Metaheurísticas para o Problema de Clusterização Automática

Stênio Sã Rosário Furtado Soares

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação Aplicada e Automação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização e Inteligência Artificial.

Orientador: Luiz Satoru Ochi

Niterói, Julho de 2004.

Metaheurísticas para o Problema de Clusterização Automática

Stênio Sã Rosário Furtado Soares

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação Aplicada e Automação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização e Inteligência Artificial.

Aprovada por:

Prof. Luiz Satoru Ochi / IC-UFF (Presidente)

Profa. Maria Cristina Silva Boeres / IC-UFF

Profa. Nair Maria Maia de Abreu / UFRJ

Prof. Nelson Francisco Favilla Ebecken / UFRJ

Profa. Simone de Lima Martins / IC-UFF

Niterói, Julho de 2004.

À minha família, bem mais valioso que tenho.

Agradecimentos

Agradeço primeiramente a Deus, que não me faltou um só momento na minha longa estrada até aqui. Agradeço aos professores do IC, em especial aos professores Luiz Satoru, Izabel Cafezeiro e Carlos Martinhon, que, com o profissionalismo e a dedicação que somente os que respiram ciência têm, acabaram motivando-me a trabalhar pensando nos rumos que levam ao doutorado.

Agradeço a todas as pessoas que de alguma forma partilharam comigo esta conquista. E fica sempre o receio de elencar tais pessoas e cometer o irreparável erro de omitir alguém. Mas, sem usar qualquer método de ordenação ou lista de prioridades, vejamos: agradeço à minha mãe, Dona Madalena Furtado, exemplo de vida que sempre me norteou e que, a cada notíciário das 8:00 apega-se aos céus rogando pela minha segurança e pedindo que se abreviem os dias até meu regresso. E neste agradecimento a ela, estendo ao meu pai, senhor Helcias Soares, e a todos os meus onze irmãos e irmãs, cujos nomes não listarei para que estes agradecimentos não se transformem numa dissertação de mestrado à parte.

Agradeço à Inegla, companheira nas horas mais difíceis e que ao longo de toda a minha trajetória do mestrado soube estar ao meu lado, mesmo quando estava a centenas de quilômetros. Cada minuto de conversa ao telefone, cada e-mail e cada carta recebida, tiveram uma nobre importância nas páginas que se seguem neste trabalho. Obrigado, querida!

Àqueles que estavam no Piauí torcendo pelo meu sucesso nesta empreitada e que sempre souberam das dificuldades enfrentadas, como a D. Algeny (e Zé Maria), a

professora Hilda Mara, a Iné(Lia), Gessilene, Soraya, Cláutenis, Ariadna, Ângela (e Zé Romero), Adriana e Diana.

Agradeço à Audea, por ter sido mãe e pai com tamanha competência e compreensão. Sem o seu esmero nos cuidados com nossos filhos, eu não teria sido capaz de privá-los da minha presença por todo o tempo que durou este trabalho. Obrigado, Audea! Ser-lhe-ei sempre grato.

Eu não poderia deixar de agradecer às pessoas que foram a minha família enquanto eu residi nesta cidade. São elas: a Mirasselva (minha irmã), que sempre tirou-me dos sufocos financeiros; a Alessandra e o Ânderson (sobrinhos maravilhosos). Agraceço ainda à D. Rozeli, Margarida, Carlinhos e o João Pedro, que por tantas vezes seguraram a minha barra quando as coisas pareciam querer desabar.

Aos colegas de curso, meu agradecimento é coletivo. As individualidades ficam na convivência com cada um, como ao Eyder, tão bem denominado pela Cris de help on-line, à Cris, à Geiza, à Idalmis, à Adriana Bechara, ao grande Jacques (o fiel escudeiro de todos no laboratório), à Luciana Pessoa, à dupla dinâmica Viviane e Daniela, à Ádria e também à Renata, claro!

Os agradecimentos nunca estão completos se não o fazemos também àqueles que dão o apoio logístico às atividades dentro do programa. E, não por isso, mas pela gratuidade da pesteza a mim concedida inúmeras vezes desde os primeiros contatos para a inscrição, meus agradecimentos à Ângela e à Isabela. Agradeço ainda ao Carlinhos, pela esforço contínuo em ajudar pelo prazer de fazer o bem, e de forma bem feita.

Espero não ter sido extensivo nem sovino nas palavras de agradecimento. Mas permita-me um agradecimento especial, daqueles que confortam a alma e nos enchem o coração. Meu muito obrigado aos meus filhos, Samuel Sã e Daniel Sã, razão da minha vida e o porquê dos desafios por mim aceitos. Valeu, filhões! Vocês são a razão desta conquista e das próximas que haveremos de construir juntos.

Resumo da Tese apresentada à UFF como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação (M.Sc.)

Metaheurísticas para o Problema de Clusterização Automática

Stênio Sã Rosário Furtado Soares

Julho/2004

Orientador: Luiz Satoru Ochi
Programa de Pós-Graduação em Computação

O Problema de Clusterização Automática (PCA) é uma generalização do Problema de Clusterização (PC) onde o número de clusters a ser obtido de uma base de dados não é pré-definido, apresentando com isso, uma complexidade maior que o PC. Embora o Problema de Clusterização já tenha sido bastante explorado por pesquisadores de diversas áreas, como matemática, estatística e computação, a maioria dos trabalhos vistos na literatura abordam o caso particular onde o número de clusters é previamente fixado. Entretanto, em muitas aplicações reais, o número de clusters é uma variável que deve ser determinada pelo algoritmo. Neste trabalho, apresentamos um Algoritmo Evolutivo Construtivo e uma algoritmo usando conceitos de *Simulated Annealing* para o PCA. O desempenho dos algoritmos propostos foram analisados por dois caminhos. Inicialmente eles foram comparados com instâncias da literatura e outras gerados aleatoriamente. Os resultados mostram uma nítida superioridade dos algoritmos propostos em termos da qualidade das soluções geradas. Numa segunda análise, avaliamos a convergência dos algoritmos propostos para valores alvos sub-ótimos previamente definidos. Os resultados mostram que os algoritmos são robustos, apresentando um desempenho muito estável em diferentes simulações.

Abstract of Thesis presented to UFF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Metaheuristics to Automatic Clustering Problem

Stênio Sã Rosário Furtado Soares

July/2004

Advisors: Luiz Satoru Ochi
Department: Computer Science

The Automatic Clustering Problem (PCA) is an generalization of the Clustering Problem (PC) where the number of clusters to be gotten of a data set is not previously known as parameter of the algorithm, presenting with this, a superior complexity to the PC. Even so the Clustering Problem already has been sufficient explored by researchers of many areas, like mathematics, statistics and computation the majority of the works seen in the literature approach the case that the number of clusters are previously fixed by the user. However in many real aplications, the number of clusters is a variable that have to be determined by the algorithm. In this work we present an Construtive Evolutionary Algorithm and a algorithm using concepts of the Simulated Annealing to PCA. The performance of the considered algorithms had been analyzed by two ways. Initially they had been compared with instances of literature and others generated aleatoriamente. The results show a clear superiority of our algorithms in terms of the quality of the generated solutions. In one second analysis, we evaluate the convergence of the algorithms considered for defined sub-excellent white values previously fixed. The results show that the algorithms are robust, presenting a very steady performance in different simulations.

Palavras-chave

1. Clusterização Automática
2. Metaherísticas
3. Algoritmo Genético
4. Simulated Annealing

Glossário

AG	:	Algoritmo Genético (*Genetic Algorithm*);
AE	:	Algoritmo Evolutivo (*Evolutive Algorithm*);
PC	:	Problema de Clusterização (*Clustering Problem*);
PCA	:	Problema de Clusterização Automática (*Automatic Clustering Problem*);
SA	:	Simulated Annealing (*Simulated Annealing*);
GRASP	:	Greedy Randomized Adaptative Search Procedures;
TS	:	Busca Tabu (*Tabu Search*)

Sumário

Resumo **v**

Abstract **vi**

Glossário **viii**

1 Introdução **1**

2 O Problema de Clusterização **5**

2.1 Tipos de Dados em Problemas de Clusterização 10

2.2 Algumas Aplicações do Problema de Clusterização 24

2.3 Análise da Complexidade de Tempo 27

3 Classificação dos Métodos de Clusterização **29**

3.1 Métodos de Particionamento 30

3.2 Métodos Hierárquicos de Clusterização 33

3.3 Metaheurísticas Aplicadas ao Problema de Clusterização 36

3.4 Características Importantes em Algoritmos de Clusterização 37

4 As Abordagens Propostas para o Problema de Clusterização Au-

tomática **45**

4.1 Considerações sobre Algoritmos Genéticos 46

4.2 Considerações sobre *Simulated Annealing* 54

4.3 O AEC - Algoritmo Evolutivo Construtivo para o PCA 57

4.3.1 A Fase de Construção 57

4.3.2 O Módulo Evolutivo do AEC 60

4.3.3 A Reconexão por Caminhos do AEC 66

4.4 O Algoritmo SAPCA - *Simulated Annealing* para o PCA 67

4.5 O Algoritmo Genético CLUSTERING 72

5 Resultados Computacionais **75**

5.1 Resultados Quanto à Qualidade Média das Soluções 79

5.2 Resultados por Imagem das Instância no R^2 83

5.3 Avaliação Probabilística dos Resultados 105

6 Conclusões e Propostas de Trabalhos Futuros **122**

Referências Bibliográficas **124**

Lista de Figuras

2.1 Plotagem de dados: atributos Massa(kg) x Altura(cm) 7

2.2 Plotagem de dados: atributos Massa(kg) x Altura(cm) x Idade 8

2.3 Plotagem de dados: variáveis Massa(g) x Altura(cm) 13

2.4 Distância Euclidiana e Distância de Manhattan no espaço R^2 17

3.1 Modelos de Clusterização Hierárquica 34

3.2 Exemplo de dados com clusters de formato e densidade diferentes . . 39

3.3 Exemplo de dados com ruídos 41

4.1 Algoritmo Genético genérico 47

4.2 Esboço do Método da Roleta 50

4.3 *crossover* de um ponto 51

4.4 *crossover* de dois ponto 52

4.5 Algoritmo *Simulated Annealing* genérico 56

4.6 Pseudocódigo do AEC 64

4.7 Mutação por sobreposição de janelas 65

4.8 Pseudocódigo do SAPCA 70

4.9 Busca local *Not_ Window* 71

5.1 Resultado para a instância 350p5c: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 84

5.2 Resultado para a instância: 2000p11c (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 86

5.3 Resultado para a instância 157p: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 88

5.4 Resultado para a instância face: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 90

5.5 Resultado para a instância 2face: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 92

5.6 Resultado para a instância *3dens*: AEC, AEC-RC, SAPCA e CLUSTERING 93

5.7 Resultado para a instância Convdensity: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 95

5.8 Resultado para a instância Numbers: (a) AEC (b) AEC-RC 97

5.9 Resultado para a instância Numbers: (a) SAPCA (b) CLUSTERING 98

5.10 Resultado para a instância Numbers2: (a) AEC (b) AEC-RC 100

5.11 Resultado para a instância Numbers2: (a) SAPCA (b) CLUSTERING 102

5.12 Resultado para a instância Moreshapes: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING 104

5.13 Análise Probabilística para a instância RuspiniData 107

5.14 Análise Probabilística para a instância 200Data 109

5.15 Análise Probabilística para a instância IrisData 111

5.16 Análise Probabilística para a instância CancerData 113

5.17 Análise Probabilística para RuspiniData sem método de construção . 115

5.18 Análise Probabilística para a instância 200Data sem método de construção 117

5.19 Análise Probabilística para IrisData sem método de construção 119

5.20 Análise Probabilística para CancerData sem método de construção . . 120

Lista de Tabelas

2.1 Exemplo de dados com 4 atributos 7

2.2 Contingência para dados de tipo binários 18

5.1 Características das instâncias usadas 76

5.2 Distribuição dos pontos da instância 200Data 76

5.3 Descrição dos atributos da instância Wisconsin Breast Cancer Database 77

5.4 Desempenho médio: SAPCA, AEC-RC, AEC e CLUSTERING . . . 81

5.5 Resultado quanto ao tempo médio em segundos: SAPCA, AEC-RC, AEC e CLUSTERING 82

Capítulo 1

Introdução

O desenvolvimento de técnicas de agrupamento, conhecidas como métodos de clusterização, constitui uma área de pesquisa que vem sendo bastante explorada, com contribuições recebidas de pesquisadores de diversas áreas. Todas as pesquisas sobre métodos de agrupamento convergem para as limitações computacionais no que se refere a automação dos processos de agrupamento em termos de viabilidade das soluções encontradas e do tempo requerido para a obtenção de tais soluções. O desenvolvimento de algoritmos de clusterização configura-se, então, não mais como um problema que depende exclusivamente da natureza do estudo ao qual o agrupamento é utilizado, mas, além disso, depende do conhecimento de técnicas que possibilitem a construção de algoritmos que ofereçam soluções de boa qualidade em tempo hábil.

Em várias aplicações, o número k de grupos (denotado clusters) obtidos por um algoritmo deve ser estabelecido a priori . Nestes casos, o problema se configura da seguinte forma: dado um conjunto S de objetos, encontre os k clusters ($1 \leq k \leq |S|$) que otimizem uma função objetivo $f(.)$, que recebe como entrada uma partição de S e retorna um valor real ou inteiro associado à qualidade desta partição. Obviamente, devemos ter mecanismos para quantificar uma solução para o problema. Para tanto, busca-se minimizar a dissimilaridade entre os objetos pertencentes a um mesmo cluster e maximizar a dissimilaridade entre os objetos de clusters diferentes.

O Problema de Clusterização é objeto de pesquisa bastante explorado na área de

Estatística. Neste tipo de abordagem, baseado em conceitos de distribuição normal, variância, regressão, desvio etc, procuram-se determinar através de equações, os modelos presentes no conjunto de dados analisado. Busca-se, portanto, estabelecer hipóteses sobre a distribuição dos dados através de equações baseadas no teorema de Bayes [42]. Um entrave desta abordagem é que as técnicas usadas para se construir um modelo ou uma hipótese podem falhar conforme seja a distribuição dos dados de cada conjunto de entrada.

Um fator complicador para o problema de clusterização é que, em muitos casos reais, além de não se ter nenhuma idéia quanto à composição dos clusters existentes no conjunto de entrada S, nada se pode dizer quanto ao número k ideal de clusters de que S é formado, ou seja, não se pode determinar a piori o valor de k para um determinado conjunto de dados S. Nestes casos, o problema torna-se mais complexo, pois, além de encontrar uma boa clusterização, deve-se encontrar simultaneamente o número ideal de clusters. Esta classe do Problema de Clusterização é conhecida como **Problema de Clusterização Automática (PCA)**, que constitui o objeto de estudo deste trabalho.

Além do problema de encontrar o número ideal de clusters, um outro fator que contribui para a obtenção de resultados de baixa qualidade é o fato de que em muitas aplicações reais, os clusters presentes numa base de dados não apresentam formato esférico e geralmente têm densidades variadas, o que, para algumas aplicações pode não ser interessante. Entenda-se por formato esférico aqueles casos em que, dados três objetos quaisquer a, b e c de uma base de dados, estes objetos só são considerados pertencentes a um mesmo cluster se as similaridades entre cada par destes objetos forem relativamente próximas. Não sendo possível, por exemplo, termos a similaridade entre a e b muito inferior à similaridade entre b e c. Este aspecto é considerado um fator decisivo na escolha da abordagem adotada.

Já que a solução do problema de clusterização é uma das partições possíveis para o conjunto de entrada S, encontrar aquela partição que agrupa os objetos de forma que cada subconjunto seja a melhor representação (ou generalização) de todos os seus elementos é um problema combinatório e certamente pode desencorajar o uso

de algoritmos determinísticos exaustivos para se garantir a obtenção de uma solução ótima à medida que a cardinalidade de S cresce. Uma alternativa promissora para driblar esta dificuldade é o uso de heurísticas ou metaheurísticas, que são técnicas aproximadas de otimização combinatória que visam encontrar uma boa solução para problemas combinatórios em um tempo aceitável, ainda que não se tenha garantia de que a solução encontrada seja uma solução ótima.

Várias abordagens para o Problema de Clusterização fazem uso de metaheurísticas como Busca Tabu (*Tabu Search*) [5, 11, 40, 87, 90, 89, 110], *Greedy Randomized Adaptive Search Procedures* (*GRASP*) [82, 89, 97, 98], Algoritmos Evolutivos [8, 31, 43, 61, 73, 115, 116], *Simulated Annealing* [21, 90], Colônia de Formigas (*Ant Colony*) [106] e Redes Neurais [84].

Neste trabalho, apresentamos duas abordagens para solução do PCA que procuram minimizar o impacto negativo que o crescimento do conjunto de entrada provoca no tempo de processamento e na qualidade das soluções geradas. Para isso, leva-se em conta o fato de que, na maioria dos casos reais, o conjunto de objetos dado como entrada, se vistos como pontos no espaço R^p, onde p é o número de atributos de cada objeto, apresenta regiões mais densas, compostas por vários objetos, separadas por regiões de poucos (ou nenhum) objetos, que chamamos regiões esparsas. A primeira abordagem é um Algoritmo Evolutivo Construtivo que incorpora uma heurística de construção e uma busca local num Algoritmo Genético tradicional. A segunda abordagem é um algoritmo baseado em conceitos de *Simulated Annealing* que, usando a mesma heurística de construção empregada na abordagem evolutiva e dois procedimentos de busca local, obteve soluções de alta qualidade na maioria das instâncias testadas.

O restante deste trabalho está assim dividido: no capítulo 2 apresentamos o Problema de Clusterização, onde enfatizamos os tipos de dados manuseados em tal problema, a complexidade de tempo e algumas aplicações do mesmo. No capítulo 3, apresentamos uma classificação dos principais algoritmos propostos para solução do Problema de Clusterização e os fatores importantes no desenvolvimento de algoritmos para este problema. Os algoritmos propostos são mostrado no capítulo 4, onde

fazemos uma explanação geral sobre Algoritmos Genéticos - AG's e sobre os módulos propostos nas duas abordagens sugeridas. Os resultados computacionais são apresentados no capítulo 5, onde demonstramos a eficiência dos algoritmos propostos comparando-os a uma abordagem genética da literatura que apresenta-se como uma boa estratégia para solução do PCA. As conclusões e sugestões de trabalhos futuros são dadas no capítulo 6.

Capítulo 2

O Problema de Clusterização

Agrupar objetos é uma ação própria do ser humano e pode ser verificada claramente já nos primeiros anos de vida do indivíduo. Segundo Jean Piaget, um dos mais conceituados teóricos da área de aprendizagem infantil, a mente humana é dotada de estruturas cognitivas denominadas esquemas (*schemata*, plural de *schema*). Tais esquemas funcionam como fichas em arquivos de informação. Nos primeiros meses de vida, dispomos de poucos esquemas, os quais apresentam informações bastante generalizadas acerca do universo ao qual estamos inseridos. Através de processos de assimilação dos estímulos do ambiente e de acomodação dos esquemas, vamos ampliando a rede de informações registradas nestes esquemas, que nos permitirá diferenciar objetos do universo conhecido [72].

Ainda na teoria piagetiana, o que explica o fato de que uma criança vendo a imagem de um cavalo, ao ser questionada sobre do que se trata, ela responder tratar-se de um cachorro, é o fato de que naquele estágio de maturidade, o esquema de que a criança dispõe que mais se assemelha ao estímulo apresentado (a imagem do cavalo) é o esquema que a mesma tem de um cachorro. Com o passar do tempo, novos esquemas podem ser inseridos na mente e também é possível (e necessário) a adequação de esquemas já existentes.

Observa-se, portanto, que nossa habilidade de *diferenciar* objetos, e num processo inverso, de *agrupar* objetos, é iniciada logo nos primeiros anos de vida. Uma

condição necessária para que o indivíduo interaja com o ambiente é que o mesmo aprenda sobre o universo de que é composto este ambiente. Neste processo de aprendizagem, utilizamos os orgãos dos sentidos para assimilar as informações que registraremos nos esquemas. Através dos orgãos dos sentidos, os estímulos a que somos submetidos levam-nos a ampliar e/ou melhorar nossa rede de esquemas e amprimorar a capacidade de classificar e agrupar tais estímulos segundo as informações nesta rede.

Pelo sistema auditivo, diversos *tipos* de sons são assimilados, aos quais são associadas outras informações que podem ser absorvidas através de outros sistemas, como a visão, olfato ou o tato. O que nos permite, por exemplo, diferenciar vozes masculinas de femininas, silvos de aves de espécies diferentes, associar o som de uma canção ou o aroma de um perfume a uma determinada pessoa. Isto é possível, graças a associação que fazemos através das informações de que dispomos na rede de esquemas.

No entanto, é fácil constatar que nossos sistemas perceptivos não são suficientemente eficazes para efetuarmos qualquer tipo de classificação ou associação. Podemos, com relativa facilidade, verificar grupos de objetos ao avaliarmos suas características plotadas no espaço R^2 ou R^3. No entanto, se quisermos avaliar os objetos através da análise de mais de três características (também refereciadas na literatura como atributos, aspectos, propriedades ou tuplas), o sistema cérebro-visão torna-se insuficiente. Nenhuma ferramenta poderia apresentar a plotagem de forma inteligível.

Para exemplificar esta situação, considere as características: idade, altura, massa e número de filhos para 10 indivíduos, conforme a Tabela 2.1. A plotagem dos dados segundo apenas os atributos peso e altura é apresenta na Figura 2.1. Percebe-se claramente a formação de quatro grupos de individuos: grupo 1, composto por Ana, Augusto e José; grupo 2, contendo por Carolina, Marina, Lennon e Julian; o grupo 3, formado por Renata e Flávio; e finalmente, o grupo 4, composto apenas por Maria.

Ainda no exemplo citado, se resolvermos avaliar os indivíduos segundo as variáveis peso, altura e idade, uma outra configuração dos grupos será formada. Note

Nome	Massa (kg)	Altura (cm)	Idade	No. Dependentes
Ana	95	193	49	7
Augusto	94	197	52	5
Carolina	67	159	26	1
Flávio	28	145	12	0
José	87	202	23	2
Julian	65	165	25	2
Lennon	68	154	24	3
Maria	27	110	6	0
Marina	48	159	34	1
Renata	33	144	9	0

Tabela 2.1: Exemplo de dados com 4 atributos

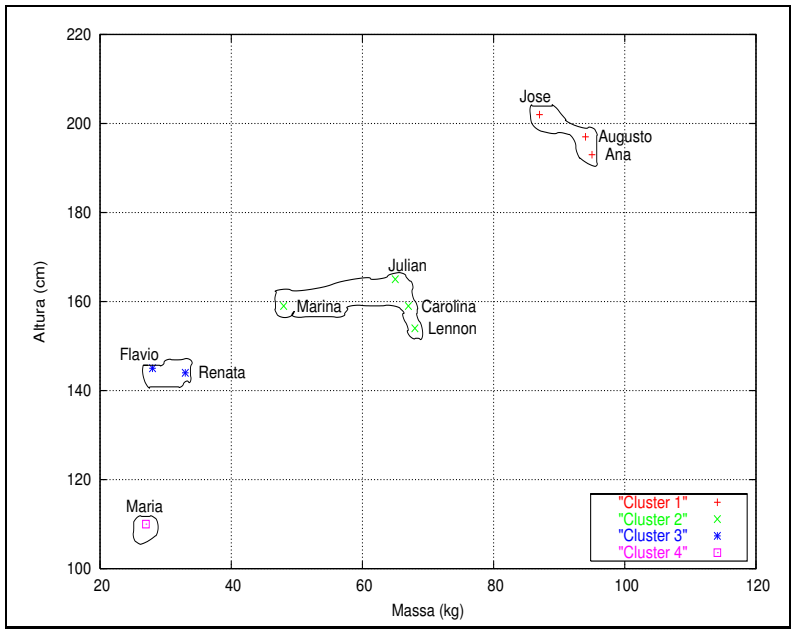

Figura 2.1: Plotagem de dados: atributos Massa(kg) x Altura(cm)

pela Figura 2.2, que os grupos agora são três, apresentando a seguinte configuração: grupo A, composto por Augusto , Ana e José; grupo B, composto por Marina, Julian, Carolina e Lennon; e finalmente o grupo C, formado por Maria, Flávio e Renata.

Naturalmente, na maioria das situações reais o número de variáveis necessárias

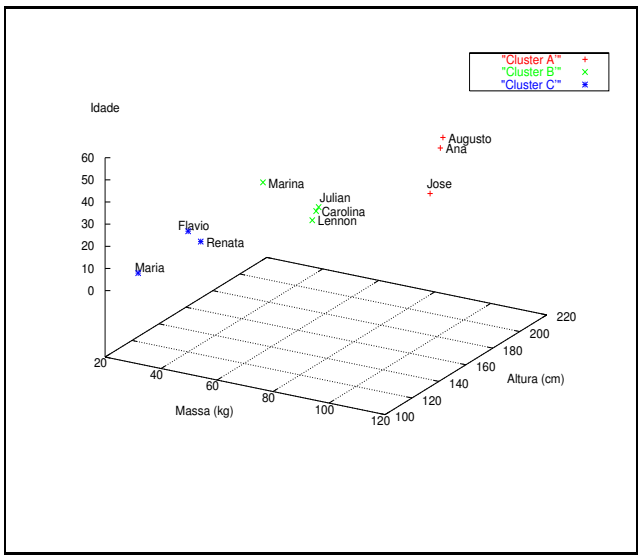

Figura 2.2: Plotagem de dados: atributos Massa(kg) x Altura(cm) x Idade

para caracterizar um dado objeto é superior a três. Imagine se nós tivéssemos que avaliar os grupos originados da Tabela 2.1, segundo todas as características apresentadas (massa, altura, idade e número de dependentes). Neste caso, como trata-se de uma avaliação no espaço R^4, não poderíamos dispor da imagem como recurso, o que implicaria na impossibilidade da apresentação dos dados de forma gráfica e, conseqüentemente, a inaplicabilidade do sistema cérebro-visão. Isto nos obrigaria a estudar detalhadamente as relações entre os objetos desta base de dados para podermos concluir algo sobre a formação e a composição dos grupos ali existentes.

Considere agora o conjunto universo formado por todos os animais e o problema de agrupá-los segundo algumas características. Dependendo das características levantadas, alguém poderá efetuar os agrupamentos de acordo com a espécie, ou de acordo com o tamanho, cor, número de patas, tipo de alimentação, tipo de reprodução etc. Para avaliarmos se os grupos apresentados estão bem caracterizados, deveremos ter em mente uma forma de como quantificar a qualidade do agrupamento obtido segundo os atributos ou características de tais objetos. Isto nos leva a concluir que o problema de agrupar objetos, denotado Problema de Clusterização - PC, requer uma função que expresse a qualidade da solução obtida.

Uma maneira de resolver este problema seria, de alguma forma, *avaliar* cada uma das partições do conjunto de objetos apresentado e verificar qual delas melhor define

os grupos (denominados clusters) ali formados, segundo as características dos objetos. Entretanto, para avaliarmos a qualidade de uma clusterização, faz-se necessário que definamos os eixos sobre os quais se fará a busca pela clusterização mais fiel aos dados avaliados, ou seja, precisamos definir quais as variáveis ou características a serem consideradas para a análise, o que não constitui uma tarefa simples. Além disso, podemos ver que o número de partições de um conjunto cresce exponencialmente com o número de elementos. Isto nos obriga a automatizar o processo de clusterização e procurar abreviar a obtenção da solução, o que tem feito surgir nos últimos anos inúmeros algoritmos com diversas abordagens diferentes.

A idéia de avaliar cada uma das partições do conjunto de entrada e verificar aquela que melhor caracteriza os grupos faz surgir uma pergunta: "o que é uma boa clusterização?". A resposta a esta questão obriga-nos a estabelecer medidas de qualidade de clusterização, que somente serão bem interpretadas quando definimos o que é clusterização.

Segundo [52], clusterização é o processo de tomar um conjunto de objetos físicos ou abstratos e agrupá-los em classes de objetos similares. Cada classe encontrada é denominada **cluster**. Portanto, um cluster é um conjunto de objetos similares entre si e ao mesmo tempo dissimilares a objetos pertecentes a outros clusters. A denominação *Clusterização* é mais comum na área de Reconhecimento de Padrões. Entretanto, em Inteligência Artificial denomina-se *Aprendizagem de Máquina* do tipo conceitual e em Processamento de Imagem, usa-se o termo *Segmentação*.

O Problema de Clusterização é, portanto, o problema de tomar um conjunto, na maioria das vezes finito, de objetos e agrupá-los em subconjuntos normalmente disjuntos, de forma que os elementos pertencentes a um mesmo subconjunto sejam similares entre si e, ao mesmo tempo, os elementos pertencentes a subconjuntos diferentes apresentem alta dissimilaridade. Em alguns casos, poderemos ter a formação de subconjuntos não disjuntos, como no caso de clusterização fuzzy.

A descrição do Problema de Clusterização em um espaço p-dimensional R^p, onde p é o número de atributos de um objeto, pode ser vista basicamente como sendo o problema de dividir um conjunto de n objetos em k ($k \leq n$) grupos objetivando

maximizar simultaneamente a similaridade entre objetos pertencentes a um mesmo cluster (intra-cluster) e a dissimilaridade entre os objetos pertencentes a clusters diferentes (inter-clusters). Formalmente, o problema pode ser descrito como: dado S um conjunto de n objetos $\{x_1, x_2, \ldots, x_n\}$, encontre o conjunto $C = \{C_1, \ldots, C_k\}$ de clusters onde a similaridade entre os objetos de um mesmo cluster C_i seja maximizada e a similaridade entre objetos de clusters diferentes seja minimizada, sujeito às seguintes condições:

$$C_i \neq \emptyset \text{ para } i = 1, \ldots, k \tag{2.1}$$

$$C_i \cap C_j = \emptyset \text{ para } i, j = 1, \ldots, k \text{ e } i \neq j \tag{2.2}$$

$$\bigcup_{i=1}^{k} C_i = S \tag{2.3}$$

A igualdade 2.3 poderá não ser verificada dependendo da abordagem adotada, ou seja, poderemos ter objetos da entrada que não foram associados a nenhum dos clusters encontrados. Tais objetos são chamados de *outliers* da base de dados, que, dependendo da aplicação e da abordagem adotada para resolver o problema, podem ser considerados, cada um, um cluster composto de um único objeto.

Da definição dada, podemos tentar responder a questão referente ao que vem a ser uma boa clusterização. Vê-se que um cluster é tão bem caracterizado quanto menor for o somatório das dissimilaridades entre os objetos que o formam e quanto maior for o somatório das dissimilaridades entre seus objetos e aqueles pertencentes a outros clusters. Desta forma, podemos dizer que uma boa clusterização é aquela que minimiza a dissimilaridade intra-cluster e maximiza a dissimilaridade inter-cluster.

2.1 Tipos de Dados em Problemas de Clusterização

Pela própria definição de Problema de Clusterização, podemos ver que há necessidade de calcularmos a dissimilaridade entre objetos da entrada, que é obtida mediante os atributos que definem cada um destes objetos. Como já mencionamos antes, podemos nos referir a objetos como pontos, e a seus atributos como variáveis. Por questão de padronização, usaremos de agora em diante os termos pontos

e variáveis (ou coordenadas), já que se aproximam mais da linguagem matemática. Quando for necessário nos referirmos ao elemento do mundo real ao invés da sua abstração, usaremos os termos objeto e atributo.

Em clusterização, os dados podem estar basicamente em dois formatos: categórico ou numérico. Dados no formato categórico podem, por exemplo, incluir para cada ponto (elemento) informações (atributos) como: cor preferida, estado civil, bairro onde reside, etc. Por outro lado, algumas bases de dados são compostas exclusivamente por dados numéricos, como ilustrado na Tabela 2.1, enquanto outras são compostas por dados categóricos e numéricos.

Entretanto, para qualquer tipo de dados, normalmente estes são inicialmente convertidos para valores numéricos antes de serem usados por um algoritmo. Desta forma, neste trabalho, vamos enfocar os problemas de clusterização com dados numéricos ou, no caso de dados categóricos, estes sempre serão convertidos para dados numéricos. Nestes casos, a dissimilaridade pode ser medida como a distância entre dois pontos. Todo algoritmo de clusterização opera sobre um conjunto $S = \{x_1, x_2, \ldots, x_n\}$ de pontos na forma de vetores de variáveis em um espaço p-dimensional, onde cada coordenada define uma variável do ponto.

Os tipos de dados mais comuns em problemas de clusterização são apresentados a seguir. Um estudo mais detalhado pode ser obtido em [52].

- **Dados do Tipo Intervalo-Escalar**

Dados numéricos do tipo intervalo-escalar são aqueles que definem quantidades que seguem uma escala linear, tais como altitude, idade, pressão, massa, força ou número de dependentes. Este tipo de dados permite-nos estabelecer vários intervalos dentro de uma escala que, se forem de tamanhos iguais, definem variações de quantidades iguais.

Antes de adentrarmos nas métricas usadas para cálculo de dissimilaridade, analisemos a forma como os dados (informações dos objetos) são armazenados na memória do computador. Considere um conjunto de n pontos com p coordenadas.

Estes dados podem ser armazenados na foma de uma matriz n x p, onde as n linhas correspondem aos pontos e as p colunas correspondem às coordenadas. Esta matriz pode ser vista como:

$$\text{n objetos} \overbrace{\begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1p} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{ip} \\ \vdots & & \vdots & & \vdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{np} \end{bmatrix}}^{p \text{ atributos}}$$

onde o valor x_{ij} representa o valor do j-ésimo atributo (coordenada) do i-ésimo objeto (ponto).

Voltando ao exemplo dado pela Tabela 2.1, uma forma de armazenar os dados através de uma matriz n x p tem como representação a matriz P apresentada a seguir, onde o número $n = 10$ de linhas indica o tamanho do conjunto de entrada e o número $p = 4$ de colunas indica o número de coordenadas ou propriedades do indivíduo (massa (kg), altura (cm), idade e número de dependentes).

$$\text{P} = \begin{bmatrix} 95 & 193 & 49 & 7 \\ 94 & 197 & 52 & 5 \\ 67 & 159 & 26 & 1 \\ 28 & 145 & 12 & 0 \\ 33 & 144 & 9 & 0 \\ 48 & 159 & 34 & 1 \\ 27 & 110 & 6 & 0 \\ 68 & 154 & 24 & 3 \\ 65 & 165 & 25 & 2 \\ 87 & 202 & 23 & 2 \end{bmatrix}$$

Neste caso, poderíamos ter uma coluna para armazenar o nome do indivíduo. Algumas vezes é interessante que se armazene o rótulo do objeto, entretanto o rótulo não é considerado uma variável no cálculo das dissimilaridades. Desta forma, optamos por referenciar um objeto apenas pelo índice da linha associado ao mesmo na matriz de entrada.

Um fator importante a ser considerado em dados do tipo intervalo-escalar é a unidade utilizada para cada atributo. No caso da Tabela 2.1, definimos como unidade de massa o kilograma (kg) e como unidade de altura o centímetro (cm). A plotagem apresentada na Figura 2.1 é influenciada pelo eixo referente à variável altura, que tem valores absolutos maiores que aqueles do eixo referente à variável massa. Se expressarmos a massa em gramas, teríamos a matriz de dados expressa por:

$$\text{P' } = \begin{bmatrix} 95000 & 193 & 49 & 7 \\ 94000 & 197 & 52 & 5 \\ 67000 & 159 & 26 & 1 \\ 28000 & 145 & 12 & 0 \\ 33000 & 144 & 9 & 0 \\ 48000 & 159 & 34 & 1 \\ 27000 & 110 & 6 & 0 \\ 68000 & 154 & 24 & 3 \\ 65000 & 165 & 25 & 2 \\ 87000 & 202 & 23 & 2 \end{bmatrix}$$

o que nos levaria a uma outra plotagem dos dados, apresentada na Figura 2.3.

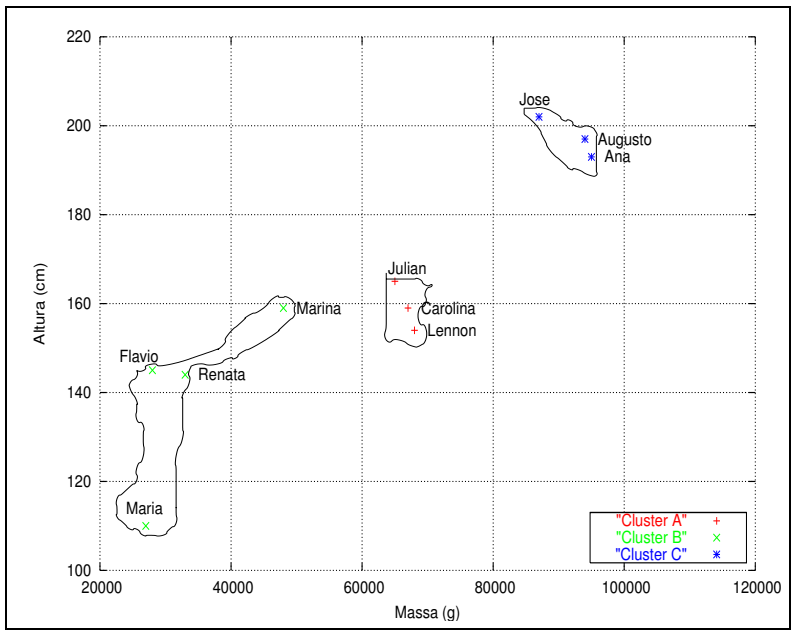

Figura 2.3: Plotagem de dados: variáveis Massa(g) x Altura(cm)

Como podemos ver, os clusters apresentados não são os mesmos obtidos quando a unidade de massa utilizada era o kilograma. Isto ocorre porque, neste caso, o eixo referente à coordenada altura passa a ter seus valores absolutos com menor

expressão no vetor de coordenadas que define o ponto, ou seja, o peso atribuído à variável massa é muito maior que o peso dado à variável altura, uma vez que os valores absolutos destas variáveis estão distorcidos pelas unidades utilizadas na entrada dos dados.

Afim de evitar que uma variável influencie na clusterização devido a unidade adotada, faz-se necessário padronizar os dados antes de efetuar a clusterização. Para isso, calculamos o desvio médio absoluto s_f definido na Equação 2.4.

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \cdots + |x_{nf} - m_f|) \tag{2.4}$$

onde $x_{1f}, \ldots, x_{nf}$ são as n medidas da variável f, e m_f é a média das medidas de f, isto é, $m_f = \frac{1}{n}(x_{1f} + x_{2f} + \cdots + x_{nf})$.

Assumindo s_f diferente de zero, padronizamos as medidas calculando, para todos os pontos, o *z-escore* de cada variável, dado por:

$$z_{if} = \frac{x_{if} - m_f}{s_f} \tag{2.5}$$

Podemos admitir que s_f é diferente de zero porque, caso contrário, a variável f teria o mesmo valor para todos os pontos, fato que nos permitiria removê-la da análise dos dados sem perda de informação. Entretanto, para efeito de execução de um algoritmo de clusterização, é necessário que se verifique esta condição antes de tentar calcular z_{if}. Desta forma, todas as p variáveis de cada um dos n pontos são substituídos pelo *z-escore* definido pela Equação 2.5 na matriz de entrada caso $s_f \neq 0$.

Geralmente, opta-se por utilizar o desvio médio absoluto e não o desvio padrão devido ao fato de que o desvio padrão é mais sensível a dados corrompidos, conhecidos como ruídos, uma vez que no cálculo do desvio médio absoluto não temos os desvios elevados ao quadrado, o que no caso de ruídos levaria a uma distorção considerável nos dados, fato que atribui um caráter diferencial à capacidade de um algoritmo manusear dados nestas condições. No Capítulo 3 é apresentada a relevância de um algoritmo de clusterização poder lidar com bases de dados contendo ruídos.

A padronização dos dados de entrada evita o problema decorrente da escolha de unidades a serem adotadas. Entretanto, em algumas aplicações é interessante que não se adote esta estratégia. Isto ocorre quando uma determinada variável deve ter peso maior devido a natureza da aplicação. Por exemplo, se os dados manuseados no problema dizem respeito a um estudo sobre obesidade, pode ser iteressante que se atribua um peso maior à variável massa.

Uma outra situação em que não é interessante padronizar os dados de entrada é quando todas as variáveis já estão na mesma unidade. Uma padronização dos dados de entrada, neste caso, acabaria por distorcer as informações ali contidas, uma vez que a própria unidade já define uma relação entre as variáveis. Conforme já mencionamos, um estudo detalhado dos efeitos de padronização de dados em Problemas de Clusterização pode ser obtido em [67] e [52].

Até aqui, tratamos os dados de entrada como uma matriz n x p. Entretanto, uma outra forma de armazenamento de dados para efeito de clusterização é através de matrizes n x n, onde n é o número de pontos do conjunto de entrada, e o elemento x_{ij} da matriz indica a dissimilaridade entre o i-ésimo e j-ésimo pontos. Para tanto, faz-se necessário que se estabeleça qual a métrica usada para se obter as dissimilaridades. A métrica mais utilizada é a *distância euclidiana*, definida como: dados dois pontos $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ e $x_j = (x_{j1}, x_{j2}, \ldots, x_{jp})$,

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2} \tag{2.6}$$

Quando os dados estiverem padronizados, substituímos x por z na Equação 2.6, onde z é o *z-score* das variáveis x calculado pela Equação 2.5. A Equação 2.6 corresponde à menor distância geométrica entre os pontos x_i e x_j. No exemplo apresentado na Tabela 2.1, a matriz de dissimilaridade obtida usando a Equação 2.6 será a matriz D dada a seguir.

Note que a matriz D, por ser simétrica, pode ser substituída por uma matriz triangular, diminuindo o consumo de memória e melhorando a complexidade de espaço. Entretanto, em muitas aplicações, a métrica utilizada pode não ser simétrica. Se, por exemplo, estivermos usando o custo que é sair de uma cidade para outra como

o tempo gasto entre duas cidades A e B, é bem possível termos custos diferentes conforme seja o sentido adotado para o trajeto. Nestes casos, é necessário que toda a matriz seja armazenada, não apenas uma de suas triagulares (superior ou inferior).

$$D=\begin{bmatrix} 0.00 & 30.77 & 7.35 & 61.66 & 12.81 & 33.33 & 30.89 & 17.15 & 11.45 & 28.98 \\ 30.77 & 0.00 & 28.62 & 38.79 & 42.25 & 63.21 & 60.95 & 40.16 & 38.60 & 30.12 \\ 7.35 & 28.62 & 0.00 & 60.89 & 14.90 & 35.71 & 34.23 & 15.10 & 10.44 & 23.87 \\ 61.66 & 38.79 & 60.89 & 0.00 & 70.36 & 94.35 & 91.32 & 67.26 & 68.52 & 52.74 \\ 12.81 & 42.25 & 14.90 & 70.36 & 0.00 & 25.00 & 23.45 & 11.22 & 6.40 & 31.69 \\ 33.33 & 63.21 & 35.71 & 94.35 & 25.00 & 0.00 & 5.92 & 32.88 & 27.96 & 54.32 \\ 30.89 & 60.95 & 34.23 & 91.32 & 23.45 & 5.92 & 0.00 & 32.80 & 27.22 & 54.17 \\ 17.15 & 40.16 & 15.10 & 67.26 & 11.22 & 32.88 & 32.80 & 0.00 & 8.19 & 22.32 \\ 11.45 & 38.60 & 10.44 & 68.52 & 6.40 & 27.96 & 27.22 & 8.19 & 0.00 & 27.04 \\ 28.98 & 30.12 & 23.87 & 52.74 & 31.69 & 54.32 & 54.17 & 22.32 & 27.04 & 0.00 \end{bmatrix}$$

Matrizes n x p são referenciadas na literatura como matrizes *two-mode*, já que as linhas se referem a uma entidade (os pontos) e as colunas a outra (as variáveis). Por outro lado, matrizes n x n são ditas *one-mode*, visto que linhas e colunas se referem a uma mesma entidade [117].

Uma outra métrica utilizada em algumas aplicações é a *distância de Manhattan*, definida pela Equação 2.7:

$$d(i,j) = (|x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|) \tag{2.7}$$

Esta métrica mede a distância entre dois pontos x_i e x_j de forma a percorrer a trajetória definida pelas coordenadas dos mesmos. No caso do espaço R^2, o triângulo retângulo definido pelas coordenadas x e y e os respectivos eixos tem como menor distância a hipotenusa, que é dada pela Equação 2.6, como pode ser visto na Figura 2.4. A *distância de Manhattan* definida pela Equação 2.7 é dada pela soma dos catetos deste triângulo. Algumas aplicações exigem o uso da *distância de Manhattan*, como no cálculo de caminho mais curto entre pontos de uma rede em que os nós encontram-se dispostos em logradouros divididos em quadras. Um algoritmo de clusterização que adota esta métrica é apresentado em [28].

Vale ressaltar que tanto a *distância euclidiana* como a *distância de Manhattan* gozam das seguintes propriedades:

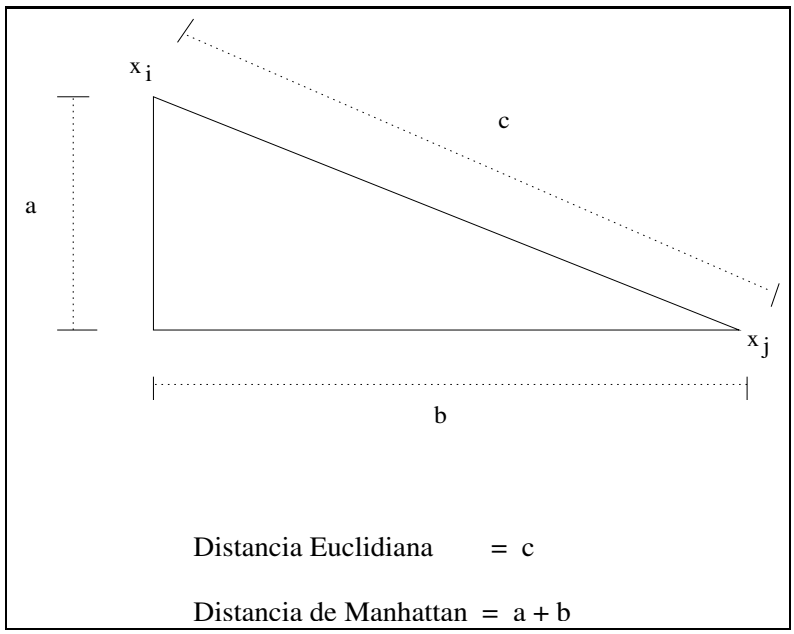

Figura 2.4: Distância Euclidiana e Distância de Manhattan no espaço R^2

P1: $d(i,j) \geq 0$;

P2: $d(i,i) = 0$;

P3: $d(i,j) = d(j,i)$;

P4: $d(i,j) \leq d(i,k) + d(k,j)$; para todo ponto i, j e k.

Podemos citar ainda, a métrica conhecida como *distância de Minkowski*, que é uma generalização das equações 2.6 e 2.7. Sua expressão é definida como:

$$d(i,j) = \sqrt[q]{(x_{i1} - x_{j1})^q + (x_{i2} - x_{j2})^q + \cdots + (x_{ip} - x_{jp})^q} \tag{2.8}$$

com $q \geq 1$ e $q \in R$. No caso da Equação 2.6, q é igual a 2, e no caso da Equação 2.7, q é igual a 1.

- **Dados do Tipo Binário**

Variáveis do tipo binário são aquelas que podem assumir apenas dois estados. Em problemas de clusterização que envolvem informações de pessoas, pode-se ter variáveis que definem o sexo, se a pessoa é fumante ou não-fumante, se já teve uma doença ou não etc. Para este tipo de dado, podemos definir 0 para uma situação e 1 para a outra.

		Objeto j		
		1	0	Soma
Objeto i	1	q	r	q + r
	0	s	t	s + t
	Soma	q + s	r + t	p

Tabela 2.2: Contingência para dados de tipo binários

Para computar a dissimilaridade de pontos que apresentam dados deste tipo, não podemos simplesmente computar as métricas como se estes dados refletissem quantidades, tal como fazemos com dados do tipo intervalo-escalar. Para tanto, se todas as variáveis binárias têm o mesmo peso, construímos uma matriz de dissimilaridade com base na Tabela 2.2, conhecida como tabela de contingência para dados binários, onde q é o número de variáveis que têm valor igual a 1 para os dois pontos i e j; r é o número de variáveis que têm valor igual a 1 para o ponto i e igual a 0 para o ponto j; s é o número de variáveis que têm valor igual a 0 para o ponto i e igual a 1 para o ponto j; e t é o número de variáveis que têm valor igual a 0 para ambos os pontos i e j. O número total de variáveis é $p = q + r + s + t$.

Quando os valores 0 ou 1 são igualmente importantes, dizemos tratar-se de uma variável binária *simétrica*, e a similaridade entre variáveis simétricas é chamada ***"similaridade invariante"***. Para este tipo de variável, o coeficiente de dissimilaridade entre dois pontos i e j mais utilizado é o *simple matching coeficient* e expresso pela Equação 2.9.

$$d(i,j) = \frac{r+s}{q+r+s+t} \tag{2.9}$$

Quando os valores 0 ou 1 têm importâncias diferentes para a aplicação ou pela natureza da variável que a mesma encerra, dizemos tratar-se de uma variável binária assimétrica. Seria, por exemplo, uma variável para identificar se um paciente é soro-positivo ou não para uma dada doença. A similaridade baseada em variáveis com esta característica é denominada ***similaridade variante*** e o coeficiente de dissimilaridade mais comumente usado é o *coeficiente de Jaccard* [52], dado pela Equação 2.10.

$$d(i,j) = \frac{r+s}{q+r+s} \tag{2.10}$$

- **Dados do Tipo Categórico**

Variáveis do tipo categórico são aquelas variáveis que definem atributos não escalares e que podem assumir vários estados. Como exemplo, podemos citar cor dos cabelos, em que o domínio é formado por mais de dois valores possíveis como: castanho, preto, ruivo e loiro. Neste caso específico, o domínio apresenta quatro estados possíveis para uma variável.

Em geral, denota-se por M o número de estados possíveis de uma variável do tipo categórico. Quando $M = 2$, temos o caso de variável do tipo binário. A cada um dos M estados, atribuímos um número entre 1 e M, de tal forma que cada estado seja rotulado por um número neste intervalo.

Convém observar que não há diferença quanto ao peso diferenciado de um estado sobre outro conforme seja o número associado ao mesmo. Assim, no caso do nosso exemplo, poderíamos ter o estado 1 associado a cabelo castanho, o estado 2 referente a cabelo preto, o estado 3 a cabelo ruivo e, finalmente, o estado 4 associado a cabelo loiro. Como o valor absoluto atribuído ao estado não indica intensidade, o estado 4 não é mais influente que os demais estados por ser numericamente maior.

Podemos converter uma variável categórica em M variáveis binárias. Entretanto, esta estratégia só é encorajadora para valores de M próximos de 2. Por exemplo, se uma variável categórica x referente à nacionalidade de uma pessoa tiver um domínio formado por 20 países, para convertermos esta variável em variáveis binárias, precisaríamos de pelo menos 20 variáveis binárias para registrar a nacionalidade do indivíduo, das quais apenas uma estaria no estado 1 e as dezenove restantes estariam no estado 0. Isto definitivamente não é iteressante em termos de desempenho computacional.

A maneira mais comumente empregada para medir a similaridade ou dissimilaridade entre dois pontos i e j que são caracterizados por variáveis categóricas é através do ***simple matching*** [109] *apud* [52]. Para o cálculo da dissimilaridade, o *simple matching* é dado por:

$$d(i,j) = \frac{p-u}{p} \tag{2.11}$$

onde u é o número de variáveis para as quais i e j apresentam o mesmo estado, e p é o número total de variáveis.

- **Dados do Tipo Ordinal**

Variáveis do tipo ordinal consistem de dados categóricos cujos estados são ordenados em uma sequência previamente definida. Desta forma, os rótulos dos estados (que variam de 1 até M) não são atribuídos aleatoriamente. Este tipo de variável é muitas vezes empregada para definir aceitação acerca de algum domínio. Por exemplo, pode-se ter uma variável cujo estado 1 seja associado à opção péssimo; o estado 2 associado a regular; o estado 3 associado a bom; e o estado 4 associado a ótimo.

Repare que há que se levar em conta não somente o número de variáveis com o mesmo estado, mas também a distância entre estes estados. Se para uma variável deste tipo, o ponto i apresenta o estado 4, o ponto j apresenta o estado 3 e um terceiro ponto k apresenta o estado 1, é razoável supor que o ponto i seja mais similar ao ponto j que ao ponto k.

É fácil verificar que é possível converter variáveis do tipo intervalo-escalar em variáveis do tipo categórico. Para tanto, basta que se estabeleçam faixas de valores dentro da escala adotada e atribuir um rótulo conforme as faixas estabelecidas. Um exemplo comum desta transformação é especificação de faixas etárias, onde define-se cada faixa pelos limites inferior e superior e em seguida atribui-se um rótulo (criança, adolescente, jovem, adulto e idoso).

O cálculo da dissimilaridade envolvendo variáveis do tipo ordinal é similar ao que se usa no caso de variáveis do tipo intervalo-escalar. Entretanto, é importante que os valores assumidos por tais variáveis estejam no intervalo $[0, 1]$, uma vez que fora estabelecida previamente uma ordem de precedência para tais variáveis. Para tanto, considere um conjunto S de n pontos que apresentam uma coordenada associada a uma variável f do tipo ordinal, onde o valor de f para o i-ésimo ponto é x_{if}. A variável f tem M_f estados r_{if}, com $r_{if} \in \{1, 2, \ldots, M_f\}$.

Para mapearmos a faixa de cada variável f no intervalo $[0, 1]$, substituímos todos

os estados r_{if} do i-ésimo ponto na f-ésima variável por:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{2.12}$$

Após esta substituição para todos os pontos, podemos utilizar a Equação 2.6 para o cálculo da dissimilaridade, tal qual fazemos para variáveis do tipo intervalo-escalar.

- **Dados do Tipo Escalar Proporcional**

Variáveis do tipo intervalo-escalar não são adequadas para aquelas situações em que, para uma dada grandeza, intervalos de tamanhos numericamente iguais, não definem variações iguais. Nestes casos, a variação é não linear e exige uma outra forma de manuseio. Este tipo de variável é muitas vezes utilizado para registrar crescimento populacional ao longo de um tempo t de estudo.

Geralmente, o cálculo das dissimilaridade para este tipo de variáveis é regido por equações da forma Ae^{Bt} ou Ae^{-Bt}, onde A e B são constantes positivas e t é o tempo de observação das variações das medidas. Para se calcular a dissimilaridade entre dois pontos x_i e x_j, onde x_{if} e x_{jf} são os valores da variável f do tipo escalar proporcional para x_i e x_j respectivamente, existem três alternativas que podem ser adotadas [52]:

1) Tratar as variáveis como sendo do tipo intervalo-escalar, o que pode gerar resultados distorcidos dada a natureza do problema;

2) Aplicar tranformação logaritmica sobre a variável f de tal forma que x_{if} seja substituído por $y_{if} = \log(x_{if})$, para em seguida utilizar a mesma equação usada para a dissimilaridade em dados do tipo intervalo-escalar (Equação 2.6);

3) Considerar f como variável do tipo ordinal e tratar cada um dos ranques como variáveis do tipo intervalo-escalar.

Destas três estratégias, as duas últimas são as mais utilizadas [52].

- **Dados de Tipos Mistos**

Como observado nos tipos apresentados até aqui, o cálculo das dissimilaridades entre pontos se dá de acordo com o tipo de dado que caracteriza tais pontos. Entretanto, em aplicações do mundo real, é muito comum termos bases de dados cujos atributos dos objetos são expressos por variáveis de tipos diferentes. Como por exemplo, em aplicações na área de Biologia, podemos ter interesse em clusterizar um conjunto de tipos de folhas que são caracterizadas por variáveis como cor predominante, tamanho, família, se apresenta pigmentação mista ou não, se há registro de determinado tipo de doença para aquela espécie etc. Em casos como este, é inadequado adotar uma das equações já apresentadas como se todos os dados fossem de um único tipo.

Uma alternativa não muito adequada seria realizar tantas clusterizações quantos tipos diferentes de váriáveis se tiver no conjunto de entrada, onde cada clusterização seria aplicada ao conjunto de variáveis de um mesmo tipo utilizando a dissimilaridade adequada para, em seguida, o próprio usuário escolher aquela que parecer mais conveniente à aplicação. Entretanto, o fato desta estratégia segmentar as características dos dados de entrada, embora ofereça facilidade na implementação, a torna pobre em termos de resultados obtidos, haja vista o fato de não se verificar as dissilaridades levando-se em conta todas as informações referentes aos pontos que compõem a base de dados. O ideal então, seria processar todas as variáveis simultaneamente, levando-se em conta os tipos diferentes de dados que são utilizados para caracterizar os pontos.

Um outro fator a ser considerado em bases de dados reais é a ocorrência de casos em que para algum ponto $x_i \in S$, o valor referente a alguma variável f não exista. Nestes casos, dizemos que a váriável x_{if} é ausente para o ponto x_i. Tratamos estes casos da mesma forma como lidamos com pontos que são caracterizados por variáveis de tipos diferentes. Porém, para fazer uso da mesma abordagem para os dois casos, é conveniente combinar todas as variáveis em uma matriz de dissimilaridade expressando seus valores dentro do intervalo $[0, 1]$ para efeito de evitar problemas referentes à unidade utilizada, conforme já mencionamos. Feito isto, a dissimilaridade

entre dois pontos x_i e x_j cujas variáveis são de tipos mistos é definida por:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{2.13}$$

onde $\delta_{ij}^{(f)}$ é um indicador de que a variável f para o ponto x_i ou x_j não tem valor registrado ou de que f é uma variável do tipo binária assimétrica, sendo que $\delta_{ij}^{(f)} = 0$ nas seguintes condições:

a) quando x_{if} ou x_{jf} é uma variável ausente;

b) quando $x_{if} = x_{jf} = 0$ e a variável f é do tipo binária assimétrica;

Caso contrário, $\delta_{ij}^{(f)} = 1$.

No cálculo da dissimilaridade entre dois pontos x_i e x_j cujas variáveis são de tipos mistos, dado pela Equação 2.13, a dissimilaridade $d_{ij}^{(f)}$ é calculada conforme o tipo de dado da variável f, como segue:

i) se f é uma variável do tipo binário ou categórico, então $d_{ij}^{(f)} = 0$ se $x_{if} = x_{jf}$, caso contrário $d_{ij}^{(f)} = 1$;

ii) se f é uma variável do tipo intervalo-escalar, então $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}}$, onde h indica os casos em que a variável f não é ausente.

iii) se f é uma variável do tipo ordinal ou do tipo escalar-proporcional, utilizamos a Equação 2.12 e calcula-se $d_{ij}^{(f)}$ como se f fosse uma variável do tipo intervalo-escalar.

Desta forma, podemos executar um algoritmo de clusterização independentemente de termos pontos cujas coordenadas sejam variáveis de tipos diferentes ou ainda, termos valores ausentes para uma determinada coordenada, o que, na realidade, é o que se pode esperar em bases de dados do mundo real. Neste trabalho, abordaremos apenas os casos em que a base de dados já se encontra pronta para processamento. Além disso, assumimos que os pontos são definidos por variáveis do tipo numérico escalar e a métrica usada para o cálculo de dissimilaridade é a

distância euclidiana definida pela Equação 2.6.

2.2 Algumas Aplicações do Problema de Clusterização

O Problema de Clusterização tem sido aplicado ao longo de décadas por pesquisadores de diversas áreas. Dependendo do campo da ciência, muitas são as denominações dadas a este de problema, como: **classificação automática**, **taxonomia automática**, **análise tipológica**, **análise de clusters** e **agrupamentos**.

Nos últimos anos, o número de publicações sobre clusterização tem tido um crescimento considerável tendo em vista as aplicações destes algoritmos de clusterização terem ganho espaço nos mais variados setores das pequenas e grandes organizações. Seja para customizar estratégias de marketing, processar imagens, otimizar configurações de máquinas em linhas de montagem, escalonar tarefas em ambientes multiprocessados, no emparelhamento de genes ou como auxílio na diagnose de doenças, o Problema de Clusterização está presente sob diversas formas.

Na área da Biologia, clusterização pode ser usada em taxonomia, emparelhamento de genes de acordo com a funcionalidade ou ainda, no agrupamento de populações de organismos [15]. Aplicações no campo da Biologia têm sido muito importante especialmente nos estudos sobre Ecologia, que vêm tendo ampla divulgação nos últimos anos [17].

Além disso, diversos trabalhos que fazem uso de técnicas de Clusterização têm sido apresentados na solução de problemas de reconhecimento de padrões, que têm aplicações variadas [9, 58, 108]. Em [8] é apresentado um algoritmo para classificação de imagem usando conceitos de Algoritmos Evolutivos.

Na medicina, Reconhecimento de Padrões através de clusterização pode ser usado para auxílio no diagnóstico de tipos de tumores e outras doenças através da formação de agrupamentos feitos sobre uma base de dados com informações de pacientes cujo histórico já é conhecido. Em [57] é feito um estudo comparativo sobre as técnicas

de clusterização aplicadas à bases de dados clínicos. Em [12] é apresentada uma outra abordagem de clusterização aplicada à medicina para classificação de genes de células vasculares de tecidos musculares com o objetivo de conhecer melhor as alterações genéticas em pacientes com arterosclerose. Um outro trabalho interessante é um método de clusterização baseado em Lógica Fuzzy para auxílio no diagnóstico de doenças através de análise de imagens laboratoriais, proposto em [79]. David Cuesta-Frau et al. [26] aplicaram clusterização para análise de sinais de eletrocardiogramas. No Brasil, uma aplicação de clusterização na área médica foi desenvolvida para classificar e clusterizar pacientes com esquizofrenia [103]. Outras aplicações de clusterização na medicina são também encontradas em [14, 24, 29, 55, 56, 86, 95, 125].

Em Mineração de Dados, o número elevado de coordenadas dos pontos que compõem a base de dados de entrada e o tamanho desta base de dados são fatores que muitas vezes limitam a aplicação de alguns algoritmos originalmente desenvolvidos para pequenas bases de dados. Aplicações desta natureza são bastante comuns em Marketing e Mineração de Web Site. Em alguns casos, usar clusterização para dividir os pontos em grupos similares e representá-los em clusters pode implicar em abrir mão da fineza dos detalhes que trazem estes pontos individualmente. Entretanto, com o uso algoritmos de clusterização como pré-processamento, obtém-se uma simplificação da base de dados para uma posterior aplicação de outras técnicas específicas de Mineração de Dados, como Regras de Associação. Isto pode se constituir em uma estratégia vantajosa e necessária para o propósito da aplicação.

Neste contexto, dois trabalhos merecem destaque por primarem pela garantia da escalabilidade dos algoritmos apresentados. O primeiro é o algoritmo ROCK (*Robust Clustering Algorithm for Categorical Attributes*) [49], no qual, a similaridade entre dois pontos é baseada no número de coordenadas para os quais estes pontos têm o mesmo valor. Além disso, o algoritmo ROCK pode operar sobre dados do tipo categórico, o que é de fundamental importância em Mineração de Dados. O segundo algoritmo é o BIRCH (*Balanced Iterative Reducing and Clustering Using Hierarchies*) [128], que faz uma pré-clusterização para reduzir a cardinalidade do conjunto de entrada através de uma estrutura de árvore denominada *CF-tree*, que armazena as informações sobre a configuração da distribuição dos pontos. Outras

abordagens empregadas para clusterização de grandes bases são apresentadas em [3, 33, 34, 50, 108, 114].

Nas áreas de Engenharias de Produção e Industrial, uma especificação do Problema de Clusterização é o problema de otimização conhecido como *problema de formação de células de manufatura* (PFCM). O PFCM é uma etapa crucial de projetos de geração de células de produção em sistemas de manufatura. Este problema é basicamente descrito por um conjunto de partes (produtos a serem manufaturados) e um conjunto de máquinas. O objetivo é agrupar as partes similares (quanto às máquinas necessárias para executá-las) em grupos denominados famílias e agrupar máquinas em grupos denominados células.

O PFCM tem sido resolvido de forma eficiente por Algoritmos Evolutivos. Resende et al. [48] propuseram um AG com busca local. Em 2004, Trindade e Ochi [113] apresentaram um Algoritmo Evolutivo construtivo com busca local que apresentou na média, resultados superiores aos obtidos em [48].

Vale ressaltar que no PFCM, o número de clusters (grupos) não é previamente fixado, ficando esta informação como uma variável a ser decidida pelo algoritmo de clusterização [65, 121]. Em [11], é apresentado um algoritmo baseado em *Tabu Search* para problemas de roteamento de veículo, que tem aplicação em várias áreas. Também com aplicação em várias áreas com a denominação de problema de alocação de recursos, uma boa referência é o algoritmo proposto por Osman e Christofides [90] e com aplicação para os setores industrial, comercial e de serviços. Podemos citar ainda, nesta classe de aplicações, os trabalhos [23, 41, 70, 91, 118].

Diversas outras aplicações do problema de clusterização podem ser encontradas na literatura, tais como Processamento de Linguagem [78, 120], Ciências Ambientais [17, 51, 85, 129], Epidemiologia [105], Bioquímica e Biomédica [20, 25, 53, 62, 102, 107]. Na área de Tratamento de Imagem destacamos os trabalhos apresentados em [32, 36, 71, 73, 74, 75, 122], este último, focado especificamente em clusterização para transmissão de imagens através de tecnologia *wireless*. Encontramos trabalhos ainda na área de Processamento Paralelo [2] e Ciências Sociais [69, 111]. Este último, trabalho de Landmann e Lourenço, utiliza clusterização para auxílio no estudo sobre

mortalidade infantil no Estado do Rio de Janeiro.

Como podemos ver, a aplicação do problema de clusterização independe do campo da ciência, uma vez que em todas áreas de pesquisa e produção, sempre manuseamos informações sobre objetos do mundo real. Tais informações configuram-se como conjuntos de dados que muitas vezes apresentam características intrínsecas que nos impedem de interpretá-los de forma a tirar algum conhecimento acerca da disposição dos mesmos. Isto explica o crescimento expressivo de publicações sobre o Problemas de Clusterização que objetivam melhorar a qualidade, a aplicabilidade e a robustez dos algoritmos propostos.

2.3 Análise da Complexidade de Tempo

Como visto na seção anterior, existem várias aplicações para o Problema de Clusterização, o que leva à especificações deste problema com características bastante peculiares. Entretanto, para um estudo da complexidade de tempo deste problema na sua configuração mais geral, considere inicialmente o caso em que se conhece o número k de clusters a ser obtido. Neste caso, o número de soluções viáveis $N(f)$ para o Problema de Clusterização é dado por:

$$N(f) = \left(\frac{1}{k!}\right) \sum_{j=0}^{k} (-1)^{k-j} \left(\begin{array}{c} k \\ j \end{array} \right) (j)^n \tag{2.14}$$

que é o número de combinações possiveis de n elementos tomados k a k [22] *apud* [1].

Temos, então, o número de clusters dado como parâmetro de entrada, ou seja, antes de procurar a clusterização, define-se o número de clusters. Um exemplo clássico de algoritmo que só funciona para o caso onde o número de clusters é previamente fixado é o *k-means* [76], cuja complexidade é $O(nki)$, onde n é o número de pontos, k é o número de clusters e i é o número de iterações executadas. Determinar os valores ideais para os parâmetros k e i constitui uma operação tediosa, já que na maioria das aplicações reais, não se tem nenhum conhecimento a priori da distribuição dos dados do conjunto de entrada, o que torna esta abordagem inadequada para estes

casos.

No caso em que o valor de k não é conhecido, que denota-se Problema de Clusterização Automática - PCA, como o espaço de busca é formado por todas as clusterizações possíveis para cada valor de k no intervalo $[1, n]$, o valor de $N(f)$ para o PCA é dado por:

$$N(f) = \sum_{k=1}^{n} \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \left(\begin{array}{c} k \\ i \end{array} \right) (k-i)^n \tag{2.15}$$

Assim, o espaço de busca do PCA é muito superior ao do PC, que apresenta uma complexidade exponencial para o tamanho do conjunto de entrada. Além disso, uma vez que a escolha do parâmetro k que define o número de clusters é importante na definição da complexidade do problema, convém citar os trabalhos de [18, 124], que provaram que para algumas funções objetivo, o Problema de Clusterização torna-se NP-completo para $k > 3$.

Com a complexidade do problema definida como NP-completo, conclui-se que o uso de abordagens que examinem exaustivamente todo o espaço de busca pode se constituir em um problema que a tornará inaplicável para aqueles casos em que o conjunto de entrada apresenta cardinalidade elevada. Nestes casos, o uso de Algoritmos Heurísticos apresenta-se como uma alternativa promissora. O uso de Algoritmos Heurísticos poderá prover soluções de boa qualidade (não necessariamente uma solução ótima) a um custo computacional aceitável [19].

A Heurística utilizada na solução de um problema pode falhar, principalmente se o espaço de busca deste problema for muito abrangente. Além disso, algumas heurísticas comportam-se bem para uma natureza do problema, enquanto para outras têm desempenho simplório. Como forma de suprir uma heurística de recursos adicionais na busca por soluções de melhor qualidade, muitas idéias foram empregadas para tornar mais inteligentes estas buscas heurísticas, o que fez surgir algumas metaheurísticas que têm apresentado bons resultados em muitos problemas de Otimização Combinatória, e o Problema de Clusterização tem sido bastante contemplado neste sentido, como será visto no Capítulo 3, onde apresentamos uma classificação dos algoritmos de clusterização desde as primeiras tentativas até o atual estado da arte.

Capítulo 3

Classificação dos Métodos de Clusterização

Nas últimas décadas, o estudo de métodos de clusterização levou ao surgimento de diversas abordagens para tratar problemas de agrupamento, que conforme já mencionamos, encontram aplicações em diversas áreas do conhecimento. E por se tratar de uma área fértil de pesquisa dentro de Otimização Combinatória, os métodos apresentados para tal problema muitas vezes utilizam técnicas diferentes ou ainda, apresentam uma reunião de técnicas já existentes. Num universo de abordagens tão diverso, classificar os algoritmos de clusterização acaba tornando-se uma tarefa difícil e polêmica, visto que muitas vezes não podemos dizer que um algoritmo constrói a clusterização utilizando apenas uma das técnicas conhecidas da literatura.

Nesta seção, procuramos apresentar uma classificação geral dos métodos de clusterização que contempla as principais abordagens para este problema. Não há intensão em dissecar a classificação de algoritmos de clusterização de forma a criar modelos pré-definidos de métodos. Ao invés disso, procuramos dar uma visão das abordagens existentes na literatura que possa oferecer os parâmetros necessários para uma comparação de métodos segundo a técnica empregada.

Em geral, os métodos de clusterização encontram-se classificados em dois grandes grupos: *métodos hierárquicos* e *métodos de particionamento*. Tal classificação,

entretanto, não cobre todos os métodos conhecidos. Assim, para uma classificação mais abrangente, além dos dois grupos apresentados, pode-se citar ainda os *métodos baseados em densidade*, *métodos baseados em grades* e o *método baseado em modelos*. Outras classes de algoritmos de clusterização são mostradas em [13, 52], como métodos baseados em co-ocorrência de dados de tipo categórico, métodos baseados em restrições, dentre outros. Não podemos deixar de ressaltar que existem ainda os métodos que integram idéias de várias abordagens diferentes. Tais métodos costumam ser denominados de *métodos híbridos de clusterização*.

3.1 Métodos de Particionamento

Os métodos de particionamento, também conhecidos como *k-clustering*, englobam aqueles algoritmos que recebem como entrada um conjunto S de pontos e um parâmetro k ($1 \leq k \leq n$) indicando o número de clusters a ser encontrado, onde $n = |S|$ é o número de elementos do conjunto de entrada, e procuram iterativamente agrupar todos os pontos de S em k clusters de forma a otimizar uma função custo f, muitas vezes chamada de *função de similaridade* [52]. A otimização da função f procura assegurar que os pontos pertencentes a um mesmo cluster apresentem alta similaridade e os pontos pertencentes a clusters diferentes tenham alta dissimilaridade.

Um algoritmo de particionamento, portanto, classifica os n pontos do conjunto de entrada em k grupos de tal forma que cada grupo contenha no mínimo um ponto e que cada ponto pertença a um único grupo. Entretanto, nem todo valor de k leva à clusterização natural, ou seja, àquela clusterização que reflete fielmente os grupos presentes no conjunto de entrada. Como veremos a seguir, existem abordagens que procuram encontrar o valor ideal do parâmetro k.

Dentre os algoritmos de particionamento, o mais popular é o algoritmo *K-means* [76]. Proposto em 1967, o algoritmo *K-means* representa cada cluster pelo centróide do mesmo. O centróide C_m de um cluster C é o ponto $C_m = (C_{m1}, C_{m2}, \ldots, C_{mp})$ onde cada coordenada C_{mj} é dada pela média das coordenadas j de todos os pontos

pertencentes ao cluster C, como dado na Equação 3.1.

$$C_{mj} = \frac{\sum_{i}^{t} x_{ij}}{t} \tag{3.1}$$

No algoritmo *K-means*, a função de custo f leva em consideração a média das distâncias entre cada ponto e o centróide do cluster ao qual pertence, devendo esta ser minimizada. Inicialmente, são escolhidos aleatoriamente k pontos para representar o centróide de cada um dos k clusters. Cada ponto do conjunto S é associado ao cluster cujo centróide lhe é mais similar, sendo que, cada vez que um ponto é associado a um cluster, o centróide daquele cluster é recalculado. Este processo é reiniciado para outros centróides escolhidos até que um critério de parada seja atendido, como um número i de iterações ou um número de iterações sem melhora na função custo f.

A complexidade do algoritmo *K-means* é $O(nki)$, onde n é o número de pontos do conjunto S, k é o número de clusters a serem formados e i é o número de iterações do algoritmo, ou seja, i é o número de vezes em que são selecionados os centróides iniciais para a geração de uma solução. Com esta complexidade, pode-se dizer que o *K-means* tem uma escalabilidade relativamente boa. Entretanto, um ponto fraco do algoritmo *K-means* é a dependência da escolha inicial dos centróides dos clusters, o que pode levá-lo à soluções muito pobres. Sobre este aspecto, o algoritmo de Bradley e Fayyad [16] procura encontrar os melhores centróides iniciais através de várias tentativas sobre pequenas amostras retiradas do conjunto de entrada. Um estudo comparativo das estratégias de inialização do *K-means* pode ser encontrado em [93].

Uma vez que o *K-means* tem uma função de similaridade que opera em relação ao centro do cluster, este método não pode ser aplicado em bases de dados onde não seja possível determinar o ponto central, como pontos com atributos do tipo categórico. Nestes caso, uma forma alternativa seria utilizar uma variante do *K-means* conhecida como método *K-modes*, que ao invés de lidar com o centroíde do cluster, passa a avaliar a freqüência dos valores de cada variável j de todos os pontos do cluster de forma a determinar a moda da variável j, que é dada pelo valor de

maior freqüência para tal variável. Para tanto, uma outra medida de similaridade deve ser adotada.

Outra deficiência do algoritmo *K-means* é a necessidade de se prefixar o número de clusters, o que a princípio o torna não aplicável às situações em que o valor ideal de k deve ser encontrado pelo algoritmo. Isto pode ser contornado parcialmente se executarmos o algoritmo várias vezes para diferentes valores de k e tomarmos a melhor solução dentre todas as execuções. No entanto, isto compromete consideravelmente o algoritmo no que se refere à escalablidade. Uma abordagem híbrida apresentada em [9] lida com esta limitação aplicando conceitos de Algoritmos Evolutivos, onde cada indivíduo da população é uma solução encontrada através do *K-means*.

Algumas bases de dados apresentam pontos que não são similares a nenhum outro ponto. Tais pontos, denominados *outliers*, constituem uma armadilha para o algoritmo *K-means* cair em ótimos locais ainda distantes de um ótimo global, pois conseguem distorcer consideravelmente o centróide do cluster. Esta deficiência do método *K-means* levou ao desenvolvimento de uma outra abordagem de partição bastante explorada, conhecida como *K-medoids*. Diferente do *K-means*, ao invés de tentar associar os pontos aos clusters representados por um centróide, o *K-medoids* tenta encontrar incialmente os k pontos mais representativos para cada um dos k clusters a serem formados. Tais pontos são denominados *medoids*, os quais o algoritmo procura iterativamente substituir por pontos não-*medoids* na tentativa de melhorar a qualidade da solução atual.

Representação de clusters por *medoids* apresenta vantagens sobre a representação por centróides, uma vez que pode ser utilizado para qualquer tipo de atributos e, além disso, a determinação do *medoid* é guiada por um maior número de pontos, o que o torna menos sensível a entrada de dados contendo *outliers*. Entretanto, embora o *K-medoids* seja mais eficiente que o *K-means* no que se refere às instâncias contendo *outliers*, não se observa uma boa eficiência do mesmo quando submetido à bases de dados de dimensões elevadas. Neste sentido, podemos destacar os algoritmos CLARA (*Clustering LARge Applications*) de Kaufman e Rousseeuw [67] e

CLARANS (*Clustering Large Applications based upon RANdomized Search*) de Ng e Han [88] como os mais populares algoritmos de partição para grandes instâncias, que apresentam melhores soluções que o algoritmo PAM (*Partitional Around Medoids*) também de Kaufman e Rousseeuw.

O algoritmo CLARA emprega algoritmo PAM apenas em uma amostra dos pontos selecionada aleatoriamente do conjundo de entrada S, o que torna desnecessário manter todos os dados na memória principal, havendo a necessidade de manter apenas os dados da amostra. Entretanto, a qualidade da clusterização gerada depende do tamanho desta amostra. Além disso, se um dos *medóids* do conjunto S não pertencer à amostra, o algoritmo falhará na clusterização, ou seja, apresentará uma solução que na verdade é um ótimo local.

O algoritmo CLARANS procura evitar ótimos locais comuns no CLARA através da construção do conjunto de *medoids* e da análise da vizinhança dos mesmos em amostras escolhidas de forma aleatória. Somente quando os pontos vizinhos não conseguem melhorar a solução quando tornam-se *medoids*, o algoritmo seleciona outra amostra para novamente buscar os *medoids*, até que um critério seja atendido. Diferente do CLARA e dos demais algoritmos de partição já citados, o CLARANS é capaz de encontrar o número ideal k de clusters usando o *coeficiente silhueta* [67], propriedade que cada ponto tem que indica o quanto o mesmo encontra-se bem associado ao cluster ao qual pertence.

3.2 Métodos Hierárquicos de Clusterização

Os métodos de hierárquicos clusterização operam sobre o conjunto de entrada S de pontos e procuram construir uma árvore de clusters denominada *dendograma*, na qual, cada nó é um cluster que pode ter outros clusters filhos, pais ou irmãos, de forma que cada ponto de S é associado a um único nó da árvore em um dos seus níveis. Isto permite que os dados sejam manuseados com um nível de granularidade melhor que os oferecidos por algoritmos de partição.

A literatura apresenta ainda uma divisão dos métodos hierárquicos de cluste-

rização em dois grupos de acordo com a forma como o dendograma é construído. Quando a árvore é construída da raiz para as folhas (estratégia *bottom-up*), dizemos que o algoritmo é **hierárquico por aglomeração**. Caso contrário (estratégia *top-down*), o algoritmo é dito **hierárquico por divisão** [13, 64].

Um algoritmo de clusterização hierárquica por aglomeração inicia o processo de clusterização criando n clusters, cada um contendo um dos n pontos do conjunto S. Iterativamente, dois ou mais clusters são reunidos o mais apropriadamente possível, segundo a (dis)similaridade dos seus pontos para formar um único cluster.

Diferente dos métodos hiérarquicos por aglomeração, um algoritmo de clusterização do tipo hierárquico por divisão inicia a clusterização criando um único cluster contendo todos os n pontos do conjunto S, que será dividido a cada nível em dois (ou mais) clusters para criar um novo nível na árvore. A cada nível que vai sendo criado na árvore, os clusters são novamente divididos para gerar outros clusters de nível inferior. A Figura 3.1 mostra os dois tipos de clusterização hierárquica. Como pode ser visto, os dois tipos diferem quanto ao ponto de partida da construção do dendograma.

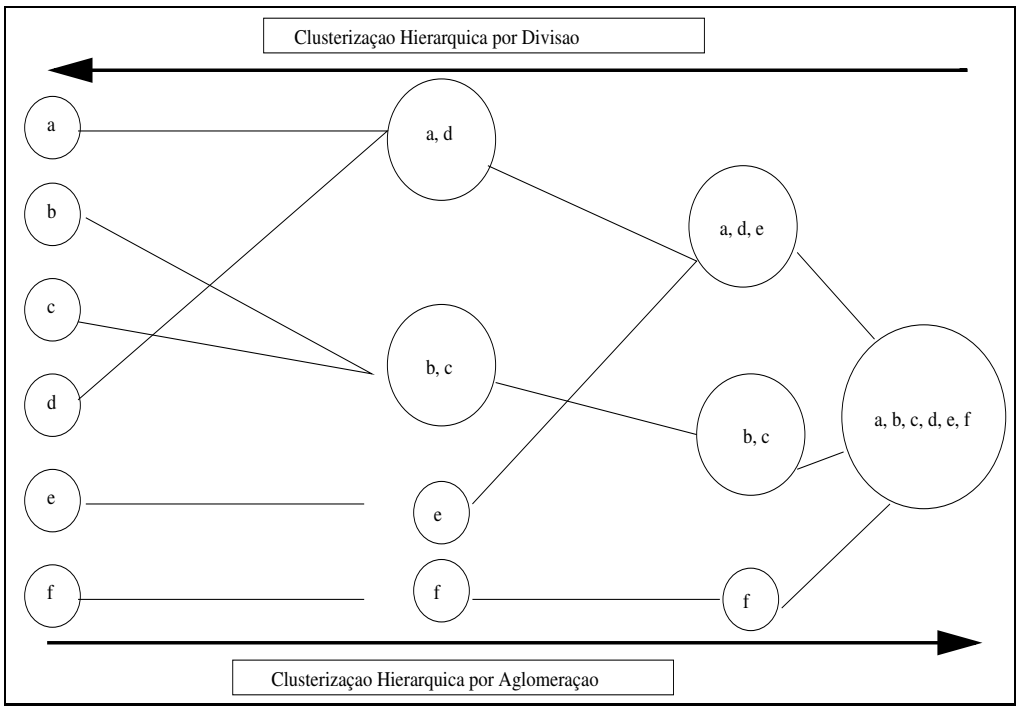

Figura 3.1: Modelos de Clusterização Hierárquica

Tanto na estratégia *top-down* como na *bottom-up* uma condição de parada deve ser atingida para que o processo de agrupar ou dividir clusters seja interrompido para se ter a solução final. Muitas vezes, esta condição é a obtenção de um número k de clusters. Pode-se ainda, a exemplo do que vimos nos métodos de partição, executar várias vezes o algoritmo com diferentes valores de k para em seguida retornar a solução mais adequada.

Clusterização hierárquica pode ser vista como um problema de particionamento em grafos. Uma vez que a entrada para o algoritmo pode ser uma matriz de conectividade A_{nXn} na qual o elemento a_{ij} indica a (dis)similaridade entre os pontos i e j do conjunto de entrada S, pode-se considerar um grafo $G(X, E)$ cujo conjunto de vértices X são os pontos de S e o conjunto de arcos E são as (dis)similaridades a_{ij}. Ao associarmos a matriz A ao grafo G, podemos ver claramente que encontrar partições em G equivale a efetuar clusterização dos pontos representados pela matriz A. Em [31, 104] são apresentados excelentes trabalhos sobre a utilização de particionamento em grafos para este problema.

Dentre vários algoritmos hierárquicos apresentados na literatura, alguns merecem destaque, como o algoritmo BIRCH (*Balanced Iterative Reducing and Clustering Using Hierarchies*) [128], desenvolvido para aplicações em grandes bases de dados, assim como o algoritmo CURE (*Clustering Using REpresentatives*) [50], um algoritmo aglomerativo que além de manter o bom desempenho quando submetido a grandes instâncias onde cada ponto apresenta atributos do tipo numérico-escalar, encontra clusters de diferentes formas e é insensível a *outliers*, o que representa boas características para um algoritmo de clusterização. Entretanto, o algoritmo CURE não é capaz de identificar o número ideal de clusters.

Um outro algoritmo hierárquico do tipo aglomerativo é o ROCK (*Robust Clustering Algorithm for Categorical Attributes*) [49], que consegue manter as características positivas do algoritmo CURE além de poder lidar com atributos do tipo categórico, recebendo como parâmetro de entrada o número k de clusters desejado. Pode-se citar ainda o algoritmo CHAMELEON [66] que apresenta características do ROCK e do CURE. O CHAMELEON constrói um grafo de conectividade corres-

pondendo aos k (parâmetro de entrada) vizinhos mais próximos, para cada ponto para em seguida realizar o particionamento sobre o grafo construído [13].

3.3 Metaheurísticas Aplicadas ao Problema de Clusterização

Existem muitos algoritmos de clusterização que utilizam abordagens diversas para concluir o processo de clusterização e retornar a solução do problema. Vários fatores influem na escolha da técnica a ser usada para tornar híbrido o algoritmo, como a natureza dos dados, tamanho do conjunto de entrada etc. Uma análise de tais fatores faz-se necessária para decidir qual a mais adequada. Muitas vezes opta-se por utilizar alguma metaheurística para auxiliar na obtenção da solução, abrindo-se mão da garantia de se obter a solução ótima em troca da diminuição do tempo de processamento.

Dentre as metaheurísticas, os Algoritmos Genéticos (AG's) [8, 9, 13, 43, 46, 47, 64, 115, 116] estão entre as mais usadas na solução do Problema de Clusterização. É importante ressaltar que, tão importante quanto a metaheurística, é a heurística usada para a construção dos clusters, que pode diferir para uma mesma metaheurística.

O algoritmo CLUSTERING apresentado por Tzeng e Yang [116] é um Algoritmo Genético montado sobre um algoritmo hierárquico que constrói a clusterização e ao mesmo tempo identifica o melhor número de clusters para o conjunto S de pontos. Tal algoritmo constrói um grafo a partir da matriz de conectividade e então, após identificar os componentes conexos do grafo gerado, procura agrupar tais componentes de forma a maximizar uma função objetivo $f(.)$.

O algoritmo de Badyopadhyay e Maulik [8] usa Algoritmos Genéticos para classificação de imagem, uma aplicação do Problema de Clusterização. Já o algoritmo de Tzeng e Yang apresentado em [115] apresenta um AG nos moldes do CLUSTERING.

Além dos algoritmos evolutivos, podemos citar o algoritmo de Gulay e Demet [11]

para o problema de roteamento de veículo (uma especificação do problema de clusterização) empregando Busca Tabu (*Tabu Search*). Busca Tabu também foi usada em [90] num algoritmo híbrido com *Simulated Annealing*. Além destes trabalhos, Busca Tabu adaptativa para o Problema de Clusterização também pode ser encontrada em [5, 40]. Em [87] encontramos um algoritmo de clusterização através de Busca Tabu para pontos cujas coordenadas são variáveis do tipo categórico, enquanto em [21] uma versão paralela de Simulated Annealing é descrita.

Existem ainda, muitos algoritmos na literatura que utilizam GRASP (*Greedy Randomized Adaptive Search Procedures*) [38, 100, 101] para diversas aplicações que na verdade são especificações do Problema de Clusterização. Por exemplo, o problema de Roteamento de Circuito Virtual, para o qual Resende e Ribeiro apresentaram um GRASP com Reconexão por Caminhos (RC) [98], e também o problema conhecido por *Job Shop Scheduling* abordado no trabalho de Aiex [4]. Outros trabalhos podem ser encontrados em [7, 14, 65, 79, 83, 91, 92, 125, 126, 127] e algumas formas híbridas que combinam metaheurísticas diferentes para construir a clusterização são dadas em [30, 37, 46, 47, 99].

A classificação das técnicas de clusterização aqui apresentada não cobre todas as abordagens da literatura. Entretanto, nesta seção procuramos dar uma visão generalizada dos principais métodos empregados no Problema de Clusterização. Como já foi citado, para uma melhor granularidade na *clusterização* dos diversos métodos da literatura, sugerimos consultar [13, 52, 67], entretanto, é unânime a consideração de que nenhuma classificação apresentada é fiel e justa, face ao grande número de aspectos a serem considerados.

3.4 Características Importantes em Algoritmos de Clusterização

Com o volume e a diversidade de informações crescendo desenfreadamente nas grandes corporações e nos mais variados campos da pesquisa, é natural nos deparar-

mos com problemas de clusterização de características diversas, cada um com suas peculiaridades e restrições. Para um melhor entendimento das características que se espera de um bom algoritmo de clusterização nas diversas aplicações, apresentamos nesta seção aquelas consideradas de maior importância.

No processo de concepção e desenvolvimento de um algoritmo de clusterização, vários aspectos devem ser levados em conta para que se obtenha uma solução viável para o propósito da aplicação. Jiawei Han e Micheline Kamber [52] citam alguns dos aspectos que merecem destaque no desenvolvimento de algoritmos de clusterização em Mineração de Dados. Entretanto, todos os aspectos mencionados são o que se espera para a grande maioria das aplicações, não apenas para problemas em Mineração de Dados. Assim, podemos dizer que as características mais desejáveis em algoritmos de clusterização são as que seguem:

- **Escalabilidade**

É comum termos algoritmos de clusterização que apresentam soluções viáveis para instâncias pequenas de 200, 300 ou 500 pontos, mas que quando submetidos a instâncias maiores, contendo até milhões de pontos, têm um desempenho medíocre. Para a maioria das aplicações reais, exige-se que o algoritmo seja capaz de apresentar soluções boas independente do tamanho da entrada.

Algumas estratégias costumam ser usadas para diminuir a sensibilidade do algoritmo ao tamanho do conjunto de entrada. Uma delas é procurar identificar as regiões mais densas da instância e tirar proveito das características de tais regiões, como é o caso do algoritmo BIRCH [128], do algoritmo ROCK [49] e do algoritmo CURE [50], além das abordagens apresentadas em [6, 7, 35, 119]. O uso de amostragem reduz a sensibilidade do algoritmo ao tamanho do conjunto de entrada na medida em que não se trabalha sobre o universo total dos dados. Entretanto, vale ressaltar que esta redução de dependência da cardinalidade não ocorre de maneira linear, uma vez que à medida que o conjunto de entrada cresce, o desempenho do algoritmo depende também do crescimento do tamanho da amostra para que esta seja capaz de refletir as características do conjunto de entrada, o que, para algumas

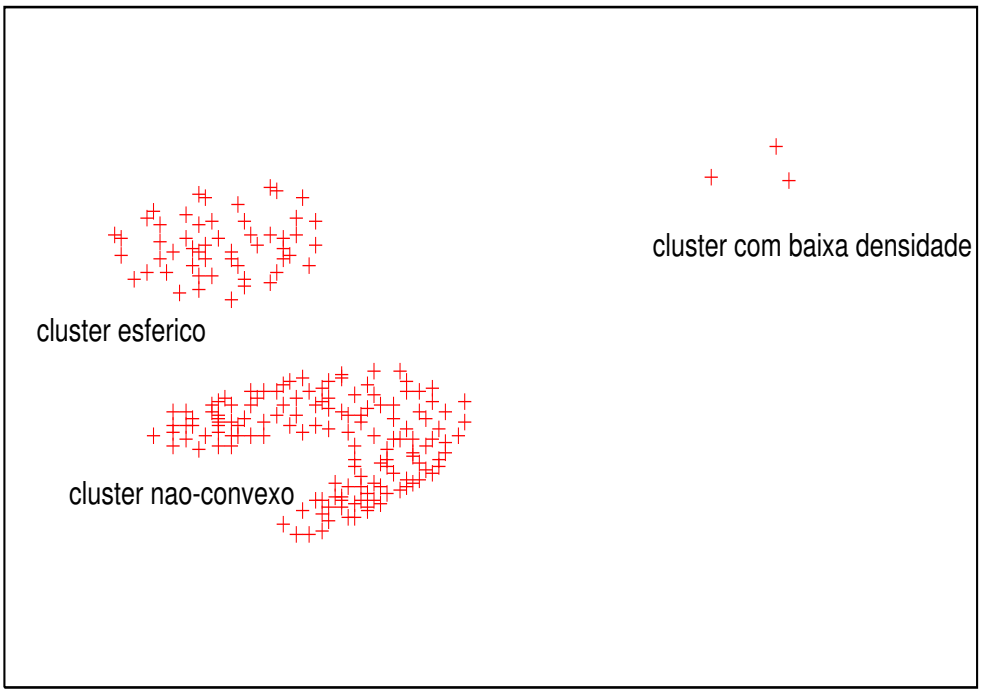

Figura 3.2: Exemplo de dados com clusters de formato e densidade diferentes

aplicações do mundo real, acaba tornando a abordagem ineficaz.

- **Obtenção de clusters de formas arbitrárias**

No processo de clusterização, para que o algoritmo decida em qual cluster um determinado ponto irá pertencer é necessário que se verifique a qual dos clusters este ponto é mais similar. Muitos algoritmos de clusterização adotam como medida de similaridade as distâncias Euclidiana ou de Manhattan. Ocorre que, algoritmos baseados nestas medidas de similaridade tedem a produzir clusters de formato esférico com densidade similares. A Figura 3.2 apresenta uma entrada com estas características.

É desejável que um algoritmo de clusterização seja hábil a encontrar clusters de formatos variados, já que em muitas aplicações nenhum conhecimento a priori sobre a distribuição dos pontos pode ser obtido. Em [10] é apresentado um algoritmo baseado em particionamento iterativo de Grafos de Vizinhança Relativa em que os clusters gerados têm forma arbitrária e não são sensíveis a *outliers*.

Além da abordagem citada anteriormente, citamos ainda os algoritmos referenciados no item anterior, que além de apresentarem soluções viáveis para grandes bases de dados, constroem clusters de formas varidas, o que os faz constituir boas estratégias para aplicações em Mineração de Dados.

- **Calibragem de parâmetros sem conhecimento prévio do domínio**

A calibragem de parâmetros de um algoritmo de clusterização pode tornar-se uma tarefa tão árdua como o próprio desenvolvimento da abordagem usada no mesmo. Muitas vezes os parâmetros que são definidos para uma instância não se aplicam a uma outra, devendo o usuário final fazer uso da sua experiência na aplicação para abreviar o processo de calibragem cada vez que uma nova instância é submetida ao algoritmo.

É claro que esta situação pode comprometer substancialmente a qualidade e a aplicabilidade do algoritmo em problemas reais. O que se deseja é um algoritmo que seja o mínimo possível dependente da experiência do usuário, ou seja, o próprio algoritmo deve conseguir identificar os clusters baseado apenas nas informações referentes aos objetos, representadas no conjunto de entrada pelos atributos que formam as coordenadas de cada ponto.

- **Habilidade para manipular dados com ruídos**

Ruídos em bases de dados são aqueles pontos cujos valores das variáveis associados às coordenadas apresentam erros ou distorções. Dados com ruídos são muito comuns em aplicações reais. Seja por erros de digitação ou por falhas na coleta dos dados, é sempre possível termos em uma massa de dados alguns pontos cujas informações foram corrompidas. Um exemplo, é o caso de termos pontos dados por coordenadas que apresentam valores fora de um domínio específico, como idade, altura ou massa com valores negativos, o que pode ser visto na Figura 3.3. Nestes casos, dependendo da abordagem adotada pelo algoritmo, podemos ter soluções que são influenciadas muito mais por ruídos do que pelos dados sem erros. Portanto, o desenvolvimento de um algoritmo que seja imune a ruídos é pré-requisito para se

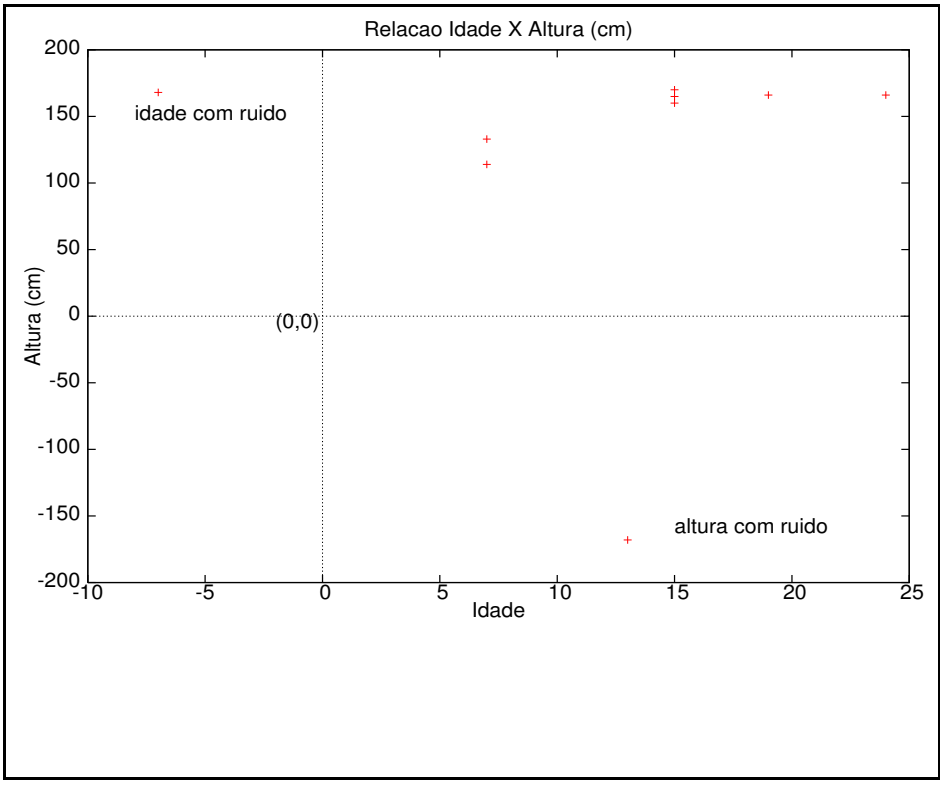

Figura 3.3: Exemplo de dados com ruídos

ter uma solução aplicável em bases de dados reais, a menos que se tenha longas fases de pré-processamento para fins de enxugar as instâncias antes de submetê-las ao algoritmo. Algumas abordagens para manuseio de dados com ruídos podem ser encontradas em [49, 54, 114, 123, 127]

- **Capacidade de processar bases de dados de dimensões elevadas**

Se por um lado a escalabilidade é uma propriedade desejável em um algoritmo de clusterização, a habilidade de lidar com objetos caracterizados por um elevado número de atributos não é menos importante. Podemos ver a dimensionalidade elevada como uma forma específica de escalabilidade, já que o que se espera é que o algoritmo não perca em eficiência à medida que seja necessário lidar com instâncias com milhões de pontos e/ou instâncias cujos pontos são formados por um elevado número de coordenadas.

Alguns algoritmos são muito eficientes quando lidam com pontos com duas ou três dimensões, mas que, para bases de dados cujos os pontos têm um número elevado

de coordenadas, a qualidade das soluções apresentadas cai consideravelmente. Com o grande volume de informações de que se dispõe, é comum as organizações manterem um *DataWarehouse*[1] como fonte de informações cruciais para as tomadas de decisões. O uso de *DataWarehouse* pode nos obrigar a lidar com pontos de um espaço de dimensões elevadas.

É importante então, que o número de dimensões da base de dados não interfira como fator negativo na performance do algoritmo. Para tanto, muitas vezes é necessário que se opte por abordagens que conceitualmente se comportam melhor para estes tipos de bases de dados, como os algoritmos de clusterização baseados em densidade ou em grades. Algumas estratégias que tratam desta questão podem ser encontradas na literatura, como pode ser visto em [9, 57, 114]. Além disso, podemos encontrar soluções paralelas para o Problema de Clusterização para aplicações que lidam com bases de dados com elevadas dimensões, como pode ser verificado em [21, 45].

- **Habilidade de lidar com diferentes tipos de atributos**

Vimos no Capítulo 2 diversas métricas utilizadas no cálculo da similaridade de objetos em Problemas de Clusterização. Ao tentarmos definir atributos ou propriedades de objetos, vemos facilmente que alguns destes atributos apresentam-se como variáveis sobre domínios diversos. Alguns algoritmos são desenvolvidos para lidar com um único tipo de variável, como tipo numérico-escalar, tipo binário, tipo categórico etc. Muitas vezes, no entanto, faz-se necessária a análise dos objetos através de atributos que definem variáveis cujos tipos de dados são diferentes.

Considere os atributos referentes ao objeto cliente de uma empresa de cartão de crédito. Vários aspectos podem ser levados em consideração para efeito de análise de clusters. Como por exemplo, faixa salarial, cor, sexo, se o cliente tem carro ou não e o grau de instrução. Note que estes atributos do objeto cliente apresentam

[1]Inmon [63] define *DataWarehouse* como sendo um conjunto de dados orientados por assunto ou negócio, integrados, variáveis com o tempo e não-voláteis que fornecem suporte ao processo de decisão do negócio.

tipos diferentes de dados e precisam ser avaliados pelo algoritmo em conjunto. Não se pode pensar em desenvolver um algoritmo para este tipo de aplicação que lide com apenas um determinado domínio.

Como já foi mencionado, no caso de se ter uma base de dados onde os tipos dos atributos são diferentes e/ou tenha-se algum atributo ausente, pode-se tratar estes dados de forma a poder utilizar métricas já conhecidas para variáveis do tipo intervalo escalar.

- **Seleção de variáveis**

Ao manusearmos bases de dados do mundo real, deparamo-nos com objetos que podem ter diversos atributos que na verdade não são relevantes para o processo de clusterização. No caso da Tabela 2.1, por exemplo, não seria importante levarmos em consideração uma variável como número de telefone, caso este atributo fizesse parte das características do indivíduo daquela tabela. Muitas vezes, a própria aplicação direciona a escolha das variáveis relevantes. Deve-se ressaltar que a escolha das variáveis a serem avaliadas não necessariamente implica somente em considerá-las ou não. Pode-se, por exemplo, fazer uso de pesos diferentes para cada uma das variáveis que caracterizam o objeto.

Nestes casos, um estudo detalhado da aplicação e dos dados de entrada precisa ser feito, o que poderá requerer intervenção do usuário, que precisará fazer uso da sua experiência na aplicação para definir pesos diferentes para uma mesma variável conforme seja o caso expresso em cada base de dados. No entanto, um algoritmo não supervisionado é o que se busca na expressa maioria das aplicações do mundo real. Um estudo detalhado sobre seleção de variáveis para algoritmos de clusterização pode ser encontrado em [94].

Uma outra característica bastante importante em um algoritmo de clusterização é a capacidade do mesmo de encontrar o número ideal de clusters. Dada as inúmeras aplicações reais em que não se tem nenhum conhecimento prévio quanto ao número de clusters presentes em uma base de dados, o estudo de algoritmos com esta característica se constitui numa generalização do Problema de Clusterização conhecida

como Problema de Clusterização Automática - PCA, como já mencionamos. Este trabalho apresenta duas propostas para o PCA, motivo pelo qual abordaremos esta propriedade apenas no Capítulo 4, onde apresentamos nossa abordagem para esta generalização do PC.

Capítulo 4

As Abordagens Propostas para o Problema de Clusterização Automática

Como vimos no Capítulo 2, dependendo da aplicação o Problema de Clusterização pode apresentar-se de diferentes formas, incluindo objetivos buscados e restrições consideradas. Algumas características são essenciais em algoritmos de clusterização, como: escalabilidade, capacidade de processar dados de tipos diversos e insensibilidade à bases dados com ruídos. Em muitas aplicações reais, uma das características mais desejáveis em um algoritmo de clusterização é a capacidade de encontrar, além da clusterização, também o número ideal de cluster.

Uma alternativa comumente empregada para encontrar a melhor clusterização num PCA é fazer uso de um algoritmo cujo número k de clusters é um parâmetro de entrada e executá-lo várias vezes com diferentes valores de k até encontrar uma solução aceitável. É de se questionar, entretanto, se esta estratégia é viável, pois em muitos casos, o tempo de processamento é fator crucial na escolha da técnica a ser empregada.

Na Seção 2.3 vimos que o espaço de busca do PCA é muito maior que o apresentado pelo Problema de Clusterização tradicional, no qual o número de clusters é

definido como parâmetro de entrada. O número de trabalhos apresentados na literatura para o PCA é ainda reduzido, comparado com o que existe para o PC. Em [9] é apresentado um AG cuja população é um conjunto de soluções com diferentes números de clusters. Outras abordagens são utilizadas na tentativa de desenvolver um algoritmo que resolva os dois problemas ao mesmo tempo: encontrar o número ideal de clusters e efetuar a clusterização. A maioria das aplicações para o PCA se encontra no Problema de Particionamento em Grafos [31] e em Problemas de Formação de Células de Manufatura - PFCM [113].

Apresentamos neste trabalho, dois algoritmos heurísticos para a solução do PCA. O primeiro, denominado *Algoritmo Evolutivo Construtivo* para o PCA (AEC), e o segundo é uma abordagem que emprega alguns conceitos de *Simulated Annealing*, denominado *Simulated Annealing para o PCA* (SAPCA). Ambos integram um método construtivo e um módulo de busca local.

O AEC e o SAPCA foram desenvolvidos para resolver o PCA em bases de dados cuja cardinalidade seja consideravelmente elevada. Pela forma de gerar os clusters iniciais, o AEC e o SAPCA reduzem a cardinalidade do conjunto de entrada para efeitos de redução do tempo total de processamento.

4.1 Considerações sobre Algoritmos Genéticos

O uso de conceitos ligados à Biologia no que tange à genética das populações e à Teoria da Evolução Natural [27] em algoritmos foi proposto pela primeira vez em 1975 por J. H. Holland [59] e, embora estes conceitos que remotam Charles Darwin sejam hoje por demais questionados pelos próprios estudiosos do campo da Biologia, algumas características destas teorias consolidaram-se como uma poderosa arma para a obtenção de soluções aproximadas de problemas de elevada complexidade [81].

Algoritmos Genéticos representam uma classe de Algoritmos Evolutivos usados para resolver problemas de Otimização Combinatória imitando as leis de seleção natural e genética de populações. Cada iteração do algoritmo é chamada de geração

do AG e o número total de gerações é um critério de parada do mesmo. Um conjunto de soluções chamado população é mantido a cada geração e a cada indivíduo da população é associado um valor numérico que indica sua aptidão (ou *fitness*), que é calculada segundo uma função de custo associada ao problema.

Um AG, na sua forma tradicional, constitui-se inicialmente de um mecanismo de geração aleatória de indivíduos, da avaliação de cada individuo (cálculo da aptidão), de uma estratégia de reprodução (cruzamento), de uma estratégia de mutação e de uma condição de sobrevivência do individuo de uma geração para outra. O esquema apresentado na Figura 4.1 mostra o pseudocódigo de um A.G genérico, que recebe como parâmetros o tamanho da população, o número de gerações e as taxas de cruzamento e mutação.

```
1. proc Algotimo_Genetico(LENG_POP, N_GERA, P_CROSS, P_MUT)
2.    i = 0;
3.    repita
4.       Inicialize população(LENG_POP);
5.       Avalie cada indivíduo;
6.       Crie novos indivíduos por cruzamento com probabilidade P_CROSS;
7.       Aplique mutação com probabilidade P_MUT ao novos indivíduos;
8.       Avalie novos indivíduos;
9.       Gere novos indivíduos;
10.      i = i + 1;
11.   até i = N_GERA;
12. fim  Algoritmo;
```

Figura 4.1: Algoritmo Genético genérico

Devemos enfatizar que num AG, há a necessidade de diversificar a população de indivíduos para que possamos explorar o maior número de regiões promissoras, já que não temos nenhuma idéia da configuração do ótimo global do problema. Isto nos leva à necessidade de construírmos algoritmos que ao longo de sua execução evitem

perda demasiada de tempo em ótimos locais devido à baixa taxa de diversificação da população corrente.

A construção de um AG requer ainda, que se tenha um mecanismo de codificação do indivíduo na forma de uma cadeia (*string*)de símbolos sobre um alfabeto[1] $\sum$. Geralmente, temos $\sum = \{0, 1\}$, podendo ainda este alfabeto ser um subconjunto de $\mathbb{Z}$ ou de $\mathbb{R}$. Além disso, um AG requer uma função que decodifique a cadeia codificada e retorne o valor da aptidão do indivíduo, obtida segundo a qualidade da solução associada ao mesmo.

Uma vez que os indivíduos que formam a população de um AG são cadeias codificadas, onde cada elemento da cadeia representa uma característica da solução associada a este indivíduo, é salutar diversificar a população para explorar o máximo possível o espaço de busca. Isto exige o emprego de operadores que garantam a obtenção deste propósito. Entretanto, é importante também que o algoritmo possa convergir para soluções de alta qualidade. Para tanto, um AG faz uso de operadores de seleção e cruzamento que possibilitam a intensificação da busca em regiões já exploradas e também aplica operadores de mutação e, possivelmente, inversão como mecanismo de diversificação. O uso dos operadores de cruzamento ou mutação sem critérios de seleção não reflete nenhum direcionamento do algoritmo rumo à soluções de boa qualidade. Neste caso, a obtenção de soluções de boas qualidade seria condicionada ao acaso. O operador de seleção, portanto, tem fundamental importância na eficiência de um AG.

O uso de AG's prevê a definição de alguns parâmetros. Tais parâmetros exigem uma calibração que pode variar tanto de aplicação para aplicação como também de instância para instância sobre uma mesma aplicação. Um destes parâmetros é o tamanho da população, que pode ser variável ou fixo. Via de regra, o número de indivíduos é fixo durante toda a execução do algoritmo. Além do tamanho da população, podemos citar ainda como parâmetros, a taxa de cruzamento, que

[1]Na Teoria de Linguagens Formais, um alfabeto $\sum$ é qualquer conjunto finito e não vazio de símbolos [60], o que é uma definição bastante e suficiente para definirmos alfabeto para a codificação de indivíduos em um AG, embora os alfabetos unários (com um único símbolo) não tenha aplicação prática.

reflete a probabilidade de se efetuar um cruzamento entre dois indivíduos após a seleção; a taxa de mutação, que indica a probalidade que cada elemento da cadeia do indivíduo tem de sofrer mutação; o número de gerações, que determina o número de iterações que o algoritmo executará; e ainda, o percentual de reposição, que expressa a quantidade de indivíduos que sobreviverão de uma geração para outra.

Voltando à questão dos operadores genéticos de cruzamento, mutação e seleção, apresentamos a seguir uma breve esplanação sobre tais operadores. Não é nossa intenção esmiuçar os conceitos e técnicas envolvidos, mas procurar suprir o leitor do que há de essencial para entender o funcionamento e a importância de tais operadores para a eficiência de um AG.

- **Operador de Seleção**

O cruzamento entre indivíduos pertencentes à população de um AG se dá após a escolha dos mesmos, onde tal escolha deve procurar gerar descendentes melhores que os pais. Baseado nos conceitos de genética das populações, acredita-se que o cruzamento entre indíviduos mais adaptados ao seu ambiente pode gerar filhos ainda melhores, entretanto, pode acontecer exatamente o contrário.

Um bom método de seleção deve priorizar a escolha de indivíduos para cruzamento que tenham valores elevados da função de custo mas, ao mesmo tempo, deve garantir alguma probabilidade de selecionar indivíduos com baixa aptidão. Assim, devemos imaginar o operador de seleção como uma função que recebe um conjunto de indivíduos (a população) com suas respectivas valores de aptidão e retorna um dos indivíduos deste conjunto segundo uma probabilidade que leva em conta tais valores.

O método de seleção mais utilizado é o Método da Roleta, que consiste em tomar o valor da soma das aptidões de todos os indivíduos como sendo a área M de uma circunferência e atribuir a cada um destes indivíduos uma fatia desta área proporcional ao valor de sua aptidão. Para selecionar o indivíduo que fará parte do cruzamento, sorteia-se um valor v dentro do intervalo $[0, M]$. Daí, partindo do primeiro indivíduo da população, segue-se até encontrar o indivíduo cujo valor da

aptidão, acrescido ao somatório das aptidões dos seus antecessores, seja maior ou igual a v. A Figura 4.2 mostra o esboço do funcionamento do Método da Roleta.

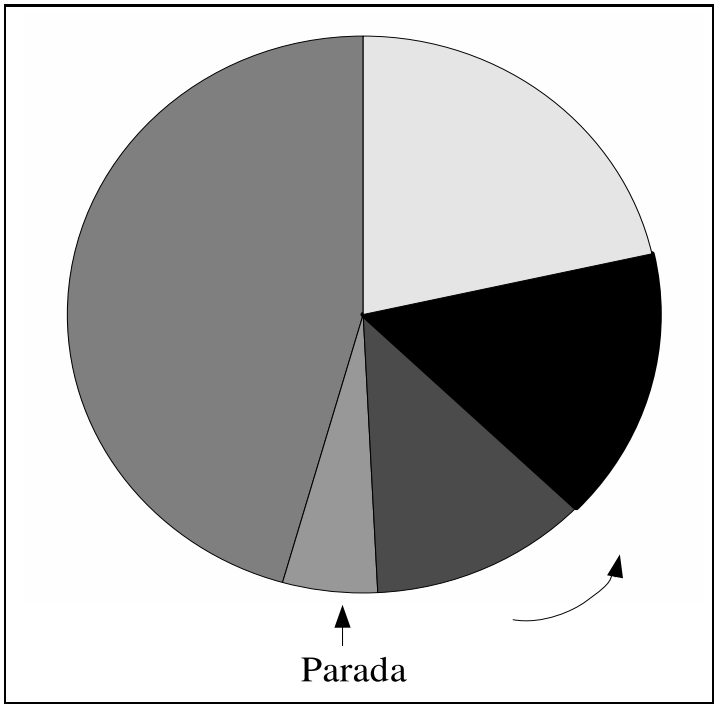

Figura 4.2: Esboço do Método da Roleta

Um problema bastante comum apresentado pelo Método da Roleta é o efeito causado pela presença de algum indivíduo cujo valor obtido pela função de aptidão é demasiadamente elevado em relação aos demais, o que chamamos de super indivíduos. A presença de super indivíduos na população e a estratégia utilizada para substituir os indivíduos da população pelos filhos obtidos de cada cruzamento pode fazer com que ocorra uma convergência prematura do algoritmo para ótimos locais ainda distantes de um ótimo global, uma vez que super indivíduos tendem a ser selecionados para cruzamento com maior freqüência e, com isso, inserir na população descendentes muito semelhantes.

Se utilizarmos um modelo elitista para substituir os indivíduos da população pelos filhos obtidos de cruzamento e/ou mutação, devemos trocar o indivíduo de menor aptidão por um filho gerado sempre que este filho apresenta melhor qualidade. Isto pode diminuir a diversidade dos indivíduos da população caso um mesmo indivíduo seja selecionado para cruzamento várias vezes, o que fará com que seu

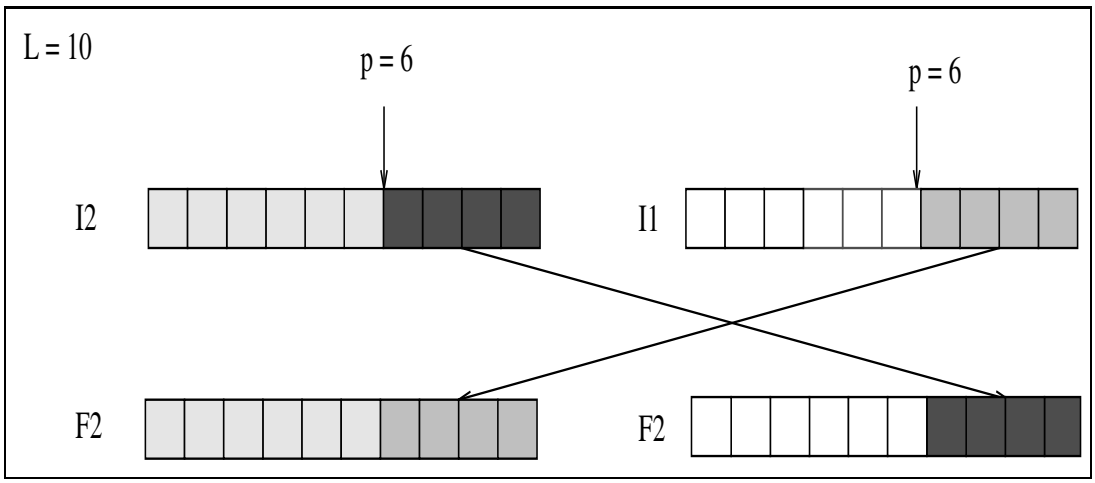

Figura 4.3: *crossover* de um ponto

código genético esteja presente em muitos indivíduos da população.

É comum termos implementações em que todos os filhos são gerados e somente após o último cruzamento a substituição dos indivíduos da população atual pelos descendentes é feita. Esta estratégia não parece muito proveitosa, uma vez que os descendentes não participam dos cruzamentos feitos após a obtenção dos mesmos. Isto emprobrece a busca mas, pode ao mesmo tempo, evitar a convergência prematura em uma mesma geração.

- **Operador de Cruzamento**

O operador de cruzamento, também referido na literatura como *crossover*, consiste na troca de material genético (elementos da cadeia codificada) entre dois indivíduos selecionados da população para a geração de descendentes.

O operador de *crossover* mais simples é o de um ponto, onde um valor inteiro p dentro do intervalo $[1, L-1]$, onde L é o tamanho da cadeia que o indivíduo codifica, é escolhido e as duas partes formadas à direita e à esquerda de p são trocadas entre os indivíduos. Na Figura 4.3, apresentamos este tipo de cruzamento, onde o tamanho da cadeia L é 10 e o ponto de corte p sorteado é 6.

Uma característicasas importante de um operador de *crossover* é a capacidade de combinar características desejáveis para a solução do problema que já encontram-se

presentes nos indivíduos selecionados. Entretanto, pode-se ver claramente que o operador de *crossover* de um ponto não consegue alterar blocos intermediários das cadeias dos pais mantendo fixas as extremidades das mesmas. Isto pode levar o algoritmo a não explorar eficientemente o espaço de busca. Para evitar este tipo de limitação, pode-se optar por operadores de dois pontos, três pontos e multipontos [81].

No *crossover* de dois pontos, são escolhidos aleatoriamente dois inteiros p e q dentro do intervalo $[1, L-1]$ com $p \neq q$. Estes pontos definem uma sub-cadeia S_{pq}, também chamada janela, em cada um dos pais. O cruzamento se dá pela troca de janelas entre os pais para a geração de seus descendentes, conforme pode ser visto na Figura 4.4.

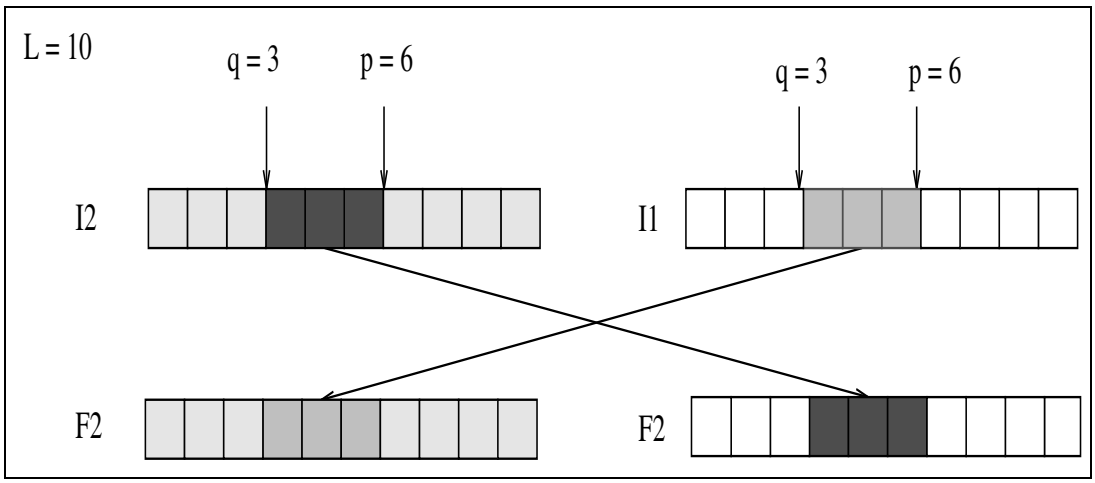

Figura 4.4: *crossover* de dois ponto

Seguindo a mesma idéia do *crossover* de dois pontos, podemos ter *crossover* de vários pontos de corte, conhecido com o *crossover* multiponto. Entretanto, é de se questionar a eficiência de um operador multiponto, uma vez que com a troca de muitas janelas entre os indivíduos selecionados para cruzamento, pode-se facilmente perder blocos da cadeia que contribuem para a qualidade das soluções já obtidas. Num caminho inverso ao operador *crossover* multiponto, pode-se trabalhar no sentido de se aplicar técnicas de mineração sobre as cadeias que codificam os indivíduos da população no intuito de identificar aqueles blocos de cadeia mais comuns nos melhores indivíduos da população, para assim, guiar a convergência do

AG.

- **Operador de Mutação**

Na natureza, indivíduos podem sofrer alterações genéticas que não se dariam apenas por meio do cruzamento entre seus descendentes. Estas alterações ocorrem através da modificação do material genético ocorridas devido a fenômenos diversos. Tais alterações são denominadas mutações, e em alguns casos pode provocar no indivíduo um ganho considerável em termos de capacidade de adaptação ao meio e, por consequência, capacidade de transmitir suas características para seus descendentes.

Em um AG, a ocorrência de mutação se dá segundo uma probabilidade ρ dada como parâmetro de entrada. O mecanismo é normalmente acionado cada vez que um descencente é gerado pelo operador de cruzamento e é aplicado em cada elemento da cadeia que codifica este descendente. A alteração imposta à cadeia original depende da codificação utilizada. Para uma codificação binária, a mutação de um elemento da cadeia implica simplesmente em invertê-lo de 0 para 1 ou vice-versa. Já para uma representação inteira ou real, as alterações devem ser planejadas de forma a garantir que a cadeia do indivíduo sofra perturbações significativas mas que não violem qualquer restrição que a modelagem possa apresentar.

Como ilustração, consideremos um indivíduo I cuja codificação é binária, gerado do cruzamento entre dois outros indivíduos da população selecionados pelo operador de seleção. Sendo a cadeia de I dada por [0101110101], para cada elemento desta cadeia será gerado um número aleatório p entre 0 e 1 que definirá a probabilidade de aceitação. Nos casos onde $p \geq \rho$, o elemento será invertido de zero para um ou vice-versa. Assim, caso esta situação se verifique para o terceiro elemento de I, teríamos um novo indivíduo I_{mut}, cuja cadeia seria [01**1**1110101]. Vê-se então que, mesmo o cruzamento não conseguindo gerar descendentes melhores que os pais, isto torna-se possível através do operador de mutação.

- **Operador de Inversão**

Além dos operadores já citados, podemos encontrar ainda o operador de inversão,

que consiste em aplicar mutação em todos os genes do indivíduo com probalilidade 1. Entretanto, em seres vivos esta técnica demonstrou ser inviável, gerando descendentes com características letais [96]. Em termos de AGs, o uso de operadores de inversão está restrito à algumas aplicações, fato que contribui para a dificuldade de se encontrar referências sobre o emprego deste operador. Para exemplificar o funcionamento da inversão, tomando o mesmo indivíduo $I = [0101110101]$ apresentado anteriormente, após o uso deste operador, teríamos o indivíduo $I^{-1} = [1010001010]$.

Vê-se então que o operador de inversão altera radicalmente a codificação do indivíduo, o que pode levar a perdas de blocos da cadeia original com características promissoras já incorporadas à população através de outros operadores. No entanto, é razoável aceitar a possibilidade de aplicar inversão aos piores indivíduos desta população apenas nas primeiras gerações do AG. Embora nada se possa garantir em termos de obtenção de ganhos, o uso deste operador pode constituir-se numa boa estratégia para algumas aplicações.

4.2 Considerações sobre *Simulated Annealing*

A técnica de *Simulated Annealing* é baseada em um processo usado na área da Física pelo qual a matéria é aquecida e em seguida resfriada contínuas vezes até atingir o congelamento, um estado de estabilidade de energia das partículas que a compõem. Entretanto, a manutenção das propriedades estruturais do sólido durante o processo dependem da taxa em que varia a temperatura durante o processo de aquecimento/resfriamento.

Publicado em 1953, o algoritmo de Metropolis *et al.* [80] simula a mudança na energia do sistema durante o processo de recozimento até que o sistema convirja para o estado estável. O uso desta técnica em problemas de Otimização Combinatória foi inicialmente proposto por Kirkpatrick *at al.* [68].

A técnica de *Simulated Annealing* usada na Física pode ser mapeada para a metaheurística *Simulated Annealing* usada na solução de problemas de Otimização associando os estados do sistema ao conjunto de soluções do problema; a energia do

sistema, ao custo da solução; a mudança de estado, à exploração da vizinhança; a temperatura do sólido, a um parâmetro de controle; e o estado final do congelamento, à solução obtida para o problema.

Muitas heurísticas de busca local apresentam dificuldades para sair de ótimos locais. Entretanto, o que se espera é que a mesma seja capaz de encontrar soluções de boa qualidade independente da solução inicial. Diante disso, para solucionar este impasse, pode-se imediatamente pensar em aplicar uma heurística de busca local sobre várias soluções iniciais. No entanto, neste caso, deparamo-nos com o problema de não dispor de um critério de parada que independa do problema e/ou do conjunto de entrada. O *Simulated Annealing* evita falhas comuns em heurísticas de busca local tradicionais que, para muitos problemas, não conseguem sair de ótimos locais ainda distantes de um ótimo global. Para isso, as soluções obtidas pela heurística de busca local são aceitas pelo algoritmo com probabilidade $p(\delta E)$, dada pela Equação 4.1, podendo assim, sair de uma solução atual para outra solução que não necessariamente implique em ganho na função objetivo.

$$p(\delta E) = expo(-\delta E/kT) \tag{4.1}$$

Na Equação 4.1, k é uma constante física conhecida como *constante de Boltzmann* e T é a temperatura atual do sistema. A mudança na temperatura do sistema é feita segundo uma função de redução $\alpha(T)$.

O que o algoritmo de Metropolis *at al.* faz é gerar perturbações através da mudança da temperatura e calcular o impacto destas perturbações na energia do sistema, caso a energia tenha uma queda, a perturbação é aceita com probabilidade 1. Caso contrário, a aceitação é feita segundo uma probabilidade calculada pela equação 4.1, o que quer dizer que, mesmo havendo um aumento na energia das partículas da matéria submetida ao recozimento, este novo estado pode ser aceito como estado corrente. Trazendo para o campo da Otimização Combinatória, isto difere das técnicas de busca conhecidas como *descent strategies*, que aceitam apenas as soluções que impliquem em ganho na função objetivo.

Para um melhor entendimento do *Simulated Annealing*, a Figura 4.5 apresenta o

pseudocódigo desta técnica para um problema de minização (vide linhas 9 e 10). Os parâmetros do algoritmo são o conjunto de soluções viáveis e o número de iteração. Podemos notar, no entanto, que outros parâmetros precisam ser definidos, como a temperatura inicial, a condição de parada e a função de decremento da temperatura. Na linha 13 é feito o teste de aceitação da solução obtida no caso da mesma ter um custo maior que a atual.

```
1.  proc Algotimo_Simulated_Annealing (S, N_ITERA)
2.     Selecione uma solução s_0 ∈ S;
3.     Selecione uma temperatura t > 0;
4.     Selecione uma função de redução da temperatura α(t);
5.     repita
6.        repita
7.           Selecione uma solução s ∈ S;
8.           δ = f(s) − f(s_0;
9.           se δ < 0 então
10.             s_0 = s;
11.          senão
12.             gere randomicamente x no intervalo [0, 1];
13.             se x < exp(−δ/t) então
14.                s_0 = s;
15.       até numero de iteração = N_ITERA;
16.       t = α(t);
17.    até condicão de parada igual verdadeiro;
18.    retorne s_0;
19. fim Algoritmo;
```

Figura 4.5: Algoritmo *Simulated Annealing* genérico

4.3 O AEC - Algoritmo Evolutivo Construtivo para o PCA

Tendo sido apresentados os conceitos básicos referentes a Algoritmos Genéticos, podemos agora entender o funcionamento do AEC - Algoritmo Evolutivo Construtivo para o Problema de Clusterização Automática. O AEC é composto por dois módulos: o módulo de construção, que procura gerar indivíduos iniciais que apresentem boa qualidade, refletida através de valores elevados para a função de aptidão de cada elemento da população, um módulo de busca local e o AG propriamente dito, que utiliza, além dos operadores genéticos tradicionais, um operador de mutação desenvolvido especificamente neste trabalho.

O AEC difere da maioria dos AG's existentes para o Problema de Clusterização, uma vez que é capaz de resolver o problema de agrupamento e também resolver o problema de encontrar o número ideal de clusters.

4.3.1 A Fase de Construção

A fase de construção utilizada pelo AEC pode ser vista como uma estratégia desenvolvida para lidar com clusterização em grandes bases de dados, como é o caso da maioria das aplicações reais. Além disso, a estratégia aqui proposta pode reduzir consideravelmente os tempos computacionais exigidos por um AG. De fato, sabe-se que o tamanho da cadeia que codifica um indivíduo em um AG pode influir no tempo de processamento do mesmo. Assim, dependendo da codificação utilizada para o Problema de Clusterização, pode ser interessante substituir grupos de objetos cuja similaridade é considerada alta, por um único objeto que represente o grupo.

Ao apresentarmos uma fase de construção, nosso propósito é permitir que o módulo genético do AEC tenha o tempo de processamento abreviado através da geração de uma população inicial cujos indivíduos apresentem alta qualidade, evitando que o algoritmo desperdice tempo de processamento tentando encontrar os melhores indivíduos em relação a um ótimo global partindo de soluções iniciais muito pobres.

A construção das soluções é realizada agrupando-se os pontos em clusters iniciais que expressem as características da distribuição dos dados do conjunto de entrada S. Para isso, a visão global de vizinhança de um algoritmo de clusterização torna-se um fator importante na decisão de qual cluster é ideal para se inserir cada ponto do conjunto de entrada.

Conforme já visto no Capítulo 2, podemos tomar cada objeto x_i do conjunto de entrada S como uma tupla $(x_{i1}, x_{i2}, \ldots, x_{ip})$, onde cada coordenada x_{ij} está relacionada com um atributo do objeto, permitindo-nos, então, considerar um objeto como um ponto no espaço R^p. A substituição de um conjunto C de t pontos similares por um único ponto C_m que os represente pode ser feita tomando-se o centróide do conjunto C, dado pela Equação 3.1.

Constantemente vemos na literatura uma associação entre o Problema de Clusterização e o Problema de Particionamento em Grafos [8, 13, 31]. Se considerarmos, porém, cada ponto do conjunto de entrada como um nó em um grafo, o custo computacional não será muito diferente se optarmos por uma ou por outra abordagem. Entretanto, sabemos que em bases de dados elevadas é comum termos regiões densas do espaço p-dimensional separadas por regiões esparsas contendo poucos ou nenhum ponto.

Considere então os pontos do conjunto de entrada S como vértices de um grafo construído conforme o conceito de Grafo de Vizinhança Relativa - GVR [112]. O Grafo de Vizinhança Relativa do conjunto S, denotado $GVR(S)$, é o grafo associado à matriz de adjacência M dada por:

$$\mathbf{M[i,j]} = \begin{cases} 1 & \text{se } d(i,j) \leq max[d(x_i, x_k), d(x_j, x_k)], \forall x_k \in S, k \neq i, j \\ 0 & \text{caso contrário} \end{cases} \tag{4.2}$$

onde $d(i,j)$ denota a distância euclidiana entre os ponto x_i e x_j dada pela Equação 2.6 e todos os objetos i são considerados.

A Equação 4.2 diz que dois pontos x_i e x_j são ditos *vizinhos relativos* ($M[i,j] = 1$) se não existir nenhum outro ponto x_k cuja distância em relação x_i e x_j seja inferior a $d(i,j)$. O $GVR(S)$ é o grafo G(V,E) onde o conjunto de vértices V é formado por todos os pontos do conjunto S e o conjunto de arcos E é formado por todos os arcos

e_{ij} cujo valor de $M[i,j]$ é igual a 1.

Como podemos observar, os componentes conexos de $GVR(S)$ são agrupamentos que refletem uma visão de vizinhança mínima, ou seja, considerando m como sendo o número de componentes conexos obtidos de $GVR(S)$, podemos esperar, como número máximo de clusters o próprio m e como número mínimo 2, já que a clusterização que gera o próprio S como solução não nos interessa.

Para concluir a fase de construção do AEC, tomamos os componentes conexos de $GVR(S)$ e obtemos o conjunto $G = \{G_1, G_2, \ldots, G_m\}$, onde cada G_i é um componente conexo candidato a se tornar um cluster inicial, e V_i, para $i = 1, \ldots, m$ é o centróide do componente conexo G_i. Devemos, então, efetuar os agrupamentos possíveis entre $G_{i's}$ de forma a termos a clusterização final. Isto é feito no sentido de otimizar uma função objetivo $f(.)$, tarefa realizada pelo módulo evolutivo do AEC.

Antes de explicarmos como se constrói a população inicial do AEC, é importante entendermos a modelagem utilizada para a codificação do indivíduo. Para tanto, considere uma cadeia binária $r = (r_1, r_2, \ldots, r_m)$, representando uma solução (semente). Se $r_i = 1$, a componente conexa $G_i \in G$ fará parte da solução como cluster pai (cluster raiz). Caso contrário ($r_i = 0$), G_i é considerado um cluster filho e será associado posteriormente a um dos clusters pais indicados pelos r'_is que constam na cadeia como sendo iguais a 1. Como exemplo, sobre a cadeia $r = [01010001]$ de uma entrada que gerou um conjunto $G = \{G_1, G_2, G_3, G_4, G_5, G_6, G_7, G_8\}$, onde cada G_i é um componente conexo de $GVR(S)$, podemos dizer que a solução apresentará três clusters, inicialmente G_2, G_4 e G_8 (clusters pais). Os demais clusters (G_1, G_3, G_5, G_6 e G_7) são os clusters filhos, cujos pontos serão anexados a um dos clusters pais para formar com este um novo cluster conforme o critério de similaridade adotado.

Na geração da cadeia que codifica uma solução, procuramos priorizar os componentes conexos com maior cardinalidade. Assim, a geração da população inicial do AEC se dá de tal forma que a probabilidade de ocorrer o valor 1 em cada elemento r_i da cadeia r é proporcional à cardinalidade do componente conexo $G_i \in G$. Entretanto, como podem existir bases de dados com pontos que não são similares a nenhum outro ponto do conjunto de entrada (*outliers*), ou ainda, entradas que apre-

sentam clusters com densidades muito variadas, optamos por usar esta abordagem com uma probabilidade $\rho = 0.75$, ficando 25% de possibilidade do indivíduo ter toda a sua cadeia gerada de forma totalmente aleatória. Esta estratégia funciona como o método de seleção denominado *Método da Roleta*, já mencionado anteriormente. Entretanto, a roleta aqui utilizada tem o propósito único de dar mais possibilidade aos componentes conexos mais densos de surgirem como clusters pais no processo de clusterização.

Para a clusterização ser concluída, utilizaremos o módulo evolutivo do algoritmo. Para isso, faremos um detalhamento do funcionamento do AEC no que se refere aos operadores utilizados e às estratégias utilizadas para tirar proveito da geração de componentes conexos para a construção de uma população de indivíduos cujas aptidões apresentam boas chances de se aproximarem de uma solução ótima. Finalmente, mostraremos como uma busca local baseada na técnica de Reconexão por Caminhos (Path Relinking) é incluída no AEC.

4.3.2 O Módulo Evolutivo do AEC

O módulo evolutivo do AEC compõe-se de um Algoritmo Genético cuja população inicial é gerada segundo as densidades dos componentes conexos obtidos na fase de redução de cardinalidade, conforme visto anteriormente. Como podemos ver, o número de $1's$ na cadeia do indivíduo fornecerá o número de clusters da solução gerada pelo mesmo. Assim, podemos ver facilmente que dentro do intervalo $[1, m]$, com m dado pelo número de componentes conexos, o número ideal de clusters a ser encontrado pelo algoritmo pode ser bem menor quando comparado ao número de combinações possíveis dentro do intervalo $[1, n]$ dado pela cardinalidade do conjunto de entrada S. Isto sem dúvida é um fator estimulante para se explorar as características dos componentes conexos obtidos, de forma a evitar que o AG se detenha a avaliar soluções que estejam muito distantes de um ótimo global.

A possibilidade de tendermos a um ótimo local ainda distante de um ótimo global com esta estratégia não chega a afetar o desempenho do algoritmo, uma vez

que além de não abandonarmos a geração aleatória, foi desenvolvido um operador de mutação especial que oferece uma diversificação considerável da população e, assim, o algoritmo consegue cobrir de maneira bastante eficiente as regiões do espaço de busca que porventura tenham sido excluídas na fase de geração dos indivíduos.

Agora, apresentamos o processo de formação dos clusters, ou seja, a transformação de uma semente em solução propriamente dita. Para tanto, após a geração da semente, considere $Y = \{G_1, G_2, \ldots, G_t\}$ o conjunto formado pelos componentes conexos $G_{i's}$ associados aos elementos cujo valor é 1 na cadeia do indivíduo. Uma solução $C = \{C_1, \ldots C_t\}$ é inicialmente formado pelos t clusters iniciais $G_{i's} \in Y$ e os centróides iniciais $H_{i's}$ dos clusters da solução são, inicialmente, os centróides $V_{i's}$ destes clusters iniciais, onde $|C_i| = |G_i|$ para $i = 1, 2, ..., t$ e $|G_i|$ denota o número de pontos pertencentes a G_i.

No processo de clusterização, os componentes $G_{i's}$ em $Z = \{G_1, G_2, ..., G_m\} - Y$ são analisados um de cada vez de tal forma que o cluster $C_j \subset C$ irá anexar os pontos do componente conexo $G_i \subset Z$ se $\|V_i - H_j\| \leq \|V_i - H_q\|$ para $1 \leq q \leq t$ e $q \neq j$. Quando algum G_i é associado a um cluster C_j, a nova cardinalidade de C_j, denotado por $|C'_j|$, e o novo centróide de C_j, denotado por H'_j são recalculados pelas equações 4.3 e 4.4, respectivamente.

$$H'_j = \frac{H_k * |C_j| + V_i * |G_i|}{|C_j| + |G_i|} \tag{4.3}$$

$$|C'_j| = |C_j| + |G_i| \tag{4.4}$$

Após todos os componentes conexos cujos índices na cadeia apresentam valor zero terem sido associados aos clusters pais indicados pelos índices cujo valor é igual 1, a clusterização está concluída e podemos então avaliar a qualidade da mesma. Para isso, utilizamos o critério apresentado em [67], conhecido como média da *silhueta* dos pontos do conjunto de entrada.

Antes de explicarmos os mecanismos dos operadores do AEC, definimos a função de aptidão calculada sobre a solução associada a um determinado indivíduo. Para tanto, seja i um ponto pertencente ao cluster C_w construído na clusterização, onde $|C_w| = M > 1$. Considere a dissimilaridade média de i em relação a todos os

pontos $j \in C_w$ dada pela Equação 4.5, onde $d_{i,j}$ é a distância euclidiana descrita pela Equação 2.6.

$$a(i) = \frac{1}{M-1} \sum_{j=1}^{M} d_{i,j} \qquad \forall j \neq i \tag{4.5}$$

Nos casos em que C_w possuir um único elemento, definimos $a(i) = 0$.

Considere ainda, cada um dos clusters $C_k \in C$ com $k \neq w$. A dissimilaridade média do ponto i em relação a todos os pontos de C_k é mostrada na Equação 4.6, onde M é o número de pontos do cluster C_k.

$$d(i, C_k) = \frac{1}{M} \sum_{i=1}^{M} d_{i,j} \qquad \forall j \in C_k \tag{4.6}$$

Denota-se $b(i)$ o menor valor dentre todos os $d(i, C_k)$, o que é obtido pela Equação 4.7. Note que $b(i)$ é obtido pelo cluster que seja o vizinho mais próximo do ponto $i \in C_w$

$$b(i) = \min_{C_k \neq C_w} d(i, C_k) \tag{4.7}$$

Definimos agora o valor da silhueta do ponto i, dada por:

$$s(i) = \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)} \tag{4.8}$$

Para avaliarmos a qualidade de uma clusterização, calculamos a média da silhueta $s(i)$ de todos os n pontos do conjunto S. Desta forma, a aptidão (*fitness*) de um indivíduo é dada por:

$$max(f) = \frac{1}{n} \sum_{i=1}^{n} s_i \tag{4.9}$$

sujeito a

$$i \in S \tag{4.10}$$

Repare que em nenhum momento precisamos definir o número k de clusters a ser obtido. Isto porque o valor mais adequado para k é atingido ao otimizarmos a função objetivo [67].

Definida a função objetivo, consideramos agora uma população de indivíduos codificados como uma cadeia binária de tamanho igual ao número de componentes conexos obtidos de $GVR(S)$. Após a fase de construção da população, todos os

indivíduos (que ainda estão na fase de semente) são levados a gerar uma solução para o PCA conforme foi apresentado anteriormente. Depois da clusterização ter sido concluída, calculamos o valor da aptidão de cada indivíduo. Para efetuar a seleção dos indivíduos para cruzamento, o operador utilizado é o Método da Roleta clássico. Entretanto, como a equação que define a aptidão do indivíduo admite valores dentro do intervalo $[-1, 1]$, portanto podendo assumir valores negativos, aumentamos em uma unidade o valor da aptidão f de cada indivíduo. Ao final do processamento, subtraímos uma unidade do valor de f do indivíduo campeão.

O operador de *crossover* utilizado no AEC é o *crossover* de dois pontos, que atua sobre dois indivíduos com genótipos diferentes. Os indivíduos são submetidos ao *crossover* com uma probabilidade p_c. Os dois pontos de corte definem os segmentos das cadeias dos pais que serão trocados entre os mesmos para gerar novos indivíduos. Para ilustrar o funcionamento do operador de *crossover* de dois pontos para a representação de indivíduo utilizada, considere os dois indivíduos $I_1 = 1100010001$ e $I_2 = 0011100110$ e os dois pontos de corte escolhidos aleatoriamente $p = 2$ e $q = 5$. Os segmentos (ou janelas) a serem trocados são $w_{1_{p,q}} = 1000$ e $w_{2_{p,q}} = 0111$. Após o cruzamento, os descendentes gerados de I_1 com I_2 são $F_1 = 0\mathbf{1000}01010$ e $F_2 = 1\mathbf{0111}10001$.

Algumas observações podem ser feitas sobre este exemplo, dando-nos uma visão da eficiência deste operador para a obtenção do número k de clusters. Primeiro, pela codificação dos pais, podemos ver que na fase de construção do algoritmo foram gerados 10 componentes conexos. Além disso, o indivíduo I_1 apresenta uma clusterização formada por quatro clusters (número de 1's na cadeia), enquanto o indivíduo I_2 gera uma outra clusterização, composta de cinco clusters ($k = 5$). Após o cruzamento, verificamos que podemos gerar indivíduos que apresentam $k = 3$ (F_1) ou $k = 6$ (F_2), diversificando satisfatoriamente a população inicial. Como foi apresentado em [67], o valor de k mais adequado resulta da clusterização com maior valor obtido pela Equação 4.9. Desta forma, partindo de uma população cujos indivíduos já representam soluções de boa qualidade, podemos abreviar a obtenção da melhor clusterização logo nas primeiras gerações do AEC.

```
1. proc AEC(S, LENG_POP, N_GERA, P_CROSS, P_MUT, FLAG_PATH)
2.    Gera_GVR (S);
3.    Gera_Comp_Conexo (GVR (S));
4.    Populacao = Gera_Solucao(LENG_POP);
5.    para k=1 ate N_GERA faça
6.       i = 1; flag_mut = falso;
7.       enquanto i < LENG_POP / 2 faça
8.          Seleciona (p1,p2);
9.          q = random(100);
10.         se (q >= P_CROSS * 100)  então
11.            i = i+1;
12.            se crossover(p1,p2,P_ CROSS,f1,f2) então
13.               Mutacao (f1,f2,P_ MUT, flag_ mut);
14.               not(flag_ mut);
15.               se (Avalia_ Solucao(f1,f2)) então
16.                     Insere_ best(f1,f2);
17.               fim se
18.            fim se
19.         fim se
20.      fim enquanto
21.      se FLAG_PATH = 1 então
22.         Path_ Relinking ();
23.      fim se
24.      Atualiza_ Pop();
25.   fim para
26.   retorna (Melhor_ Solucao;)
27. fim  AEC
```

Figura 4.6: Pseudocódigo do AEC

Foram desenvolvidos dois operadores de mutação para o AEC cujo emprego é alternado a cada cruzamento. O primeiro é o operador tradicional, no qual cada elemento da cadeia pode sofrer mutação com probabilidade igual a p_m, passando do

valor zero para um, ou vice-versa.

O segundo operador de mutação funciona através de sobreposição de janelas. Para exemplificar seu funcionamento, considere o indivíduo F_1 gerado do *crossover* apresentado anteriormente. Dentro do intervalo $[1, |F_1|]$, onde $|F_1|$ é o tamanho da cadeia do indivíduo, são escolhidos aleatoriamente dois pontos de corte p e q, com $p < q$, e um ponto de inserção t. Os dois pontos de corte definem uma janela $w_{p,q}$, como no operador *crossover*. Definida a janela, um outro indivíduo é gerado a partir de F_1 da seguinte forma: considerando a cadeia de F_1 como uma lista circular, a partir do ponto de inserção t, a janela $w_{p,q}$ será inserida na cadeia original de F_1, gerando um novo indivíduo $F_1(mut)$. Tomemos como exemplo os pontos $p = 3$, $q = 7$ e $t = 6$. Teríamos a seguinte situação: $F_1 = 0100001010$, $w_{3,7} = 00001$ e $t = 6$. Substituindo $w_{3,7}$ na cadeia original de F_1 a partir da sexta posição, um novo indivíduo $F_1(mut) = 01000\mathbf{00001}$ é obtido. A Figura 4.7 mostra o efeito deste operador para diferentes valores de t.

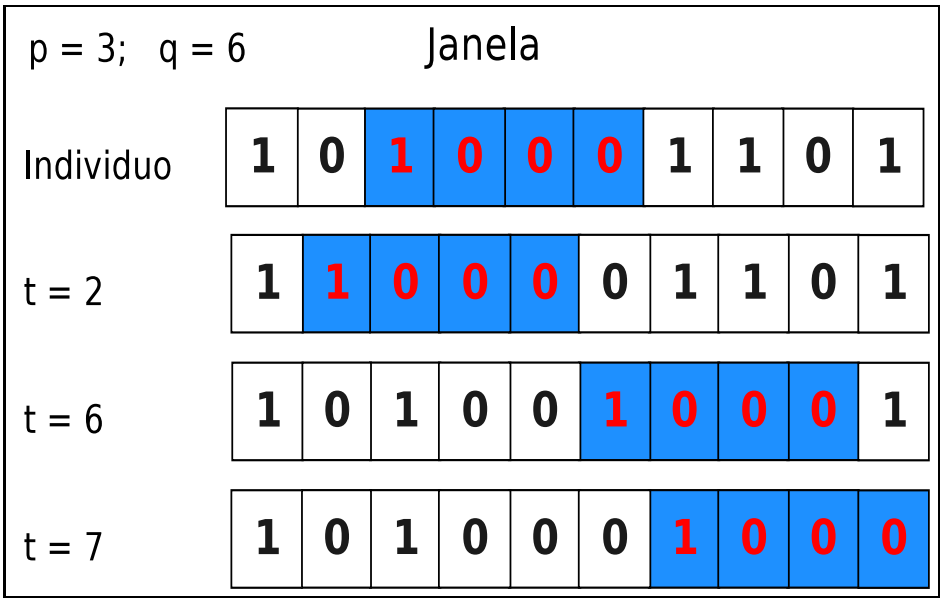

Figura 4.7: Mutação por sobreposição de janelas

Após a aplicação dos operadores de *crossover* e mutação, cada descendente é inserido na população caso o valor obtido pela função de aptidão seja superior ao indivíduo da população atual que apresenta a menor aptidão. Além disso, a cada geração, a taxa de reposição é de 95%, ou seja, apenas 5% da população é mantida

para a próxima geração.

A Figura 4.6 mostra o pseudocódigo do AEC. Na linha 2 é feita uma chamada ao método de construção do Grafo de Vizinhança Relativa que toma como parâmetro o conjunto S de pontos dado como entrada. Em seguida, na linha 3, o método $Gera\_Comp\_Conexo$ recebe o GVR e constrói os componentes conexos que serão utilizados para a geração da população inicial do algoritmo, o que é feito na linha 4. Os passos entre as linhas 5 e 25 efetuam a execução das gerações do AEC, cujo número é indicado pelo parâmetro N_{GERA}. O número de cruzamentos em cada geração foi fixado como a metade do tamanho da população, ou seja, $\frac{LENG\_POP}{2}$. Na linha 13 faz-se a chamada ao método de mutação, que recebe um flag ($flag\_mut$) que indica qual dos dois operadores de mutação será acionado. Na linha seguinte, este flag é alterado para permitir a alternância destes operadores a cada cruzamento.

Ainda sobre Figura 4.6, a linha 15 indica a chamada da função de avaliação dos indivíduos gerados pelo *crossover* seguida de um dos operadores de mutação. Esta avaliação consiste em gerar a solução associada a cada uma dos descendentes $f1$ e $f2$ e calcular o valor da função de aptidão dada pela Equação 4.8. No caso dos indivíduos descendentes apresentarem aptidão superior aos dois piores indivíduos da população, estes serão substituídos, o que é feito na linha 16.

4.3.3 A Reconexão por Caminhos do AEC

O uso de busca local aplicada em Algoritmos Genéticos tem encontrado bastante aceitação por parte dos pesquisadores desta área, fazendo crescer o interesse no desenvolvimento de Algoritmos Evolutivos. Neste contexto, a Reconexão por Caminhos (RC) proposta por Glover et al. [44] apresenta-se como uma técnica de busca alternativa em relação às buscas tradicionais.

A RC consiste em traçar o caminho entre uma solução base e uma solução destino (alvo) que apresentam boa qualidade e avaliar as soluções intermediárias obtidas ao longo do trajeto com o intuito de tentar encontrar soluções melhores que a origem e destino. O princípio básico da RC é o de que entre duas soluções de qualidade pode

existir uma terceira melhor que os extremos.

Existem várias questões envolvendo o uso de Reconexão por Caminhos, como por exemplo, o sentido da trajetória adotado. Neste caso, dadas as soluções s_1 e s_2, pode-se percorrer o caminho de s_1 para s_2, o caminho inverso, ou ainda, pode-se percorrer as duas trajetórias simultaneamente. Além disso, pode-se optar por não percorrer a trajetória completa em qualquer dos sentidos.

A versão do AEC que incorpora Reconexão por Caminhos, denotada AEC-RC, adota o caso em que se percorrem os dois sentidos não simultaneamente. Optamos por suprir o algoritmo com esta técnica de busca local como forma de aprimorar a capacidade do mesmo de não falhar nas instâncias consideradas mais difíceis, como será mostrado no capítulo 5, onde apresentamos os resultados obtidos. No entanto, sabemos que os tempos de processamento podem sofrer acréscimos significativos em alguns casos, mas se avaliarmos a robustez como uma característica essencial em um algoritmo de clusterização, não podemos dispensar este recurso mesmo que seu diferencial seja mais explicitamente relevante apenas nas instâncias consideradas mais difíceis

Na figura 4.6, a condição colocada na linha 21 seleciona ou rejeita a função que aciona o método de Reconexão por Caminhos. Por último, na linha 24, a população é atualizada de tal forma que, apenas os melhores indivíduos (até um total de 5% de LENG_POP) sobrevivem para a próxima geração, como já mencionado. Após ter executado todas as gerações, a solução representada pelo indivíduo de maior aptidão é retornada.

4.4 O Algoritmo SAPCA - *Simulated Annealing* para o PCA

Além do AEC, desenvolvemos uma outra abordagem não-evolutiva para o PCA. O algoritmo SAPCA - *Simulated Annealing* para o PCA - é um algoritmo que, mesmo não sendo uma versão da técnica conhecida como *Simulated Annealing*, foi

assim batizado tendo em vista que as idéias aplicadas no mesmo para resolver o Problema de Clusterização foram inspiradas nesta técnica.

O SAPCA difere do *Simulated Annealing* tradicional em alguns aspectos fundamentais. Enquanto o *Simulated Annealing* usa a variação da qualidade da solução atual (energia do sistema) na função de aceitação de uma solução vizinha para uma dada temperatura t, conforme a equação 4.1, o SAPCA tem um comportamento guloso na aceitação das soluções vizinhas, ou seja, o SAPCA é um representante da já mencionada técnica de busca local conhecida como *descent strategies*. Assim, no SAPCA, uma solução obtida pelas estruturas de busca local somente será aceita caso haja um aumento na qualidade da solução. Além disso, a temperatura inicial do sistema a cada iteração do algoritmo é sempre igual à energia inicial do mesmo. Como a nossa função objetivo é de maximização, teremos a temperatura caindo à medida que a energia do sistema aumenta. Assim, a principal diferença é na função de probabilidade $p(\delta E)$ de aceitação da solução no processo de busca, que no caso do SAPCA é dada por $p(\delta E) = 0$.

Podemos pensar que o fato de se aceitar a solução de forma gulosa pode levar o SAPCA a prender-se em ótimos locais ainda distantes de um ótimo global. Entretanto, deve-se levar em consideração que a eficiência da heurística de construção evita o desperdício de iterações na avaliação de regiões do espaço de busca consideradas pouco promissoras. O SAPCA resolve este problema acionando a estratégia de busca local para várias soluções iniciais obtidas pelo módulo de construção. Entretanto, isso poderia ocasionar o problema do algoritmo não ter um critério de parada que não sofra influência da natureza do conjunto de entrada. Para tanto, usamos como iteração do algoritmo o resfriamento do sistema, que tem a temperatura inicial t dada pelo valor da função de aptidão aplicada à solução inicial. Desta forma, o algoritmo é executado até que t seja igual a zero, sendo que, a cada iteração, uma solução inicial é obtida através do módulo de construção e a temperatura é novamente calculada.

A Figura 4.8 mostra o pseudocódigo do algoritmo SAPCA. O Algoritmo recebe como parâmetros o conjunto de pontos S e as constantes R_1 e R_2 usadas nas funções

de redução da temperatura. Inicialmente, usando a mesma heurística de construção empregada no AEC, o algoritmo gera uma solução e a avalia para obter a temperatura inicial do sistema, dada pelo valor obtido pela Equação 4.8, que é armazernado na variável T_1, conforme mostra a linha 4. A solução a ser retornada pelo algoritmo é sempre armazenada na variável $Best$, que inicialmente recebe a solução obtida pela heurística de construção através do comando da linha 5.

Note que a Figura 4.8 apresenta dois laços de repetição aninhados. No laço mais externo, são feitas as chamadas à busca local $Window\_Insert$ e no laço mais interno, são feitas as chamadas à busca local Not_Window. As duas variáveis T_1 e T_2, que armazenam valores obtidos segundo a Equação 4.8, são usadas nos critérios de parada dos laços externo e interno, respectivamente. No laço mais externo, a cada iteração o decremento do valor de T_1 se dá segundo a função de resfriamento dada por $\alpha_1(T_1) = T_1 - (T_1/R1)$, conforme a linha 20. Por outro lado, o valor de T_2 é reduzido segundo a função $\alpha_2(T_2) = T_2 - (T_2/R2)$, conforme a linha 18. As linhas 19 e 22 testam os critérios de parada, onde usamos a constante APROX para multiplicar pelo valor de T_1 e T_2. Uma vez que os valores obtidos pela Equação 4.8 estão no intervalo [-1, 1], nós adicionamos uma unidade ao valor inicial de T_1 e T_2, embora não conste no código apresentado na Figura 4.8.

```
1.  proc SAPCA(S, R1, R2, APROX )
2.     Gera_GVR (S);
3.     Sol_ Temp = Gera_ Solucao (GRV);
4.     T1 = Avalia (Sol_ Temp);
5.     Best = Sol_ Temp;
6.     repita
7.        Sol_ Temp = Busca_ Window_ Insert (Sol_ Temp);
8.        se (Avalia (Sol_ Temp) > Avalia (Best)) então
9.           Best = Sol_ Temp;
10.       fim se;
11.       T2 = Avalia (Best);
12.       Sol_ Temp = Best;
13.       repita
14.          Sol_ Temp = Busca_ Not_ Window (Best);
15.          se (Avalia (Sol_ Temp) > Avalia (Best)) então
16.             Best = Sol_ Temp;
17.          fim se;
18.          T2 = α2(T2);
19.       até (T2 * APROX ≥ 1)
20.       T1 = α1(T1);
21.       Sol_ Temp = Gera_ Solucao (GVR);
22.    até (T1 * APROX ≥ 1)
23.    retorna (Best;)
24. fim SAPCA
```

Figura 4.8: Pseudocódigo do SAPCA

Cada vez que uma nova iteração do laço externo inicia, a busca $Window\_Insert$ parte de uma nova solução, obtida na linha 21. Isto oferece ao algoritmo a possibilidade de sair de ótimos locais. Além disso, o número de vezes em que a busca local Not_Window é acionada sobre a mesma solução inicial é muito superior à $Window\_Insert$, visto que na primeira não há chamada ao procedimento de construção ($Gera\_Solução(GVR)$), como ocorre na $Window\_Insert$.

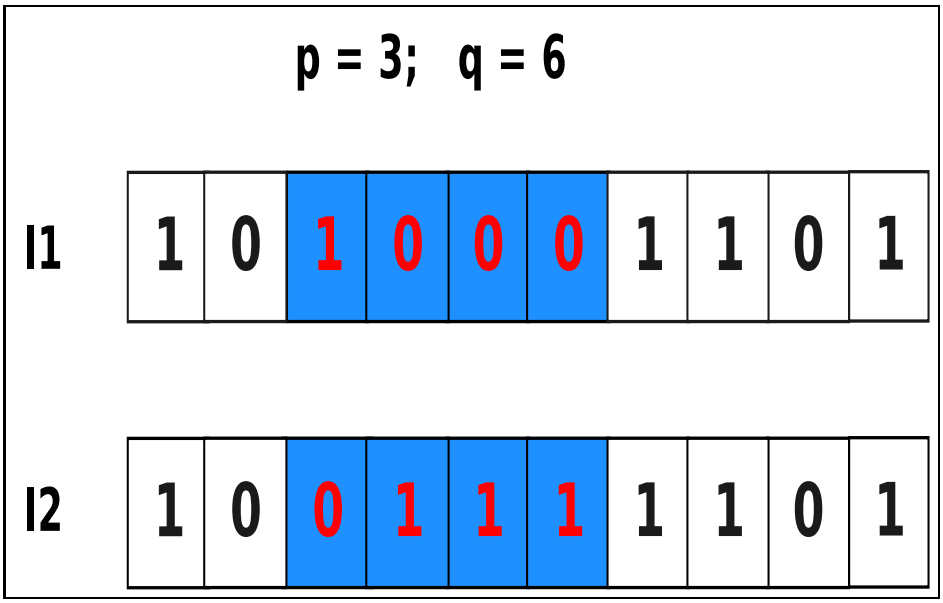

Figura 4.9: Busca local *Not_ Window*

O primeiro procedimento de busca, chamado $Window\_Insert$ é semelhante ao operador de mutação por sobreposição de janela apresentado na figura 4.7. A outra estrutura de busca, chamada Not_Window, é baseada na inversão de janelas. Este segundo método de busca local consiste em tomar uma janela da *string* solução r (obtida tal como na busca $Window\_Insert$) e aplicar uma operação de negação sobre cada elemento da mesma, de tal forma que o elemento que contém valor 0 passa a ter valor 1 e vice-versa. Da mesma forma que a primeira estratégia, a nova solução é avaliada e substituirá a atual caso apresente qualidade superior. Como exemplo, para $r = [0001011110]$, supondo $p = 2$, $q = 6$ obtidos aleatoriamente dentro do intervalo $[1, |r|]$, temos a janela $w_{pq} = [00101]$ e a solução obtida após a aplicação do operador lógico de negação sobre w_{pq} em r é $r' = [0\mathbf{11010}1110]$. Os elementos em negrito indicam aqueles que foram alterados durante a aplicação do procedimento. A Figura 4.9 apresenta o efeito desta busca sobre a cadeia original

do indivíduo.

4.5 O Algoritmo Genético CLUSTERING

Além dos três algoritmos propostos AEC, AEC-RC e SAPCA, implementamos para efeito de comparação, o Algoritmo Genético proposto para o PCA denominado CLUSTERING. Desenvolvido por Tzeng e Yang [116], além dos parâmetros comuns de um AG genérico, este algoritmo recebe como entrada três parâmetros u, ws e wl, que são definidos experimentalmente e são utilizados no processo de obtenção do melhor valor para o número k de clusters. Além disso, o CLUSTERING opera com uma fase de construção baseada em componentes conexos, a exemplo do AEC. No entanto, o grafo de vizinhança G, gerado na fase de construção do CLUSTERING, é criado segundo uma visão de vizinhança baseada na média d_{av} de todas as distâncias d_{ij} entre cada ponto i e seu vizinho j mais próximo.

Assim, uma matriz de adjacência M é construída de forma que $M[i,j] = 1$ se $d_{ij} \leq u * d_{av}$, o que nos faz perceber que, dependendo da distribuição dos pontos do conjunto entrada, o valor de u poderá distorcer completamente a solução já na fase de construção, constituindo um dos gargalos deste procedimento. Construído o grafo de vizinhança associado à matriz M, obtém-se o conjunto $B = \{B_1, \ldots B_m\}$, onde cada B_i é um componente conexo de G. Desta forma, cada componente conexo B_i é um cluster inicial, e denotamos V_i como seu centróide.

A codificação usada pelo CLUSTERING para cada indivíduo da população é a mesma utilizada no AEC: uma cadeia binária r de tamanho m. A ocorrência de 1 no i_ésimo elemento de r indica que o componente conexo B_i iniciará a clusterização como cluster pai. Caso contrário, B_i será associado a um dos clusters pais indicados na cadeia do indidivíduo, como ocorre no AEC. Entretanto, a principal diferença entre o método de construção proposto e o utilizado no CLUSTERING é a probabilidade associada à cardinalidade do componente conexo no instante em que a cadeia que codifica o indivíduo é construída. No caso do CLUSTERING, a probabilidade de ocorrer 1 ou 0 é a mesmal (0.5), não levando em conta que os

componentes conexos de maior cardinalidade são mais significativos. Além disso, a visão de vizinhança do AEC é menos sensível à distribuição dos pontos do conjunto de entrada.

Após o processo de clusterização ter sido concluído, cada indivíduo da população gera uma solução $C = \{C_1, \ldots, C_t\}$, com $1 < t \leq m$. Denota-se S_i o centróide do cluster C_i. Para um melhor entendimento do cálculo da função de aptidão utilizada pelo CLUSTERING, considere um cluster $C_i \in C$ obtido na clusterização. Denota-se $D_{intra}(C_i)$ o somatório das distâncias entre o centróide V_k de cada componente conexo $B_k \in C_i$ ao centróide S_i do cluster C_i multiplicada pela cardinalidade de B_k, dado pela Equação 4.11. Denota-se, ainda, $D_{inter}(C_i)$ como sendo o somatório das distâncias entre o centróide V_k do componente conexo $B_k \in C_i$ mais próximo do centróide S_j do cluster j, para todos os clusters $C_j \in C$, com $j \neq i$, conforme dado pela Equação 4.12.

$$D_{intra}(C_i) = \sum_{B_k \subset C_i} ||V_k - S_i|| * |B_k| \tag{4.11}$$

$$D_{inter}(C_i) = \sum_{B_k \subset C_i} (\min_{j \neq i} ||V_k - S_j||) * |B_k| \tag{4.12}$$

A função de aptidão usada no CLUSTERING é então dada Equação 4.13.

$$max(f) = \sum_{i=1}^{i=k} D_{inter}(C_i) * w - D_{intra}(C_i) \tag{4.13}$$

sujeito a:

$$C_1 \cup \cdots \cup C_k = S \tag{4.14}$$

$$C_i \cap C_j = \emptyset \tag{4.15}$$

onde w será encontrado mediante uma busca binária no intervalo dado pelos parâmetros ws e wl.

Podemos observar pela Equação 4.13 que, quando w assume valores pequenos, dá-se prioridade à uma clusterização formada por muitos clusters de composição mais

compacta, enquanto no caso de w assumir valores elevados, prioriza-se a formação de um número reduzido de clusters cujos pontos encontram-se mais dispersos.

O CLUSTERING segundo seus autores apresenta complexidade $O(n^2 + GNm^2)$, onde n é o número de objetos do conjunto de entrada, G é o número de gerações do AG, N é o número de indivíduos da população e m é o número de componentes conexos encontrados na primeira fase do algoritmo. Entretanto, para obter o número ideal de clusters, o AG é executado várias vezes através de uma busca binária num intervalo dado como parâmetro de entrada.

Os agoritmos implementados foram, portanto, o AEC, o AEC-RC, o CLUSTERING e o SAPCA. Embora nós tenhamos implementado também algoritmos clássicos como o $Kmeans$ e o $Kmedoides$, a análise dos resultados não deve prender-se a estas abordagens, uma vez que o PCA não é tratado pelas mesmas.

Capítulo 5

Resultados Computacionais

Para efeito de comparação dos algoritmos, três formas de avaliação foram usadas e são apresentadas a seguir. Todos os algoritmos foram implementados usando a linguagem C++ com o compilador *gcc* versão 3.2.2 no ambiente **Linux Mandrak** versão 9.1. Nos testes de contagem de tempo foram utilizadas máquinas com processadores **AMD Athlon(TM) 1.3 Ghz** com 256 Mb de memória RAM.

Utilizamos como medida de qualidade de resultado o valor apresentado pela solução de cada algoritmo expresso pela função de aptidão utilizada pelo AEC. Isto porque, segundo Kaufman & Rousseeuw [67], a solução expressa por esta função está tão próxima da solução ideal quanto seu valor se aproximar de 1. Desta forma, uma vez que a função de aptidão usada pelo algoritmo da literatura (CLUSTERING) não se baseia na silhueta do ponto, após o mesmo ter encontrado a solução, esta é avaliada para que seu valor seja expresso com base na Equação 4.9.

Das 17 instâncias mostradas na tabela 5.1, quatro são instâncias conhecidas da literatura e estão destacadas em negrito. Destas quatro, a *RuspiniDataSet*, na tabela denotada RuspiniData, e a instância $200Data$ são artificias no espaço R^2 e estão definidas em [67]. A instância RuspiniData é composta de 75 pontos distribuídos em quatro clusters. Por outro lado, a instância 200Data consiste de uma distribuição normal onde cada objeto o_i é um vetor (μ_x, μ_y) com desvio padrão

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

Tabela 5.1: Características das instâncias usadas

ρ para x e y. A distribuição gera três grupos com a configuração apresentada na tabela 5.2 dada em [67].

grupo 1	120 pontos	$\mu_x = 0$	$\mu_y = 10$	$\rho = 1.7$
grupo 2	60 pontos	$\mu_x = 20$	$\mu_y = 12$	$\rho = 0.7$
grupo 3	20 pontos	$\mu_x = 10$	$\mu_y = 20$	$\rho = 1.0$

Tabela 5.2: Distribuição dos pontos da instância 200Data

Ainda na tabela 5.1, a coluna **Número de grupos** indica o número ideal de clusters nos casos em que esta informação é conhecida ou mesmo quando se pode observar através da imagem dos agrupamentos inerentes à mesma. É importante enfatizar que estas informações não são repassadas aos algoritmos para que estes obtenham este número durante a sua execução. O sinal de interrogação indica as

instâncias em que a quantidade ideal de clusters que pode ser verificada através de imagem no espaço R^2 é dúbia ou subjetiva. Além disso, as instâncias cujos nomes não são destacados em negrito foram geradas artificialmente. Para a geração de tais instâncias artificiais no espaço R^2, desenvolvemos uma ferramenta denominada *Dots* através da qual pode-se especificar uma faixa de valores sobre a qual os pontos serão gerados.

O aplicativo *Dots* constitui-se de uma interface gráfica através da qual, sobre um sistema de eixos cartesianos, pode-se construir um conjunto S de pontos dando a forma que se desejar a cada clusters apenas através do uso do *mouse*. A cada clique do *mouse* sobre uma região do plano formado pelos eixos, as coordenadas deste ponto são armazenadas. Desta forma, podemos construir as instâncias com características variadas de densidade, formato ou presença de *outliers*.

As outras duas bases da literatura são reais. A primeira delas é a *Wisconsin Breast Cancer Database* [77], denotada na tabela 5.1 por CancerData, composta de 699 pontos formados por 9 atributos que caracterizam tumores no tratamento de câncer. O significado de cada atributo pode ser visto consultando a tabela 5.3.

Atributo	**Descrição**
A1	Medida da espessura da massa da célula
A2	Uniformidade do tamanho da célula
A3	Uniformidade da superfície da célula
A4	Adesão marginal da célula
A5	Tamanho da célula epitelial
A6	Núcleos abertos
A7	Cromatina
A8	Núcleos normais
A9	Mitoses

Tabela 5.3: Descrição dos atributos da instância Wisconsin Breast Cancer Database

Dos 699 pontos da instância Wisconsin Breast Cancer, 16 foram excluídos por apresentarem atributos ausentes. Assim, o conjunto de pontos submetido aos algoritmos é composto de 683 elementos. Por se tratar de um conjunto de dados reais

cuja distribuição já é conhecida, sabemos de antemão a classificação dada para cada ponto como sendo de um de dois tipos possíveis, onde 458 referem-se a dados de tumores do tipo benigno e 241 do tipo maligno, sendo que esta informação serviu apenas para medirmos a eficiência dos algoritmos, já que a classificação não pertence ao escopo de abordagens para Problemas de Clusterização.

A outra base de dados reais da literatura é a *Iris Plants Database* [39], denotada na tabela como IrisData, e é composta de 150 pontos no espaço R^4 divididos em três grupos de 50 objetos. Cada grupo refere-se a uma classificação de tipo de íris de planta: íris Setosa, íris Versicolour e íris Virginica. Os atributos dos pontos dizem respeito às medidas de largura e comprimento da pétala e sépala do objeto representado pelo ponto. Os dois últimos grupos não são linearmente independentes.

Antes da avaliação dos resultados, convém ressaltar alguns detalhes quanto à calibragem dos parâmetros utilizados. Neste sentido, o número máximo de gerações do AEC, do AEC-RC e do CLUSTERING foi fixado inicialmente em 100. Entretanto, os testes mostraram que para a maioria das instâncias, tanto o AEC como o AEC-RC não necessitam de um número tão elevado (100 gerações) para encontrar soluções de alta qualidade, já que, em média, a geração na qual a sua melhor solução foi encontrada ficou sempre abaixo de 50. Este fato não se deve a alguma incapacidade do algoritmo de sair de ótimos locais, o que é comprovado pela qualidade das soluções apresentadas principalmente para as instâncias em que se conhece os melhores resultados da literatura, dos quais, podemos adiantar que foram todos atingidos pelos algoritmos propostos.

O tamanho da população foi fixado em 10 indivíduos para o AEC e para o AEC-RC, enquanto a calibragem deste parâmentro para o CLUSTERING seguiu o indicado pelos autores, fixando-se então em 50 indivíduos. Os parâmetros referentes a taxa de crossover e de mutação foram idênticos para os três AG's, onde a taxa de crossover usada foi de 80% e a taxa de mutação foi de 5%. A fixação dos parâmetros u, w_s e w_l utilizados pelo CLUSTERING foi feita conforme recomendam seus respectivos autores.

No caso do SAPCA, o valor da constante APROX foi fixado em 10^6, que é o

mesmo valor de aproximação utilizado nos AG's para indicar a precisão no cálculo da função de aptidão. O valor da constante $R1$ usada na função de decréscimo da temperatura T_1 foi fixado em 8.0 e a constante $R2$ foi a metade deste valor. Portanto, a busca local $Window\_Insert$, no pior caso, quando a função de avaliação da solução inicial retornar o valor 1.0, teremos 104 iterações, já que acrescentamos uma unidade no valor da função de avaliação. Para a busca local Not_Window, como o valor de $R2$ foi fixado em 4.0, no pior caso terá 49 iterações. No entanto, como para cada iteração da busca $Window\_Insert$ há uma chamada para a busca Not_Window, o número de iterações efetivamente executadas desta última é, no pior caso, 5096. Se levarmos em conta que, para cada iteração de ambas as heurísticas de busca, uma única solução é avaliada, o número de chamadas à função de avaliação é bem inferior ao número de soluções avaliadas por qualquer uma das abordagens evolutivas implementadas neste trabalho.

5.1 Resultados Quanto à Qualidade Média das Soluções

Nos resultados apresentados nesta seção, procuramos demonstrar a capacidade de cada algoritmo em encontrar soluções de alta qualidade tendo os parâmetros fixados conforme mencionado no Capítulo 4. Todas as instâncias, com excessão da $2000p11c$ e da $CancerData$ foram testadas executando-se cada algoritmo 100 vezes, Esta exceção se deve ao tempo de processamento do algoritmo CLUSTERING ser muito elevado. Assim, para estas duas instâncias cada algoritmo executou apenas 30 vezes.

Para todas as demais instâncias, após as 100 execuções foi calculada a média dos resultados obtidos pelos algoritmos segundo a função de aptidão dada pela Equação 4.8 bem como foi calculada a média dos tempos de processamento. Uma vez que a função objetivo é de maximização, para cada linha da tabela 5.4, o maior valor indica o desempenho médio do algoritmo mais eficiente. Os valores em negrito indicam a melhor média obtida.

Na tabela 5.4, a coluna **Best** indica o valor da solução obtida na literatura [61, 67]. No caso das instâncias 200Data e Ruspini, a melhor solução obtida pelos quatro algoritmos dentre as 100 execuções é a mesma referenciada na literatura como a melhor.

A instância IrisData, que é composta por três agrupamentos, nenhum dos algoritmos conseguiu separar os três grupos de 50 pontos conforme a natureza dos dados. Entretanto, levando-se em conta que as classes iris Versicolour e iris Virginica não são linearmente independentes, os resultados apresentados pelos algoritmos propostos e também pelo CLUSTERING em termos que qualidade da melhor solução retornada podem ser considerados de boa qualidade, já que este mesmo resultado é citado em outras abordagens sobre o PCA como sendo as melhores soluções obtidas até então [61].

Para a instância CancerData, o AEC, o AEC-RC e o SAPCA encontraram a clusterização ideal, tendo separado os dois clusters de forma que, reparando o conjunto de entrada, podemos perceber que todos os pontos foram agrupados conforme os rótulos indicados no conjunto original. Já o algoritmo CLUSTERING não resolveu o problema de encontrar o número ideal de clusters e falhou na clusterização em todas as execuções, tendo no melhor resultado uma solução com 17 clusters (veja tabela 5.4).

Todos os resultados obtidos obtidos pelos algoritmos AEC, AEC-RC e SAPCA apresentados na tabela 5.4 para as instâncias conhecidas da literatura (200Data, Ruspini, IrisData e CancerData) são referenciados como as melhores soluções encontradas [67, 61], isto demonstra a eficiência das abordagens apresentadas em termos de qualidade da solução.

Podemos ver pela tabela 5.4 que das 17 instâncias, o AEC-RC conseguiu melhorar a solução apresentada pelo AEC em sete, ficando empatado nas demais. Isto nos permite afirmar a eficiência do módulo de Reconexão por Caminhos. Além disso, o algoritmo SAPCA apresenta-se como uma excelente abordagem, tendo em vista que o desempenho médio do mesmo em 16 instâncias esteve entre os três melhores, perdendo apenas na instância $Numbers2$, que apresenta uma distribuição de clusters

Instância	Best	SAPCA	AEC-RC	AEC	CLUSTERING
200Data	**0.823**	**0.823**	**0.823**	**0.823**	0.541
CancerData	**0.596**	**0.596**	**0.596**	0.592	0.374
IrisData	**0.686**	**0.686**	**0.686**	**0.686**	0.601
RuspiniData	**0.737**	**0.737**	**0.737**	**0.737**	0.550
1000p6c	?	**0.735**	0.727	0.708	0.367
157p	?	**0.667**	**0.667**	0.666	0.657
2000p11c	?	**0.658**	0.611	0.602	0.287
2face	?	**0.666**	**0.666**	**0.666**	0.513
350p5c	?	**0.758**	**0.758**	**0.758**	0.568
3dens	?	**0.762**	**0.762**	**0.762**	0.742
97p	?	**0.710**	**0.710**	**0.710**	0.706
Convdensity	?	**0.854**	**0.854**	**0.854**	0.818
Convexo	?	**0.667**	**0.667**	**0.667**	0.618
Face	?	**0.511**	**0.511**	**0.511**	0.402
Moreshapes	?	**0.731**	0.725	0.720	0.436
Numbers	?	**0.546**	0.542	0.540	0.417
Numbers2	?	0.527	**0.565**	0.562	0.513

Tabela 5.4: Desempenho médio: SAPCA, AEC-RC, AEC e CLUSTERING

com formatos bem diversos.

O algoritmo da literatura CLUSTERING, embora seja apresentado pelos autores como tendo resultados satisfatórios, na média das execuções obteve o pior desempenho. Além disso, o tempo de processamento do CLUSTERING comprometeu a apresentação dos resultados para as instâncias $2000p11c$ e $CancerData$ no que se refere ao número reduzido de execuções avaliadas, que foram 30. Entretanto, pelos resultados apresentados pelo algoritmo, mesmo com este número reduzido de testes, podemos comprovar a limitação desta abordagem a medida que cresce o conjunto de dados.

A tabela 5.5 mostra os tempos médios de execução para os quatro algoritmos.

Instância	SAPCA	AEC-RC	AEC	CLUSTERING
200Data	**19,32**	98,15	89,58	508,73
CancerData	5.679,53	3.888,30	**3.648,02**	20.523,29
IrisData	311,31	119,17	**91,17**	583,04
RuspiniData	21,4	20,87	**16.26**	507,52
1000p6c	**537,28**	1.475,71	1.309,89	16.370,43
157p	**23,64**	40,06	35,37	739,01
2000p11c	**4.365,85**	5.829,00	5.166,50	28.294,57
2face	**51.40**	90,03	79,27	3.024,95
350p5c	**69,67**	280,66	247,85	5.319,80
3dens	**10,08**	39,32	34,18	2.069,69
97p	**12,02**	21,12	20.17	299,12
Convdensity	**15,03**	70,08	61,96	1.394,18
Convexo	**49,06**	89,85	79.36	1.043,82
Face	**101,09**	195,41	172,09	6.524,39
Moreshapes	**103,62**	519,04	457,84	11.530,23
Numbers	**212,28**	475,10	416,47	12.166,79
Numbers2	**317,78**	690,01	603,89	26.944,17

Tabela 5.5: Resultado quanto ao tempo médio em segundos: SAPCA, AEC-RC, AEC e CLUSTERING

O algoritmo SAPCA teve o melhor desempenho (menor tempo médio) em 14 das 17 instâncias testadas. No entanto, tendo em vista que o AEC e AEC-RC chegaram à melhor solução nas primeiras gerações na maioria das execuções graças ao uso da heurística de construção, acreditamos que o número de gerações fixado em 100 tenha prejudicado o desempenho dos mesmos na comparação com o desempenho do SAPCA. Na prática, este parâmetro poderia ser reduzido para 50 gerações. Isto certamente faria os tempos de processamento das duas abordagens evolutivas serem reduzidos sensivelmente sem, com isso, prejudicar o desempenho em termos de média das soluções obtidas. Além disso, observa-se uma superioridade de desempenho considerável do AEC e do AEC-RC sobre o SAPCA nas duas instâncias cuja

dimensão é maior que dois.

O algoritmo CLUSTERING teve os tempos altos graças a sua heurística usada para a obtenção do melhor número de clusters, na qual, várias chamadas ao módulo genético são feitas alternando-se o parâmetro w usado no cálculo da função objetivo.

5.2 Resultados por Imagem das Instância no R^2

Várias bases de dados além das descritas na tabela 5.1 no espaço R^2 podem ser construídas pelo gerador de pontos *Dots*. O desempenho dos algoritmos aqui analisados pode ser também avaliado através da análise da imagem plotada com base na clusterização obtida por cada algoritmo.

Na avaliação dos resultados por imagens no R^2, os testes foram feitos de forma que cada algoritmo foi executado 100 vezes e a melhor solução foi tomada para a plotagem. Inicialmente, mostraremos os casos em que as instâncias apresentam clusters de formato esférico com densidades semelhantes, instâncias estas consideradas fáceis. Como o comportamento dos três algoritmos propostos neste trabalho foi o mesmo para todas as instâncias com esta característica, apresentaremos apenas um destes resultados.

A Figura 5.1 mostra a plotagem dos quatro algoritmos para a instância $350p5c$. Como podemos observar, o resultado baseado na saída das três abordagens propostas indicado pela figura (a) apresenta os 5 clusters bem definidos, enquanto a clusterização resultante do algoritmo CLUSTERING na figura (b) mostra pequenos agrupamentos que este algoritmo não conseguiu reunir em um só cluster, o que representa uma dificuldade do algoritmo em sair de ótimos locais.

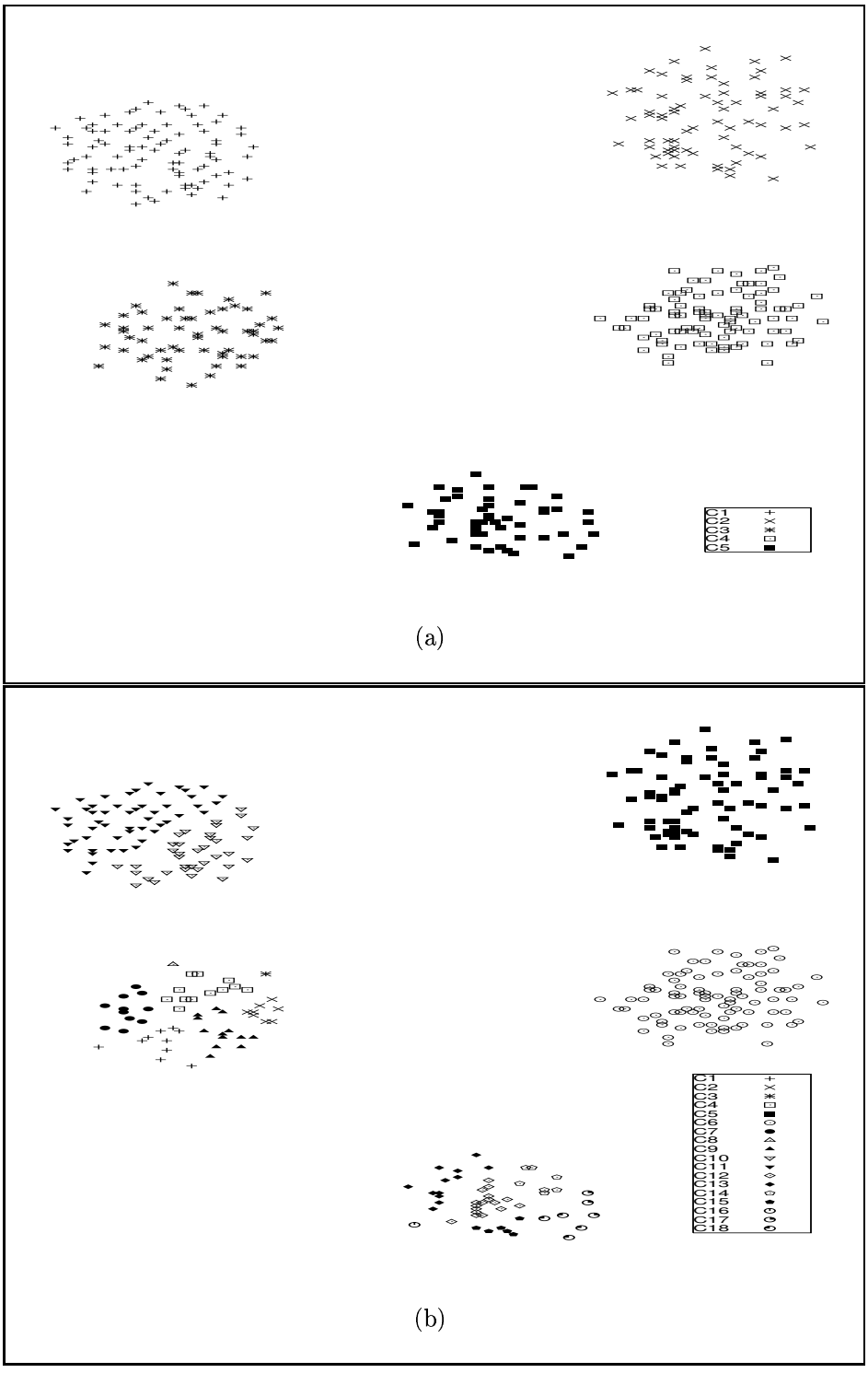

Figura 5.1: Resultado para a instância 350p5c: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

Apenas para ilustrar que o AEC, o AEC-RC e o SAPCA não perdem em eficiência à medida que a base de dados cresce, na Figura 5.2 mostramos o resultado da instância $2000p11c$, na qual percebe-se que enquanto os três algoritmos citados conseguem convergir para o melhor agrupamento contendo 11 clusters (número ideal para esta instância), o CLUSTERING mostra 49 clusters formados, resultado pior do que o apresentado pelo mesmo para a instância $350p5c$.

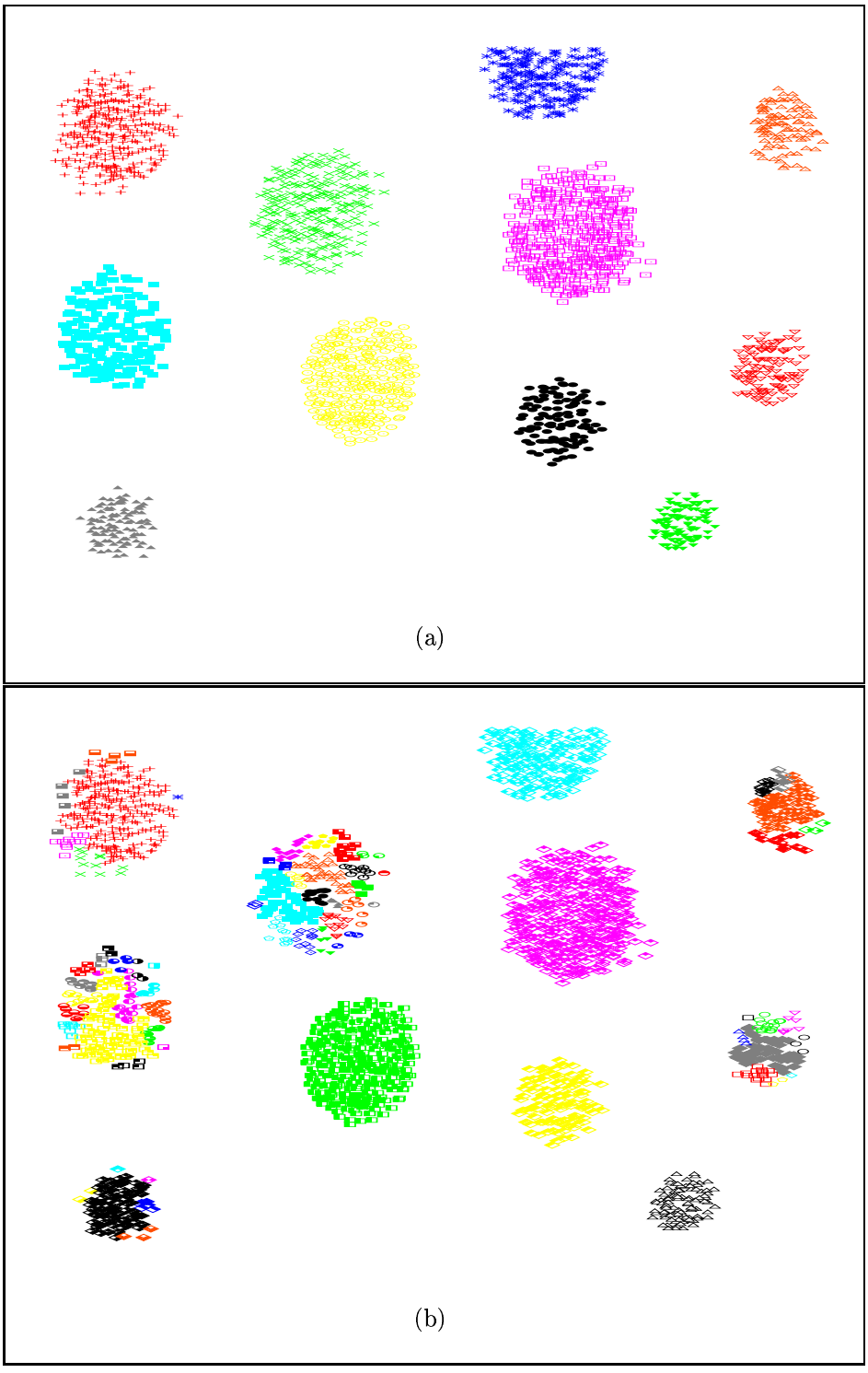

Figura 5.2: Resultado para a instância: 2000p11c (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

Os resultados apresentados pelos algoritmos para a instância $1000p6c$ foram semelhantes aos indicados pelas Figuras 5.1 e 5.2. Portanto, não mostramos a clusterização desta instância. Mostraremos agora os resultados para aquelas instâncias que apresentam clusters de diferentes formas e densidades.

Para a instância $157p$, os resultados mostrados na Figura 5.3 (a) representam a saída das três abordagens propostas. Já a Figura 5.3 (b) é a clusterização obtida pelo CLUSTERING. Podemos ver que todos os algoritmos apresentaram resultados praticamente idênticos, tendo o CLUSTERING conseguido neste caso melhorar sua qualidade ao associar corretamente todos os pontos centrais da figura.

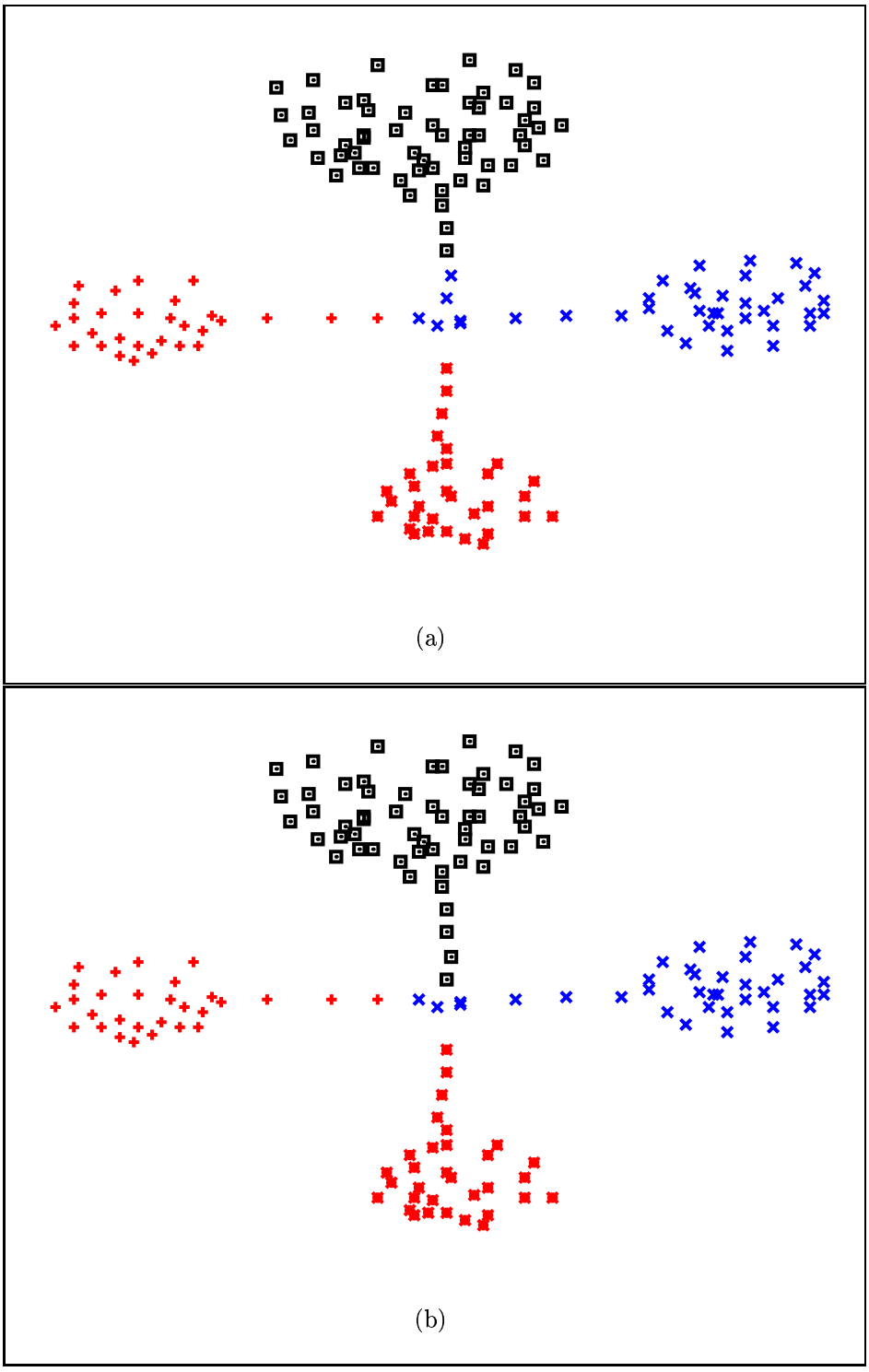

Figura 5.3: Resultado para a instância 157p: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

Duas instâncias foram desenhadas através do uso da ferramenta *Dots* com o intuito de verificar a mudança de comportamento dos algoritmos quando inserimos um novo cluster numa instância já existente. Assim, a instância $face$ (Figura 5.4) tem os pontos dispostos de tal forma que esboçam um desenho de uma face e nesta mesma massa de dados inserimos pontos que graficamente esboçam o algarismo **2**, fazendo surgir a instância $2face$ (Figura 5.5).

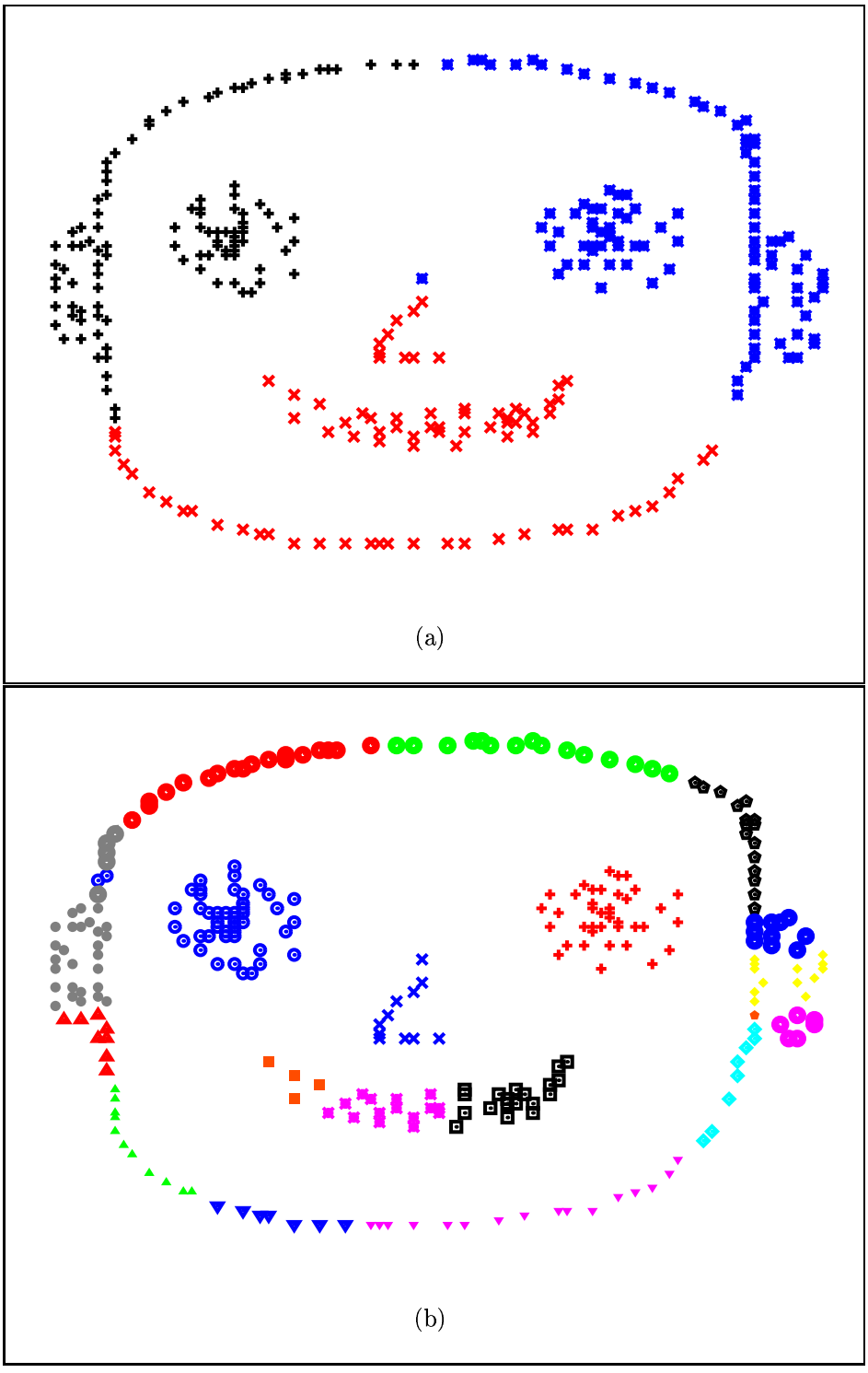

Figura 5.4: Resultado para a instância face: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

No caso da instância *face*, a Figura 5.4 nos mostra três clusters encontrados pelos três algoritmos propostos. Pode-se perceber a influência do uso da distância euclidiana como medida de dissimiladade ao vermos que os clusters são formados de maneira que a linha que forma o contorno da face juntamente com o que seriam as orelhas foi dividida exatamente nas regiões onde outros grupos de pontos estão dispostos. Já no caso do algoritmo CLUSTERING (Figura 5.4 (b)), que apresentou 20 clusters para esta instância, os agrupamentos formados mais uma vez indicam sua fragilidade ao lidar com instâncias com clusters de diferentes densidades. Note, no entanto, que os pontos referentes aos dois círculos do que seriam os olhos, bem como a região do nariz foram bem separados pelo algoritmo. Contudo, o esboço da cabeça e o formato da boca levaram o algoritmo a falhar no resultado de forma a apresentar como solução agrupamentos ininteligíveis.

Uma solução possívelmente mais aceitável como uma boa clusterização para a instância *face* seria a formação de um clusters para cada olho, um cluster para cada orelha, um outro para o nariz, outro para a boca e um último para o contorno da cabeça. Entretanto, não se tem garantia de que, segundo as funções de aptidão usadas pelos algoritmos, esta solução teria um valor superior ao que foi obtido.

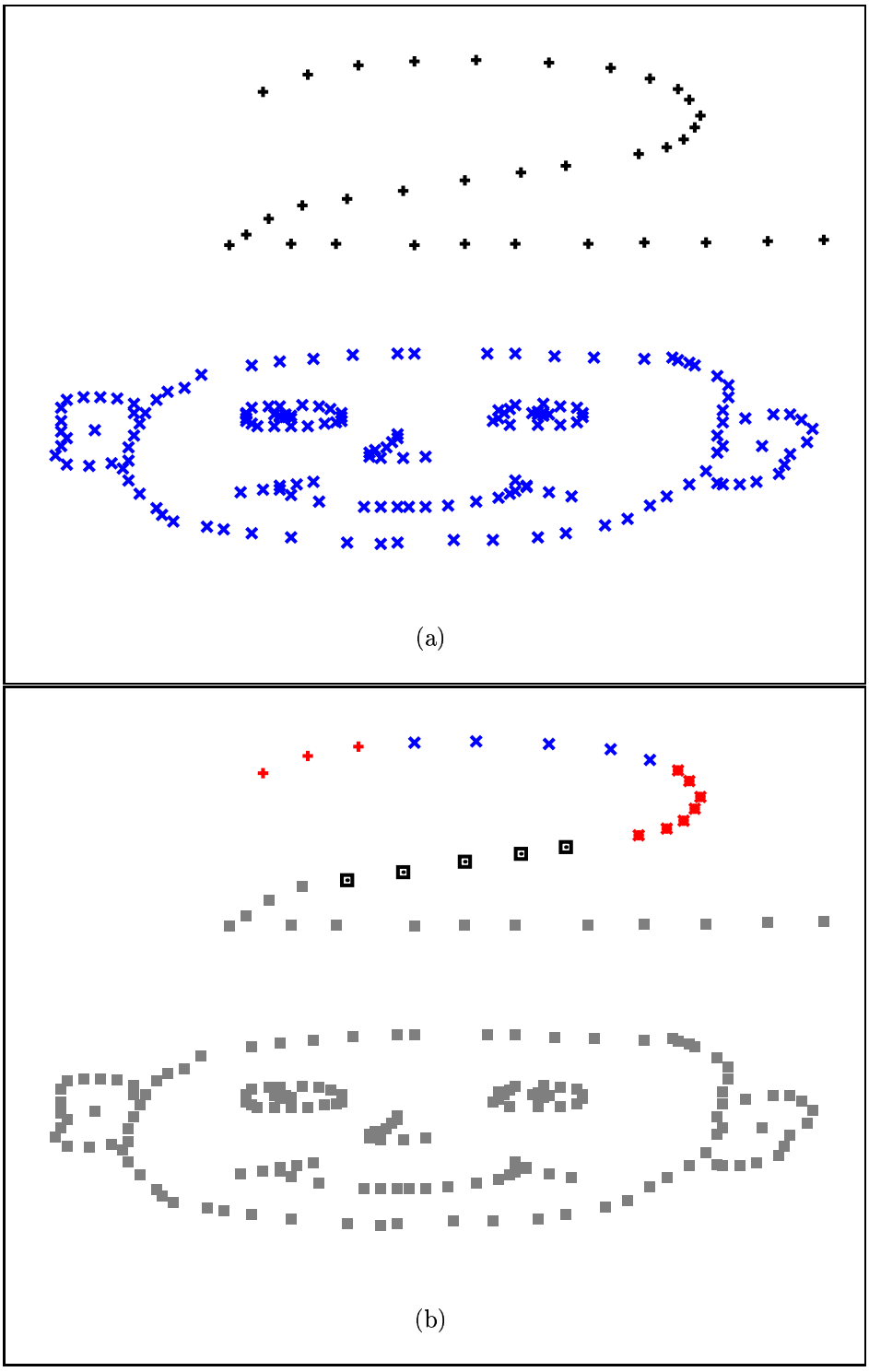

Figura 5.5: Resultado para a instância 2face: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

Ao inserirmos novos pontos no conjunto de dados da instância *face*, de forma a criar a nova instância *2face* (Figura 5.5), onde todos os pontos do conjunto anterior devem ser agrupados em um único cluster, os algoritmos mudam de comportamento. Neste caso, novamente tivemos os resultados do AEC, AEC-RC e SAPCA semelhantes, como mostrado na Figura 5.5 (a). A Figura 5.5 (b) mostra claramente que o CLUSTERING sofreu influência da diferença de densidade dos dois agrupamentos, pois o cluster que mostra o esboço da face é muito mais denso que o cluster que representa o algarismo 2. Este fato acabou levando algoritmo da literatura a fazer com que o cluster mais denso anexe para si pontos que na verdade pertencem ao cluster menos denso.

A instância *3dens* apresenta uma distribuição caracterizada por três densidades diferentes. Os resultados apresentados pelos quatro algoritmos foram equivalentes e estão apresentados numa única plotagem na figura Figura 5.6. Podemos ver que nenhum dos algoritmos foi capaz de separar os três pontos que se encontram na parte inferior da figura dos pontos que se encontram na parte central.

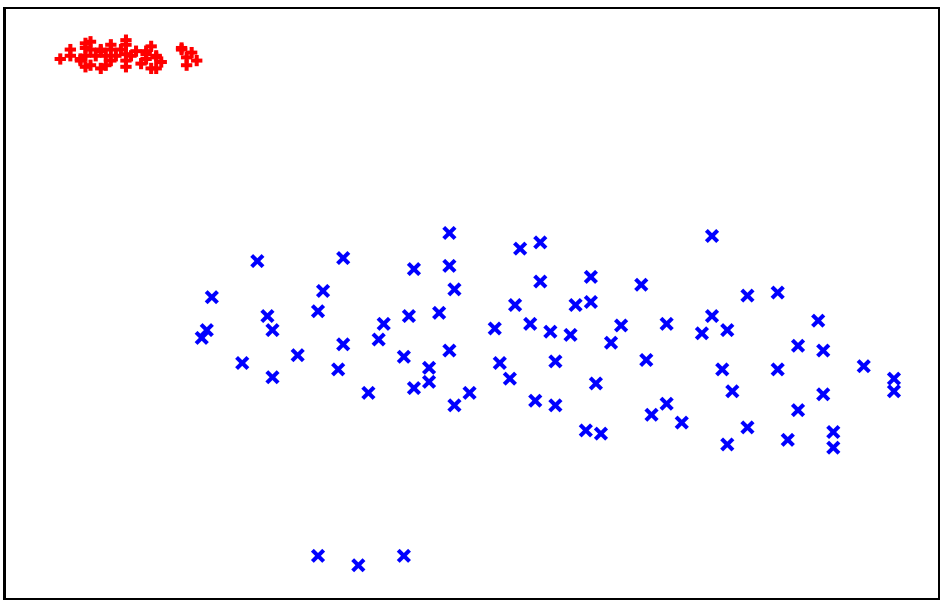

Figura 5.6: Resultado para a instância *3dens*: AEC, AEC-RC, SAPCA e CLUSTERING

Na Figura 5.7 podemos ver o melhor desempenho dos algoritmos propostos em relação ao da literatura quando submetemos os mesmos à instâncias contendo clusters de tamanho diferentes e alguns clusters não convexos. Uma vez que os clusters na figura apresentam densidades variadas, o uso do parâmetro que guia o CLUSTERING na construção dos clusters na fase de construção acaba gerando soluções iniciais de baixa qualidade com clusters muito dispersos, enquanto a instância compõe-se de clusters mais coesos. No entanto, nenhuma das abordagens propostas conseguiu separar o cluster central, que apresenta apenas três pontos.

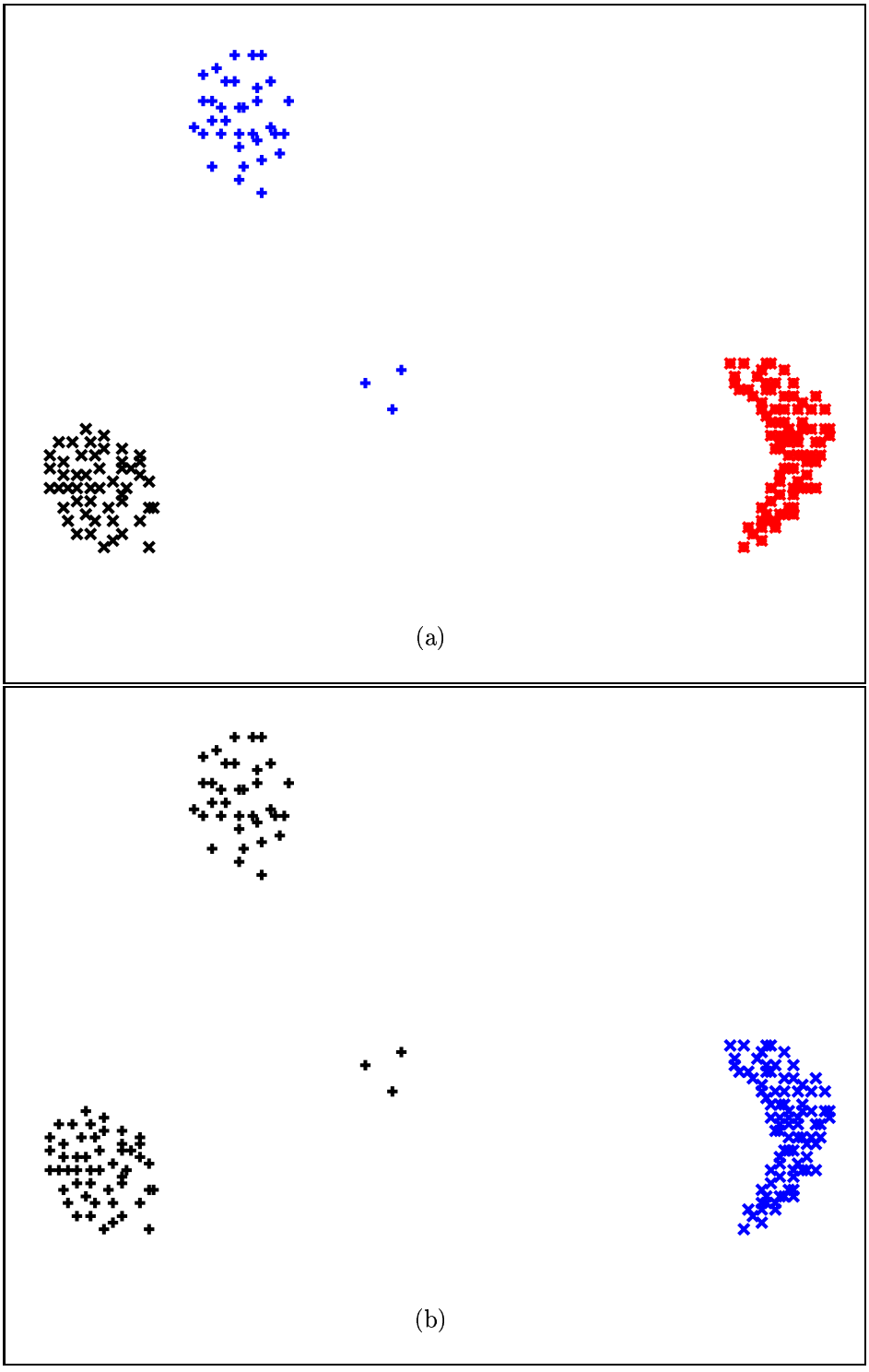

Figura 5.7: Resultado para a instância Convdensity: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

Como podemos observar nas figuras 5.8 e 5.9, que mostram a plotagem da instância *Numbers*, o resultado apresentado pelo AEC (Figura 5.8(a))e pelo AEC-RC (Figura 5.8(b)) apresenta uma significativa superioridade em relação ao SAPCA (Figura 5.9(a))e ao CLUSTERING (Figura 5.9(b)), que apresentou a pior solução. Os pontos que fazem o contorno do algarismo **5** na Figura 5.9 (a) apresentam dois clusters e na Figura 5.9 (b) apresentam três. É fácil verificar em ambas as figuras que a proximidade dos pontos referentes ao algarismo **9** em relação aos pontos dos algarismos **1** e **4** fez com que nenhum dos algoritmos conseguisse separá-los perfeitamente.

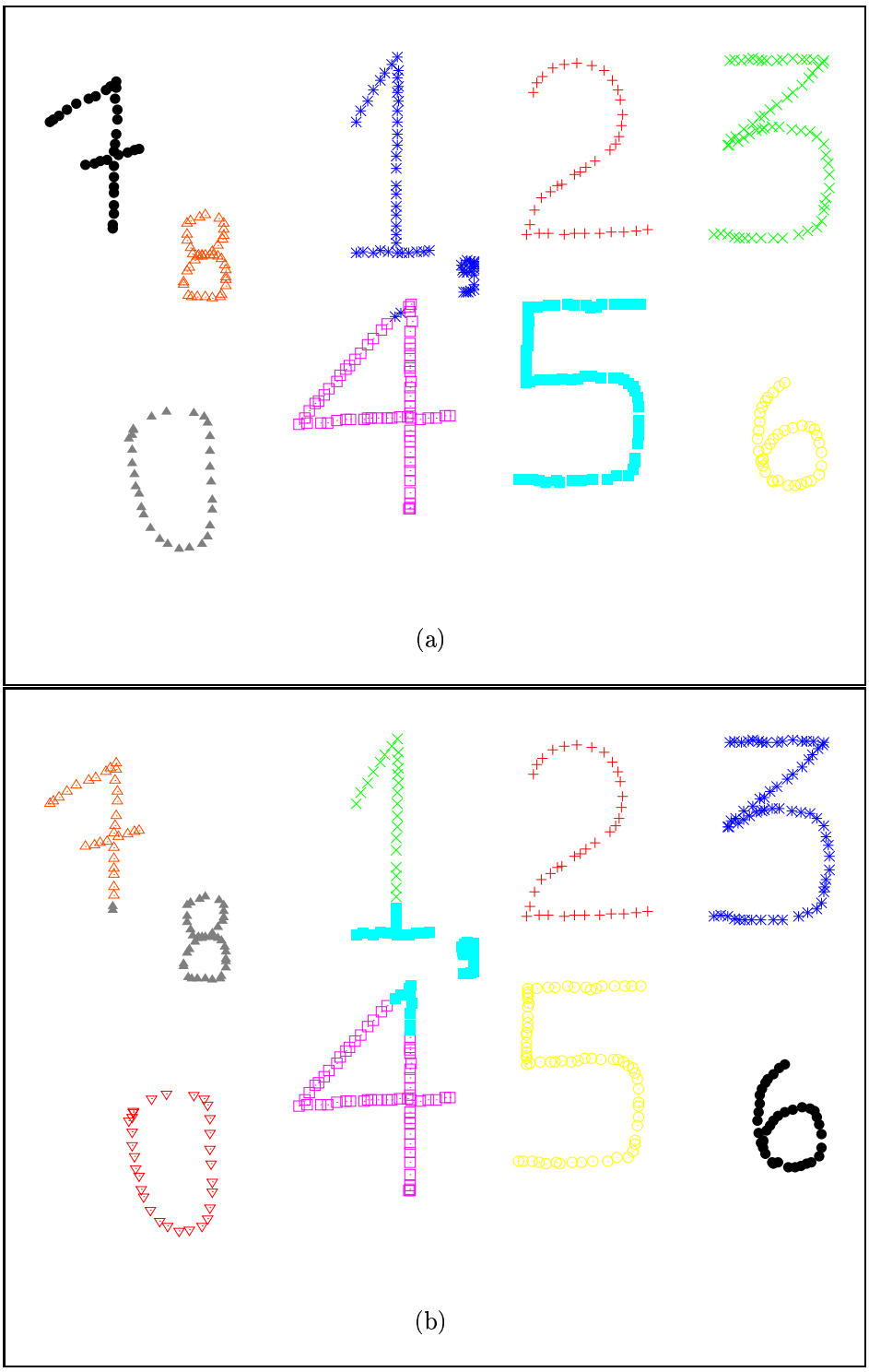

Figura 5.8: Resultado para a instância Numbers: (a) AEC (b) AEC-RC

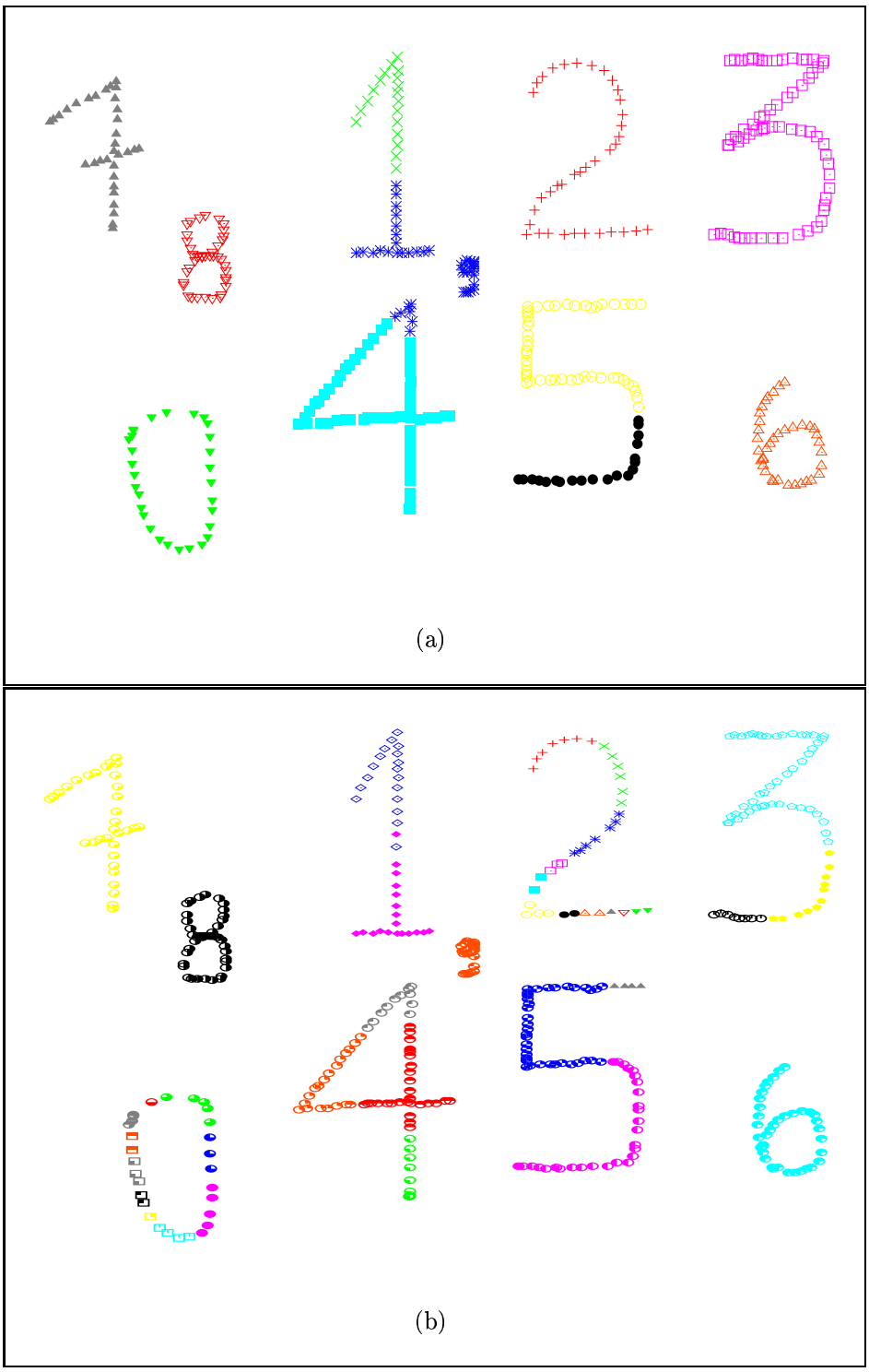

Figura 5.9: Resultado para a instância Numbers: (a) SAPCA (b) CLUSTERING

Ao retirarmos os pontos referentes ao número **9** na Figura 5.8 e redesenharmos este algarismo noutra região, criamos a instância $Numbers2$ (Figura 5.10). Como podemos observar, o AEC-RC conseguiu melhorar a solução apresentada pelo AEC, que, mesmo obtendo o número ideal de clusters (dez), agrupou em um só cluster alguns pontos pertencentes aos agrupamentos referentes ao algarismo **2** no cluster formado pelos pontos do algarismo **5**.

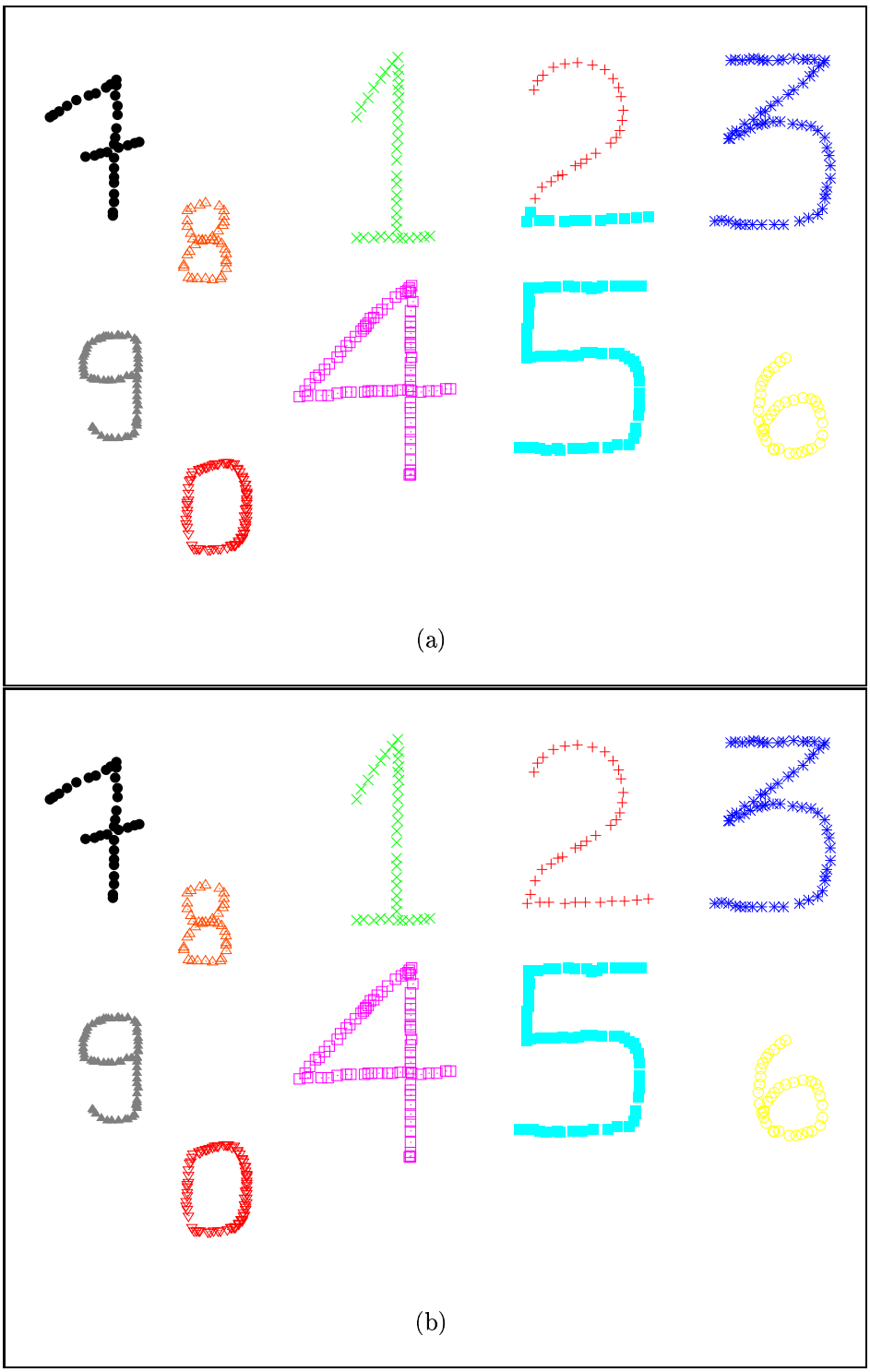

Figura 5.10: Resultado para a instância Numbers2: (a) AEC (b) AEC-RC

Pode-se perceber ainda que o melhor desempenho novamente foi do AEC-RC, pois o SAPCA (figura 5.9(a)), que noutras instâncias apresentava boas soluções, não separou corretamente os pontos referentes aos algarismo **7, 8, 1** e **4**, embora tenha identificado o número ideal de clusters, a exemplo do AEC. O desempenho do CLUSTERING (figura 5.9(b)) foi demasiadamente ruim, tendo o mesmo apresentado um total de 49 clusters, quando o número ideal é 10 clusters, confirmando sua ineficiência nos casos em que o formato dos clusters não é esférico.

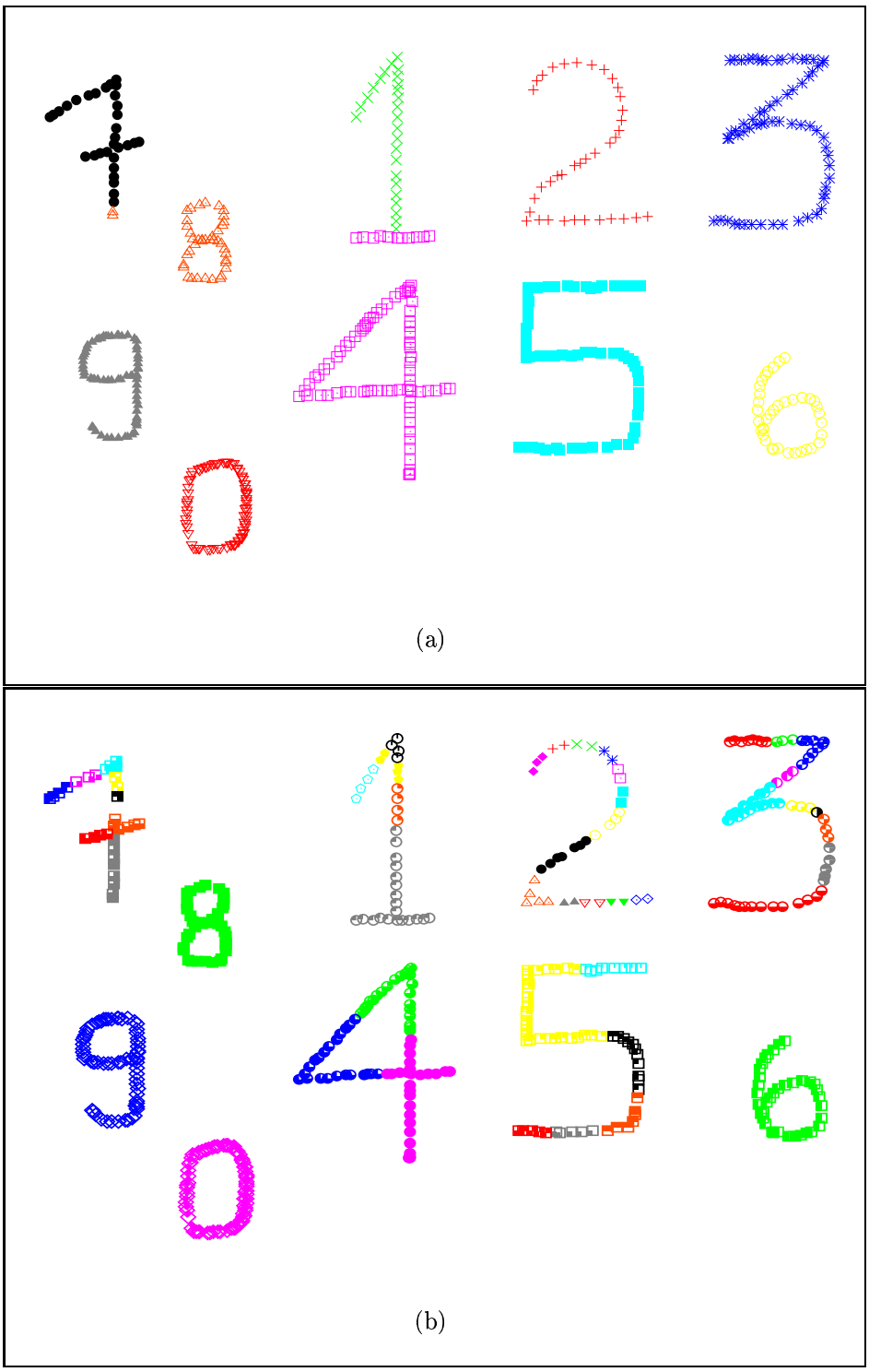

Figura 5.11: Resultado para a instância Numbers2: (a) SAPCA (b) CLUSTERING

Na instância *Moreshapes* (Figura 5.12) nós procuramos desenhar um conjunto de clusters com formatos bem variados e inserimos um *outlier* para verificarmos como os algoritmos se comportariam quando submetidos a uma instância com estas características. A Figura 5.12 (a) mostra o resultado obtido pelos algoritmos propostos. Podemos notar que, embora os algoritmos não tenham identificado a presença de um ponto isolado, a presença do mesmo não prejudicou a clusterização para os demais pontos. Enquanto o CLUSTERING (Figura 5.12 (b)) apresentou 19 clusters, os algoritmos propostos apresentaram seis agrupamentos onde o único aspecto negativo foi o fato dos mesmos não terem identificado o ponto inserido como *outlier*.

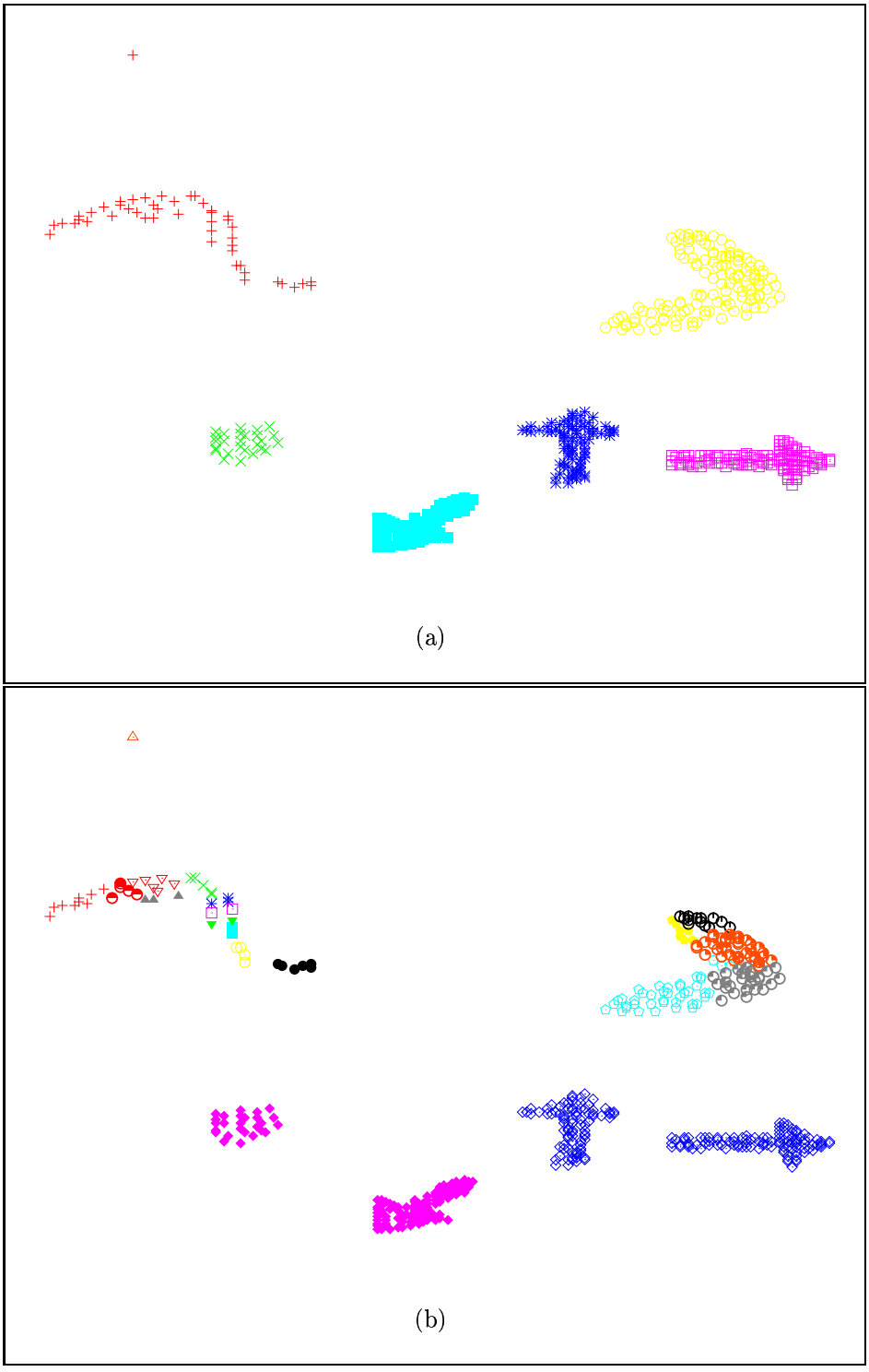

Figura 5.12: Resultado para a instância Moreshapes: (a) AEC, AEC-RC e SAPCA (b) CLUSTERING

Pelos resultados observados nesta seção, podemos perceber uma nítida superioridade dos três algoritmos aqui propostos em relação ao da literatura. Um aspecto importante é o fato dos algoritmos propostos apresentarem desempenho praticamente idênticos nestes testes, mostrando que o fator dominante são os módulos aqui propostos, como a fase construtiva, a busca local e a reconexão por caminhos (RC), que possibilitou ao AEC-RC alguns resultados de melhor qualidade que aqueles apresentados pelo AEC e SAPCA.

Apresentamos a seguir uma forma de avaliar metaheurísticas a partir de análises probabilísticas que procuram oferecer subsídios quanto a sua convergência para valores ótimos.

5.3 Avaliação Probabilística dos Resultados

Uma outra abordagem para análise de desempenho de metaheurísticas foi proposta em [4]. Esta abordagem consiste em tomar os tempos de processamento que cada algoritmo requer para atingir um valor alvo na função de aptidão. Foram estabecidos neste trabalho três alvos para cada instância utilizada para este modelo de avaliação: um alvo considerado fácil, dado pela média das soluções do pior algoritmo; um alvo considerado médio, dado pela média dos resultados de todos os algoritmos; e, finalmente, um alvo considerado difícil, dado pela média dos resultados do melhor algoritmo. Desta forma, no instante em que cada algoritmo encontra uma solução maior ou igual ao alvo fixado, o tempo é registrado e uma nova execução começa. Em outras palavras, um novo critério de parada é inserido para cada algoritmo, que consiste na obtenção de uma solução cujo valor atinja ou ultrapasse o alvo.

Cada algoritmo foi executado 100 vezes para cada instância usando os mesmos parâmetros. Os tempos foram tomados e dispostos em ordem crescente numa lista L. A cada tempo de CPU t obtido, foi associada a probabilidade $p_i = (i - \frac{1}{2})/100$, onde i é a ordem que t aparece na lista ordenada L. A plotagem foi feita então tomando cada ponto (t_i, p_i). Caso o algoritmo não tenha sucesso em encontrar o alvo por 100 execuções consecutivas, admitimos que o mesmo é incapaz de atingí-lo,

o que o faz ficar fora da análise para aquele alvo.

Uma vez que este tipo de avaliação requer o triplo de execuções em relação à análise apenas através de imagens, por questão de praticidade, nos restringiremos às instâncias da literatura. A análise do desempenho dos algoritmos para as outras instâncias não fica prejudicada, uma vez que as distorções não são tão consideráveis.

Para a instância *RuspiniData*, o alvo fácil foi fixado em 0.550; o alvo médio em 0.680; e o alvo difícil em 0.737. Os tempos de execução dos quatro algoritmos e a probabilidade de cada um atingir o alvo são apresentados na Figura 5.13. O gráfico da figura para o alvo 0.55 nos mostra a eficiência do AEC-RC e do AEC em relação ao SAPCA e ao CLUSTERING. Ambos apresentam 80% de chance de atingir o alvo em apenas um centésimo de segundo, enquanto neste tempo o SAPCA alcança cerca de 45%.

O comportamento dos algoritmos sofre uma pequena alteração nesta instância quando elevamos o alvo para 0.68, como pode ser visto no gráfico central da Figura 5.13. Para este alvo, o algoritmo SAPCA comportou-se muito bem, com 83% de chance de encontrar o alvo em um tempo inferior a um segundo. No entanto, podemos observar que o AEC supera SAPCA a partir da probabilidade 0.83. É de se destacar que até o AEC-RC, que sabemos consumir tempo de processamento extra associado à sua busca local, supera SAPCA quando analisamos a convergência entre 80% e 100%. A exemplo do desempenho apresentado para o alvo fácil, o CLUSTERING teve o pior desempenho no alvo médio.

Para o alvo 0.73 pode-se observar na Figura 5.13 que tanto o AEC como AEC-RC perdem para o SAPCA se tomarmos como probabilidade de referência $p_i < 0.9$. Após este valor, no entanto, o AEC-RC sem religamento de caminhos supera o SAPCA, que passa a comportar-se de forma similar ao AEC-RC.

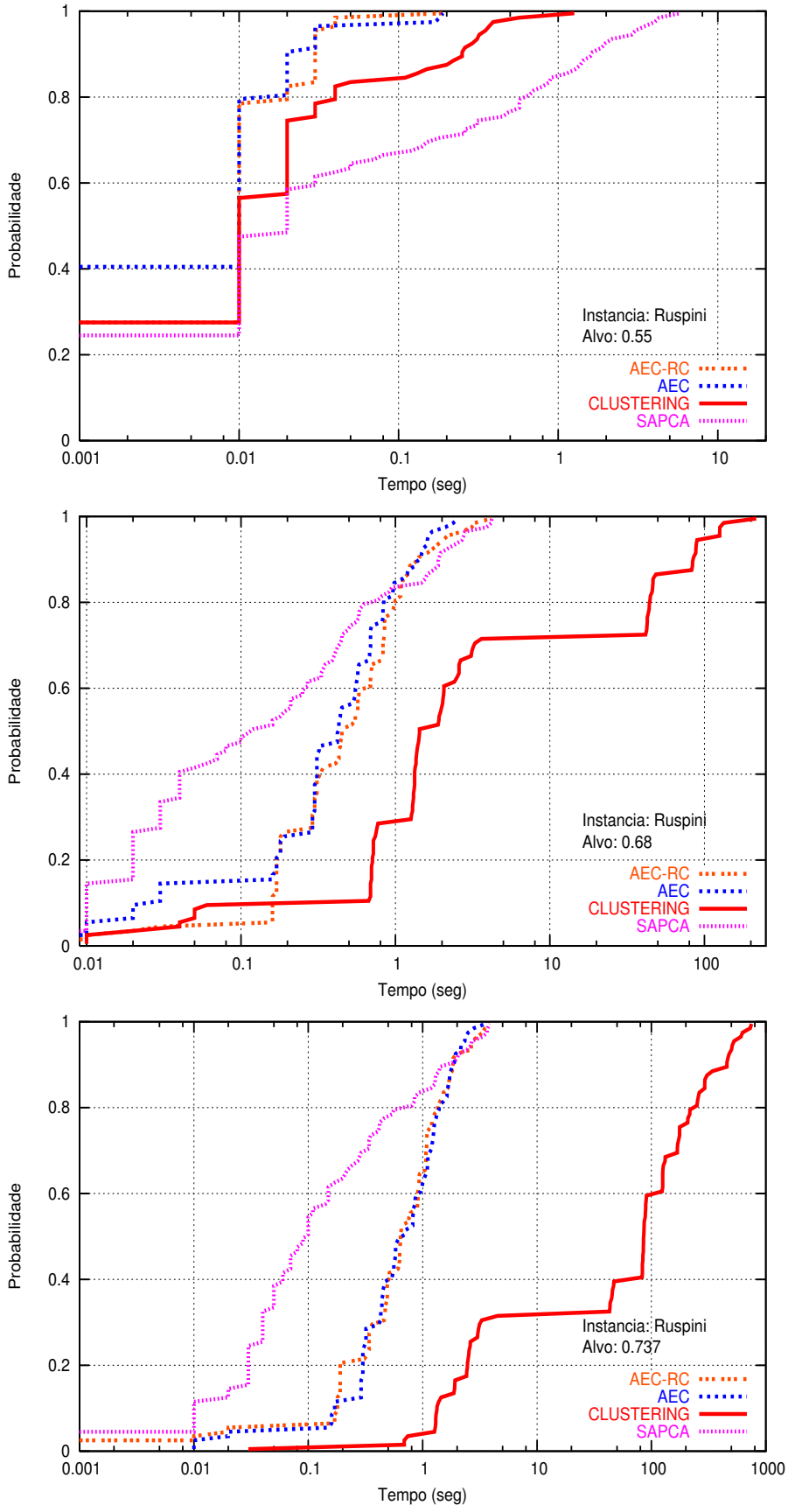

Figura 5.13: Análise Probabilística para a instância RuspiniData

No caso da instância 200*Data* (Figura 5.14), o alvo fácil foi fixado em 0.540; o médio em 0.710; e o alvo difícil em 0.823. A Figura 5.14 apresenta a plotagem dos resultados para estes três alvos. Pode-se ver que, para o alvo fácil, os três algoritmos propostos apresentam-se com probabilidades bem próximas de atingir o alvo num tempo inferior a um décimo de segundo. No entanto, percebe-se um desempenho superior do AEC e do AEC-RC em relação ao SAPCA e ao CLUSTERING antes do primeiro décimo de segundo de processamento, sendo que o AEC-RC, mesmo com o tempo de busca local associado ao religamento, sempre supera o SAPCA para uma probabilidade entre 50% e 95%.

Para o alvo médio, o comportamento dos quatro algoritmos não sofreu alterações significativas. Pode-se ver, entretanto, que o CLUSTERING ultrapassa a barreira de 1000 segundos para que se assegure os 100% de chance do mesmo atingir o alvo. As inclinações das curvas de cada algoritmo para os alvos fácil e médio indicam que o incremento feito no alvo não repercutiu tanto no comportamento das abordagens comparando-as com o que apresentaram para o alvo difícil.

O desempenho do AEC e do AEC-RC é superado pelo SAPCA para o alvo difícil. Podemos ver na Figura 5.14 que o SAPCA atinge o alvo com 100% de chance antes de 10 segundos, ao passo que o AEC e o AEC-RC em 10 segundos apresentam cerca de 90% de chance. O CLUSTERING ultrapassa os 10.000 segundos, sendo que no melhor tempo deste algoritmo, as três abordagens propostas já teriam terminado de executar no tempo mais tardio. Além disso, nota-se pela inclinação da curva referente ao algoritmo da literatura que o mesmo foi muito mais sensível à elevação do alvo de médio para difícil.

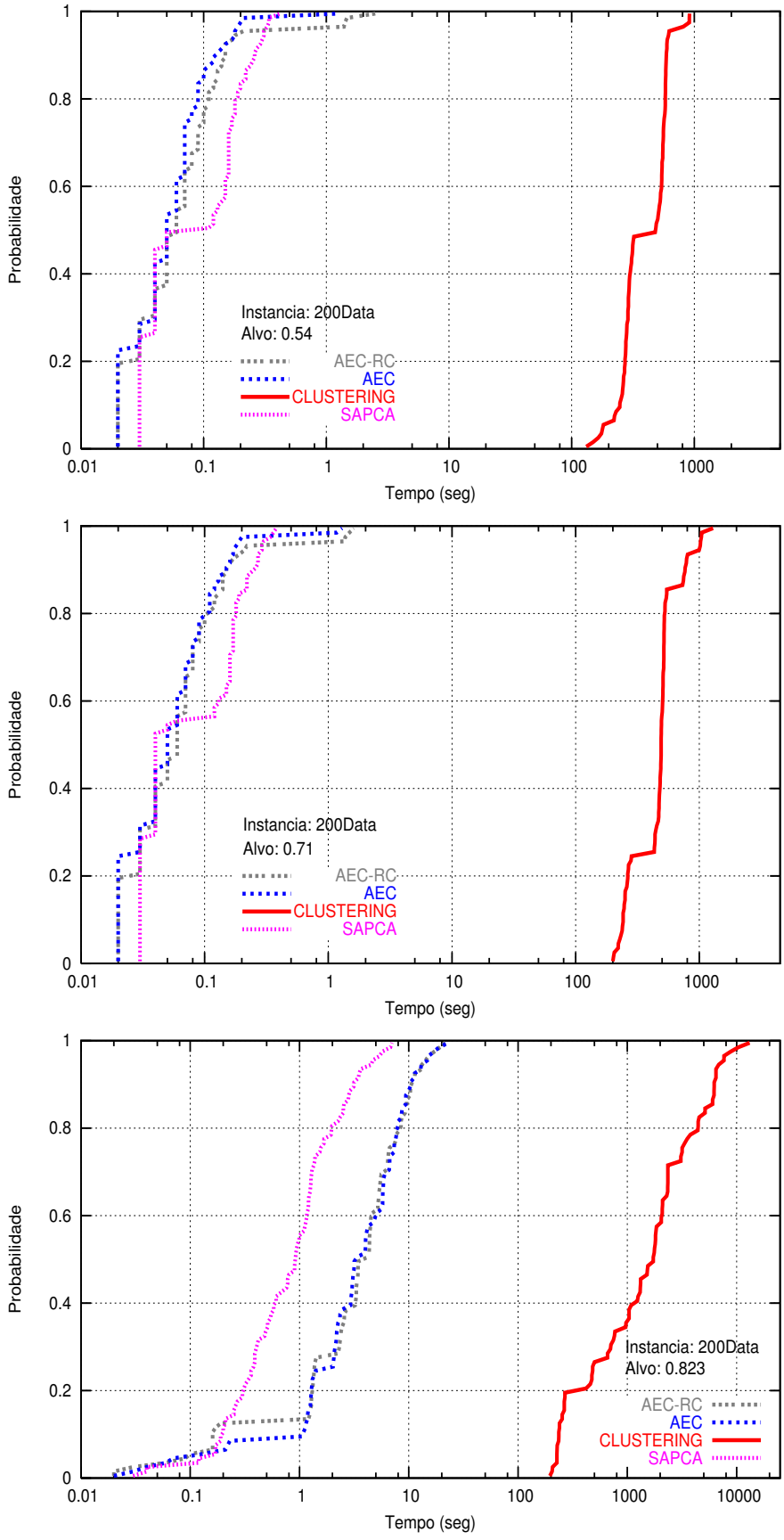

Figura 5.14: Análise Probabilística para a instância 200Data

Apresentaremos agora os resultados para a instância IrisData. A Figura 5.15 mostra os gráficos para os três alvos fixados. No gráfico superior (alvo fácil), pode-se perceber que para taxas de convergência entre 50% e 90% os dois AG's superam o SAPCA, sendo que a superioridade do AEC é estendida até quase 100%.

Ainda na Figura 5.15, o gráfico ao centro nos mostra que as abordagens evolutivas conseguiram superar o SAPCA já a partir da taxa de convergência de 25%, mantendo esta superioridade até os 80%. Entre 80% e 90%, percebe-se uma superioridade do AG com Reconexão de Caminho. No caso do gráfico inferior, em que mostramos o desempenho para o alvo difícil, novamente podemos ver que a superioridade dos AG's ocorre na região de probabilidades acima de 0.5. Entretanto, o AEC consegue atingir o alvo com 100% de chance antes de qualquer outro algoritmo.

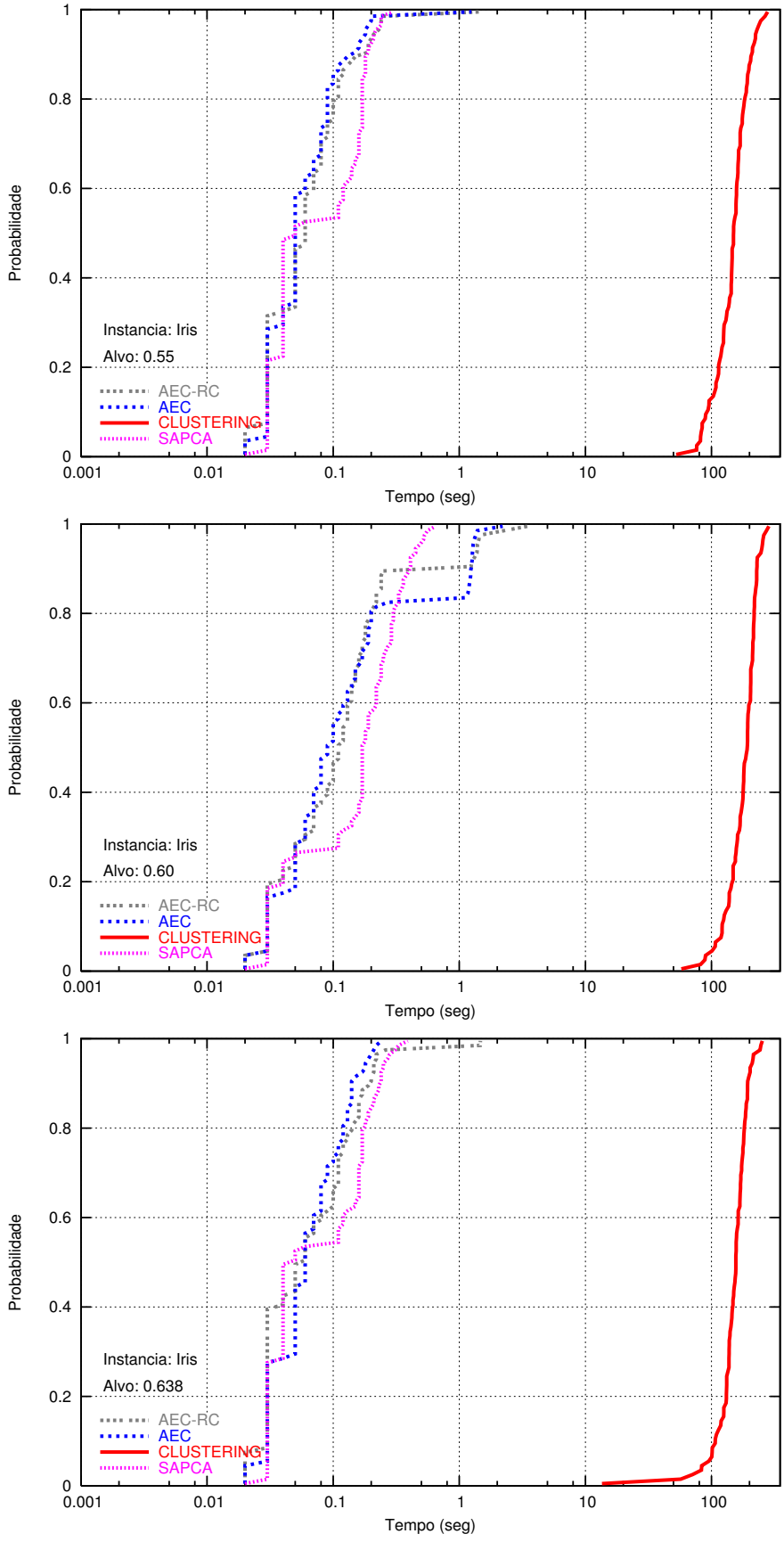

Figura 5.15: Análise Probabilística para a instância IrisData

Os resultados apresentados na figura 5.16 são referentes à instância Wisconsin Breast Cancer Data Set. O algoritmo CLUSTERING não conseguiu atigir os alvos médio e difícil. Para o alvo fácil, o SAPCA em nenhuma das execuções ultrapassou a barreira de 1 segundo, sendo assim, considerado o mais robusto dos quatro algoritmos. No entanto, esta tendência não se verifica para todo o tempo de processamento referente aos alvos médio e difícil.

O comportamento do CLUSTERING nos permite afirmar que esta abordagem não apresenta-se como uma boa estratégia. Pode-se ver que o tempo que o CLUSTERING leva na sua execução mais rápida é suficiente para afirmarmos que qualquer uma das três abordagens propostas já tenha obtido a solução qualquer que seja o alvo. Além disso, é de se notar a queda considerável de desempenho do CLUSTERING em relação aos resultados apresentados nas instâncias no espaço R^2. Esta tendência se confirma com os resultados para a instância CancerData, que é composta por pontos no espaço R^9, onde o CLUSTERING não foi capaz de encontrar os alvos médio e difícil.

Para os alvos médio e difícil, destacamos a importância do uso de Reconexão por Caminhos. Podemos notar pelo gráfico referente ao alvo difícil que mesmo com a Reconexão de Caminhos consumindo mais tempo, o AEC-RC a partir de 60% de taxa de convergência, teve um desempenho praticamente idêntico ao AEC, que não gasta tempo excedente em busca local. Este fato foi o que motivou o uso de Reconexão por Caminhos, pois nos testes em que o parâmetro referente ao número de geração era o critério de parada, em algumas execuções o AEC não chegava na melhor solução citada na literatura.

Nos testes efetuados até o momento, observamos que um fator que influencia o desempenho dos dois algoritmos evolutivos aqui propostos é o algoritmo de construção da população inicial. Para permitir uma verificação da importância do método de construção da população inicial utilizado, apresentamos agora os gráficos de desempenho do AEC, do AEC-RC e do CLUSTERING comparando-os com o AEC sem o procedimento de construção aqui proposto, denotado AEC-SC e, da mesma forma, com o AEC-RC sem o método de construção, denotado AEC-RCSC.

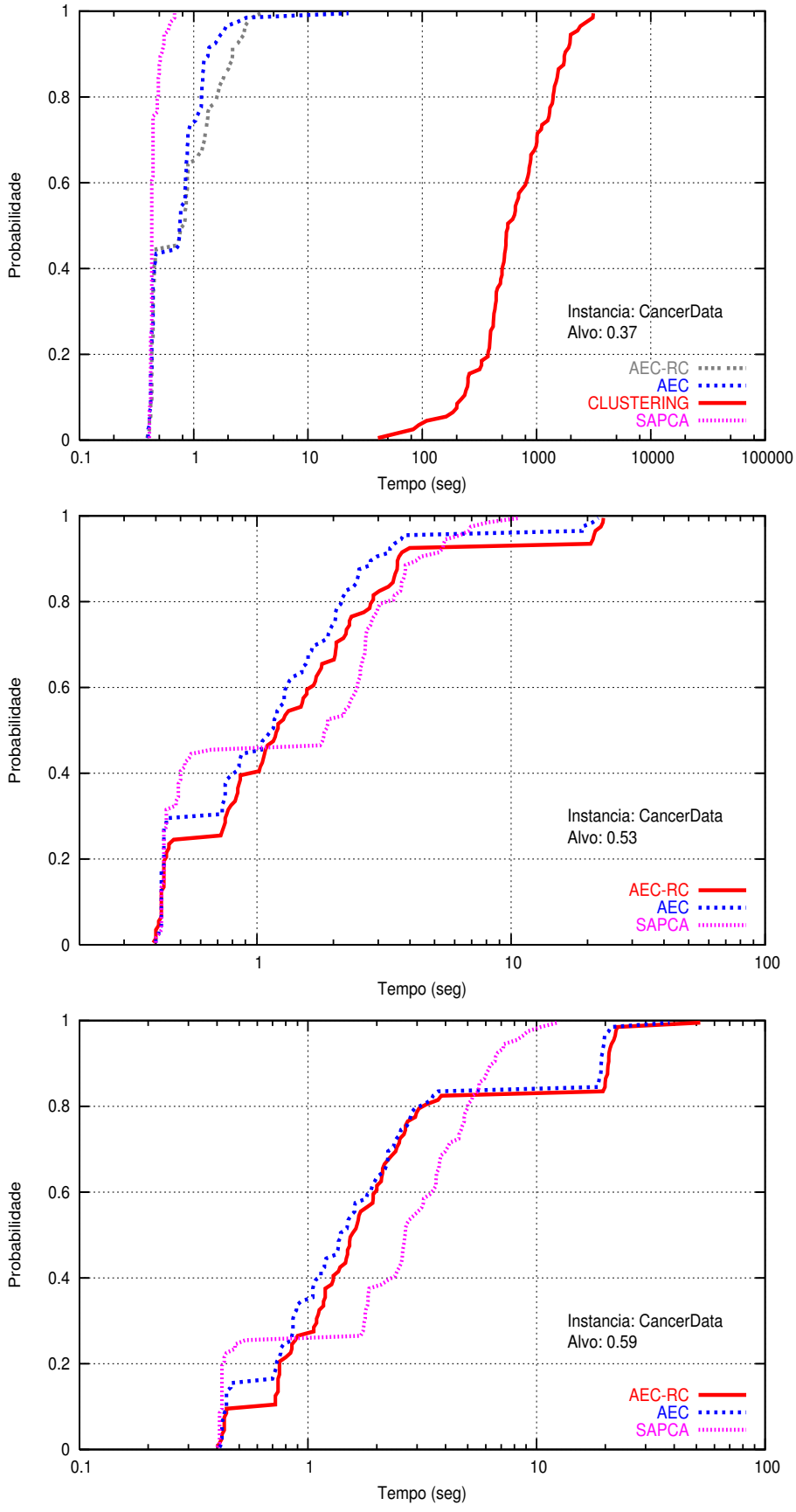

Figura 5.16: Análise Probabilística para a instância CancerData

Inicialmente, na Figura 5.17 mostramos o desempenho das cinco abordagens para a instância *RuspiniData*. Para o alvo fácil, obtivemos o melhor desempenho praticamente empatado pelos algoritmos AEC e AEC-RC, seguidos do CLUSTERING e, por último, os dois algoritmos sem o procedimento de construção. Note que o menor tempo requerido pelos algoritmos sem método de construção é praticamente o mesmo tempo que os algoritmos com construção utilizam pra as execuções mais tardias. Por outro lado, a análise do gráfico de desempenho para o alvo médio permite-nos observar uma melhora considerável no desempenho dos algoritmos sem procedimento de construção, que para probabilidades acima de 0.7 superaram o algoritmo da literatura.

Para o alvo difícil, o CLUSTERING tem uma queda de desempenho considerável em relação aos algoritmos sem construção. Acreditamos que o uso do operador de mutação através da sobreposição de janelas aqui proposto é o fator que contribuiu para este desempenho, pois os operadores de crossover e de mutação tradicionais que o CLUSTERING utiliza são os mesmos adotados pelos algoritmos evolutivos propostos. Isto permite que o AEC-SC e o AEC-RCSC atinjam o alvo difícil até pouco mais de 10 segundos com 100% de chance, enquanto o CLUSTERING requer, para esta probabilidade, praticamente 1000 segundos. Mesmo assim, mais uma vez os dois melhores algoritmos foram o AEC e AEC-RC, que fazem uso da heurística de construção.

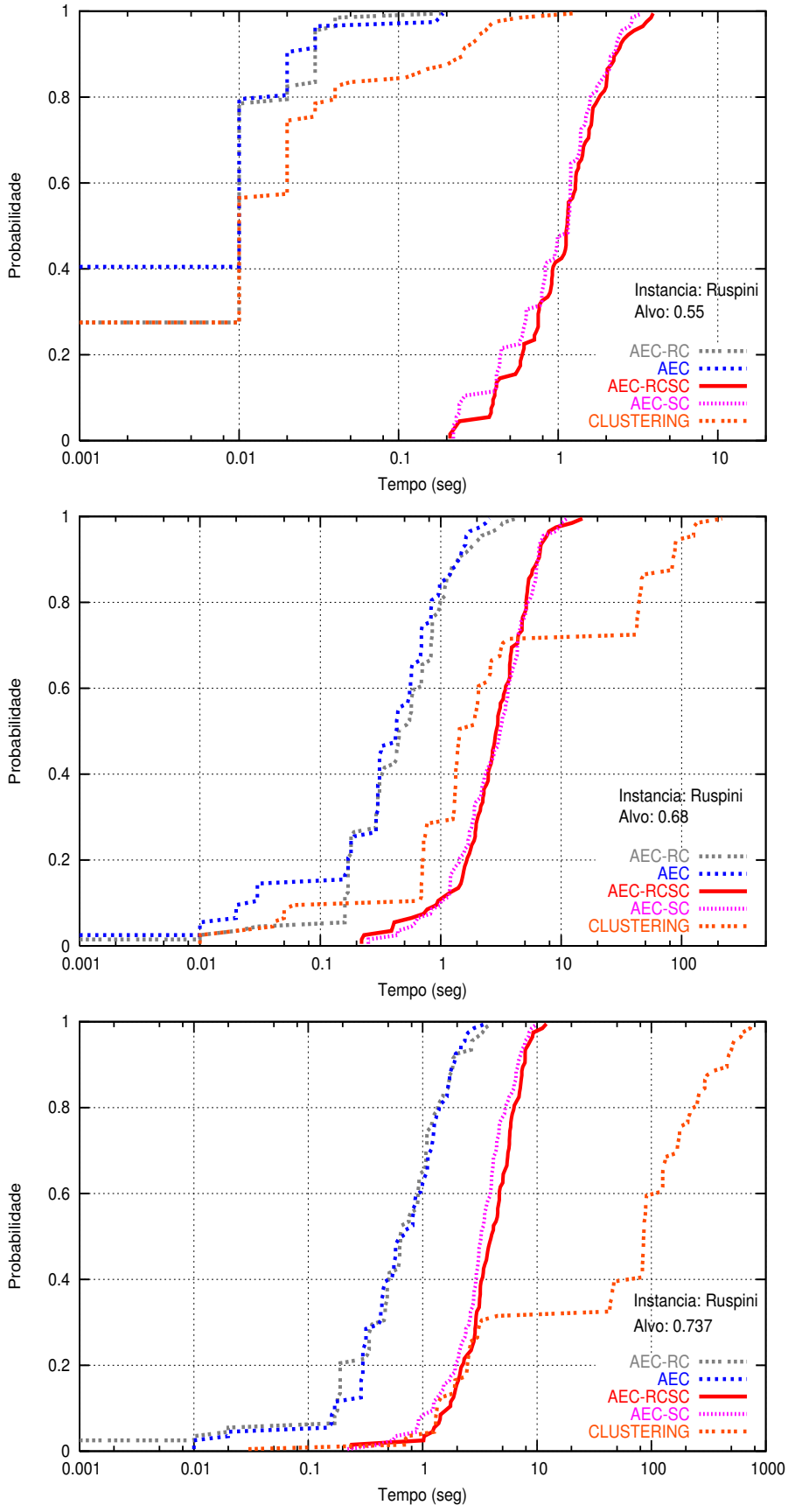

Figura 5.17: Análise Probabilística para RuspiniData sem método de construção

Para a instância 200Data (Figura 5.18), notamos um equilíbrio ainda maior entre as abordagens sem método de construção. Podemos perceber que para o alvo fácil, o algoritmo mais eficiente é o AEC, seguido do AEC-RC. O terceiro melhor desempenho foi obtido pelas duas abordagens evolutivas sem método de construção, que apresentaram suas curvas no gráfico praticamente coincidentes. Mais uma vez enfatizamos o fato das abordagens com construção conseguirem chegar na solução alvo com probabilidade sempre acima das abordagens sem construção. Além disso, o CLUSTERING não conseguiu superar nenhum do algoritmos propostos com ou sem método de construção independentemente do alvo. No caso do alvo médio, novamente o AEC e o AEC-RC apresentaram o primeiro e segundo melhor desempenho, respectivamente. Entretanto notamos uma pequena melhora do AEC-RCSC em relação ao AEC-SC em probabilidades próximas de 1,0. Este comportamento não se verifica para o alvo difícil. Contudo, em probabilidades um pouco acima de 0.8, há um equilíbrio entre os dois algoritmos sem procedimentos de construção.

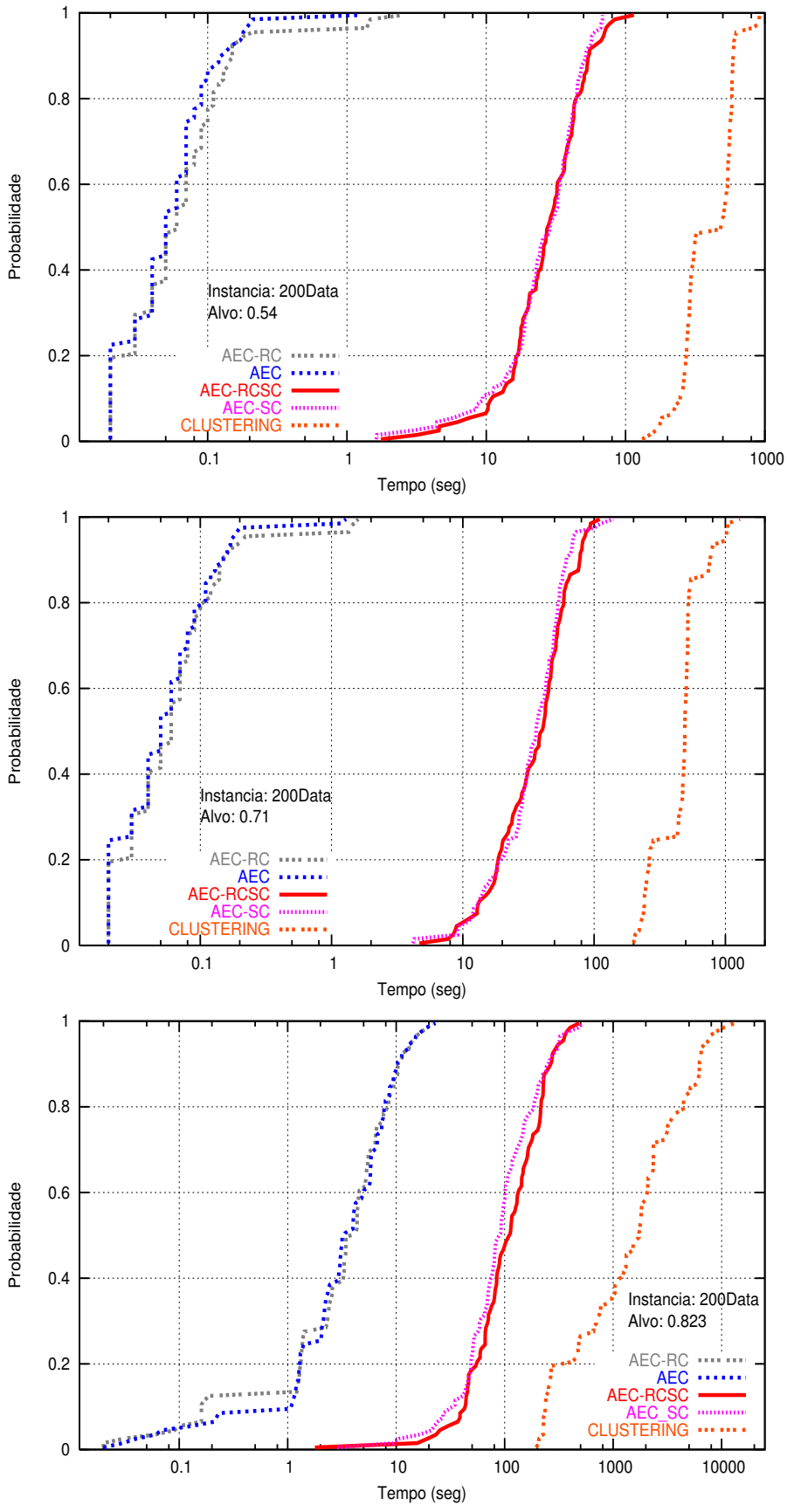

Figura 5.18: Análise Probabilística para a instância 200Data sem método de construção

As outras duas instâncias da literatura são apresentadas a seguir. A Figura 5.19 nos mostra os resultados para a instância $IrisData$. O comportamento do AEC-SC e do AEC-RCSC para esta instância é similar ao apresentado pelos mesmos na instância $200Data$.

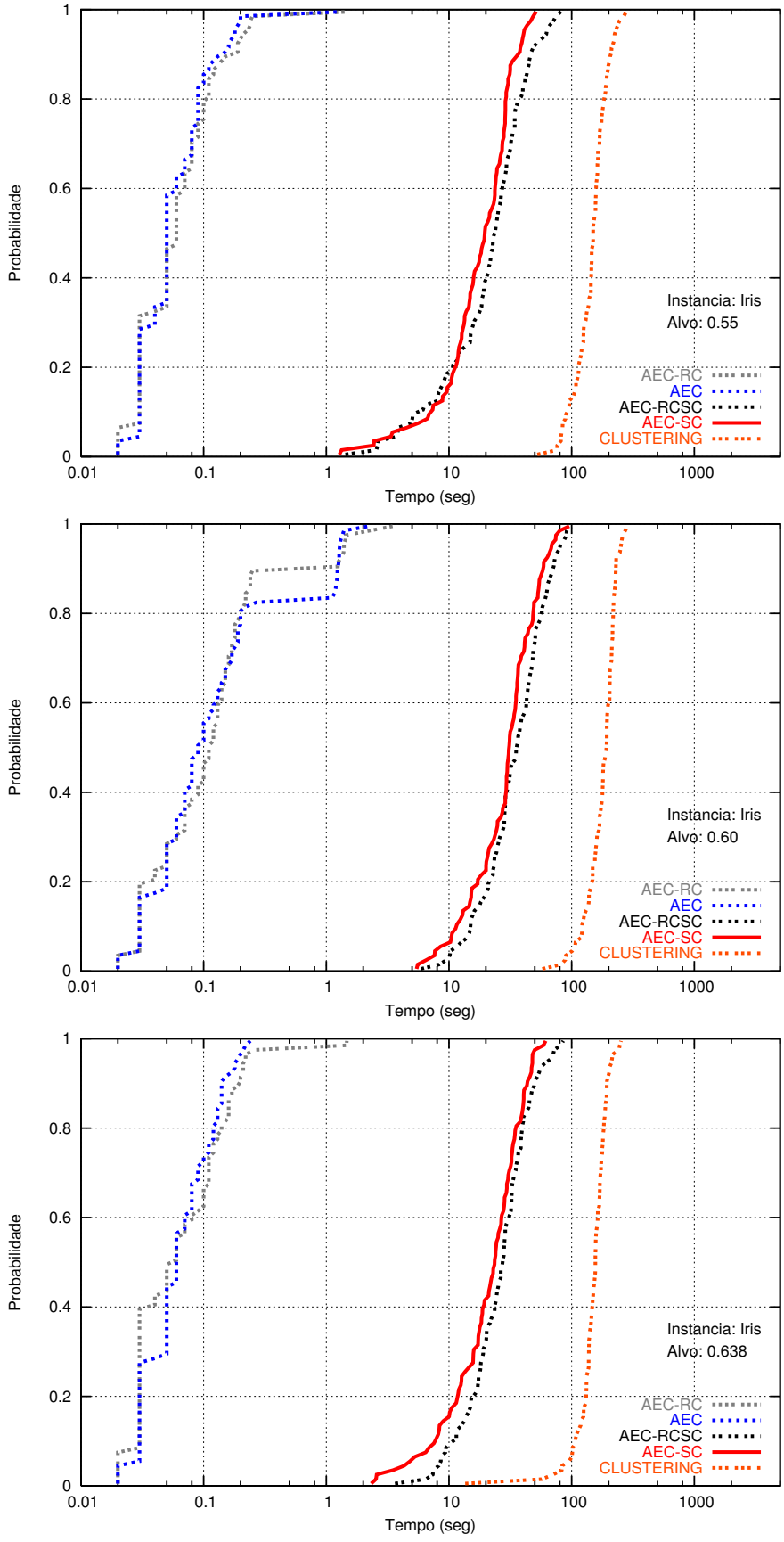

Figura 5.19: Análise Probabilística para IrisData sem método de construção

Tendo em vista que o tempo de processamento consumido pelos algoritmos sem método de construção é muito grande (ficando acima dos 20.000 segundos), não apresentamos os gráficos referentes aos alvos médio e difícil para a instância *CancerData*. A Figura 5.20 mostra o desempenho dos algoritmos para o alvo fácil sobre esta instância.

Percebe-se na Figura 5.20 que, analisando a taxa de convergência para valores próximos de 100%, observamos que no tempo em que o AEC e o AEC-RC apresentam 100% de chance de atingir o alvo, os dois algoritmos sem procedimento de construção apresentam 0% de chance. Observa-se ainda que os resultados do AEC-SC e AEC-RCSC são similares para probabilidade próxima de 1.0, embora ao longo da curva haja intervalos em que o AEC-SC supera o AEC-RCSC, o que nos faz acreditar que a Reconexão por Caminhos, mesmo oferecendo custo adicional, constitui uma boa estratégia. Vale ressaltar novamente que esta instância apresentou, para os testes com o número de gerações como critério de parada, algumas execuções em que o AEC não conseguia obter a melhor solução conhecida da literatura, fato não ocorrido para o AEC-RC.

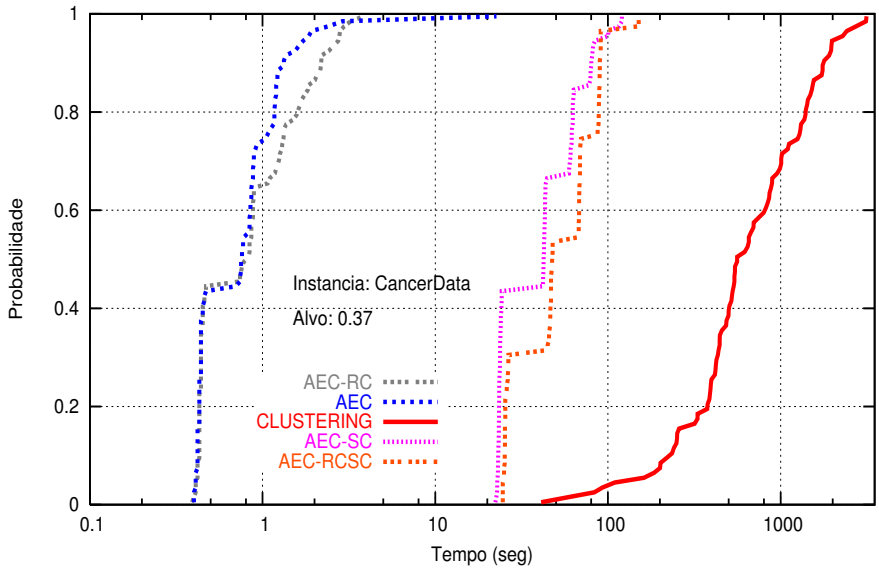

Figura 5.20: Análise Probabilística para CancerData sem método de construção

Percebemos que o módulo de construção aqui proposto é de grande importância para o desempenho dos AG's. Para instâncias maiores, percebemos também

que os AG's propostos fornecem soluções melhores mesmo sem usar o módulo de construção, quando comparados com o CLUSTERING.

Capítulo 6

Conclusões e Propostas de Trabalhos Futuros

Neste trabalho, apresentamos novas heurísticas para resolver aproximadamente o Problema de Clusterização Automática (PCA). As nossas contribuições são módulos que, incluídos num Algoritmo Genético ou num *Simulated Annealing* conseguem obter soluções de melhor qualidade qua as versões tradicionais de AG's.

Dentre as contribuições, destacamos o algoritmo construtivo para gerar uma população de soluções iniciais que pode ser usada para qualquer metaheurística existente na literatura.

Outras contribuições incluem: um módulo de Reconexão por Caminhos (RC) incluído nos nossos AE's e o gerador de problemas testes denominado *Dots*.

O desempenho dos algoritmos propostos quando comparados com o algoritmo CLUSTERING da literatura mostrou uma nítida superioridade do AEC, AEC-RC e SAPCA tanto na qualidade das soluções geradas como no tempo computacional exigido. Numa comparação entre os algoritmos propostos, observou-se uma superioridade do AEC-RC e do SAPCA em relação ao AEC.

Na média, o SAPCA obteve mais vitórias, mas em algumas instâncias difíceis o

AEC-RC obteve melhores resultados. Isso ocorreu também na análise das imagens obtidas das soluções de algumas instâncias no espaço R^2, onde existiu sempre uma superioridade do AEC-RC ou do SAPCA.

Em termos de trabalhos futuros, podemos colocar os seguintes tópicos:

- Desenvolver outras metaheurísticas tipo GRASP, VNS e Busca Tabu utilizando o método de construção aqui proposto e a Reconexão por Caminhos para o PCA;
- Desenvolver versões paralelas de metaheurísticas procurando reduzir os tempos computacionais para instâncias de grande porte, como também tentar, com a comunicação entre processadores, tornar estes algoritmos mais cooperativos e adaptativos, buscando com isso, obter soluções de melhor qualidade que as obtidas pelas suas versões seqüenciais.

Referências Bibliográficas

[1] Abramowitz, M. *Handbook of Mathematical Functions.* No. 55 in Mathematical Applied Series. National Bureau of Standards, 1964.

[2] Aguilar, J., and Gelenbe, E. Task assignment and transaction clustering heuristics for distributed systems. *Information Sciences 97* (March 1997), 199–219.

[3] Ahuja, R. K., Orlin, J. B., and Sharma, D. Very large-scale neighborhood search. *International Transactions in Operational Research 7* (September 2000), 301–317.

[4] Aiex, R., Binato, S., and Resende, M. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing 29* (2003), 393–430.

[5] Al-Fawzan, M. A., and Al-Sultan, K. S. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. *Computers & Industrial Engineering 44* (January 2003), 35–47.

[6] Ananthanarayana, V. S., Murty, M. N., and Subramanian, D. K. Efficient clustering of large data sets. *Pattern Recognition 34* (December 2001), 2561–2563.

[7] Asharaf, S., and Murty, M. N. An adaptive rough fuzzy single pass algorithm for clustering large data sets. *Pattern Recognition 36* (December 2003), 3015–3018.

[8] Bandyopadhyay, S., and Maulik, U. Genetic clustering for automatic evolution of clusters and application to image classification. *Patter Recognition 35* (May 2001), 1197–1208.

[9] Bandyopadhyay, S., and Maulik, U. An evolutionary technique based on k-means algorithm for optimal clustering in r^n. *Information Sciences 146* (October 2002), 221–237.

[10] Bandyopadhyay, S., and Maulik, U. A novel approach to clustering problem. *Information Sciences 146* (October 2002), 221–237.

[11] Barbarosoglu, G., and Ozgur, D. A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research 26* (March 1999), 255–270.

[12] Beauchamp, N. J., van Achterberg, T. A. E., and Engelse, M. A. Gene expression profiling of resting and activated vascular smooth muscle cells by serial analysis of gene expression and clustering. *Genomics 82* (April 2003), 288–299.

[13] Berkhin, P. Surveys of clustering data mining techniques. Accrue Software,inc, San Jose, CA, 2002.

[14] Betanzos, A. A., Varela, B. A., and Martínez, A. C. Analysis and evaluation os hard and fuzzy clustering segmentation techniques in burned patient images. *Image Vivion Computing 18* (April 2000), 1045–1054.

[15] Bolshakova, N., and Azuaje, F. Cluster validation techniques for genoma expression data. *Signal Processing 83* (September 2003), 825–833.

[16] Bradley, P., Fayyad, U., and Reina, C. Scaling clustering algorithms to large databases. In *4th ACM Special Interest Group on Knowledge Discovery in Data and Data Mining - SIGKDD* (New York - USA, 1998), pp. 91–99.

[17] Brown, D. H., and Bolker, B. M. The effects of disease dispersal and host clustering on the epidemic threshold in plants. *Bulletin of Mathematical Biology 66* (March 2004), 341–371.

[18] Brucker, P. On the complexity of clustering problems. In *Optimization and Operations Research*, M. Beckmenn and H. P. Kunzi, Eds., vol. 157 of *Lectures Notes in Economics and Mathematical Systems*. Springer, 1978, pp. 45–54.

[19] Campello, R. E., and Maculan, N. *Algoritmos e Heurísticas: Desenvolvimento e avaliação de performance*. EDUFF: Editora da Universidade Federal Fluminense, 1994.

[20] Chamarthya, P., Stanley, R. J., and Cizek, G. Image analysis techniques for characterizing disc space narrowing in cervical vertebrae interfaces. *Computerized Medical Imaging and Graphics 28* (January-March 2004), 39–50.

[21] Chandy, J., Kim, S., Ramkumar, B., Parkes, S., and Banerjee, P. An evaluation of parallel simulated annealing strategies with application to standard cell placement. *IEEE Trans. on Comp. Aid. Design of Int. Cir. and Sys 16* (1997).

[22] Chiou, Y.-C., and Lan, L. W. Genetic clustering algorithms. *European Journal of Operational Research 135* (2001), 413–427.

[23] Chung, Y. D., and Kim, M. H. A wireless data clustering method for multipoint queries. *Decision Support Systems 30* (March 2001), 469–482.

[24] Cohn, D. E., Babb, S., Whelan, A. J., Mutch, D. G., Herzog, T. J., Rader, J. S., Elbendary, A., and Goodfellow, P. J. Atypical clustering of gynecologic malignancies: A family study including molecular analysis of cadate genes. *Gynecologic Oncology 77* (April 2000), 18–25.

[25] Cotta, C., and Moscato, P. A memetic-aided approach to hierarchical clustering from distance matrices: application to gene expression clustering and phylogeny. *Biosystems 72* (November 2003), 75–97.

[26] Cuesta-Frau, D., Pérez-Cortés, J. C., and Adreu-García, G. Clustering of electrocardiograoph signals in computer-aided holter analysis. *Computer Methods and Programs in Biomedicine 72* (September 2003), 197–196.

[27] Darwin, C. *The Origins of Species by Means of Natural Selection.* Penguin Classics, 1859.

[28] de Souza, R. M. C. R., and de A T de Carvalho, F. Clustering of interval data based on city-block distances. *Pattern Recognition Letters 25* (February 2004), 353–365.

[29] Deb, K., and Reddy, A. R. Reliable classification of two-class cancer data using evolutionary algorithms. *Biosystems 72* (November 2003), 111–129.

[30] Devillez, A., Billaudel, P., and Lecolier, G. V. A fuzzy hybrid hierarchical clustering method with a new criterion able to find the optimal partition. *Fuzzy Sets and Systems 128* (June 2002), 323–338.

[31] Dias, C. R., and Ochi, L. S. Efficient evolutionary algorithms for the clustering problem in directed graphs. In *IEEE-CEC Congress on Evolutionary Computation em CD-ROM* (Canberra - Austrália, 2003), pp. 983–990.

[32] Drew, M. S., and Au, J. Clustering of compressed illumination-invariant chromaticity signatures for efficient video summarization. *Image and Vision Computing 21* (August 2003), 705–716.

[33] Ester, M., Kriegel, H. P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial database with noise. In *International Conference on Knowledge Discovery in Databases and Data Mining(KDD-96)* (1996), pp. 226–231.

[34] Ester, M., Kriegel, H. P., and Xu, X. A database interface for clustering in large spatial databases. In *International Conference on Knowledge Discovery in Databases and Data Mining (KDD-95)* (1995), pp. 94–99.

[35] Estivill-Castro, V., and Lee, I. Argument free clustering for large spatial point-data sets via boundary extraction from delaunay diagram. *Computers, Environment and Urban Systems 26* (July 2002), 315–334.

[36] Fang, J.-T., Shyu, W.-L., Wu, C.-S., and Chen, K.-J. Clustering source output bits and equalizing bit error sensitivity to improve the quality and

robustness of transmitted images over wireless channels. *Signal Processing: Image Communication 18* (November 2003), 947–956.

[37] Festa, P., Pardalos, P., Resende, M. G., and Ribeiro, C. GRASP and VNS for Max-Cut. In *Proceedings of the IV Metaheuristics International Conference (MIC2001)* (2001), J. Sousa, Ed., pp. 371–376.

[38] Festa, P., and Resende, M. GRASP: An annotated bibliography. In *Essays and Surveys on Metaheuristics*, C. Ribeiro and P. Hansen, Eds. Kluwer Academic Publishers, 2002, pp. 325–367.

[39] Fisher, R. The use of multiple measurements in taxonomic problems. *Annual Eugenics 7* (1936), 179–188.

[40] França, P. M., Sosa, N. M., and Pureza, V. An adaptive tabu search algorithm for the capacitated clustering problem. *International Transactions in Operational Research 6* (November 1999), 655–678.

[41] Friedmann, R., and Sanddorf-Köhle, W. G. Volatility clustering and nontrading days in chinese stock markets. *Journal of Economics and Business 54* (March-April 2002), 193–217.

[42] Fukunaga, K. *Introduction to Statistical Pattern Recognition.* Academic Press, Inc, 1990.

[43] Garai, G., and Chaudhuri, B. B. A novel genetic algorithm for automatic clustering. *Pattern Recognition Letters 25* (January 2004), 173–187.

[44] Glover, F., Laguna, M., and Martí, R. Fundamentals of scatter-search and path relinking. *Control ans Cybernetics 36* (2000), 653–684.

[45] Goil, S., and Choudhary, A. N. Efficient parallel classification using dimensional aggregates. In *Large-Scale Parallel Data Mining* (1999), pp. 197–210.

[46] Gonçalves, J., and Resende, M. A hybrid genetic algorithm for manufacturing cell formation. Tech report, Internet and Network Systems Research

Center, AT&T Labs Research, Florham Park, NJ, 2002. Submitted to *Computers and Industrial Engineering.*

[47] Gonçalves, J., Mendes, J., and Resende, M. A hybrid genetic algorithm for the job shop scheduling problem. *European J. of Operational Research* (2004). To appear.

[48] Gonçalves, J. F., and Resende, M. G. C. A hybrid genetic algorithm for manufactoring cell formation. In *Computer and Industrial Engineering* (2002).

[49] Guha, S., Rastrogi, R., and Shim, K. Rock: A robust clustering algorithm for categorical attributes. *Information Systems 25* (July 2000), 345–366.

[50] Guha, S., Rastrogi, R., and Shim, K. CURE: A clustering algorithm for large databases. *Information Systems 26* (March 2001), 35–58.

[51] Guo, H., Renaut, R., Chen, K., and Reiman, E. Clustering huge data sets for parametric pet imaging. *Biosystems 71* (September 2003), 81–92.

[52] Han, J., and Kamber, M. *Data Mining: Concepts and Techiniques.* Morgan Kaufmann, 2001.

[53] Hanson, C. E., and Wieczorek, W. F. Alcohol mortality: a comparison of spatial clustering methods. *Social Science & Medicine 55* (September 2002), 791–802.

[54] He, Z., Xu, X., and Deng, S. Discovering cluster-based local outliers. *Patter Recognition Letters 24* (December 2002), 1641–1650.

[55] Hemminki, K., and Granström, C. Familial clustering of ovarian and endometrial cancers. *European Journal of Cancer 40* (January 2004), 90–95.

[56] Hemminki, K., and Li, X. Association of brain tumours with other neoplasms in families. *European Journal of Cancer 40* (January 2004), 253–259.

[57] Hirano, S., Sun, X., and Tsumoto, S. Comparison of clustering methods for clinical databases. *Information Sciences 159* (March 2004), 155–165.

[58] Hodge, V. J., and Austin, J. Hierarchical word clustering automatic thesaurus generation. *Neurocomputing 48* (October 2002), 819–846.

[59] Holland, J. H. *Adaptation in Nature and Artificial Systems.* University of Michigan Press - MI, 1975.

[60] Hopcroft, J. E., Ullman, J. D., and Motwani, R. *Introdução à Teoria de Autômatos, Linguagens e Computação.* Editora Campus, 2002.

[61] Hruschka, E. R. Algoritmos genéticos de agrupamento para extração de regras de redes neurais. Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2001.

[62] Huang, J., Shimizu, H., and Shioya, S. Clustering gene expression pattern and extracting relationship in gene network based on artificial neural networks. *Journal of Bioscience and Bioengineering 96* (December 2003), 421–428.

[63] Inmon, W. H. *Building the Data Warehouse.* John Wiley & Sons, 1993.

[64] Jain, A. K., and Flynn, P. J. *Algorithms for Clustering Data.* Prentice-Hall, England Cliffs, NJ, 1988.

[65] Josien, K., and Liao, T. W. Simultaneous grouping of parts and machines with an integrated fuzzy clustering method. *Fuzzy Sets and Systems 126* (January 2001), 1–21.

[66] Karypis, G., and Kumar, V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *COMPUTER 32* (2002), 68–75.

[67] Kaufman, L., and Rousseeuw, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis.* A Wiley-Interscience publicatiion, 1990.

[68] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220, 4598* (1983), 671–680.

[69] Kuate-Defo, B., and Diallo, K. Geography of child mortality clustering within african families. *Health & Place 8* (June 2002), 93–117.

[70] LAGENDIJK, A. *The Dynamics of Industrial Clustering. International Comparisons in Computing and Biotechnology*, vol. 30 of *Research Policy*. Oxford University Press, February 2001.

[71] LEE, S.-L., AND CHUNG, C.-W. Hyper-rectangle based segmentation and clustering of large video data sets. *Information Sciences 141* (March 2002), 139–168.

[72] LEONARD, B. J. W. *Inteligência e Afetividade da Criança na Teoria de Piaget*. Pioneira Thomson Learning, 1990.

[73] LI, C.-T., AND CHIAO, R. Multiresolution genetic clustering algorithm for texture segmentation. *Image and Vision Computing 21* (October 2003), 955–966.

[74] LO, C.-C., AND WANG, S.-J. A histogram-based moment-preserving clustering algorithm for video segmentation. *Pattern Recognition Letters 24* (October 2003), 2209–2218.

[75] LU, H., AND TAN, Y.-P. Unsupervised clustering of dominant scenes in sports video. *Pattern Recognition Letters 24* (November 2003), 2651–2662.

[76] MACQUENN, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symp. Math. Statist. Prob.* (1967), L. L. Cam and J. Neyman, Eds., pp. 281–297.

[77] MANGASARIAN, O. L., AND WOLBERG, W. H. Cancer diagnosis via linear programming. *SIAM News 23* (1990), 1–18.

[78] MARTENSEN, H., MARIS, E., AND DIJKSTRA, T. Phonological ambiguity and context sensitivity: On sublexical clustering in visual word recognition. *Journal of Memory and Language 49* (October 2003), 375–395.

[79] MASULLI, F., AND SCHENONE, A. A fuzzy clustering based segmentation system as support to diagnosis in medical imaging. *Artificial Intelligence in Medicine 16* (October 1999), 129–147.

[80] Metropolis, M., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and teller, E. Equation of state calculation by fast computing machines. *Jornal of Chemistry and Physics 21* (1953), 1087–1091.

[81] Michalewicsj, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.

[82] Mitchell, J. E., Pardalos, P. M., and Resende, M. G. C. Interior point algorithms for combinatorial optimization. In *Handbook of combinatorial optimization*, D.-Z. Du and P. M. Pardalos, Eds., vol. 1. Kluwer Academic Publishers, 1998, pp. 189–298.

[83] Miyamoto, S. Information clustering based on fuzzy multisets. *Information Processing & Management 39* (March 2003), 195–213.

[84] Mulder, S. A., and Wunsch, D. C. Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks. *Neural Networks 16* (June-July 2003), 827–832.

[85] Murphy, K. J., and Dickinson, G. Aquatic plant communities and predictors of diversity in a sub-tropical river floodplain: the upper Rio Paraná, Brazil. *Aquatic Botany 77* (December 2003), 257–276.

[86] Nakanishi, N., Nishina, K., and Okamoto, M. Clustering of components of the metabolic syndrome and risk for development of type 2 diabetes in japanese male office workers. *Diabetes Research and Clinical Practice 63* (March 2004), 185–194.

[87] Ng, M. K., and Wong, J. C. Clustering categorical data sets using tabu search techniques. *Pattern Recognition 35* (December 2002), 2783–2790.

[88] Ng, R., and Han, J. Efficient and effective clustering methods for spatial data mining. In *Proceedings of International Conference on Very Large Data Base* (1994), J. Sousa, Ed., pp. 144–155.

[89] Ochi, L. S., Souza, M. J. F., and Maculan, N. A GRASP - TABU SEARCH algorithm to solve a School Timetabling Problem. *Combinatorial Optimization Book Series, Metaheuristics: Computer Decision 15* (2003), 659–672.

[90] Osman, I. H., and Christofides, N. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research 1* (July 1994), 317–336.

[91] Ozer, M. User segmentation of online music services using fuzzy clustering. *Omega 29* (April 2001), 193–206.

[92] Pedrycz, W. Fuzzys clustering with a knowledge-based guidance. *Pattern Recognition Letters 25* (March 2004), 469–480.

[93] Peña, J. M., Lozano, J. A., and Larrañaga, P. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters 20* (October 1999), 1027–1040.

[94] Questier, F., Walczak, B., and Massart, D. L. Feature selection for hierarchical clustering. *Analitica Chimica Acta 466* (June 2002), 311–324.

[95] Quin, P., Agerbo, E., and Mortensen, P. B. Suicide risk in relation to family history of completed suicide and psychiatric disorders: a nested case-control study based on longitudinal registers. *The Lancet 360* (Octuber 2002), 1126–1130.

[96] Reinhold, V. N. *Handbook of Genetic Algorithms.* Lawrence Davis, 1991.

[97] Resende, L. I. P., and Resende, M. G. C. A GRASP for frame relay permanent virtual circuit routing. In *Proceedings of the III Metaheuristics International Conference (MIC99)* (1999), P. Hansen and C. C. Ribeiro, Eds., pp. 397–401.

[98] Resende, M., and Ribeiro, C. C. A grasp with path-relinking for private virtual circuit routing. *Networks 41*, 1 (2003), 104–114.

[99] Resende, M., and Werneck, R. A hybrid heuristic for the p-median problem. Tech report, Internet and Network Systems Research Center, AT&T Labs Research, Florham Park, NJ, 2002. Submitted to *J. of Heuristics*.

[100] Resende, M. G. C. Grasp: A bibliography. In *Proceedings of the III Metaheuristics International Conference (MIC99)* (1999), P. Hansen and C. C. Ribeiro, Eds., pp. 403–410.

[101] Resende, M. G. C. Greedy randomized adaptive search procedures (GRASP). In *Encyclopedia of Optimization*, vol. 2. Kluwer Academic Publishers, 2001, pp. 373–382.

[102] Rosato, V., Pucello, N., and Giuliano, G. Evidence for cystine clustering in thermophilic proteomes*1. *Trends in Genetics 19* (November 2003), 607–608.

[103] Rosenthal, M., Carvalho, L. A. V., Laks, J., and Maculan, N. Searching and analyzing neuropsychological patterns in schizophrenia with artificial neural networks. Tech Report 499, COPPE - SISTEMAS, 1999.

[104] Schrader, R. Approximations to clustering and subgraph problems on trees. *Discrete Applied Mathematics 6* (1983), 301–309.

[105] Sciutto, E., and Martínez, J. J. Familial clustering of taenia solium cysticercosis in the rural pigs of mexico: hints of genetic determinants in innate and acquired resistance to infection. *Veterinary Parasitology 116* (October 2003), 226–229.

[106] Shelokar, P. S., Jayaraman, V. K., and Kulkarni, B. D. An ant colony approach for clustering. *Analytica Chimica Acta 37* (February 2004), 175–188. In Press, Corrected Proof.

[107] Shimoji, T., and Kanda, H. Clinico-molecular study of dedifferentiation in well-differentiated liposarcoma. *Biochemical and Biophysical Research Communications 314* (February 2004), 1133–1140.

[108] SMITH, K. A., AND NG, A. Web page clustering using a self-organizing map of user navigator patterns. *Decision Suport Systems 35* (2003), 245–256.

[109] SOKAL, R. R., AND A, P. H. *Priciples of Numeric Taxonomy.* Freeman, 1963.

[110] SUNG, C. S., AND JIN, H. W. A tabu-search-based heuristic for clustering. *Pattern Recognition 33* (May 2000), 849–858.

[111] SZWARCWALD, C. L., DE ANDRADE, C. L. T., AND BASTOS, F. I. Income inequality, residential poverty clustering and infant mortality: a study in Rio de Janeiro, Brazil. *Social Science & Medicine 55* (December 2002), 2083–2092.

[112] TOUSSAINT, G. T. The relative neighborhood graph of a finite planar set. *Pattern Recognition 12* (1980), 261–268.

[113] TRINDADE, A. R., AND OCHI, L. S. Desenvolvimento e análise experimental de um algoritmo evolutivo para o problema de clusterização em sistemas de manufatura. In *Congresso Brasileiro de Computação (IV CBcomp) - artigo aceito* - (2004).

[114] TSAI, C.-F., TSAI, C.-W., WU, H.-C., AND YANG, T. ACODF: a novel data clustering approach for data mining in large databases. *The Journal of Systems and Software, in press* (2003).

[115] TSENG, L. Y., AND YANG, S. B. A genetic clustering algorithm for data with non-spherical-shape clusters. *Pattern Recognition 33* (July 2000), 1251–1259.

[116] TSENG, L. Y., AND YANG, S. B. A genetic approach to the automatic clustering problem. *Pattern Recognition 34* (Februrary 2001), 415–424.

[117] TUCKER, L. R. The extension of factor analysis to three-dimensional matrices. *Contribitions to Mathematical Psychology*, 1.2 (1964).

[118] VAKHARIA, A. J., AND MAHAJAN, J. Clustering of objects and attributes for manufacturing and marketing applications. *European Journal of Operational Research 123* (June 2000), 640–651.

[119] Vijaya, P. A., Murty, M. N., and Subramanian, D. K. Leaders-subleaders: An efficient hierarchical clustering algorithm for large data sets. *Pattern Recognition Letters 25* (March 2004), 505–513.

[120] Waes, L. V., and Schellens, P. J. Writing profiles: the effect of the writing mode on pausing and revision patterns of experienced writers. *Journal of Pragmatics 35* (June 2003), 829–853.

[121] Wang, J. Formation of machine cells and families in cellular manufactoring systems using a linear assignment algorithm. *Automatica 39* (April 2003), 1607–1615.

[122] Wang, L., and Bai, J. Threshold selection by clustering gray levels of boundary. *Pattern Recognition Letters 24* (August 2003), 1983–1999.

[123] Wei, C.-P., Lee, Y.-H., and Hsu, C.-M. Empirical comparison of fast partitioning-based clustering algorithms for large data set. *Expert Systems with Applications 24* (2003), 351–363.

[124] Welch, J. W. Algorithmic complexty: three NP-hard problems in computational statistics. *Journal os Statistical Computation ans Simulation 15* (1983), 17–25.

[125] Yang, M.-S., Hu, Y.-J., Lin, K. C.-R., and Lin, C. C.-L. Segmentation techniques for tissue differentiation in mri of ophthalmology using fuzzy clustering algoritms. *Magnetic Resonance Imaging 20* (January 2002), 173–179.

[126] Yang, M.-S., Hwang, P.-Y., and Chen, D.-H. Fuzzy clustering algorithms for mixed feature variables. *Fuzzy Sets and Systems 141* (january 2004), 301–317.

[127] Yang, T.-N., and Wang, S.-D. Competitive algorithms for the clustering of noisy data. *Fuzzy Sets and Systems 141* (january 2004), 281–299.

[128] Zang, T., Ramakrishnan, R., and livny, M. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD Conference on Management of Data* (Montreal - Canadá, 1996), pp. 103–114.

[129] Öhman, K., and Lämås, T. Clustering of harvest activities in multi-objective long-term forest planning. *Forest Ecology and Management 176* (March 2003), 161–171.