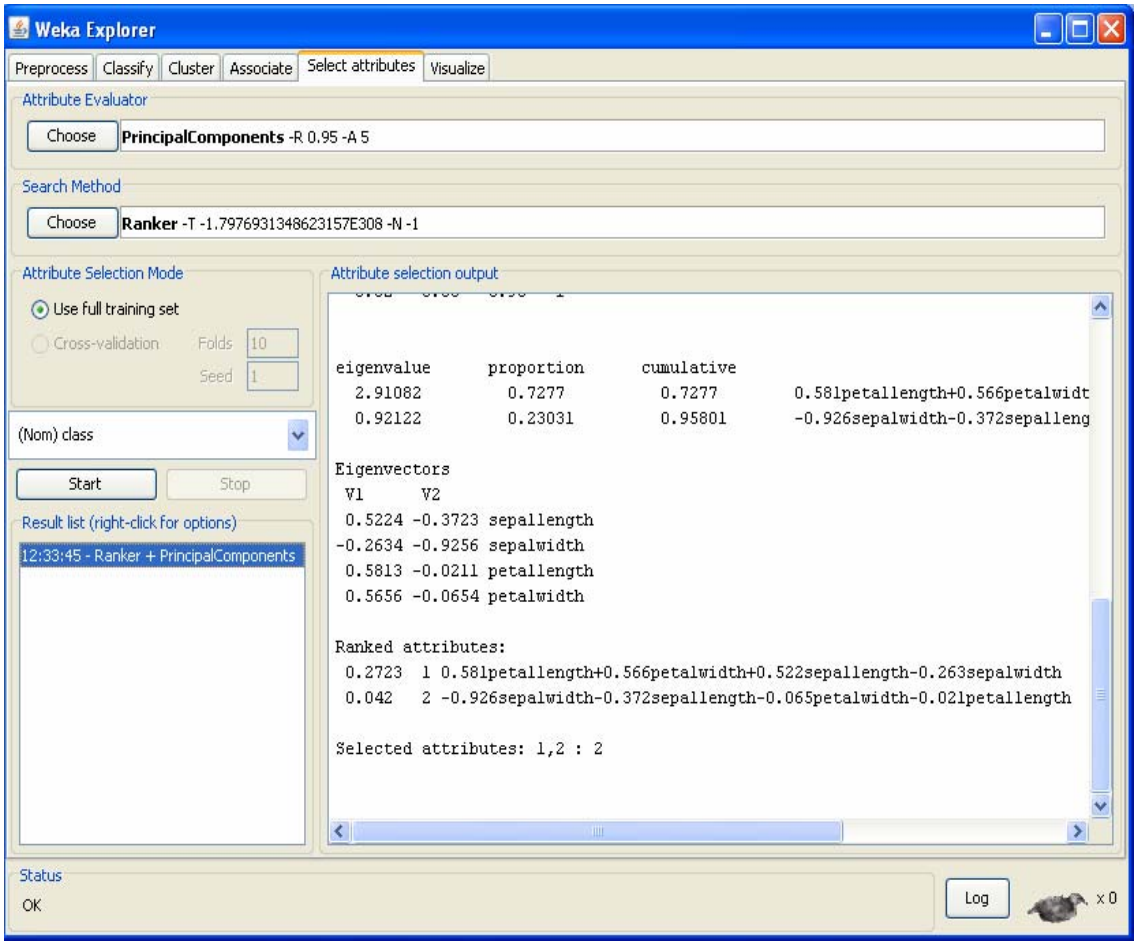

Figura 23: Seleção de atributos Weka – base Íris

Depois de executarmos muitas combinações do otimizador AJC para a seleção das variáveis por algoritmo genético da base de dados Íris, foi possível identificar que

essa base não necessita de muitas gerações e nem a presença de muitos indivíduos em cada uma delas para achar a mesma combinação do cromossomo. Verificando-se todas as execuções, notamos que nem mesmo as taxas de cruzamento ou de mutação, sendo esta última considerada elevada, afetam a descoberta do ótimo global, que é a composição do cromossomo formado pela seleção dos dois primeiros atributos, que possuem as maiores médias aritméticas.

A execução do programa para seleção de atributos PrincipalComponents –R 0.95 –A 5, do Weka, que executa a análise das componentes principais e transformações dos dados, nos mostrou um resultado exatamente igual ao AJC para esta base de dados, ou seja, os dois atributos selecionados.

Portanto, como resultado final, encontramos os dois primeiros atributos como sendo os mais importantes para representarem as características desta base de dados, efetivando uma redução de exatos 50% dos seus atributos.

6.2.1.2 BASE DE DADOS GLASS

Base de dados inicialmente formada por 9 atributos e 214 registros.

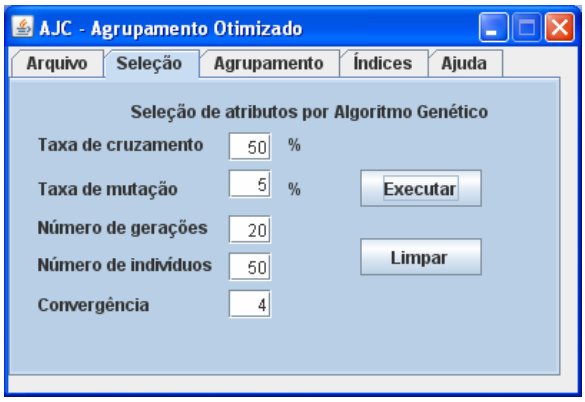

Figura 24a: Seleção de atributos AJC

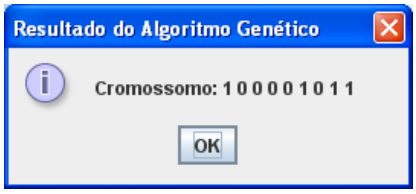

Figura 24b: Atributos selecionados

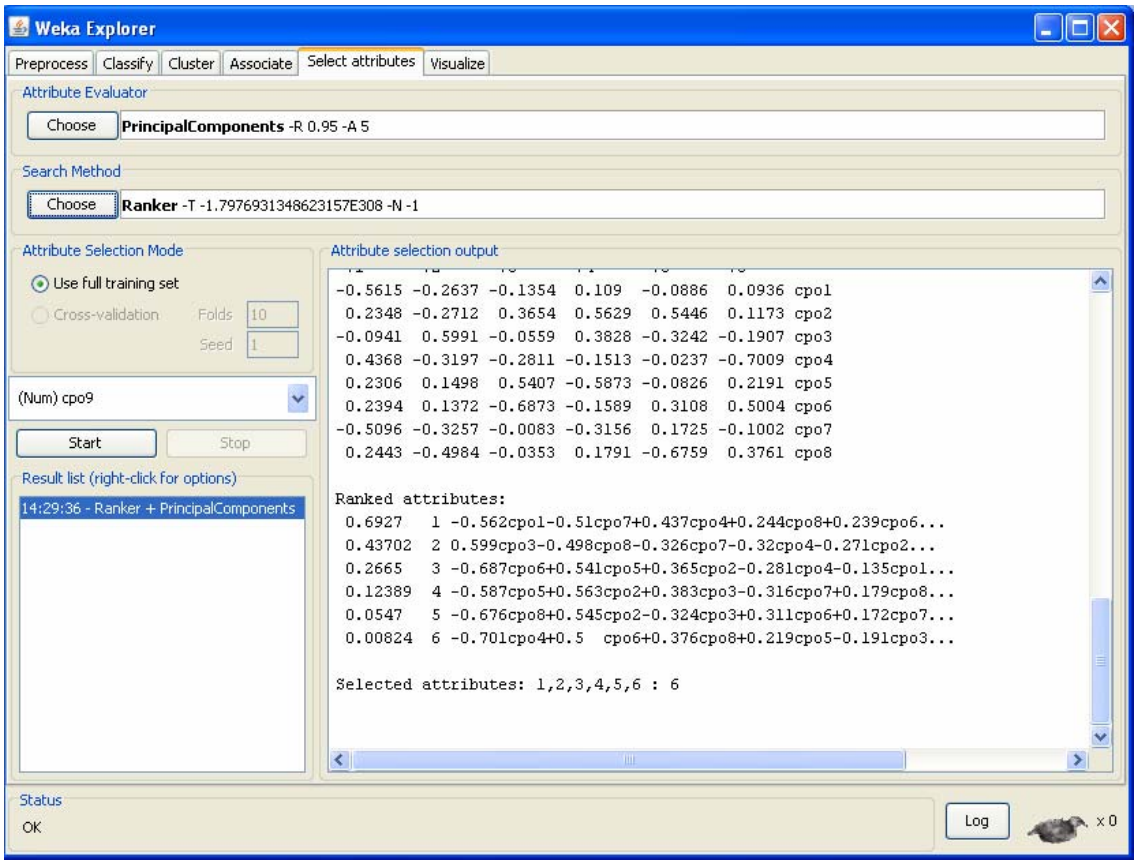

Figura 25: Seleção de atributos Weka – base Glass

Essa é uma base de difícil otimização. Em muitas execuções do AJC encontramos valores muito diversos. Quando aumentamos o número de convergência e diminuímos as gerações, encontramos um resultado que nos expressa um terceiro atributo sendo levado em consideração. Entretanto, se aumentamos o número de indivíduos e diminuímos a taxa de cruzamento, verificamos a saída de um sétimo e constante atributo e observamos a inclusão do primeiro e terceiro atributos, sendo o primeiro levado em consideração apenas nessa situação, o que nos mostra, estatisticamente, que pode ser considerado um atributo fraco para o agrupamento. Todavia, ao convergirmos para 4 atributos, nossa base parece ser constante em qualquer variação dos parâmetros de entrada, levando a crer que é uma boa convergência. Portanto, nossa seleção final de otimização contempla o primeiro, sexto, oitavo e nono atributos, que nos deixam com pouco mais de 44% dos atributos para representarem a base de dados.

A execução do programa para seleção de atributos PrincipalComponents –R 0.95 –A 5, do Weka, que executa a análise das componentes principais e transformações

dos dados, nos mostrou um resultado um pouco diferente do AJC para esta base de dados. Se por um lado o AJC encontrou 4 atributos, o algoritmo do Weka encontrou 6.

Para esta base o AJC nos mostra ser mais efetivo na redução dos atributos do que o algoritmo para seleção de atributos por componentes principais do Weca.

6.2.1.3 BASE DE DADOS HEART

Base de dados Heart-Cleveland inicialmente formada por 13 atributos e 303 registros.

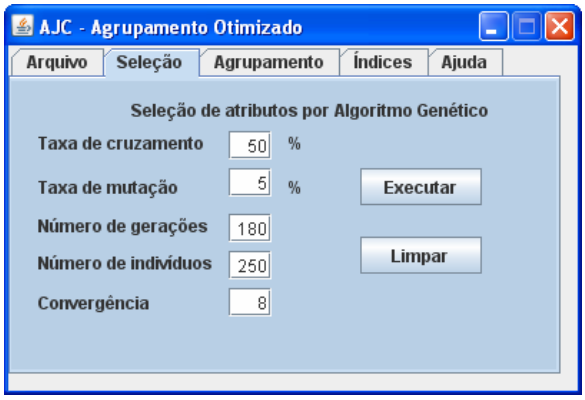

Figura 26a: Seleção de atributos AJC

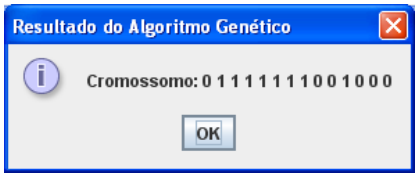

Figura 26b: Atributos selecionados

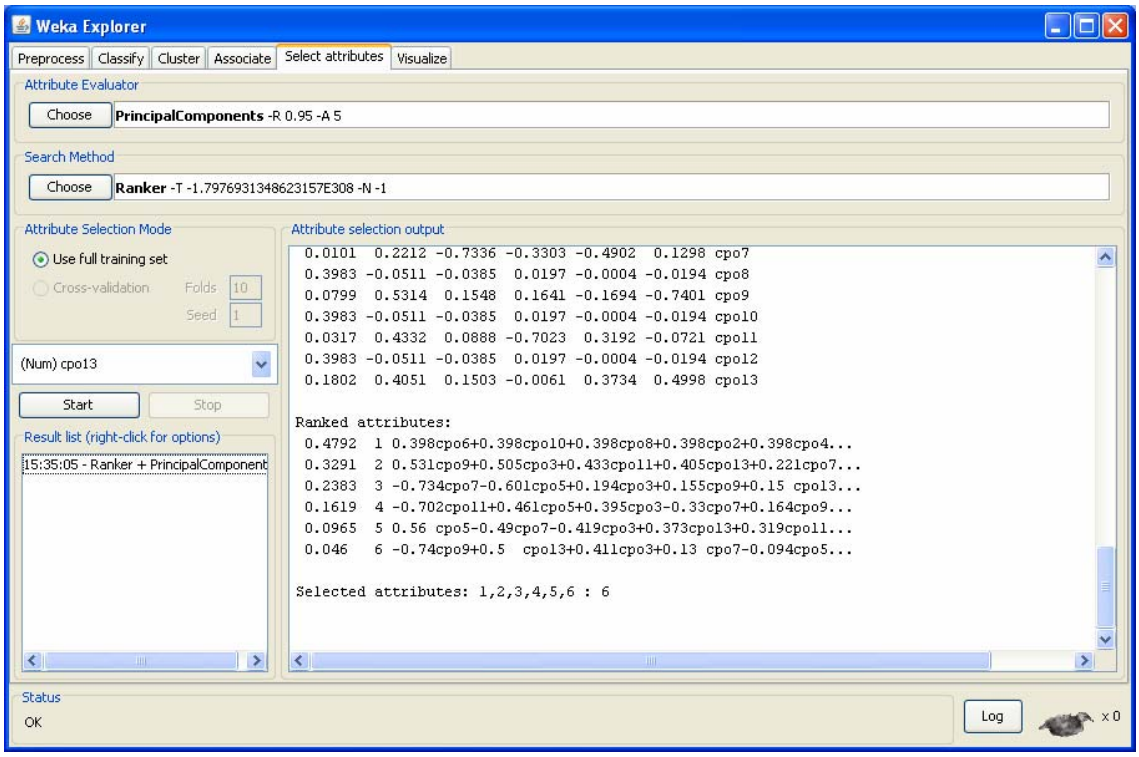

Figura 27: Seleção de atributos Weka – base Heart

Os resultados obtidos com a execução do AJC para esta base mostram-se bastante consistentes. Foram testadas várias combinações de parâmetros de entrada para cada execução. Primeiramente notamos que foi preciso maior número de gerações e de indivíduos. Outro fato observado foi o de que, apesar de convergir para o maior número possível de variáveis, somente foi encontrado um resultado calculado de 8 atributos ao final das execuções. Isto foi um fator marcante e mostrou claramente a consistência entre os resultados com esta convergência, e permitiu consolidar o resultado final nestes 8 atributos. Abaixo da convergência destas 8 seleções, os cromossomos gerados nunca permaneciam os mesmos de uma execução para outra, ou seja, inconsistentes entre si.

Os atributos selecionados foram o segundo, terceiro, quarto, quinto, sexto, sétimo, oitavo e o décimo primeiro, que possibilitou uma redução de aproximadamente 43% dos atributos da base de dados.

A execução do programa para seleção de atributos PrincipalComponents –R 0.95 –A 5, do Weka, que executa a análise das componentes principais e transformações dos dados, nos mostrou um resultado um pouco diferente do AJC para esta base de dados. Se por um lado o AJC encontrou 8 atributos, o algoritmo do Weka encontrou 6, superando em 2 atributos o primeiro.

6.2.1.4 BASE DE DADOS CANA DE AÇÚCAR

Base de dados cana de açúcar inicialmente formada por 19 atributos e 193 registros.

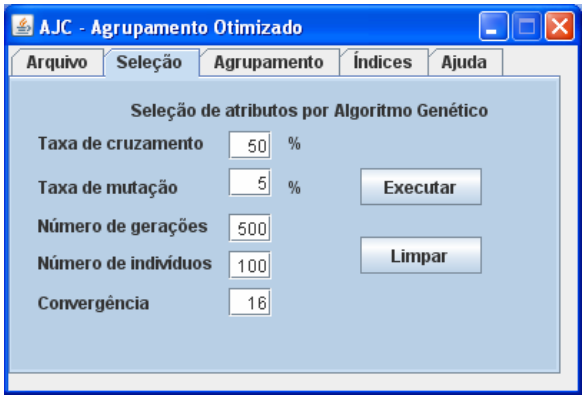

Figura 28a: Seleção de atributos AJC

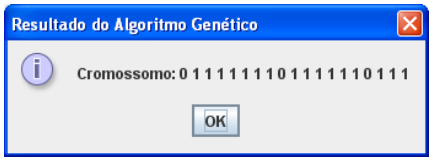

Figura 28b: Atributos selecionados

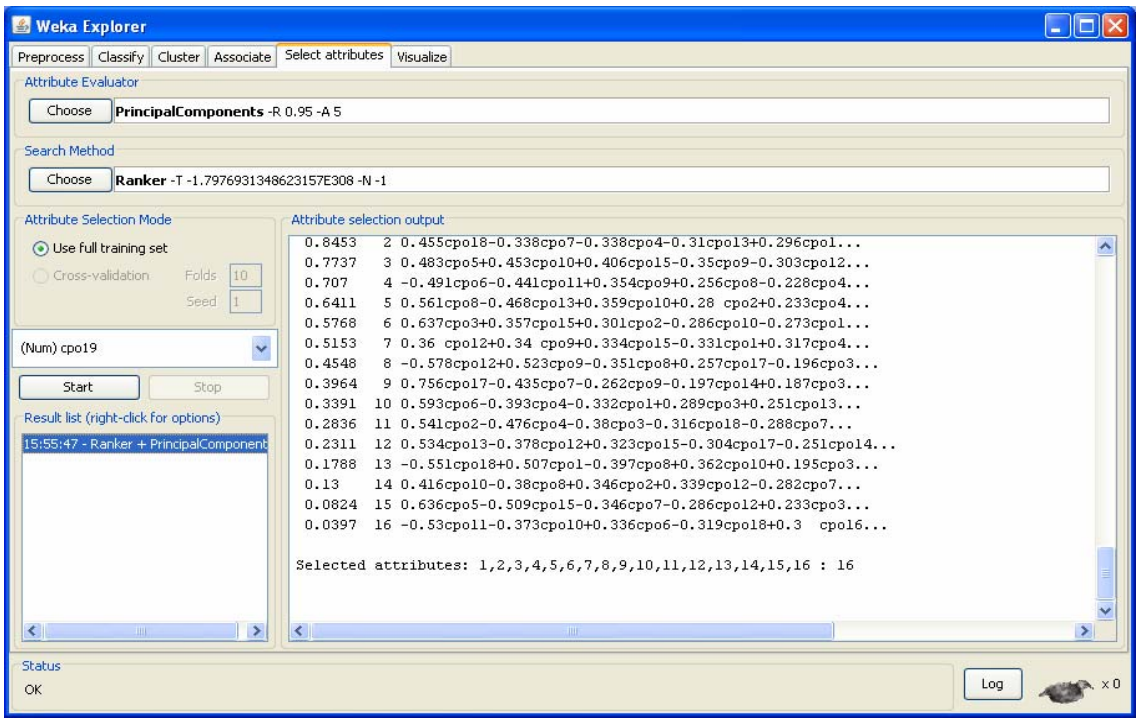

Figura 29: Seleção de atributos Weka – base Cana de açúcar

Os resultados obtidos nesta base mostram-se muito consistentes para o AJC. Muitas combinações de parâmetros de entrada foram testadas para cada execução. Primeiramente notamos que a convergência nunca ultrapassava 16 atributos. Abaixo desta convergência não obtivemos cromossomos consistentes de uma geração para outra.

O cromossomo gerado que deixou próximo da melhor consistência de execuções eliminava o primeiro, nono e décimo sexto alelo, que possibilitou uma redução de pouco mais de 25% dos atributos da base de dados.

A execução do programa para seleção de atributos PrincipalComponents –R 0.95 –A 5, do Weka, que executa a análise das componentes principais e transformações dos dados, nos mostrou um resultado exatamente igual ao AJC para esta base de dados, ou seja, os dezesseis atributos selecionados.

6.2.2 AVALIAÇÃO DO AGRUPAMENTO COM ATRIBUTOS SELECIONADOS

Nesta fase de testes utilizamos apenas os atributos escolhidos no processamento de seleção de atributos por algoritmo genético do AJC.

Vale ressaltar que, com a aplicação da otimização por seleção de atributos em uma base de dados conseguimos diminuir consideravelmente a quantidade de atributos e, conseqüentemente, gastamos pouco tempo computacional. Se por um lado ganhamos tempo e diminuímos esforço computacional, por outro, podemos perder informações que aparentemente podem ser consideradas irrelevantes pelo algoritmo de seleção de atributos. Todas as variáveis descartadas, por menor que sejam seus valores, alteram os resultados finais e, por menor que seja esta variação, ela deve ser levada em consideração e discutida com o responsável pela base de dados. Este paradigma ocorreu neste trabalho e em muitas outras pesquisas.

6.2.2.1 BASE DE DADOS ÍRIS

Para a base de dados Íris considerou-se os dois primeiros atributos, que correspondem à largura da pétala e largura da sépala, novamente usando os mesmos programas e a mesma quantidade de registros, 150.

Tabela 5

Agrupamento com atributos selecionados – base Íris

Programa	Classe 1	Classe 2	Classe 3	Registros
AJC	**50**	**48**	**52**	**150**
AJC - PBM	**70**	**80**		**150**
SearchCluster	**50**	**48**	**52**	**150**

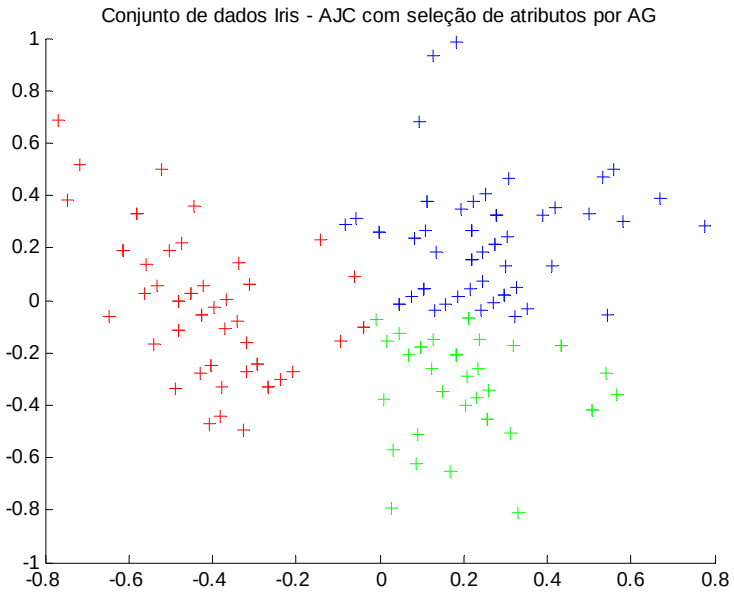

Figura 30: Distribuição dos pontos – base Íris AJC - Calinski e Harabasz

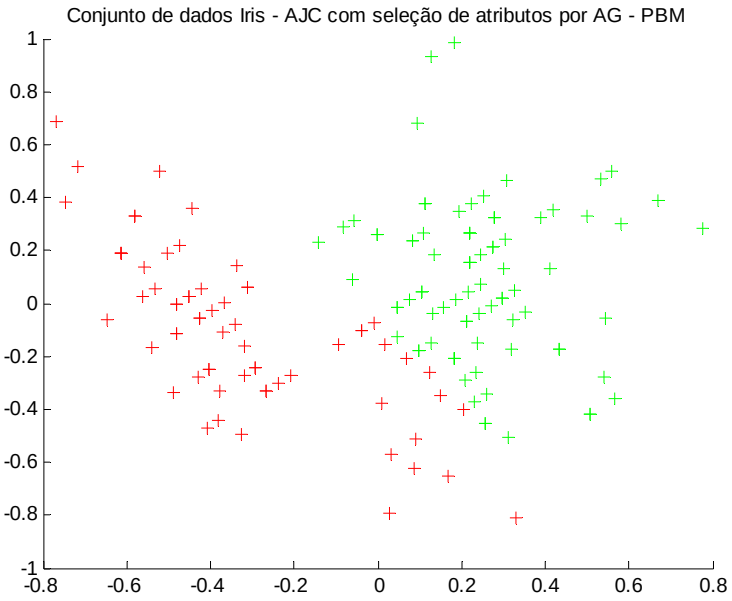

Figura 31: Distribuição dos pontos – base Íris AJC - PBM

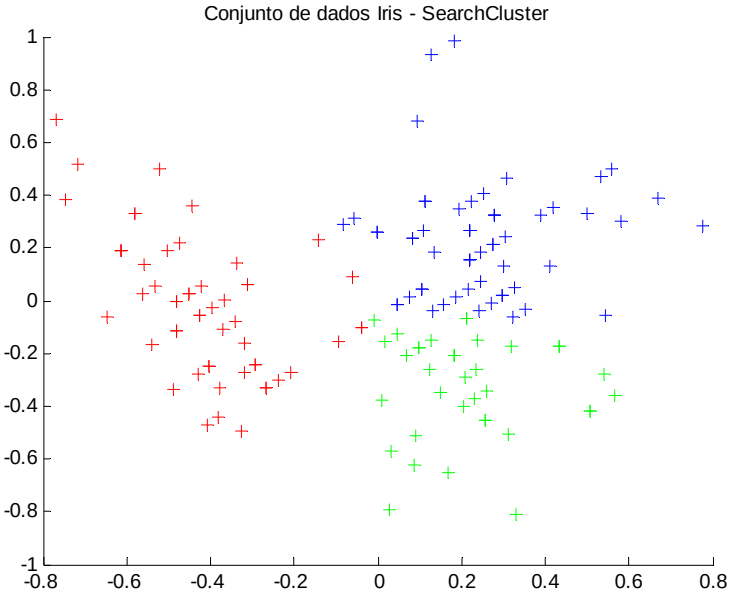

Figura 32: Distribuição dos pontos – base Íris SeachCluster

O que aconteceu depois da execução do programa otimizado com a seleção de atributos por algoritmo genético para esta base foi que o AJC, índice Calinski e Harabasz, e o SearchCluster, alcançaram o mesmo resultado em cada um dos 3 grupos.

O primeiro perdeu um registro e o segundo perdeu 2 registros, comparando-se com a base completa. Mesmo assim, obtiveram um excelente índice de 96% de agrupamentos corretos, considerando-se as informações da base disponibilizadas pelo proprietário dos dados.

O índice PBM do AJC novamente não atingiu um bom resultado, pois ficou muito distante da realidade, tanto de grupos quanto de elementos em cada grupo. Portanto, consolida-se que esse índice não possui um bom comportamento para esta base de dados.

6.2.2.2 BASE DE DADOS GLASS

Neste teste foi utilizado a base otimizada do arquivo *glass*, contemplando 4 atributos selecionados: primeiro, sexto, oitavo e nono. Permanecemos com os 214 registros.

Tabela 6

Agrupamento com atributos selecionados – base Glass

	AJC	AJC – PBM	SearchCluster
Classe 1	**57**	**180**	**18**
Classe 2	**43**	**34**	**33**
Classe 3	**59**		**23**
Classe 4	**21**		**15**
Classe 5	**5**		**22**
Classe 6	**29**		**13**
Classe 7			**8**
Classe 8			**21**
Classe 9			**18**
Classe 10			**19**
Classe 11			**2**
Classe 12			**7**
Classe 13			**15**
Registros	**214**	**214**	**214**

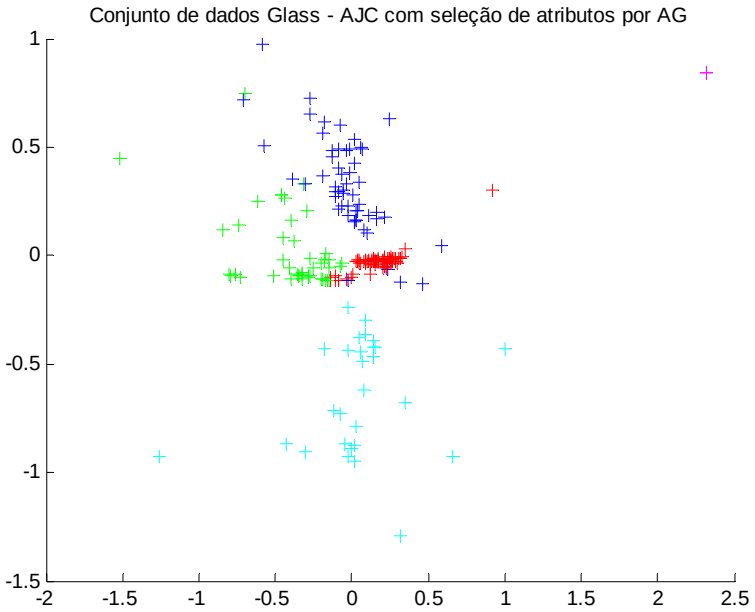

Figura 33: Distribuição dos pontos – base Glass AJC - Calinski e Harabasz

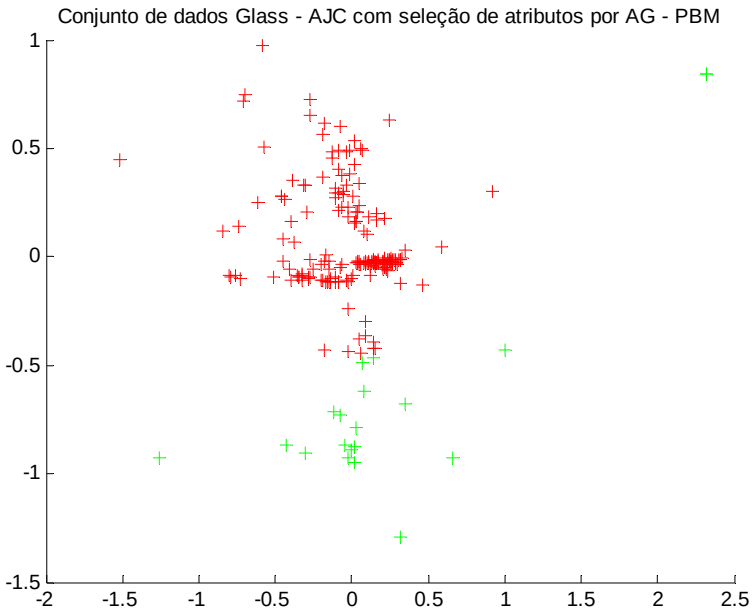

Figura 34: Distribuição dos pontos – base Glass AJC - PBM

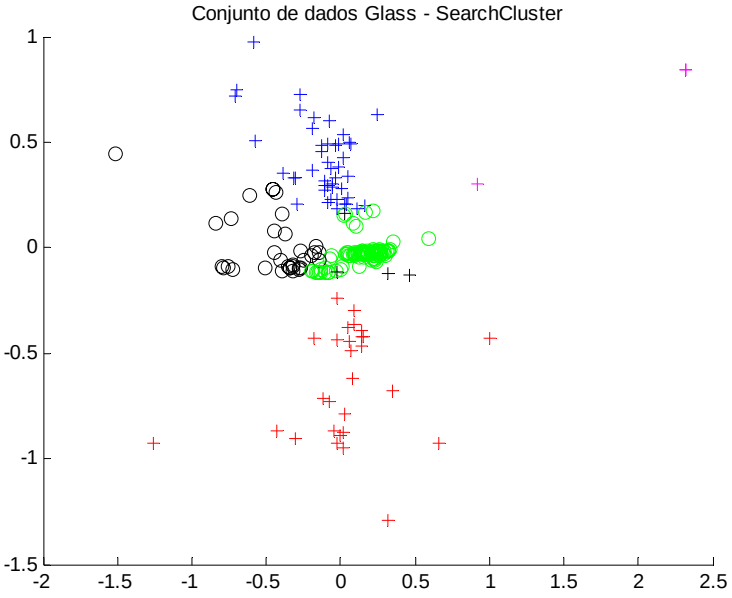

Figura 35: Distribuição dos pontos – base Glass SeachCluster

Executando os três algoritmos na base *glass* com as variáveis selecionadas pelo AJC, notamos um comportamento muito diferente em cada um. O índice PBM apenas forma 2 grupos, 180 e 34 elementos para o primeiro e segundo grupo, respectivamente. A informação do usuário para 2 grupos indica 163 e 51, para o primeiro e segundo

grupo respectivamente. O AJC, índice Calinski e Harabasz, forma 6 grupos, a mesma quantidade de agrupamentos formados em laboratório. Se com o agrupamento com os 9 atributos, descobrimos 2 grupos distintos, com a execução otimizada, chegamos a seis grupos distintos. A execução do algoritmo genético nos deu uma eliminação de aproximadamente 60% dos atributos, ou seja, com 4 atributos e, aplicando-se o agrupamento AJC sobre estes 4 atributos, o resultado superou em muito o agrupamento simples. O algoritmo SearchCluster encontrou 13 grupos distintos. Esse número de grupos é muito mais elevado do que a indicação informada pelo proprietário técnico dos dados, não parecendo existir um bom comportamento deste programa para esta base apenas para os atributos selecionados pelo AG.

Com base nos resultados encontrados, podemos perceber que o AJC, índice Calinski e Harabasz, encontra um resultado excelente, considerando-se os grupos informados pelo proprietário das informações. Já o AJC – PBM e o SearchCluster não apresentam bons resultados para esta base otimizada.

6.2.2.3 BASE DE DADOS HEART

Inicialmente possuindo 13 atributos e 303 registros, esta base de dados foi reduzida para 8 atributos selecionados pelo AG para um posterior agrupamento dos dados.

Tabela 7

Agrupamento com atributos selecionados – base Heart

Programa	Classe 1	Classe 2	Registros
AJC	**157**	**146**	**303**
AJC - PBM	**157**	**146**	**303**
SearchCluster	**157**	**146**	**303**

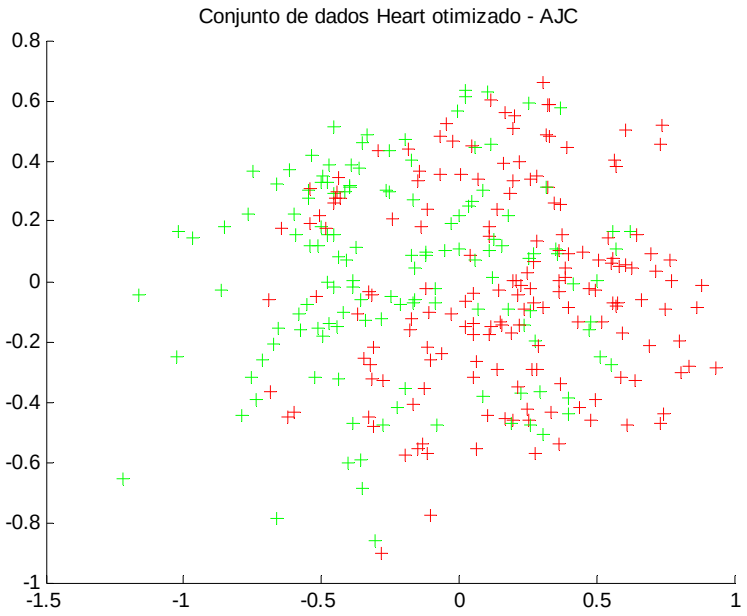

Figura 36: Distribuição dos pontos – base Heart AJC - Calinski e Harabasz

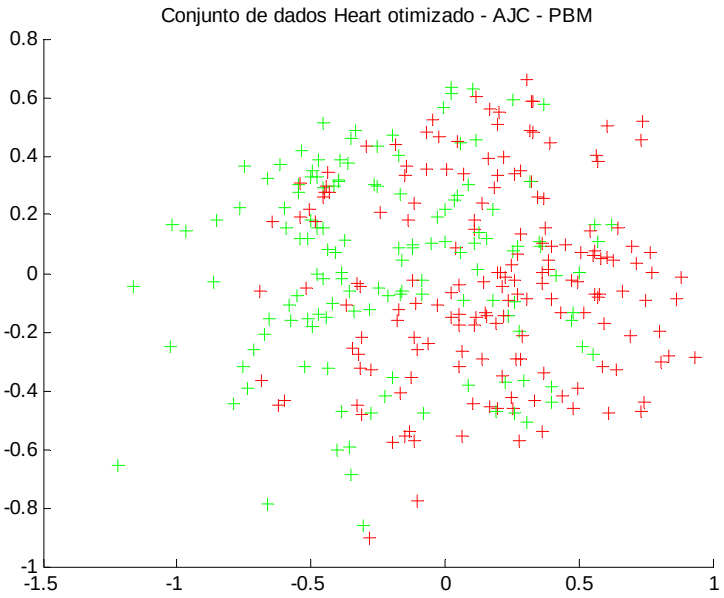

Figura 37: Distribuição dos pontos – base Heart AJC - PBM

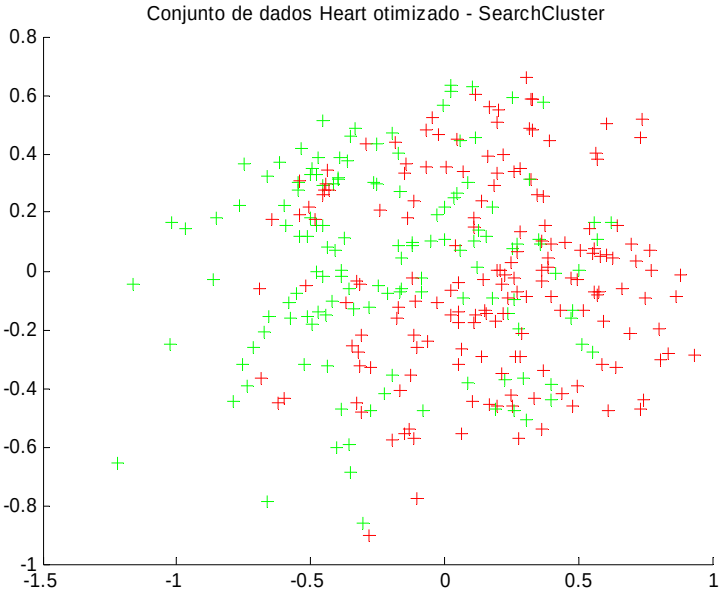

Figura 38: Distribuição dos pontos – base Heart SeachCluster

Neste caso de teste, na execução da seleção AJC obteve-se um ganho, ou seja, uma redução de atributos, de aproximadamente 43%. A partir desta base reduzida aplicamos o agrupamento do AJC e conseguimos os mesmos 2 grupos, porém com uma melhora de 2 dos falsos positivos. Isso nos deixa com quase 96% de acerto, considerando os dados disponibilizados pelo proprietário da base de dados. A execução

do agrupamento com os dados otimizados nesta base foi mais que satisfatório, pois nos mostrou que a redução dos 6 atributos nos deixou com um melhor agrupamento, portanto, esses atributos podem ser desprezados em qualquer execução ou criação de grupos sem que se perca a confiabilidade da informação.

6.2.2.4 BASE DE DADOS CANA DE AÇÚCAR

Inicialmente possuindo 19 atributos e 193 registros, esta base de dados foi reduzida para 16 atributos selecionados pelo AG para um posterior agrupamento dos dados.

Tabela 8

Agrupamento com atributos selecionados – base Cana de açúcar

	AJC	AJC – PBM	SearchCluster
Classe 1	**5**	**82**	**7**
Classe 2	**6**	**111**	**6**
Classe 3	**10**		**11**
Classe 4	**7**		**9**
Classe 5	**6**		**16**
Classe 6	**17**		**11**
Classe 7	**10**		**25**
Classe 8	**13**		**4**
Classe 9	**5**		**9**
Classe 10	**4**		**10**
Classe 11	**10**		**11**
Classe 12	**20**		**9**
Classe 13	**2**		**8**
Classe 14	**20**		**11**
Classe 15	**7**		**15**
Classe 16	**10**		**17**
Classe 17	**10**		**14**
Classe 18	**16**		
Classe 19	**15**		
Registros	**193**	**193**	**193**

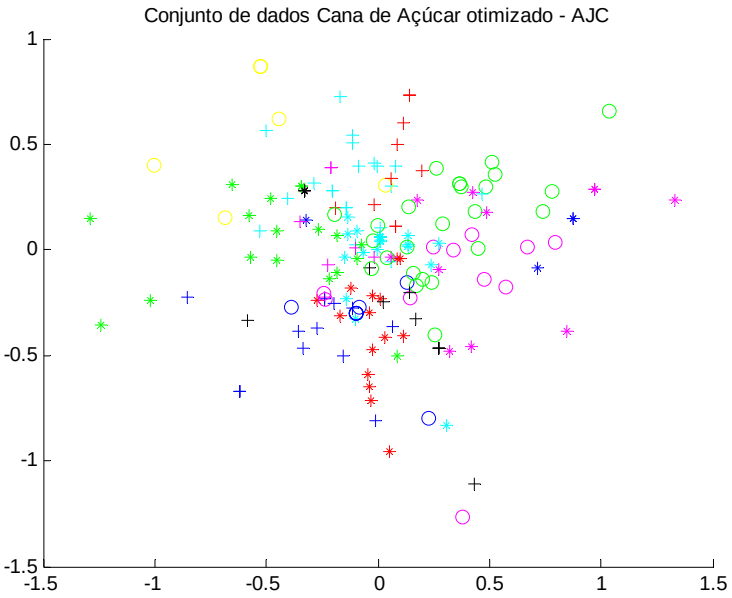

Figura 39: Distribuição dos pontos – base Cana de açúcar AJC - Calinski e Harabasz

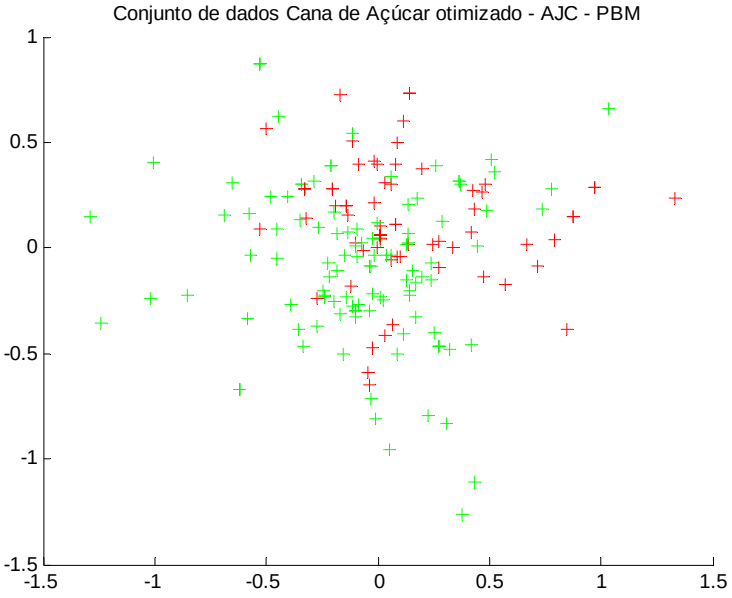

Figura 40: Distribuição dos pontos – base Cana de açúcar AJC - PBM

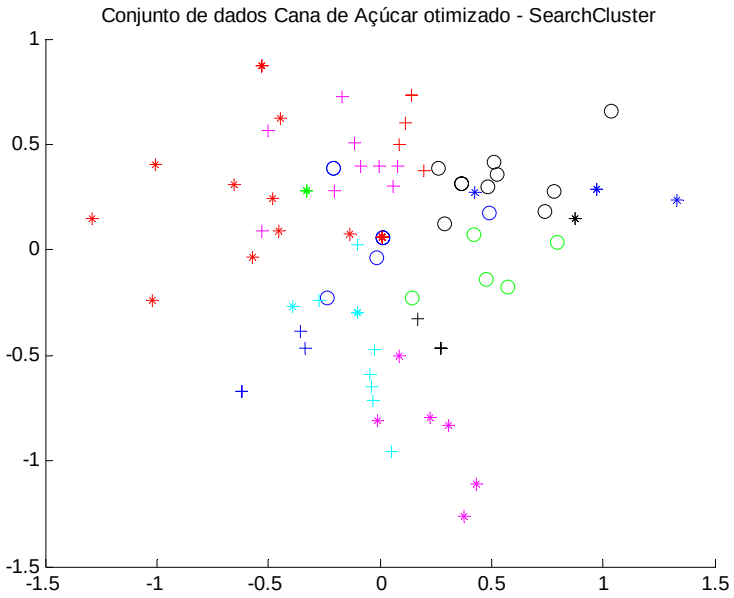

Figura 41: Distribuição dos pontos – base Cana de açúcar SeachCluster

Na execução da seleção AJC obteve-se um ganho, ou seja, uma redução de atributos, de aproximadamente 15%, correspondentes à redução de 3 variáveis. A partir desta base reduzida aplicamos os agrupamentos do AJC e conseguimos resultados muito diferentes com os dois algoritmos. A execução do agrupamento AJC, índice Calinski e Harabasz, sobre os dados otimizados pode ser considerada um sucesso, pois chegou muito próximo dos 20 grupos indicados pelo proprietário dos dados. Este índice encontrou 19 grupos diferentes. Com o AJC, índice PBM, encontramos apenas 2 grupos distintos, não sendo, portanto, muito indicado para esta base de dados. O SearchCluster encontrou 16 grupos, 3 a menos do que o AJC, índice Calinski e Harabasz, e 4 a menos do que o total estudado em laboratório pelo criador das informações desta base.

7 CONCLUSÃO E TRABALHOS FUTUROS

Descobrir informações implícitas em base de dados não é uma tarefa muito simples, que requer conhecimento mínimo dos dados, um trabalho computacional estatístico ou a participação efetiva de um especialista sobre as informações.

Por mais que existam programas implementados para mineração de dados, algoritmo genético, agrupamento de dados ou índices de grupos, na maioria das vezes haverá distorções ou variações de resultados sobre uma mesma base de dados.

Facilitar a interação do usuário com um programa computacional através de uma interface gráfica amigável estimula o uso da ferramenta e, conseqüentemente, a melhoria e o avanço desta. A interface gráfica muito amigável e de fácil entendimento do aplicativo AJC sempre esteve em primeiro lugar e, logo a seguir, a integração entre os módulos, de maneira que o AJC possibilitasse o uso isolado ou em conjunto dos módulos existentes.

Os índices implementados no AJC possuem comportamentos finais bem diferentes uns dos outros e, portanto, inclusive neste momento, a participação de um especialista deve ser levada em consideração para que se chegue o mais próximo de um resultado confiável.

Como foi comprovado, o comportamento de alguns dados para um determinado algoritmo é melhor do que para outros. Para as bases de dados estudadas, o AJC, índice Calinski e Harabasz, e o SearchCluster registraram uma grande similaridade nas respostas dos processamentos, o que nos deixa muito satisfeito, visto que é um produto em produção e muito validado no meio acadêmico. Já o AJC, índice PBM, não se comportou como esperado, pois não nos retornou resultados muito aceitáveis.

Vale ressaltar que não se tem um erro quando não se chega próximo ao resultado criado em laboratório para uma determinada base de dados. O que se pretende descobrir quando aplicamos um agrupamento de dados, otimizado ou não, é o conhecimento que não é explícito e que pode servir de suporte na tomada de decisões ou ajudar de alguma outra forma possível.

Uma questão que devemos levantar e que não podemos esquecer é o conhecimento do usuário proprietário ou especialista dos dados. Se um conhecimento prévio nos mostra que devemos levar em consideração um determinado atributo, então não podemos desconsiderá-lo por questão de relevância técnica. Mesmo que o classificador, otimizado ou não, crie grupos automáticos, todo o resultado pode ficar

completamente diferente se considerarmos o conjunto de informações passadas pelo proprietário dos dados quando processamos o algoritmo.

Embora não seja possível afirmar que os resultados encontrados sejam os melhores em cada modelo no qual foi aplicado o AG, registrou-se desempenho bastante superior em relação aos modelos brutos, ou seja, sem a seleção das variáveis. E podemos afirmar que o AJC nos deu evidências significativas de sua eficiência e eficácia.

Como um dos trabalhos futuros, deve ser realizado um melhor estudo sobre as bases de dados onde o AJC apresentou melhores resultados, visando um melhor entendimento e buscando padrões capazes de descrever melhor as bases de dados em que podemos aplicar o algoritmo com sucesso, bem como pesquisar e implementar novos índices.

Outro trabalho importante para ser incorporado ao AJC é a visualização dos resultados de forma gráfica, para que se tenha a demonstração visual do comportamento dos dados processados.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[1] JAIN, A. K., FLYN P. J., *Data Clustering: A Review*, ACM Computing Surveys, v. 31, n. 3, pp. 264-323, 1999.

[2] HALL, M. A., HOLMES, G., *Benchmarking Attribute Selection Techniques for Discrete Class Data Mining*. IEEE Transactions on Knowledge and Data Engineering, v. 15, n. 6, pp. 1437-1447, 2003.

[3] BLUM, A. L., *Selection of Relevant Features and Examples in Machine Learning*. Artificial Intelligence, v. 97, n. 1-2, pp. 245-271, 1997.

[4] DASH, M., LIU, H., *"Feature Selection for Classification"*, Intelligent Data Analysis, An International Journal, Elsevier, v. 1, n. 3, 1997.

[5] KOHAVI, R., JOHN G. H., *Wrappers for Feature Subset Selection*, Artificial Intelligence, v. 97, n. 1-2, pp. 273-324, 1997.

[6] HALL, A. M., *"Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning"*. In: Proceedings of the 17th International Conference Machine Learning, pp. 359-366, 2000.

[7] ALMUALLIM, H., DIETTERICH, T. G., *"Learning with Many Irrelevant Feature"*. In: Proceedings of the 9th International Conference on Artificial Intelligence, pp. 547-552, 1991.

[8] HOLLAND, J. H., *Genetic Algorithm.* Scientific America, New York, v. 267, n. 1, pp. 44-50, 1992.

[9] LEVINE, D., *A Parallel Genetic Algorithm for the Set Partitioning Problem*. Tech. Rep. No ANL 94/23, Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL, 1994.

[10] GOLDBERG, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading: Addison-Wesley, MA, 1989.

[11] BLEKAS, K., STAFYLOPATIS, A., *Real-coded Genetic Optimization of Fuzzy Clustering*. In Proceddings of the European Congress on Intelligence Techniques and Soft Computing (EUFIT'96), pp. 461-465, Aachen, Germany, 1996.

[12] MITCHELL, M., *An introduction to genetic algorithms*, Cambrige, MA: MIT Press, 1999.

[13] MAN, K. F., TANG, K. S., KWONG, S., *Genetic algorithms: Concepts and designs*, London: Springer, ISBN 1-85233-072-4, 1999.

[14] BONISSONE, P. P., CHEN, Y. T., GOEBEL, K. et al., *"Hybrid soft computing systems: Industrial and commercial applications"*. In: Proceedings of the IEEE, v. 87, n. 9, pp. 1641-1667, 1999.

[15] ILLICH, N., SIMONOVIC, S., *"Evolutionary algorithm for minimization of pumping cost"*, Journal of Computing in Civil Engineering, v. 12, n. 4, pp. 232-240, 1998.

[16] BACK, T., KURSAWE, F., *"Evolutionary algorithms for fuzzy logic: A brief overview"*. In: Proceedings of th 5th Internationsl Conference IPMU: Information Processing and Management of Uncertainty in Knowledge-Based Systems, vol. 2, pp. 659-664, 1994.

[17] CORDÓN, O., HERRERA, F., HOFFMANN, F. et al., *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, ISBN 981-02-4016-3, World Scientific, Singapore, 2001.

[18] MIKI, M., HIROYASU T., KANERO, K. et al., *"A Parallel Genetic Algorithm with Distributed Environment Multiple Population Scheme"*. In: Proceedings of the 3nd World Congress of Structural and Multidisciplinary Optimization, Doshisha University, Kyoto, v. 1, pp. 186-191, 1999.

[19] PAPPA, G. L., *Seleção de Atributos Utilizando Algoritmos Genéticos Multiobjetivos*. Dissertação de M.Sc., Pontifícia Universidade do Paraná, Curitiba, 2002.

[20] DE JONG, K. A., *An analysis of the behavior of a class of genetic adaptive systems*. PhD Thesis, Dissertation Abstract International, University of Michigan, Ann Arbor.

[21] SHARIF, M., WARDLAW, R., *"Multireservoir systems optimization using genetic algorithms"*. Case study, Journal of Computing in Civil Engineering, v. 14, n. 4, pp. 255-263, 2000.

[22] GREFENSTETTE, J., *Optimization of control parameters for genetic algorithms*. IEEE Transactions on Systems, Man, and Cybernetics, v. 16, n. 1, pp. 122-128, 1986.

[23] SPEARS, W. M., *Crossover or mutation?* In: WHITLEY, L. D. (Ed.) Foundation of genetic algorithms, Los Altos, CA: Morgan Kaufmann, v. 2, pp. 221-237, 1993.

[24] SPEARS, W. M., *"Adapting Crossover in Evolutionary Algorithms"*. In: Proceedings of the Evolutionary Programming Conference, Cambridge: MIT Press, pp. 367-384, 1995.

[25] CIOS, K. J., PEDRYCZ, W., SWINIARSKI, R. W., *Data Mining Methods for knowledge Discovery*. Kluwer Academic Publishers, Boston, 1998.

[26] KASKI, S., *Data exploration using self-organizing maps*. Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series n. 82, Finish Academy of Technology, 1997.

[27] KANUNGO, T., MOUNT, D. M., NETANYAHU, N. S. et al., *"Computing nearest neighbors for moving points and applications to clustering"*. In:

Proceedings of 10th ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, pp. 931-932, 1999.

[28] BRADLEY, P. S., FAYYAD, U., REINA, C., *"Scaling clustering algorithms to large database"*. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98), AAAI Press, pp. 9-15, 1998.

[29] XU, X., ESTER, M., KRIEGEL, H.-P., Sander, J., *Clustering and Knowledge Discovery in Spatial Databases*. Vistas in Astronomy, v. 41, n. 3, pp. 397-403, 1997.

[30] CALINSKI, T., HARABASZ, J., *A Dendrite Method for Cluster Analysis*. Communications in Statistics, v. 3, n. 1, pp. 1-27, 1974.

[31] ANDRADE, L. P., *Procedimento Interativo de Agrupamento de Dados*. Dissertação de M.Sc., Universidade Federal do Rio de Janeiro, Engenharia Civil, Rio de Janeiro, 2004.

9 APÊNDICE

```
/*
 * AJC.java
 *
 * Criado em 19 de Dezembro de 2007, 21:40
 */

/**
* autor  José Cláudio Garcia Damaso
*/
import javax.swing.*;
import java.io.*;      // File
import java.util.*;    // Random
import java.text.*;    // DecimalFormat
//import java.lang.String;

public class AJC extends javax.swing.JFrame {
    int [] vetCromossomo;
    int executouAG=0;
    /** Creates new form AJC */
    public AJC() {
        initComponents();
    }

    private void initComponents() {
        jTabbedPaneProg = new javax.swing.JTabbedPane();
        panelArquivo = new java.awt.Panel();
        labelNome = new java.awt.Label();
        labelAtributo = new java.awt.Label();
        choiceAtributo = new java.awt.Choice();
        for (int i=0; i<500; i++)       {
            choiceAtributo.addItem(String.valueOf(i+1));
        }
        textFieldArquivo = new java.awt.TextField();
```

```
jButtonBuscar = new javax.swing.JButton();
panelOtimizador = new java.awt.Panel();
labelCruzamento = new javax.swing.JLabel();
labelMutacao = new javax.swing.JLabel();
labelGeracao = new javax.swing.JLabel();
labelTitOtimizador = new javax.swing.JLabel();
labelIndividuo = new javax.swing.JLabel();
labelConvergencia = new javax.swing.JLabel();
tFieldCruzamento = new javax.swing.JTextField();
tFieldMutacao = new javax.swing.JTextField();
tFieldGeracao = new javax.swing.JTextField();
tFieldIndividuo = new javax.swing.JTextField();
tFieldConvergencia = new javax.swing.JTextField();
jButtonExecutaAG = new javax.swing.JButton();
jButtonLimpaAG = new javax.swing.JButton();
jLabelPercCruza = new javax.swing.JLabel();
jLabelPercMut = new javax.swing.JLabel();
panelAgrupamento = new java.awt.Panel();
labelTitAgrup = new javax.swing.JLabel();
labelQtdGrupo = new javax.swing.JLabel();
choiceMinimo = new java.awt.Choice();
for (int i=0; i<499; i++)       {
   choiceMinimo.addItem(String.valueOf(i+2));
}
choiceMaximo = new java.awt.Choice();
for (int i=0; i<499; i++)       {
   choiceMaximo.addItem(String.valueOf(i+2));
}
labelMinimo = new javax.swing.JLabel();
labelMaximo = new javax.swing.JLabel();
jButtonExecutaAgrup = new javax.swing.JButton();
jRadioAgrupCal = new javax.swing.JRadioButton();
jRadioAgrupPBM = new javax.swing.JRadioButton();
panelIndice = new java.awt.Panel();
```

```
                jButtonIndiceOK = new javax.swing.JButton();
                jPanel1 = new javax.swing.JPanel();
                jRadioCalHara = new javax.swing.JRadioButton();
                jRadioPBM = new javax.swing.JRadioButton();
                panelAjuda = new java.awt.Panel();
                textAreaAjuda = new java.awt.TextArea();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
                setTitle("AJC - Agrupamento Otimizado");
                setResizable(false);
                   jTabbedPaneProg.setBackground(new java.awt.Color(191, 210, 228));
                labelNome.setText("Nome");
                labelAtributo.setText("Atributos");
                textFieldArquivo.setEnabled(false);
                textFieldArquivo.setText(" ");
                jButtonBuscar.setText("Buscar");
                jButtonBuscar.addMouseListener(new java.awt.event.MouseAdapter() {
                   public void mouseClicked(java.awt.event.MouseEvent evt) {
                      jButtonBuscarMouseClicked(evt);
                   }
                });

                javax.swing.GroupLayout panelArquivoLayout = new
javax.swing.GroupLayout(panelArquivo);
                panelArquivo.setLayout(panelArquivoLayout);
                panelArquivoLayout.setHorizontalGroup(
          panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING).addGroup(panelArquivoLayout.createSequentialGroup().addGap(20, 20,
20)
.addGroup(panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING).addComponent(labelNome,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(labelAtributo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING).addGroup(panelArquivoLayout.createSequentialGroup()
.addComponent(textFieldArquivo, javax.swing.GroupLayout.PREFERRED_SIZE, 179,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jButtonBuscar, javax.swing.GroupLayout.PREFERRED_SIZE, 76,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addComponent(choiceAtributo, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(34, 34, 34)));
                panelArquivoLayout.setVerticalGroup(
panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelArquivoLayout.createSequentialGroup()
.addGroup(panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING).addGroup(panelArquivoLayout.createSequentialGroup()
.addContainerGap().addComponent(jButtonBuscar)).addGroup(panelArquivoLayout.cr
eateSequentialGroup().addGap(68, 68, 68)
.addGroup(panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING).addComponent(textFieldArquivo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(labelNome, javax.swing.GroupLayout.DEFAULT_SIZE, 20,
Short.MAX_VALUE)))).addGap(28, 28, 28)
.addGroup(panelArquivoLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.LEADING).addComponent(labelAtributo,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(choiceAtributo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(69, 69, 69)) );
                jTabbedPaneProg.addTab("Arquivo", panelArquivo);
                labelCruzamento.setText("Taxa de cruzamento");
                labelMutacao.setText("Taxa de muta\u00e7\u00e3o");
                labelGeracao.setText("N\u00famero de gera\u00e7\u00f5es");
                labelTitOtimizador.setBackground(new java.awt.Color(191, 210, 228));
                labelTitOtimizador.setText("ALGORITMO GEN\u00c9TICO");
                labelIndividuo.setText("N\u00famero de indiv\u00edduos");
                labelConvergencia.setText("Converg\u00eancia");

tFieldCruzamento.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
                tFieldCruzamento.setText("50");
                tFieldMutacao.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
                tFieldMutacao.setText("5");
                tFieldGeracao.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
                tFieldGeracao.setText("50");
                tFieldIndividuo.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
                tFieldIndividuo.setText("100");

tFieldConvergencia.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
                jButtonExecutaAG.setText("Executar");
                jButtonExecutaAG.addMouseListener(new java.awt.event.MouseAdapter()
{
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        jButtonExecutaAGMouseClicked(evt);
                    }
                });

                jButtonLimpaAG.setText("Limpar");
                jButtonLimpaAG.addMouseListener(new java.awt.event.MouseAdapter() {
```

```
                public void mouseClicked(java.awt.event.MouseEvent evt) {
                    jButtonLimpaAGMouseClicked(evt);
                }
            });

            jLabelPercCruza.setText("%");
            jLabelPercMut.setText("%");
            jLabelPercMut.setAlignmentY(0.3F);
            javax.swing.GroupLayout panelOtimizadorLayout = new
javax.swing.GroupLayout(panelOtimizador);
            panelOtimizador.setLayout(panelOtimizadorLayout);
            panelOtimizadorLayout.setHorizontalGroup(
panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(panelOtimizadorLayout.createSequentialGroup().addGap(19, 19, 19)

.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING).addComponent(labelGeracao).addComponent(labelMutacao)
.addComponent(labelCruzamento).addComponent(labelIndividuo)
.addComponent(labelConvergencia)).addGap(11, 11, 11)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.TRAILING, false).addComponent(tFieldCruzamento,
javax.swing.GroupLayout.Alignment.LEADING, 0, 0, Short.MAX_VALUE)
.addComponent(tFieldMutacao, javax.swing.GroupLayout.Alignment.LEADING, 0, 0,
Short.MAX_VALUE).addComponent(tFieldGeracao,
javax.swing.GroupLayout.Alignment.LEADING, 0, 0, Short.MAX_VALUE)
.addComponent(tFieldIndividuo, javax.swing.GroupLayout.Alignment.LEADING, 0, 0,
Short.MAX_VALUE).addComponent(tFieldConvergencia,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING).addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelOtimizadorLayout.createSequentialGroup()
.addComponent(jLabelPercMut)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 77,
Short.MAX_VALUE)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.TRAILING, false).addComponent(jButtonLimpaAG,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addComponent(jButtonExecutaAG,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)).addGap(52, 52,
52))                .addGroup(panelOtimizadorLayout.createSequentialGroup()
.addComponent(jLabelPercCruza).addContainerGap())))
.addGroup(panelOtimizadorLayout.createSequentialGroup().addGap(117, 117, 117)
.addComponent(labelTitOtimizador).addContainerGap(150, Short.MAX_VALUE)));
            panelOtimizadorLayout.setVerticalGroup(
panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING).addGroup(panelOtimizadorLayout.createSequentialGroup().addContainerGap()
.addComponent(labelTitOtimizador).addGap(7, 7, 7)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE).addComponent(labelCruzamento)
.addComponent(tFieldCruzamento, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jLabelPercCruza))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE).addComponent(labelMutacao).addComponent(tFieldMutacao,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE).addComponent(jButtonExecutaAG)
.addComponent(jLabelPercMut))
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING).addGroup(panelOtimizadorLayout.createSequentialGroup()
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
```

```
nment.BASELINE).addComponent(labelGeracao)
.addComponent(tFieldGeracao, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE).addComponent(labelIndividuo)
.addComponent(tFieldIndividuo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelOtimizadorLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.BASELINE).addComponent(labelConvergencia)
.addComponent(tFieldConvergencia, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
.addGroup(panelOtimizadorLayout.createSequentialGroup().addGap(22, 22, 22)
.addComponent(jButtonLimpaAG))).addContainerGap(43, Short.MAX_VALUE)));
                jTabbedPaneProg.addTab("Otimizador", panelOtimizador);
                labelTitAgrup.setText("K-MEANS");
                labelQtdGrupo.setText("Quantidade de grupos:");
                labelMinimo.setText("M\u00ednimo");
                labelMaximo.setText("M\u00e1ximo");
                jButtonExecutaAgrup.setText("Executar");
                jButtonExecutaAgrup.addMouseListener(new
java.awt.event.MouseAdapter() {
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        jButtonExecutaAgrupMouseClicked(evt);
                    }
                });

                jRadioAgrupCal.setBackground(new java.awt.Color(191, 210, 228));
                jRadioAgrupCal.setSelected(true);
                jRadioAgrupCal.setText("Calinski e Harabasz");
```

```
jRadioAgrupCal.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
                jRadioAgrupCal.setMargin(new java.awt.Insets(0, 0, 0, 0));
                jRadioAgrupCal.addMouseListener(new java.awt.event.MouseAdapter() {
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        jRadioAgrupCalMouseClicked(evt);
                    }
                });

                jRadioAgrupPBM.setBackground(new java.awt.Color(191, 210, 228));
                jRadioAgrupPBM.setText("PBM");

jRadioAgrupPBM.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0,
0));
                jRadioAgrupPBM.setMargin(new java.awt.Insets(0, 0, 0, 0));
                jRadioAgrupPBM.addMouseListener(new java.awt.event.MouseAdapter()
{
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        jRadioAgrupPBMMouseClicked(evt);
                    }
                });

                javax.swing.GroupLayout panelAgrupamentoLayout = new
javax.swing.GroupLayout(panelAgrupamento);
                panelAgrupamento.setLayout(panelAgrupamentoLayout);
                panelAgrupamentoLayout.setHorizontalGroup(

panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING).addGroup(panelAgrupamentoLayout.createSequentialGroup()
.addGroup(panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
.addGroup(panelAgrupamentoLayout.createSequentialGroup().addGap(157, 157, 157)
.addComponent(labelTitAgrup))
.addGroup(panelAgrupamentoLayout.createSequentialGroup().addGap(56, 56, 56)
```

```
.addGroup(panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING).addComponent(labelQtdGrupo)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelAgrupamentoLayout.createSequentialGroup()
.addGroup(panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING).addComponent(labelMinimo).addComponent(labelMaximo))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 59,
Short.MAX_VALUE)
.addGroup(panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING, false).addComponent(choiceMaximo,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE).addGroup(panelAgrupamentoLayout.createSequentialGroup()
.addComponent(choiceMinimo, javax.swing.GroupLayout.DEFAULT_SIZE, 43,
Short.MAX_VALUE).addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED))))).addGroup(panelAgrupamentoLayout.createSequentialGroup()
.addComponent(jRadioAgrupCal)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED))
.addComponent(jRadioAgrupPBM)).addGap(85, 85, 85)))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(jButtonExecutaAgrup).addGap(27, 27, 27)) );
                panelAgrupamentoLayout.setVerticalGroup(
panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING).addGroup(panelAgrupamentoLayout.createSequentialGroup()
.addContainerGap().addComponent(labelTitAgrup).addGap(19, 19, 19)
.addComponent(labelQtdGrupo)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addGroup(panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING).addComponent(labelMinimo).addComponent(choiceMinimo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(panelAgrupamentoLayout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING).addComponent(labelMaximo).addComponent(choiceMaximo,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)).addGap(15, 15, 15)
.addComponent(jRadioAgrupCal).addGap(14, 14,
14).addComponent(jRadioAgrupPBM)              .addContainerGap(25,
Short.MAX_VALUE))
.addGroup(panelAgrupamentoLayout.createSequentialGroup().addGap(78, 78, 78)
.addComponent(jButtonExecutaAgrup).addContainerGap(101, Short.MAX_VALUE)));
                jTabbedPaneProg.addTab("Agrupamento", panelAgrupamento);
                jButtonIndiceOK.setText("OK");
                jButtonIndiceOK.addMouseListener(new java.awt.event.MouseAdapter() {
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        jButtonIndiceOKMouseClicked(evt);
                    }
                });

                jRadioCalHara.setSelected(true);
                jRadioCalHara.setText("Calinski e Harabasz");

jRadioCalHara.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
                jRadioCalHara.setMargin(new java.awt.Insets(0, 0, 0, 0));
                jRadioCalHara.addMouseListener(new java.awt.event.MouseAdapter() {
                    public void mouseClicked(java.awt.event.MouseEvent evt) {
                        jRadioCalHaraMouseClicked(evt);
                    }
                });

                jRadioCalHara.getAccessibleContext().setAccessibleParent(panelIndice);
                jRadioPBM.setText("PBM");
                jRadioPBM.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0,
0, 0));
```

```
            jRadioPBM.setMargin(new java.awt.Insets(0, 0, 0, 0));
            jRadioPBM.addMouseListener(new java.awt.event.MouseAdapter() {
               public void mouseClicked(java.awt.event.MouseEvent evt) {
                  jRadioPBMMouseClicked(evt);
               }
            });

            javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
            jPanel1.setLayout(jPanel1Layout);
            jPanel1Layout.setHorizontalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup().addContainerGap()
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING).addComponent(jRadioCalHara).addComponent(jRadioPBM))
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)));
            jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup().addContainerGap()
.addComponent(jRadioCalHara).addGap(29, 29, 29).addComponent(jRadioPBM)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)));
            javax.swing.GroupLayout panelIndiceLayout = new
javax.swing.GroupLayout(panelIndice);
            panelIndice.setLayout(panelIndiceLayout);
            panelIndiceLayout.setHorizontalGroup(
panelIndiceLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
               .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelIndiceLayout.createSequentialGroup().addGap(64, 64, 64)
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 60,
Short.MAX_VALUE).addComponent(jButtonIndiceOK).addGap(81, 81, 81)));
                panelIndiceLayout.setVerticalGroup(
panelIndiceLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                    .addGroup(panelIndiceLayout.createSequentialGroup()
.addGroup(panelIndiceLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING).addGroup(panelIndiceLayout.createSequentialGroup().addGap(78, 78,
78)
.addComponent(jButtonIndiceOK)).addGroup(panelIndiceLayout.createSequentialGrou
p()                    .addGap(56,
56,56).addComponent(jPanel1,javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,javax.swing.GroupLayout.PREFERRED_
SIZE))).addContainerGap(65, Short.MAX_VALUE))        );
                jTabbedPaneProg.addTab("\u00cdndices", panelIndice);
                textAreaAjuda.setEditable(false);
                textAreaAjuda.setText("          Essa janela tem como finalidade orientar os
usu\u00e1rios que\ndesejarem  submeter  suas  bases  de  dados a  uma
execu\u00e7\u00e3o\notimizada de um agrupamento n\u00e3o supervisionado.");

                javax.swing.GroupLayout panelAjudaLayout = new
javax.swing.GroupLayout(panelAjuda);
                panelAjuda.setLayout(panelAjudaLayout);
                panelAjudaLayout.setHorizontalGroup(
panelAjudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)          .addComponent(textAreaAjuda,
javax.swing.GroupLayout.PREFERRED_SIZE, 381,
javax.swing.GroupLayout.PREFERRED_SIZE));
                panelAjudaLayout.setVerticalGroup(
panelAjudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)          .addComponent(textAreaAjuda,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 202, Short.MAX_VALUE));
                jTabbedPaneProg.addTab("Ajuda", panelAjuda);
```

```
            javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
layout.setHorizontalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING).addGroup(layout.createSequentialGroup()
.addComponent(jTabbedPaneProg, javax.swing.GroupLayout.PREFERRED_SIZE,
386, javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)));
            layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()          .addComponent(jTabbedPaneProg,
javax.swing.GroupLayout.PREFERRED_SIZE, 230,
javax.swing.GroupLayout.PREFERRED_SIZE).addContainerGap()));
            pack();
        }
        private void jRadioAgrupPBMMouseClicked(java.awt.event.MouseEvent
evt) {
            jRadioAgrupCal.setSelected(false);
            jRadioAgrupPBM.setSelected(true);
        }
        private void jRadioAgrupCalMouseClicked(java.awt.event.MouseEvent evt)
{
            jRadioAgrupPBM.setSelected(false);
            jRadioAgrupCal.setSelected(true);
        }
        private void jRadioPBMMouseClicked(java.awt.event.MouseEvent evt) {
            jRadioCalHara.setSelected(false);
            jRadioPBM.setSelected(true);
        }
        private void jRadioCalHaraMouseClicked(java.awt.event.MouseEvent evt) {
            jRadioPBM.setSelected(false);
```

```
                jRadioCalHara.setSelected(true);
            }
            private void jButtonIndiceOKMouseClicked(java.awt.event.MouseEvent evt)
{
                String  auxString=textFieldArquivo.getText();
                int wsMinGrupo=0, wsMaxGrupo=0, wsColMax=0, wsAtributo=0, erro=0;
                if (auxString.equals(" ")) {
                   JOptionPane.showMessageDialog(null,"Arquivo deve ser
selecionado","ERRO",
                         JOptionPane.ERROR_MESSAGE);
                   erro=1;
                }

                if (erro==0) {
                   wsAtributo=Integer.parseInt(choiceAtributo.getSelectedItem());
                   if (jRadioCalHara.isSelected())
                      executaIndiceCalHar(vetCromossomo, auxString, 2, 30, wsAtributo);
                   else
                      executaIndicePBM(vetCromossomo, auxString, 2, 30, wsAtributo);
                }
            }
                private void
jButtonExecutaAgrupMouseClicked(java.awt.event.MouseEvent evt) {
                   String  auxString=textFieldArquivo.getText();
                   int wsMinGrupo=0, wsMaxGrupo=0, wsColMax=0, wsAtributo=0,
erro=0;
                   if (auxString.equals(" ")) {
                      JOptionPane.showMessageDialog(null,"Arquivo deve ser
selecionado","ERRO",
                            JOptionPane.ERROR_MESSAGE);
                      erro=1;
                   }
                   if (erro==0) {
                      wsMinGrupo=Integer.parseInt(choiceMinimo.getSelectedItem());
```

```
                wsMaxGrupo=Integer.parseInt(choiceMaximo.getSelectedItem());
                if (wsMinGrupo > wsMaxGrupo) {
                   JOptionPane.showMessageDialog(null,"Quantidade mínima de
grupos não pode ser\nsuperior à quantidade máxima","ERRO",
                         JOptionPane.ERROR_MESSAGE);
                   erro=1;
                }
             }
             if (erro==0) {
                wsAtributo=Integer.parseInt(choiceAtributo.getSelectedItem());
                    if (jRadioAgrupCal.isSelected())
                    executaAgrupamentoCal(vetCromossomo, auxString,
wsMinGrupo, wsMaxGrupo, wsAtributo);
                    else
                          executaAgrupamentoPBM(vetCromossomo, auxString,
wsMinGrupo, wsMaxGrupo, wsAtributo);
             }
          }

          private void executaAgrupamentoCal(int [] aptidaoMelhor, String arquivo,
int numMinCluster, int numMaxCluster, int colMax) {
             int i, j, nrCluster, linMax, colOriginal=colMax;
             double T, W, ist;
             double [][] atrib;
             double [][] tabCl;
             double [] medBase;
             String arquivoNovo=arquivo;
             CalcGer work   = new CalcGer();
             LerArquivo ler = new LerArquivo();
             linMax=ler.calcLinArq(arquivo, colMax);
             if (executouAG==1) {
                for (i=0, j=0; i<colMax; i++)
                   if (aptidaoMelhor[i]==9 || aptidaoMelhor[i]==1 )
                      j++;
```

```
                    colMax=j; //novo arquivo com atributos válidos -- alelos 1 ou 9
                } else {
                    aptidaoMelhor = new int[colMax];
                    for (i=0; i<colMax; i++)
                        aptidaoMelhor[i]=9;
                }
                atrib = new double[linMax][colMax+2];
                tabCl = new double[1600][colMax+2];
                medBase = new double[colMax];
                ler.trataArquivo(aptidaoMelhor, arquivo, atrib, medBase, linMax,
colOriginal);
                work.gMax=-90000.00;
                T=0;
                ist=11;
                for (i=arquivo.length(); i>=0 ; i--) {
                    if ((arquivo.charAt(i-1))=='\\') break;
                }
                arquivoNovo=arquivo.substring(0,(i));
                T=work.calculaDifGeral(atrib,medBase,linMax,colMax);
                for (nrCluster=numMinCluster; nrCluster <= numMaxCluster;
nrCluster++) {
                    W = 90000.00;
                    work.montaClusterInicial(atrib,tabCl,linMax,colMax,nrCluster);
                    W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
                    work.realocaObjetos(atrib,tabCl,linMax,colMax,nrCluster);
                    W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
                    if (work.erro == 0) break;
                    work.executaCal(arquivoNovo,atrib,linMax,colMax, W, T,
nrCluster,1);
                }
                String mensagemFinal="Numero de grupos.....: " + work.nrOtimoCl;
                mensagemFinal=mensagemFinal + "\nMenor erro calculado..: " +
work.melhorErro;
```

```
                mensagemFinal=mensagemFinal + "\nResultado da funcao...: " +
work.gMax;
                JOptionPane.showMessageDialog(null,mensagemFinal,"Resultado do
Agrupamento - Calinski e Harabasz",JOptionPane.INFORMATION_MESSAGE);


            }
            private void executaAgrupamentoPBM(int [] aptidaoMelhor, String arquivo,
int numMinCluster, int numMaxCluster, int colMax) {
                executouAG=0;
                int i, j, nrCluster, linMax, colOriginal=colMax;
                double T, W, ist;
                double [][] atrib;
                double [][] tabCl;
                double [] medBase;
                String arquivoNovo=arquivo;
                CalcGer work   = new CalcGer();
                LerArquivo ler = new LerArquivo();
                linMax=ler.calcLinArq(arquivo, colMax);
                if (executouAG==1) {
                   for (i=0, j=0; i<colMax; i++)
                      if (aptidaoMelhor[i]==9 || aptidaoMelhor[i]==1 )
                         j++;
                   colMax=j; //novo arquivo com atributos válidos -- alelos 1 ou 9
                } else {
                   aptidaoMelhor = new int[colMax];
                   for (i=0; i<colMax; i++)
                      aptidaoMelhor[i]=9;
                }
                atrib = new double[linMax][colMax+2];
                tabCl = new double[1600][colMax+2];
                medBase = new double[colMax];
                ler.trataArquivo(aptidaoMelhor, arquivo, atrib, medBase, linMax,
colOriginal);
                work.maxPBM=-90000.00;
```

```
            T=0;
            T=work.calculaDifGeral(atrib,medBase,linMax,colMax);
            work.E1=Math.sqrt(T);

            for (nrCluster=numMinCluster; nrCluster <= numMaxCluster;
nrCluster++) {
               W = 90000.00;
               work.montaClusterInicial(atrib,tabCl,linMax,colMax,nrCluster);
               W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
               work.realocaObjetos(atrib,tabCl,linMax,colMax,nrCluster);
               W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
               work.executaPBM(arquivoNovo,atrib,tabCl,linMax,colMax, W, T,
nrCluster, 1);
            }
            String mensagemFinal="Numero de grupos.....: " + work.nrOtimoCl;
               mensagemFinal=mensagemFinal + "\nMenor erro calculado..: " +
work.melhorErro;
               mensagemFinal=mensagemFinal + "\nResultado da funcao...: " +
work.gMax;
               JOptionPane.showMessageDialog(null,mensagemFinal,"Resultado do
Agrupamento - PBM",JOptionPane.INFORMATION_MESSAGE);
         }
         private void jButtonExecutaAGMouseClicked(java.awt.event.MouseEvent
evt) {
            String auxString, temp="";
            int auxNum, erro=0, wsCruzamento=0, wsMutacao=0, wsGeracao=0,
wsIndividuo=0, wsConvergencia=0;
            auxString=textFieldArquivo.getText();
            if (auxString.equals(" ")) {
               JOptionPane.showMessageDialog(null,"Arquivo deve ser
selecionado","ERRO", JOptionPane.ERROR_MESSAGE);
               erro=1;
            }
            if (erro==0) // testar campo cruzamento
```

```
            {
               try {
                  auxNum=Integer.parseInt(tFieldCruzamento.getText());
                  wsCruzamento=auxNum;
                  if ((wsCruzamento<=0) || (wsCruzamento > 100)) {
                     exibirMensagemErro("Campo Taxa de cruzamento deve ser maior
que zero e menor que cem");
                     erro=1;
                  }
               } catch (NumberFormatException e) {
                  exibirMensagemErro("Campo Taxa de cruzamento deve ser
preenchido com número        inteiro");
                  erro=1;
               }
            }
            if (erro==0) {
               try {
                  auxNum=Integer.parseInt(tFieldMutacao.getText());
                  wsMutacao=auxNum;
                  if ((wsMutacao<=0) || (wsMutacao > 100)) {
                     exibirMensagemErro("Campo Taxa de mutacao deve ser maior que
zero e menor que cem");
                     erro=1;
                  }
               } catch (NumberFormatException e) {
                  exibirMensagemErro("Campo Taxa de mutacao deve ser preenchido
com número inteiro");
                  erro=1;
               }
            }
            if (erro==0) {
               try {
                  auxNum=Integer.parseInt(tFieldGeracao.getText());
                  wsGeracao=auxNum;
```

```
                    if (wsMutacao<=0) {
                        exibirMensagemErro("Campo Numero de geracoes deve ser maior
que zero");
                        erro=1;
                    }
                } catch (NumberFormatException e) {
                    exibirMensagemErro("Campo Numero de geracoes deve ser
preenchido com número        inteiro");
                    erro=1;
                }
            }
            if (erro==0) {
                try {
                    auxNum=Integer.parseInt(tFieldIndividuo.getText());
                    wsIndividuo=auxNum;
                    if (wsIndividuo<=0) {
                        exibirMensagemErro("Campo Numero de individuos deve ser
maior que zero");
                        erro=1;
                    }
                } catch (NumberFormatException e) {
                    exibirMensagemErro("Campo Numero de individuos deve ser
preenchido com número        inteiro");
                    erro=1;
                }
            }

            if (erro==0) {
                try {
                    auxNum=Integer.parseInt(tFieldConvergencia.getText());
                    wsConvergencia=auxNum;
                    if (wsConvergencia<=0) {
                        exibirMensagemErro("Campo Convergencia deve ser maior que
zero");
```

```
                    erro=1;
                }
            } catch (NumberFormatException e) {
                exibirMensagemErro("Campo Convergencia deve ser preenchido com
número inteiro");
                erro=1;
            }
        }
        if (erro==0) {
            int auxAtributo=Integer.parseInt(choiceAtributo.getSelectedItem());
            vetCromossomo=new int[auxAtributo];
            for (int i=0; i<auxAtributo; i++) vetCromossomo[i]=9;
            erro=executaAG(vetCromossomo, auxString, auxAtributo,
wsConvergencia, wsGeracao, wsIndividuo, wsCruzamento, wsMutacao);
            if (erro==0) {
                for (int i=0; i<auxAtributo; i++)
                    temp = temp + vetCromossomo[i] + " ";
                JOptionPane.showMessageDialog(null,"Cromossomo: " +
temp,"Resultado do Algoritmo Genético",JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }
    private void exibirMensagemErro(String campo) {
JOptionPane.showMessageDialog(null,campo,"ERRO",JOptionPane.ERROR_MESSA
GE);
    }
    private void jButtonLimpaAGMouseClicked(java.awt.event.MouseEvent evt)
{
        tFieldConvergencia.setText("0");
        tFieldCruzamento.setText("0");
        tFieldGeracao.setText("0");
        tFieldIndividuo.setText("0");
        tFieldMutacao.setText("0");
        executouAG=0;
```

```
            int auxAtributo=Integer.parseInt(choiceAtributo.getSelectedItem());
            vetCromossomo=new int[auxAtributo];
            for (int i=0; i<auxAtributo; i++)
                vetCromossomo[i]=9;
        }
        private void jButtonBuscarMouseClicked(java.awt.event.MouseEvent evt) {
            JFileChooser fc= new JFileChooser();
            executouAG=0;
            if (evt.getSource() == jButtonBuscar) {
                int returnVal = fc.showOpenDialog(AJC.this);
                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File file = fc.getSelectedFile();
                    textFieldArquivo.setText(file.getPath());
                } else {
                    textFieldArquivo.setText(" ");
                }
            }
        }
        private  void executaIndicePBM(int [] aptidaoMelhor, String arquivo, int
numMinCluster, int numMaxCluster, int colMax) {
            executouAG=0;
            int i, j, nrCluster, linMax, colOriginal=colMax;
            double T, W, ist;
            double [][] atrib;
            double [][] tabCl;
            double [] medBase;
            String arquivoNovo=arquivo;

            CalcGer work   = new CalcGer();
            LerArquivo ler = new LerArquivo();
            linMax=ler.calcLinArq(arquivo, colMax);
            if (executouAG==1) {
                for (i=0, j=0; i<colMax; i++)
                    if (aptidaoMelhor[i]==9 || aptidaoMelhor[i]==1 )
```

```
                    j++;
                colMax=j; //novo arquivo com atributos válidos -- alelos 1 ou 9
            } else {
                aptidaoMelhor = new int[colMax];
                for (i=0; i<colMax; i++)
                    aptidaoMelhor[i]=9;
            }
            atrib = new double[linMax][colMax+2];
            tabCl = new double[1600][colMax+2];
            medBase = new double[colMax];
            ler.trataArquivo(aptidaoMelhor, arquivo, atrib, medBase, linMax,
colOriginal);
            work.maxPBM=-90000.00;
            T=0;
            T=work.calculaDifGeral(atrib,medBase,linMax,colMax);
            work.E1=Math.sqrt(T);
            for (nrCluster=numMinCluster; nrCluster <= numMaxCluster;
nrCluster++) {
                W = 90000.00;
                work.montaClusterInicial(atrib,tabCl,linMax,colMax,nrCluster);
                W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
                work.realocaObjetos(atrib,tabCl,linMax,colMax,nrCluster);
                W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
                work.executaPBM(arquivoNovo,atrib,tabCl,linMax,colMax, W, T,
nrCluster,0);

            }
            String mensagemFinal="Índice ótimo de grupos: " + work.nrOtimoCl;
        JOptionPane.showMessageDialog(null,mensagemFinal,"PBM",JOptionPane.IN
FORMATION_MESSAGE);    }
            private  void executaIndiceCalHar(int [] aptidaoMelhor, String arquivo, int
numMinCluster, int numMaxCluster, int colMax) {
            int executouAG=0;
            int i, j, nrCluster, linMax, colOriginal=colMax;
```

```
            double T, W, ist;
            double [][] atrib;
            double [][] tabCl;
            double [] medBase;
            String arquivoNovo=arquivo;
            CalcGer work   = new CalcGer();
            LerArquivo ler = new LerArquivo();
            linMax=ler.calcLinArq(arquivo, colMax);
            if (executouAG==1) {
               for (i=0, j=0; i<colMax; i++)
                  if (aptidaoMelhor[i]==9 || aptidaoMelhor[i]==1 )
                     j++;
               colMax=j; //novo arquivo com atributos válidos -- alelos 1 ou 9
            } else {
               aptidaoMelhor = new int[colMax];
               for (i=0; i<colMax; i++)
                  aptidaoMelhor[i]=9;
            }
            atrib = new double[linMax][colMax+2];
            tabCl = new double[1600][colMax+2];
            medBase = new double[colMax];
            ler.trataArquivo(aptidaoMelhor, arquivo, atrib, medBase, linMax,
colOriginal);

            work.gMax=-90000.00;
            T=0;
            ist=11;
            T=work.calculaDifGeral(atrib,medBase,linMax,colMax);
            for (nrCluster=numMinCluster; nrCluster <= numMaxCluster;
nrCluster++) {
               W = 90000.00;
               work.montaClusterInicial(atrib,tabCl,linMax,colMax,nrCluster);
               W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
               work.realocaObjetos(atrib,tabCl,linMax,colMax,nrCluster);
```

```
                W = work.calculaErro(atrib,tabCl,linMax,colMax,nrCluster);
                work.executaCal(arquivoNovo,atrib,linMax,colMax, W, T, nrCluster, 0);
            }
            String mensagemFinal="Índice ótimo de grupos: " + work.nrOtimoCl;
            JOptionPane.showMessageDialog(null,mensagemFinal,"Calinski e
Harabasz",JOptionPane.INFORMATION_MESSAGE);
        }
        private int executaAG(int [] aptidaoMelhor, String arquivo, int col, int
convergencia, int geracoes, int populacao, int cruzamento, int mutacao) {
            int atributo, tempInt, i, j, k, l, n, im;
            int lin, indAptidao;
            double aux, valorMin, mut, valAptidao=0,melhorValor;
            Double numDouble;
            mut=mutacao/100;
            Random rn = new Random();
            LerArquivo ler=new LerArquivo();
            lin=ler.calcLinArq(arquivo, col);
            executouAG=1; //flag para indicar para o agrupamento que o AG foi
executado
            atributo=col;
            int [] troca = new int[col];
            int [] elementoPop = new int[atributo];
            int [][] matPop = new int[populacao][atributo]; // matriz p/0 e 1 de cada
atributo de cada populacao
            int [][] popPai = new int [populacao][atributo];
            int [][] popMae = new int [populacao][atributo];
            double [] medBase;
            double [][] dados;
            medBase=new double[col];
            dados=new double[lin][col];
            Matriz objMat=new Matriz();
            DecimalFormat nf=new DecimalFormat("#0");
            double [] aptidaoPop = new double[populacao]; // o melhor em cada
populacao
```

```
            double [][] multipPop = new double[atributo][atributo]; // matriz quadrada
para cada elemento
            ler.trataArquivo(aptidaoMelhor,arquivo,dados,medBase,lin,col);
            if (ler.erro==1) {
               JOptionPane.showMessageDialog(null," " + col + " " + lin,"ERRO" +
ler.erro,JOptionPane.ERROR_MESSAGE);
               return 1;
            }
            // calcula matriz de covariancias com base nos atributos de entrada
            objMat.calcMatCovari(lin,col,dados,medBase);
            // zera valores abaixo da diag.principal e converte em valores absolutos
            for (i=0; i<col; i++) {
               for (j=0; j<col; j++)
                  if (i>j)
                     objMat.coVari[i][j]=0;
                  else
                     objMat.coVari[i][j]=Math.abs(objMat.coVari[i][j]);
            }
            // inicia a populacao dos atributos aleatoriamente com 0 ou 1

            for (i=0; i<populacao; i++) {
               for (j=0; j<atributo; j++)
                  matPop[i][j]=rn.nextInt(2); // 2 means 0 and 1
            }
            // inicia aptidoes com maior valor
            melhorValor=99999;
            for (i=0; i<populacao; i++) {
               aptidaoPop[i]=0;
            }
            for (i=0; i<atributo; i++)
               aptidaoMelhor[i]=9;

//*****************************************************************************
*********
```

```
            //***** calcula aptidao individual para cada elemento da primeira
populacao ****

//*****************************************************************************
*********
            for (i=0; i<populacao; i++) {
               for (j=0; j<atributo; j++)
                  elementoPop[j]=matPop[i][j];
               // gera matriz quadrada com linhas iguais para para ser multiplicada
               // ponto-a-ponto, pela matriz de correlacao
               for (j=0; j<atributo; j++) {
                  for (k=0; k<atributo; k++)
                     multipPop[k][j]=elementoPop[j];
               }
               // gera matriz transposta (primeira linha passa a ser uma unica coluna
               // que multiplica toda a matriz)
               for (j=0; j<atributo; j++) {
                  for (k=0; k<atributo; k++)
                     multipPop[j][k]=elementoPop[j]*multipPop[j][k];
               }
               // calcula aptidao

               for (j=0; j<atributo; j++) {
                  for (k=0; k<atributo; k++)
                     aptidaoPop[i]=aptidaoPop[i]+multipPop[k][j]*objMat.coVari[k][j];
               }
               // calcula aptidao com convergencia
               aux=0;
               for (k=0; k<atributo; k++) {
                  aux=aux+Math.abs(elementoPop[k]*elementoPop[k]);
               }
               aptidaoPop[i]=aptidaoPop[i]+(2*(Math.abs(aux-convergencia)));
               if (aptidaoPop[i]<melhorValor) {
                  melhorValor=aptidaoPop[i];
```

```
                for (j=0; j<atributo; j++)
                    aptidaoMelhor[j]=elementoPop[j];
                }
            } // fim do for com cálculo INICIAL da melhor aptidao

//*********************************************************************
*********
            //****************** inicio do processamento para cada geração
******************

//*********************************************************************
*********
            for (i=0; i<geracoes; i++) {
                // metade da populacao com melhor aptidao carregada para vetor pai
                for (j=0; j<(cruzamento*populacao/100); j++) // populacao / 2
                {
                    // localizar a melhor aptidao (menor valor)
                    indAptidao=0;
                    valAptidao=100;
                    for (k=0; k<populacao; k++)
                        if (aptidaoPop[k]<valAptidao) {
                        valAptidao=aptidaoPop[k];
                        indAptidao=k;
                        }
                    ////////////////// armazenar valores em popPai (50% melhores)
                    for (k=0; k<atributo; k++) {
                        popPai[j][k]=matPop[indAptidao][k];
                    }
                    aptidaoPop[indAptidao]=100; // este nao serah mais considerado
                    ////////////////// armazenar valores em popMae (50% aleatorios)
                    n=rn.nextInt((cruzamento*populacao/100)); // varia de 0 a
populacao/2
                    // outra metade da populacao ordenada escolhida aleatoriamente para o
vetor mae
```

```
    im=(n+j)%((cruzamento*populacao/100)); // populacao / 2
    for (k=0; k<atributo; k++) {
        popMae[im][k]=matPop[j][k];
    }
}
n=rn.nextInt(atributo);
// crossover & mutation
for (j=0; j<(cruzamento*populacao/100); j++) { // populacao / 2
    for (k=0; k<n; k++) {
        matPop[j][k]=popPai[j][k];
    }
    if (rn.nextDouble() < mut) { // valor para mutação. Teste aleatório
        tempInt=rn.nextInt(atributo);
        if (popPai[j][tempInt]==0) // inverter bit (mutation)
            matPop[j][tempInt]=1;
        else
            matPop[j][tempInt]=0;
    }
}

l=0; // indice para vetor mae
for ( ; j<(cruzamento*populacao/100); j++) { // populacao / 2
    for (k=0; k<atributo; k++) {
        matPop[j][k]=popMae[l][k];
    }
    if (rn.nextDouble() < mut) { // valor para mutação. Teste aleatório
        tempInt=rn.nextInt(atributo);
        if (popMae[l][tempInt]==0) // inverter bit (mutation)
            matPop[j][tempInt]=1;
        else
            matPop[j][tempInt]=0;
    }
}
```

```
//*************************************************************************
*********
                //***** calcula aptidao individual para cada elemento da populacao
****

//*************************************************************************
*********
                for (l=0; l<populacao; l++) {
                   for (j=0; j<atributo; j++)
                      elementoPop[j]=matPop[l][j];
                   // gera matriz quadrada com linhas iguais para para ser multiplicada
                   // ponto-a-ponto, pela matriz de correlacao
                   for (j=0; j<atributo; j++) {
                      for (k=0; k<atributo; k++)
                         multipPop[k][j]=elementoPop[j];
                   }
                   // gera matriz transposta (primeira linha passa a ser uma unica coluna
                   // que multiplica toda a matriz)
                   for (j=0; j<atributo; j++) {
                      for (k=0; k<atributo; k++)
                         multipPop[j][k]=elementoPop[j]*multipPop[j][k];
                   }
                   // calcula aptidao
                   aptidaoPop[l]=0;
                   for (j=0; j<atributo; j++) {
                      for (k=0; k<atributo; k++)

aptidaoPop[l]=aptidaoPop[l]+multipPop[k][j]*objMat.coVari[k][j];
                   }
                   // calcula aptidao com convergencia
                   aux=0;
                   for (k=0; k<atributo; k++) {
                      aux=aux+Math.abs(elementoPop[k]*elementoPop[k]);
```

```
            }
            aptidaoPop[l]=aptidaoPop[l]+(2*(Math.abs(aux-convergencia)));
            if (aptidaoPop[l]<melhorValor) {
               melhorValor=aptidaoPop[l];
               for (j=0; j<atributo; j++) {
                  aptidaoMelhor[j]=elementoPop[j];
               }
            }
         } // fim do for com cálculo para cada geração da melhor aptidao
      }
      return ler.erro;
   }
   /**
    * Programa principal, método main
    */
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new AJC().setVisible(true);
         }
      });
   }

   // Declaração de variáveis //
   private java.awt.Choice choiceAtributo;
   private java.awt.Choice choiceMaximo;
   private java.awt.Choice choiceMinimo;
   private javax.swing.JButton jButtonBuscar;
   private javax.swing.JButton jButtonExecutaAG;
   private javax.swing.JButton jButtonExecutaAgrup;
   private javax.swing.JButton jButtonIndiceOK;
   private javax.swing.JButton jButtonLimpaAG;
   private javax.swing.JLabel jLabelPercCruza;
   private javax.swing.JLabel jLabelPercMut;
```

```
    private javax.swing.JPanel jPanel1;
    private javax.swing.JRadioButton jRadioAgrupCal;
    private javax.swing.JRadioButton jRadioAgrupPBM;
    private javax.swing.JRadioButton jRadioCalHara;
    private javax.swing.JRadioButton jRadioPBM;
    private javax.swing.JTabbedPane jTabbedPaneProg;
    private java.awt.Label labelAtributo;
    private javax.swing.JLabel labelConvergencia;
    private javax.swing.JLabel labelCruzamento;
    private javax.swing.JLabel labelGeracao;
    private javax.swing.JLabel labelIndividuo;
    private javax.swing.JLabel labelMaximo;
    private javax.swing.JLabel labelMinimo;
    private javax.swing.JLabel labelMutacao;
    private java.awt.Label labelNome;
    private javax.swing.JLabel labelQtdGrupo;
    private javax.swing.JLabel labelTitAgrup;
    private javax.swing.JLabel labelTitOtimizador;
    private java.awt.Panel panelAgrupamento;
    private java.awt.Panel panelAjuda;
    private java.awt.Panel panelArquivo;
    private java.awt.Panel panelIndice;
    private java.awt.Panel panelOtimizador;
    private javax.swing.JTextField tFieldConvergencia;
    private javax.swing.JTextField tFieldCruzamento;
    private javax.swing.JTextField tFieldGeracao;
    private javax.swing.JTextField tFieldIndividuo;
    private javax.swing.JTextField tFieldMutacao;
    private java.awt.TextArea textAreaAjuda;
    private java.awt.TextField textFieldArquivo;
    // Fim da declaração de variáveis
}
```

CalcGer.java

```
import java.io.*;
import javax.swing.JOptionPane;
import java.lang.Math.*; // sqrt
public class CalcGer {
        double W, G, B, T, E1, maxPBM, gMax, s1, erro, desvPad, melhorErro;
        double[][]melhorAtrib;
        int ist, K, nrOtimoCl;
        public double calculaDifGeral (double [][] atrib, double [] medBase, int
linMax, int colMax)  {
                int i, j;
                double dif, lin, T=0;
                for (i=0; i<colMax; i++)
                        medBase[i]=medBase[i] / linMax;
                for (i=0; i<linMax; i++)
                        for (j=0; j<colMax; j++){
                                dif=atrib[i][j] - medBase[j];
                                T=T + (dif*dif);
                        }
                return T;
        }
        public void montaClusterInicial (double[][]atrib, double [][]tabCl, int
linMax, int colMax, int nrCluster)   {
                int nrObj, linCl, cont, i, lin;
                nrObj = (int) ((double)linMax / nrCluster - 0.5);
                lin = 0;
                for (cont=1; cont<=nrCluster; cont++)
                        for (i=0; i<nrObj; i++) {
                                atrib[lin][colMax+1]= (double) cont;
                                lin=lin+1;
                        }
                for (i=lin; i<linMax; i++)
                        atrib[i][colMax+1]=nrCluster;
                for (lin=0; lin<linMax; lin++) // limpar tabela
                        for (i=0; i<=colMax+1; i++)
```

```
                    tabCl[lin][i]=0;
            for (lin=0; lin<linMax; lin++){
                    linCl=(int) atrib[lin][colMax+1]-1;
                    for (i=0; i<colMax; i++)
                            tabCl[linCl][i]=tabCl[linCl][i]+atrib[lin][i];
                    tabCl[linCl][colMax]=tabCl[linCl][colMax]+1;
            }
            for (linCl=0; linCl<nrCluster; linCl++)
                    for (i=0; i<colMax; i++)

    tabCl[linCl][i]=tabCl[linCl][i]/tabCl[linCl][colMax];
        }
        public double calculaErro (double[][]atrib, double [][]tabCl, int linMax,
int colMax, int nrCluster)      {
            int linCl, i, lin;
            double dif, dist;
            erro=0;
            for (i=0; i<nrCluster; i++)
                    tabCl[i][colMax+1]=0;
            for (lin=0; lin<linMax; lin++){
                    dist=0;
                    linCl=(int) atrib[lin][colMax+1]-1;
                    for (i=0; i<colMax; i++){
                            dif  = atrib[lin][i]-tabCl[linCl][i];

                            dist = dist + (dif*dif);
                            erro = erro + (dif*dif);
                    }
                    atrib[lin][colMax]=dist;
                    tabCl[linCl][colMax+1]=tabCl[linCl][colMax+1] +
atrib[lin][colMax];
            }
            return erro;
        }
```

```
public void realocaObjetos (double[][]atrib, double [][]tabCl, int linMax,
int colMax, int nrCluster)      {
                int linCl, i, clusterOrigem, troca, lin;
                double dif=0, dist, nvNrObj, nrObjOrigem=0;
                troca=1;
                while (troca==1){
                        troca=0;
                        lin=0;
                        while (lin<linMax){
                                clusterOrigem=(int) atrib[lin][colMax+1];
                                for (linCl=0; linCl<nrCluster; linCl++){
                                        dist=0;
                                        for (i=0; i<colMax; i++){
                                                dif=atrib[lin][i] - tabCl[linCl][i];
                                                dist=dist + dif*dif;
                                        }

                                        if (dist<atrib[lin][colMax]){
                                                atrib[lin][colMax]=dist;
                                                atrib[lin][colMax+1]=linCl+1;
                                        }
                                }
                                if (atrib[lin][colMax+1]!=clusterOrigem){
                                        troca=1;
                                        linCl=clusterOrigem-1;
                                        nvNrObj=tabCl[linCl][colMax]-1;
                                        nrObjOrigem=nvNrObj;
                                        for (i=0; i<colMax; i++){
                                                tabCl[linCl][i]=tabCl[linCl][i] *
tabCl[linCl][colMax];

                                                tabCl[linCl][i]=tabCl[linCl][i] -
atrib[lin][i];

                                                if (nvNrObj!=0)
```

```
        tabCl[linCl][i]=tabCl[linCl][i]/nvNrObj;
                                        }
                                        tabCl[linCl][colMax]=nvNrObj;
                                        linCl=(int) atrib[lin][colMax+1]-1;
                                        nvNrObj=tabCl[linCl][colMax]+1;
                                        for (i=0; i<colMax; i++){
                                                tabCl[linCl][i] = tabCl[linCl][i] *
tabCl[linCl][colMax];
                                                tabCl[linCl][i] = tabCl[linCl][i] +
atrib[lin][i];
                                                tabCl[linCl][i] = tabCl[linCl][i] /
nvNrObj;
                                        }
                                        tabCl[linCl][colMax]=nvNrObj;
                                }
                                lin=lin+1;
                                if (nrObjOrigem == 1) break;
                        }
                }
        }
        public void executaCal (String arquivoNovo,double [][] atrib, int linMax,
int colMax, double erro, double T, int K, int gravar) {
                double B, G;

                B=T-erro;
                G=((linMax - K) * B) / ((K - 1) * erro);
                if (G > gMax) {
                        melhorAtrib=new double[linMax][colMax+2];
                        gMax=G;
                        nrOtimoCl=K;
                        melhorErro=erro;
                        for (int i=0; i<linMax; i++){
                                for (int j=0; j<=colMax+1; j++){
```

```
                                        melhorAtrib[i][j]=atrib[i][j];
                                }
                        }
                        if (gravar==1)
                                gravaArquivo(arquivoNovo+"Cluster"+K+".txt",
atrib, linMax, colMax);
                }
        }
        public void executaPBM (String arquivoNovo, double [][] atrib, double
[][] tabCl, int linMax, int colMax, double erro, double T, int nrCluster, int gravar) {
                int i, EK, p, s, lin, linCl;
                double PBM, DK, dif, dist, difAtrib, difAbs;
                EK=0;
                for (lin=0; lin<linMax; lin++){
                        dist=0;
                        linCl=(int) atrib[lin][colMax+1];
                        for(i=0; i<colMax; i++){
                                dif=atrib[lin][i]-tabCl[linCl][i];
                                dist+=(dif*dif);
                        }
                        EK+=Math.sqrt(dist);
                }
                dif=0;
                DK=0;
                for (p=0; p<nrCluster; p++)
                        for (s=p+1; s<nrCluster; s++){
                                for (i=0; i<colMax; i++){
                                        difAtrib=tabCl[p][i]-tabCl[s][i];
                                        dif+=(difAtrib*difAtrib);
                                }
                                difAbs=Math.sqrt(dif);  //'abs(dif);
                                if(difAbs>DK)
                                        DK=difAbs;
                        }
```

```
                    K=nrCluster;
                    PBM=(1/K)*(E1/EK)*DK;
                    PBM*=PBM;
                    if (PBM>maxPBM)    {
                            melhorAtrib=new double[linMax][colMax+2];
                            maxPBM=PBM;
                            nrOtimoCl=K;
                            for (i=0; i<linMax; i++){
                                    for (int j=0; j<=colMax+1; j++){
                                            melhorAtrib[i][j]=atrib[i][j];
                                    }
                            }
                            if (gravar==1)
                                    gravaArquivo(arquivoNovo+"Cluster"+K+".txt",
atrib, linMax, colMax);
                    }
            }
            public void gravaArquivo (String nomeArqui, double [][] atrib, int
linMax, int colMax)    {
                    BufferedWriter saida = null;
                    String concatena=" ";
                    try     {
                            saida = new BufferedWriter(new
FileWriter(nomeArqui));
                            for (int i=0; i<linMax; i++){
                                    concatena=" ";
                                    for (int j=0; j<=colMax+1; j++){
                                            if (j!=colMax) // retirar o erro da linha
                                                    concatena=concatena+"
"+atrib[i][j];
                                    }
                                    saida.write (concatena);
                                    saida.write(System.getProperty("line.separator"));
                            }
```

```
                    saida.close();
                }
                catch (IOException iO)
                {
                    JOptionPane.showMessageDialog(null,"Erro na gravação
do arquivo de saída!","ERRO",JOptionPane.ERROR_MESSAGE);
                }
            }
        }
```

Matriz.java

```
        import java.io.*;
        import java.lang.Math.*; // sqrt
        public class Matriz
        {
            static double [][] desvPad;
            static double [][] variancia;
            static double [][] coVari;
            public void calcMatCovari (int lin, int col, double [][] dados, double []
med) {
                int i, j, k;
                double soma=0;
                desvPad=new double[col][col];
                variancia=new double[col][col];
                coVari=new double[col][col];
                // calculo das medias de cada atributo
                for (i=0; i<col; i++){
                    med[i]=med[i]/lin;
                }
                // calculo de desvio padrao entre os atributos componentes
                for (i=0; i<lin; i++){
                    for (j=0; j<col; j++){
                        for (k=0; k<col; k++)  {
```

```
                desvPad[k][j]=desvPad[k][j]+(dados[i][j]*dados[i][k]);
                                }
                        }
                }
                for (i=0; i<col; i++){
                        for (j=0; j<col; j++){
                                desvPad[i][j]=desvPad[i][j]/lin;
                        }
                }
                // calculo de variancia entre os atributos componentes
                for (i=0; i<col; i++){
                        for (j=0; j<col; j++){
                                variancia[i][j]=desvPad[i][j]-(med[i]*med[j]);
                        }
                }
                // calculo de covariancia entre os atributos componentes
                        for (j=0; j<col; j++){
                                for (k=0; k<col; k++)  {
                coVari[j][k]=variancia[j][k]/Math.sqrt(variancia[j][j]*variancia[k][k]);
                                }
                        }
                }
        }
```

LerArquivo.java

```
        import java.io.*;
        import java.util.Scanner;
        import javax.swing.*;
        public class LerArquivo{
                public int erro;
                public void trataArquivo (int[]aptidaoMelhor, String arq,
double[][]dados, double[]medBase,int lin, int col)    {
                        double lido = 0;
                        int i=0, j=0, soma=0;
```

```
                String aux=" ";
                erro=0;
                for (i=0; i<col; i++)    {
                        if (aptidaoMelhor[i]==1 || aptidaoMelhor[i]==9)      {
                                medBase[soma]=0;
                                soma+=1;
                        }
                }
                try      {
                        Scanner ler = new Scanner (new File (arq));
                        for (i=0; i<lin; i++)     {
                                lido=0;
                                soma=0;
                                for (j=0; j<col; j++)    {
                                        if (ler.hasNext())        {
                                                aux = ler.next();
                                                lido = Double.parseDouble(aux);
                                                if (aptidaoMelhor[j]==1 ||
aptidaoMelhor[j]==9){
                                                        dados[i][soma]=lido;

        medBase[soma]=medBase[soma] + lido;
                                                        soma++;
                                                }
                                        }
                                }
                        }
                }
                catch (Exception e)     {
                        JOptionPane.showMessageDialog(null,"Erro na leitura do
arquivo. Pode ser algum valor faltante ou
vírgula","ERRO",JOptionPane.ERROR_MESSAGE);;
                        erro = 1;
                }
```

```
        }
        public int calcLinArq (String arq, int colMax)        {
                int i=1, j=0; double lido=0;
                String aux=" ";
                erro=0;
                try {
                        Scanner ler = new Scanner (new File (arq));
                        for (; ; i++)     {
                                for (j=0; j<colMax; j++)        {
                                        if (ler.hasNext())        {
                                                aux=ler.next();
                                                lido=Double.parseDouble(aux);
                                        }

                                }
                                if (!(ler.hasNext()))
                                        {break;}
                        }
                }
                catch (Exception e) {
                        JOptionPane.showMessageDialog(null,"Erro na leitura do
arquivo. \nPode ser algum valor faltante ou
vírgula","ERRO",JOptionPane.ERROR_MESSAGE);;
                        erro = 1;
                }
                if (j%colMax!=0) {
                        JOptionPane.showMessageDialog(null,"Erro na leitura do
arquivo. \nHá campos faltantes ou caracteres não
reconhecidos","ERRO",JOptionPane.ERROR_MESSAGE);
                        erro = 1;
                }
                return i;
        }
    }
```