FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
MESTRADO EM INFORMÁTICA APLICADA

Everlândio Rebouças Queiroz Fernandes

PROJETO MULTIOBJETIVO DE FUSORES HIERÁRQUICOS DE PARTIÇÕES DE DADOS VIA PROGRAMAÇÃO GENÉTICA

Fortaleza
2009

F363p Fernandes, Everlândio Rebouças Queiroz.
Projeto multiobjetivo de fusores hierárquicos de partições de dados via programação genérica / Everlândio Rebouças Queiroz Fernandes. - 2009.
103 f.

Dissertação (mestrado) – Universidade de Fortaleza, 2009.
"Orientação: Prof. Dr. André Luís Vasconcelos Coelho."
"Co-orientação: Profa. Dra. Katti Faceli."

1. Algoritmos genéticos. 2. Programação genética. 3. Agrupamento de dados. 4. Otimização matemática. I. Título.

CDU 681.3:004.021

FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
CENTRO DE CIÊNCIAS TECNOLÓGICAS
MESTRADO EM INFORMÁTICA APLICADA

Everlândio Rebouças Queiroz Fernandes

PROJETO MULTIOBJETIVO DE FUSORES HIERÁRQUICOS DE PARTIÇÕES DE DADOS VIA PROGRAMAÇÃO GENÉTICA

Dissertação submetida à coordenadoria do Curso de Mestrado em Informática Aplicada (MIA) da Universidade de Fortaleza como requisito parcial para obtenção do Título de Mestre em Informática.

Orientador: Prof. Dr. André Luís Vasconcelos Coelho

Co-orientadora: Profª. Dra. Katti Faceli

Fortaleza
2009

Everlândio Rebouças Queiroz Fernandes

PROJETO MULTIOBJETIVO DE FUSORES HIERÁRQUICOS DE PARTIÇÕES DE DADOS VIA PROGRAMAÇÃO GENÉTICA

Data de Aprovação __/__/____

Banca Examinadora:

Prof. André Luís Vasconcelos Coelho, D. Sc.
(Orientador: Universidade de Fortaleza – Unifor)

Rafael Duarte Coelho dos Santos, D. Sc.
(Membro Externo: Instituto Nacional de Pesquisas Espaciais – INPE)

Prof. Cícero Nogueira dos Santos, D. Sc.
(Membro Interno: Universidade de Fortaleza – Unifor)

Aos meus amados pais,

Manoel Queiroz e Maria Geiza.

AGRADECIMENTOS

Inicialmente, agradeço a Deus por estar sempre presente na minha vida, indicando o caminho correto, e por tornar todos os meus anseios possíveis.

Agradeço imensamente aos meus pais, Manoel e Geiza, que sempre incentivaram de forma marcante o meu crescimento intelectual e profissional, e pela confiança que depositam em mim. Aos meus irmãos, Éverton e Égna, pelos ensinamentos e apoio.

Ao professor André Luís Vasconcelos Coelho, pela orientação objetiva, por sua contagiante motivação pela pesquisa, pela paciência, atenção e, principalmente, pela oportunidade de realizar este trabalho.

À professora Katti Faceli, pois sua disponibilidade, empenho, conhecimentos e contribuições científicas foram essenciais para a realização deste trabalho.

Agradeço ao Banco do Nordeste do Brasil pelo apoio financeiro e por acreditar que o crescimento da instituição está intimamente ligado ao crescimento de seus funcionários.

Aos meus amigos e colegas do BNB, em especial à Célula de Qualidade de Sistemas, que contribuíram, direta ou indiretamente, com o exercício diário da confiança e da compreensão, ao longo deste trabalho.

Por fim, aos meus amigos mais próximos, que não duvidaram de minha capacidade, toleraram minhas ausências, compreenderam de forma imparcial o meu estresse e, claro, pelo carinho e amizade demonstrados.

Resumo da dissertação apresentada ao corpo docente do programa de Mestrado em Informática Aplicada da Universidade de Fortaleza, como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática.

Projeto Multiobjetivo de Fusores Hierárquicos de Partições de Dados via Programação Genética

Autor: Everlândio Rebouças Queiroz Fernandes
Orientador: Prof. Dr. André Luís Vasconcelos Coelho
Co-orientadora: Profª. Dra. Katti Faceli

Um notável avanço vem sendo recentemente obtido na área de agrupamento de dados mediante o desenvolvimento de métodos de fusão de partições. Essa abordagem, conhecida como *clustering ensembles*, consiste em combinar os resultados de múltiplos agrupamentos de uma mesma base de dados em uma única partição-consenso. Embora promissora, essa abordagem ainda é restritiva, já que uma única resposta para um problema limita a aquisição do conhecimento que poderia ser obtido considerando outras possíveis soluções (partições). Por outro lado, devido à existência de vários critérios de avaliação da qualidade de agrupamentos, pode-se modelar essa tarefa como um problema típico de otimização multiobjetivo. Nesse contexto, o presente estudo apresenta uma nova abordagem, baseada em programação genética multiobjetivo, que projeta automaticamente novos operadores hierárquicos de fusão de partições. Desse modo, um conjunto inicial de partições, obtido via a aplicação de diferentes técnicas de agrupamento, pode ser continuamente refinado através de uma população de hierarquias de fusores, que selecionam e combinam as partições originais, utilizando diferentes critérios de qualidade como funções-objetivo. Para validar a nova abordagem, em termos de eficiência e eficácia, foi implementado um protótipo e conduzido um estudo comparativo, envolvendo outros algoritmos de agrupamento (dentre os quais três são de *clustering ensembles* e dois são multiobjetivo), sobre 10 diferentes bases de dados. Os experimentos demonstram que, em geral, a ideia de se ter uma hierarquia de fusores aliada à correta seleção das partições pode proporcionar ganhos significativos em termos de eficácia e robustez.

Palavras-chave: Agrupamento de Dados. *Clustering Ensembles*. Fusão Hierárquica de Partições. Algoritmos Evolutivos Multiobjetivos. Programação Genética.

Abstract of the dissertation presented to the board of faculties of the Master Program in Applied Informatics at the University of Fortaleza, as partial fulfillment of the requirements for the Master's degree in Computer Science.

Multiobjective Design of Hierarchical Fusion Operators for Clustering Ensembles via Genetic Programming

Author: Everlândio Rebouças Queiroz Fernandes
Advisor: André Luís Vasconcelos Coelho, Ph.D.
Co-advisor: Katti Faceli, Ph.D.

A remarkable progress has been recently achieved in the area of data clustering, in part due to the development of clustering ensemble methods. In a nutshell, this approach aims at combining multiple partitions produced over the same dataset into a single consensus partition. Although promising, this approach is still restrictive in the sense that obtaining a single solution (partition) as result limits the knowledge that could be grasped from the data, which could contain several meaningful alternative solutions. On the other hand, there exist several validation criteria to assess the data partitions, each considering a distinct viewpoint. This permits to model the data clustering task as a typical multiobjective optimization problem. This strategy, which has also gained much attention in the last years, is known as multiobjective clustering. In this context, this study presents a novel hybrid approach, which is based on multiobjective genetic programming, aiming at the automatic design of novel hierarchical fusion operators for clustering ensembles. By this means, an initial set of partitions obtained via the application of different clustering techniques could be continuously refined through a population of hierarchies of fusion operators, which select and combine the original partitions, using different quality criteria as objective functions. To validate the new approach in terms of efficiency and effectiveness, we have implemented a prototype and conducted a comparative study including other clustering algorithms (three of which are of clustering ensembles and two are multiobjective in nature) over 10 different datasets. The experiments indicate that, in general, the idea of having a fusion hierarchy together with the correct selection of the data partitions can provide significant gains in terms of effectiveness and robustness.

Keywords: Data Clustering. Clustering Ensembles. Hierarchical Fusion. Multiobjective Evolutionary Algorithms, Genetic Programming.

SUMÁRIO

LISTA DE FIGURAS 11
LISTA DE TABELAS 12
1. INTRODUÇÃO 13
2. ALGORITMOS EVOLUTIVOS MULTIOBJETIVOS E PROGRAMAÇÃO GENÉTICA 17
2.1. Algoritmos Evolutivos Multiobjetivos 17
2.1.1. Otimização Multiobjetivo 18
2.1.2. Algoritmos Evolutivos 19
2.1.3. Algoritmo Evolutivo Multiobjetivo de Interesse 24
2.1.3.1. NSGA (Non-Dominated Sorting in Genetic Algorithms) 25
2.2. Programação Genética 27
2.2.1. População Inicial 30
2.2.2. Função de Aptidão 30
2.2.3. Métodos de Seleção 31
2.2.4. Operadores Genéticos 31
2.3. Considerações Finais 33
3. AGRUPAMENTO DE DADOS, FUSÃO DE PARTIÇÕES E TÉCNICAS DE VALIDAÇÃO 34
3.1. Agrupamento de Dados 34
3.1.1. Algoritmos Hierárquicos – Ligação Simples e Ligação Média 38
3.1.2. Algoritmo k-médias (KM) 40
3.1.3. Algoritmo Shared Nearest Neighbor (SNN) 41
3.2. Fusão de Partições 43
3.2.1. Geração de Partições Iniciais 44
3.2.2. Determinação da Função-Consenso 45
3.2.3. Funções-Consenso de Interesse 47

3.2.3.1. Métodos Propostos por Strehl e Ghosh (2002) 47
3.2.3.2. Hybrid Bipartite Graph Formulation (HBGF) 52
3.3. Validação de Agrupamentos 54
3.4. Algoritmos de Agrupamento de Dados Multiobjetivo de Interesse 57
3.4.1. MOCK (Multi-Objective Clustering with automatic K-determination) 57
3.4.2. MOCLE (Multi-Objective Clustering Ensemble) 59
3.5. Considerações Finais 60
4. MCHPF – Multiobjective Clustering with Hierarchical Partition Fusion 61
4.1. Abordagem Proposta 61
4.1.1. Representação das Soluções e Geração da População Inicial 63
4.1.2. Cálculo do Valor de Aptidão 65
4.1.3. Seleção 66
4.1.4. Operadores Genéticos 67
4.1.5. Elitismo e Critério de Parada 67
4.2. Configurações Alternativas do MCHPF 68
4.3. Considerações Finais 70
5. EXPERIMENTOS E RESULTADOS 71
5.1. Configuração dos Experimentos 71
5.2. Eficiência e Análise das Hierarquias Obtidas 76
5.3. Comparação do MCHPF com os Algoritmos Individuais 80
5.4. Comparação do MCHPF com as Técnicas de Combinação 83
5.5. Comparação entre as Configurações do MCHPF 84
5.6. Considerações Finais 86
6. CONCLUSÃO E TRABALHOS FUTUROS 88
BIBLIOGRAFIA 90
APÊNDICE I – ÍNDICE CR DAS DIFERENTES VERSÕES DE MCHPF 96
APÊNDICE II – CONJUNTO DE DADOS 99

LISTA DE FIGURAS

Figura 2.1 – Ilustração das técnicas de preservação da diversidade. 23
Figura 2.2 – Dois possíveis modos de se implementar o elitismo. 24
Figura 2.3 – Uma função representada na forma de árvore. 28
Figura 2.4 – Exemplo de cruzamento entre dois programas. 33
Figura 3.1 – Dendograma. 39
Figura 3.2 – Ilustração da matriz de similaridade produzida por CSPA. 50
Figura 3.3 – Ilustração da fusão de partições obtida via HGPA. 51
Figura 3.4 – Ilustração de uma fusão de partições por HBGF. 54
Figura 4.1 – Fluxograma do MCHPF. 62
Figura 4.2 – Indivíduos representando possíveis fusões de partições. 64
Figura 5.1 – Conjuntos de dados artificiais *ds4c2sc8* e *ds2c2sc13* (FACELI, 2006) 82

LISTA DE TABELAS

Tabela 3.1 – Exemplo de um conjunto de partições sendo representadas na forma de um hipergrafo 49
Tabela 4.1 – Configurações alternativas do MCHPF 69
Tabela 5.1 – Características dos conjuntos de dados 72
Tabela 5.2 – Valores de parâmetros e partições iniciais geradas 73
Tabela 5.3 – Tempo médio de execução do MCHPF 76
Tabela 5.4 – Média de CR dos indivíduos com diferentes níveis de hierarquia 77
Tabela 5.5 – Média de CR dos algoritmos individuais e do MCHPF 79
Tabela 5.6 – Média e desvio-padrão do índice CR obtidos pelas técnicas de combinação e por MCHPF 81
Tabela 5.7 – Médias de CR das cinco melhores configurações do MCHPF 85

1. INTRODUÇÃO

A tarefa de agrupamento de dados (*clustering*) consiste na distribuição de dados em grupos homogêneos, de maneira que os itens pertencentes a um mesmo grupo devem ser similares, enquanto aqueles que estão em grupos diferentes devem ser dissimilares (XU; WUNSCH, 2005; HANDL; KNOWLES, 2006). A grande vantagem da utilização de uma técnica que realize esse processo é que, ao se obter como resultado a partição dos dados, as características dos diversos grupos ficam mais explícitas, facilitando o entendimento dos dados originais e possibilitando o desenvolvimento de esquemas de classificação de novos dados (HAN; KAMBER, 2001).

Segundo Fasulo (1999), os métodos heurísticos voltados para a solução de problemas relacionados a agrupamentos de dados podem ser classificados em métodos de particionamento e métodos hierárquicos. Nos métodos de particionamento, cada objeto pertence a exatamente um grupo. Exemplos de algoritmos dessa classe são *k-means* e ISODATA. Já nos métodos hierárquicos, os grupos vão sendo formados gradativamente, gerando uma hierarquia de partições, normalmente representadas por uma estrutura em árvore. São exemplos de algoritmos hierárquicos *Shared Nearest Neighbor*, Hierárquico com Ligação Simples e Hierárquico com Ligação Média (XU; WUNSCH, 2005).

Todos os algoritmos de agrupamento de dados fazem uso de algum critério específico para gerar uma nova partição. Da variedade de critérios existentes, muitos são parcialmente complementares ou até mesmo conflitantes, de modo que podem favorecer diferentes tipos de solução (FACELI et al., 2008). Outra característica desses algoritmos é que eles podem encontrar estruturas em diferentes níveis de refinamento, dependendo dos ajustes de seus parâmetros (JAIN; DUBES, 1988).

Porém, apesar da extensa variedade de algoritmos de agrupamento existentes, não existe nenhum que, otimizando apenas um critério, seja capaz de revelar todos os tipos de estruturas (homogêneas e heterogêneas) que podem estar presentes no conjunto de dados. Além disso, não é possível saber, a priori, qual critério é mais apropriado para capturar as estruturas em uma massa de dados da qual se tem pouco conhecimento prévio (FACELI, 2006; FACELI et al., 2008).

Dessa forma, um avanço considerável no processo de agrupamento de dados vem sendo obtido por meio do desenvolvimento de métodos de fusão de partições, também conhecidos como técnicas de consenso ou de *clustering ensembles* (STREHL; GHOSH, 2002; FERN; BRODLEY, 2004; TOPCHY et al., 2005). A ideia geral da fusão consiste em criar múltiplas partições de um mesmo conjunto de dados e, então, combiná-las para formar uma partição final que represente um consenso entre as partições obtidas inicialmente. Assim, pode-se usar vários algoritmos de agrupamento, cada um gerando agrupamentos conforme seus critérios, e, em seguida, combiná-los para formar uma partição-consenso. Duas técnicas de fusão de partições bastante difundidas são as propostas por Strehl e Ghosh (2002) e por Fern e Brodley (2004).

A partição obtida a partir da combinação das várias partições iniciais deve ser consistente e de alguma forma concordar com essas partições, não ser sensível a pequenas variações nas estruturas individuais e estar em concordância com as informações externas sobre a estrutura dos dados, caso estejam disponíveis (FRED; JAIN, 2003). Além disso, Strehl e Ghosh (2002) explicam que, em seu algoritmo, a partição-consenso é elaborada sem que a função utilizada para criá-la acesse a base de dados original ou os algoritmos utilizados para criar as partições. Sendo assim, o algoritmo é útil mesmo quando a base de dados não está disponível, como, por exemplo, quando os dados em questão são sigilosos.

Contudo, um ponto negativo comum às diferentes abordagens de fusão de partições é que uma única resposta (partição) para o problema pode limitar a aquisição do conhecimento inerente ao conjunto de dados sob análise, já que esses dados podem esconder outras possíveis estruturas de partição igualmente relevantes. Por outro lado, como existem vários índices utilizados para avaliar a qualidade de um agrupamento, cada um favorecendo um tipo diferente de grupo, pode-se facilmente modelar a tarefa de agrupamento de dados como um problema típico de otimização multiobjetivo.

Dessa forma, é promissor o emprego de uma abordagem multiobjetiva, em que o alvo da busca não seja apenas uma única solução, mas um conjunto de boas soluções segundo a avaliação de diversos índices de qualidade (HANDL; KNOWLES, 2005; FACELI et al., 2008). A abordagem multiobjetivo visa encontrar uma aproximação do conjunto de soluções (partições) ótimas para o problema de agrupamento, segundo os objetivos utilizados. Tais soluções representam compromissos entre os objetivos, sendo que nenhuma delas pode ser avaliada como "melhor" do que as demais, considerando-se simultaneamente todos os índices de qualidade utilizados (objetivos).

Outro ponto importante é que, na fusão de agrupamentos tradicional, todos os agrupamentos gerados inicialmente são usados para construir o agrupamento-consenso. Com isso, a presença de partições iniciais de baixa qualidade pode influenciar negativamente o resultado da combinação. Assim, uma alternativa atraente seria a de se lançar mão de uma abordagem evolutiva que selecionasse gradativamente um subconjunto das partições iniciais mais significativas para realizar as combinações e que os resultados dessas combinações pudessem ser utilizados em novas fusões, formando assim uma hierarquia de fusões.

Neste trabalho, o principal foco está na utilização de técnicas de Programação Genética para se gerar estruturas hierárquicas de fusão de partições que produzam partições-consenso representando os melhores compromissos das várias medidas de qualidade de partições (funções de aptidão).

A Programação Genética é uma abordagem de computação evolutiva que simula a evolução de programas de computador baseando-se nos princípios darwinistas de Seleção Natural, recombinação genética (cruzamento), mutação, duplicação de genes e outros mecanismos de desenvolvimento biológico (KOZA, 1992). No contexto da fusão de partições, a ideia é simular a evolução de hierarquias de fusão na forma de árvores de fusores. Para isso, inicialmente geram-se várias partições com o uso de diferentes algoritmos de agrupamento de dados. Então, cria-se uma população inicial de hierarquias de fusão sobre essas partições, lançando-se mão de operadores de fusão de partições (*ensembles*) já disponíveis na literatura. Assim, cada indivíduo dessa população representa uma nova forma de fundir um subconjunto das partições iniciais.

Em seguida, essa população é evoluída utilizando como critérios de avaliação índices de qualidade das partições de dados, também disponíveis na literatura. Assim, ao final do processo obtém-se um conjunto de indivíduos, representados por hierarquias de fusões, cada um possivelmente contendo apenas um subconjunto das partições originais. Espera-se que as partições resultantes desses indivíduos sejam de melhor qualidade que as iniciais, segundo os critérios de qualidade preestabelecidos.

Este estudo está assim organizado: os Capítulos 2 e 3 fazem uma revisão dos temas necessários para a compreensão do assunto. Os Capítulos 4 e 5 apresentam e avaliam o método proposto. O Capítulo 6 traz a conclusão e uma proposta de trabalhos futuros. Mais especificamente:

- No Capítulo 2, há uma breve explanação sobre problemas de otimização multiobjetivo, indicando como os algoritmos evolutivos são apropriados para tratar essa classe de problemas. No contexto desse capítulo, a Programação Genética é apresentada em maiores detalhes.
- No Capítulo 3, são apresentados conceitos básicos e avançados para o entendimento das linhas de agrupamento de dados, fusão de partições e validação de agrupamentos, sendo expostos também dois métodos evolutivos de cunho multiobjetivo para fins de agrupamento de dados.
- O Capítulo 4 é dedicado à descrição do método proposto e os aspectos das diferentes instâncias desenvolvidas para ele.
- As configurações dos experimentos realizados, seus resultados e a análise desses resultados são apresentados no Capítulo 5.
- O Capítulo 6 contém as conclusões do estudo, juntamente com os principais resultados obtidos, as contribuições e as limitações da pesquisa, além da indicação de trabalhos futuros.

2. ALGORITMOS EVOLUTIVOS MULTIOBJETIVOS E PROGRAMAÇÃO GENÉTICA

A primeira seção deste capítulo aborda os algoritmos evolutivos multiobjetivos, incluindo conceitos básicos do problema de otimização multiobjetivo, aspectos dos algoritmos evolutivos e o que os torna adequados para tratar essa classe de problemas. Na mesma seção, também é apresentado o algoritmo evolutivo multiobjetivo de maior interesse para este estudo, o NSGA-II. Na segunda seção, discute-se em mais detalhes a Programação Genética, uma subclasse dos algoritmos evolutivos que opera com estruturas não-lineares de representação das soluções.

2.1. Algoritmos Evolutivos Multiobjetivos

A maioria dos problemas do mundo real envolve múltiplas medidas de desempenho (ou objetivos), que devem ser melhorados (ou atingidos) simultaneamente. Em alguns casos, quando essas medidas são não conflitantes, é possível a otimização de cada uma separado, sem que a otimização de uma interfira na otimização da outra. Contudo, essa situação raramente acontece. Frequentemente, a otimização de um objetivo interfere na dos outros, tornando muitas vezes a solução ótima para um inaceitável para outro. Dessa forma, uma solução adequada para problemas que envolvam objetivos conflitantes deve ser vista como uma relação de compromisso, oferecendo um desempenho aceitável em todos os objetivos, mas não garantindo a otimalidade para cada objetivo em particular (FONSECA; FLEMING, 1995).

Sendo assim, algumas técnicas de programação matemática foram desenvolvidas para trabalhar com múltiplos objetivos. A mais simples utiliza todos os objetivos combinando-os em uma única função composta. A determinação dessa função é

possível empregando-se métodos de soma ponderada, por exemplo. O desafio dessa estratégia está na dificuldade da correta seleção dos pesos que caracterizam o problema, algo que pode ser muito complexo. Além disso, essa otimização retorna uma única solução, em vez de um conjunto de soluções de compromisso. Por essas razões, frequentemente, prefere-se um conjunto de boas soluções considerando todos os múltiplos objetivos ao mesmo tempo (KONAK et al., 2006).

Por outro lado, os algoritmos evolutivos, classe de métodos heurísticos de otimização estocástica, apresentam boa capacidade de trabalhar com problemas de natureza multiobjetiva. Isso porque viabilizam que seja encontrado, em uma única execução, um conjunto (população) de possíveis soluções de compromisso, contrastando com as técnicas tradicionais de programação matemática que não possuem essa capacidade (COELLO COELLO, 1999). Ademais, para lidar com problemas multiobjetivos, os algoritmos evolutivos podem ser customizados pelo uso de mais de uma função-objetivo especializada e pela introdução de métodos que promovem a diversificação das soluções (KONAK et al., 2006).

Na Subseção 2.1.1, são introduzidos conceitos gerais relacionados à otimização multiobjetiva. Na Subseção 2.1.2, são discutidos os principais aspectos dos algoritmos evolutivos, enfatizando alguns de seus elementos interessantes para tratar de problemas multiobjetivos. Na Subseção 2.1.3, é apresentado o algoritmo evolutivo multiobjetivo de maior interesse para este estudo, o NSCGA-II (DEB et al., 2002).

2.1.1. Otimização Multiobjetivo

O problema de otimização multiobjetivo pode ser assim definido: dado um vetor de variáveis de decisão $y = \{y_1, y_2, ..., y_s\}$, de dimensão *s*, no espaço de busca *Y*, deve-se encontrar um vetor-solução y^* que otimize um conjunto de *m* funções-objetivo $z(y^*) = \{z_1(y^*), z_2(y^*), ..., z_m(y^*)\}$. O espaço de soluções *Y* geralmente é modelado por uma série de restrições limitadas pelas variáveis de decisão, como, por exemplo, $g_j(y^*) = b_j$ para $j = 1, ..., r$. Sem perda de generalidade, todos os objetivos neste estudo têm igual importância e são considerados como de minimização, já que objetivos desse tipo podem ser facilmente convertidos para o tipo maximização, e vice-versa, mediante multiplicação da função-objetivo por -1 (KONAK et al., 2006).

Como já mencionado, frequentemente, os problemas reais têm múltiplos e conflitantes objetivos que devem ser otimizados simultaneamente. Porém, como explicado por Konak et al. (2006), é quase impossível se encontrar uma única solução que atenda perfeitamente a todos os objetivos de um problema, de modo que uma alternativa razoável é a de se investigar um conjunto de soluções que satisfaça todos os objetivos em um nível aceitável, sem que cada uma seja "dominada" (de acordo com a definição dada a seguir) por qualquer outra.

Considerando funções-objetivo que devam ser minimizadas, diz-se que uma solução y_1 domina outra y_2 ($y_1 \succ y_2$), se e somente se $z_i(y_1) \leq z_i(y_2)$ para $i = 1, ..., m$ e $z_j(y_1) < z_j(y_2)$ para pelo menos uma função-objetivo *j*; ou seja, para que y_1 domine y_2 ela deve ser melhor que y_2 em pelo menos uma função-objetivo, e não pode ser pior em nenhuma outra.

Uma solução é um ótimo de Pareto se ela não é dominada por nenhuma outra no espaço de soluções. Portanto, um ótimo de Pareto não pode ser melhorado com respeito a um objetivo sem que haja uma piora de qualquer outro objetivo. O conjunto de todas as possíveis soluções não-dominadas em *Y* (todos os ótimos de Pareto) é referenciado como conjunto ótimo de Pareto. Os valores das funções-objetivo para as soluções do conjunto ótimo de Pareto compõem o fronte de Pareto (ou fronteira de Pareto). Para muitos problemas, a cardinalidade do conjunto ótimo de Pareto é muito grande, podendo tender para o infinito (KONAK et al., 2006).

O ideal para os algoritmos de otimização multiobjetivo seria encontrar o conjunto completo de soluções do conjunto ótimo de Pareto. Entretanto, como explicam Konak et al. (2006), nem sempre isso é possível, devido ao tamanho desse conjunto. Então, na prática, o que muitos métodos de otimização multiobjetivo fazem é investigar um conjunto de soluções que representa o conjunto ótimo de Pareto tanto quanto possível, ou seja, tentam encontrar uma aproximação do conjunto ótimo de Pareto.

2.1.2. Algoritmos Evolutivos

Os algoritmos evolutivos simulam o processo da evolução natural, baseando-se na teoria evolucionista neo-darwinista de reprodução e sobrevivência dos mais fortes. Assim, tais algoritmos trabalham com uma população de indivíduos, cada um dos quais

representando uma possível solução para um dado problema e tendo um valor de aptidão associado, mediante operadores computacionais que simulam os processos de seleção, recombinação cromossomial e mutação gênica.

Um algoritmo evolutivo pode ser caracterizado por três aspectos (ZITZLER et al., 2004):

1. um conjunto de possíveis soluções é mantido;
2. um processo de reprodução seletiva é realizado nesse conjunto; e
3. as soluções podem ser combinadas para gerar novas soluções.

Antes de aplicar um algoritmo evolutivo, propriamente dito, ao problema, faz-se necessário projetar uma representação apropriada da solução (indivíduo) e em seguida definir um mapeamento (codificação) entre os pontos do espaço de busca do problema e as instâncias da representação. Por exemplo, aplicando algoritmos genéticos (uma subclasse dos algoritmos evolutivos) a um certo problema em que o objetivo é o de se encontrar o ótimo global de uma função multidimensional desconhecida, uma possível representação da solução seria uma cadeia de caracteres ou números, sendo que cada componente específico da estrutura estaria associado a uma das variáveis do problema (KOZA, 1995). Como um valor de aptidão está associado a cada indivíduo, no caso de um problema de otimização multidimensional, esse valor seria o resultado da função-objetivo, utilizando como entradas as instâncias da estrutura.

Nos algoritmos evolutivos, geralmente, o conjunto de indivíduos que representam a população inicial são gerados aleatoriamente; ou seja, cada indivíduo tem suas variáveis iniciadas com valores aleatórios. Então, essa população é avaliada para atribuir valores de aptidão a todos os seus membros. Parte dessa população, selecionada com base na sua aptidão, criará a nova geração, sendo manipulada pelos operadores de cruzamento e mutação (KONAK et al., 2006).

No operador de cruzamento, alguns indivíduos, chamados de pais, são combinados para formar um número predefinido de novos indivíduos (prole ou *offspring*), pela combinação de partes dos pais (ZITZLER et al., 2004). Utilizando como exemplo uma cadeia de caracteres para a representação dos indivíduos, a combinação dos pais se daria via a permutação de partes dessa cadeia, de tal modo que a prole

resultante tenha o mesmo formato dos pais. Os pais são selecionados com base na sua aptidão, esperando que os novos indivíduos combinem as características dos pais, tornando-os ainda mais aptos. Esse tipo de operador genético é de fundamental importância para a classe dos algoritmos genéticos.

Já o operador de mutação introduz mudanças aleatórias em localizações específicas da estrutura. Seu papel é diferenciado de acordo com os diferentes dialetos de algoritmos evolutivos. Por exemplo, em uma implementação típica de algoritmos genéticos, a taxa de mutação, que é a probabilidade de um indivíduo sofrer alteração, é muito baixa e depende do tamanho da representação. Assim, geralmente, os indivíduos resultantes de uma operação de mutação em algoritmos genéticos não são muito diferentes dos originais (KONAK et al., 2006). O mesmo não ocorre com as estratégias evolutivas e a programação evolucionária, que são classes de algoritmos evolutivos fortemente baseadas nesse operador.

O procedimento geral de um algoritmo evolutivo é dado pela sequência de passos a seguir (BÄCK et al., 1997; FACELI, 2006):

1. Inicia-se o número da geração atual (*t*): t = 0;
2. Inicia-se a população P_t com um número de indivíduos n^P predefinidos;
3. Atribui-se o valor de aptidão a cada indivíduo da população P_t, usando a medida de aptidão adotada;
4. $t = t + 1$;
5. Seleciona-se P_t a partir de P_{t-1};
6. Manipula-se P_t através dos operadores genéticos; e
7. Se o critério de parada não tiver sido satisfeito, retornar ao passo 3.

Para problemas com um único objetivo, o conjunto de soluções da população final apresenta pelo menos um indivíduo que é mais apto que todos os outros, segundo a função-objetivo utilizada. Porém, nas situações em que há vários objetivos, não deve existir apenas uma solução, mas, sim, um conjunto de soluções de compromisso de qualidade equivalente.

Segundo Konak et al. (2006), os algoritmos evolutivos possuem algumas propriedades que os tornam atraentes para otimização multiobjetiva:

- apresentam um conjunto de possíveis soluções;
- os algoritmos que trabalham com apenas um objetivo podem ser facilmente modificados para aceitar múltiplos objetivos e encontrar um conjunto de soluções não-dominadas;
- a habilidade de busca em diferentes regiões do espaço de soluções cria a possibilidade de encontrar um conjunto bastante diverso de soluções; e
- o operador de cruzamento pode explorar a estrutura de boas soluções em relação aos diferentes objetivos para criar novas soluções não-dominadas em regiões ainda não exploradas da fronteira de Pareto.

Os algoritmos evolutivos para otimização multiobjetiva têm como principais objetivos conduzir a busca na direção do ótimo de Pareto e manter um conjunto diverso de soluções não-dominadas. O primeiro objetivo relaciona-se principalmente aos procedimentos que envolvem o modo como são atribuídos valores escalares de aptidão em um ambiente com múltiplos objetivos. O segundo diz respeito ao processo de seleção, já que se deseja evitar que a população contenha essencialmente soluções idênticas. Além disso, um terceiro fator que envolve os dois objetivos é o elitismo, que aborda principalmente a questão de como não perder boas soluções ao longo das várias gerações (ZITZLER, 2004).

Dessa forma, dentre os aspectos a serem considerados no projeto de algoritmos evolutivos multiobjetivos, destacam-se o cálculo do valor de aptidão, o processo de seleção, a diversidade populacional e o elitismo, discutidos a seguir (ZITZLER, 2004):

Cálculo do valor de aptidão e seleção

Em contraste com a otimização de um único objetivo, em que as funções-objetivo e de aptidão geralmente se confundem, em otimizações multiobjetivas, tanto o cálculo do valor de aptidão quanto a seleção devem considerar os vários critérios a serem otimizados. As três principais estratégias existentes para o cálculo da aptidão e para a seleção se baseiam em agregação, alternância de objetivos e ordenação baseada em não-dominação, explicitadas a seguir:

- Agregação: os objetivos são agregados em uma única função-objetivo parametrizada, sendo que os parâmetros variam sistematicamente durante a execução, a fim de encontrar um conjunto de soluções não-dominadas, em vez de apenas uma.
- Alternância de objetivos: a cada seleção realizada, um objetivo diferente é utilizado para eleger o melhor indivíduo.
- Ordenação baseada em não-dominação: a população é ordenada de acordo com uma regra de dominância, e o cálculo da aptidão e a seleção são realizados com base nessa ordenação.

Diversidade populacional

A maioria dos algoritmos evolutivos multiobjetivos tenta manter a diversidade da população, incorporando ao processo de seleção informações sobre a densidade do espaço de soluções. Ou seja, a chance de um indivíduo ser selecionado decresce caso a região do espaço de soluções em que ele se encontre tenha alta densidade. Os métodos usados podem sem classificados de acordo com as técnicas estatísticas de estimação de densidade (vide a Figura 2.1, extraída de (ZITZLER, 2004)):

- Método *kernel*: define a densidade estimada de uma solução por meio de uma função *kernel*, que utiliza como argumento a distância entre essa solução e todas as demais.
- Vizinhos mais próximos: para estimar a densidade na região, utiliza a distância entre os vizinhos mais próximos de uma solução.
- Histograma: usa um *hypergrid* para definir a vizinhança dentro do espaço de soluções. A densidade ao redor de um indivíduo é estimada pela quantidade de indivíduos no mesmo *box* do *grid*.

Figura 2.1 – Ilustração das técnicas de preservação da diversidade.

Além desses métodos, o algoritmo *Non-Dominated Sorting in Genetic Algorithms* (NSGA-II) (DEB et al., 2002) utiliza, para calcular a densidade local, o método *crowding distance*, descrito na Subseção 2.1.3.1.

Elitismo

O elitismo trata das consequências da perda de boas soluções durante o processo de otimização, problema causado pela aleatoriedade inerente aos algoritmos evolutivos. Um modo de tratar esse problema consiste em combinar a população corrente com a sua prole (*offspring*) e aplicar uma seleção determinística. Outro modo é manter uma população secundária, chamada de "arquivo", para onde soluções promissoras são copiadas a cada geração. O arquivo pode ser usado como uma reserva externa, separada do mecanismo de otimização, ou pode ser integrado ao algoritmo. Esses dois métodos estão ilustrados na Figura 2.2 (ZITZLER, 2004).

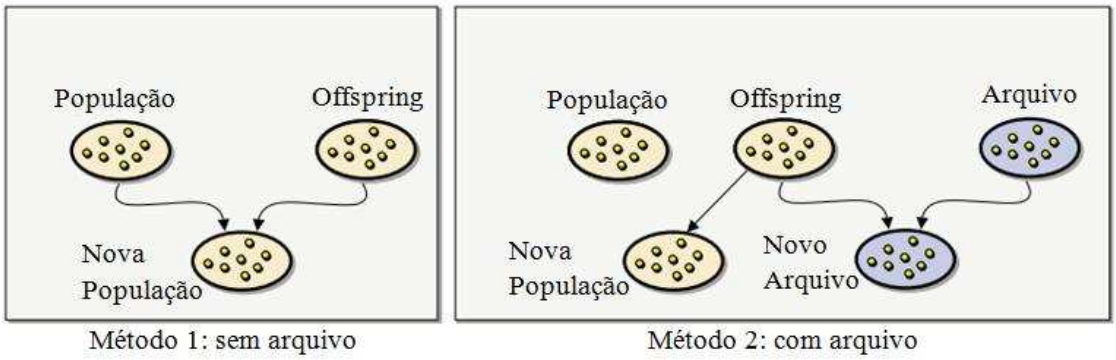


Figura 2.2 – Dois possíveis modos de se implementar o elitismo.

2.1.3. Algoritmo Evolutivo Multiobjetivo de Interesse

O algoritmo evolutivo multiobjetivo de maior interesse para este trabalho é NSGA-II (*Non-Dominated Sorting in Genetic Algorithms*) (DEB et al., 2002), pois grande parte de seus conceitos é empregado na implementação da abordagem proposta. A escolha desse algoritmo se deu por sua popularidade e por apresentar bom desempenho em problemas onde o número de objetivos não é elevado (ZITZLER et al., 2001), que é o caso deste trabalho.

2.1.3.1. NSGA (Non-Dominated Sorting in Genetic Algorithms)

NSGA-II é uma versão melhorada de seu predecessor, NSGA (SRINIVAS; DEB, 1994), um algoritmo genético multiobjetivo baseado em não-dominação. NSGA apresenta como pontos negativos a alta complexidade de implementação, a falta de elitismo e a necessidade de especificação de parâmetro para o método *fitness sharing*, usado para preservar a diversidade. Já NSGA-II possui um método mais rápido para ordenação das soluções com base na não-dominação, e emprega o conceito de *crowding distance* para manter a diversidade da população e compor um operador de comparação (*crowded comparison*).

Para se realizar a ordenação dos indivíduos com base na não-dominação, a população é distribuída em *F* frontes. Os indivíduos que não são dominados por nenhum outro são colocados no primeiro fronte (F_1). O segundo fronte (F_2) é composto pelos indivíduos que são dominados apenas por aqueles que se encontram no F_1. Já o fronte F_3 contém os indivíduos dominados apenas pelos que se encontram em F_1 e F_2, e assim por diante. Com isso cada indivíduo terá um *rank*, que corresponde ao nível de não-dominação em que ele se encontra, começando por 1 (um), correspondendo ao primeiro nível, e indo até a quantidade de frontes criados.

Crowding distance é uma estimativa de densidade de soluções em torno de uma solução em particular. Antes de se calcular essa medida, faz-se necessário normalizar os objetivos. Dado um fronte F_i, o procedimento para calcular a *crowding distance* de suas soluções contém as seguintes etapas (DEB et al., 2002):

1. Para cada função-objetivo z_j:

    - ordenam-se as soluções de F_i em ordem crescente de z_j;
    - são encontradas as soluções-limite p_1, com menor valor de z_j (z_j^{min}), e p_l, com maior valor de z_j (z_j^{max});
    - atribui-se a *crowding distance* em relação a z_j dessas soluções como sendo ∞ (infinito); e
    - para as demais soluções p_w, o valor da *crowding distance* é calculado de acordo com a Equação 2.1:

$$crd(z_j, p_w) = \frac{z_j(p_{w+1}) - z_j(p_{w-1})}{z_j^{max} - z_j^{min}}.$$

Equação 2.1

2. A *crowding distance* de uma solução *p* é dada pela soma das *crowding distances* em relação aos *m* objetivos: $cdr(p) = \sum_{j=1}^{m} cdr(z_j, p)$.

O operador de comparação (*crowded comparison*), $\prec_n$, usado em vários estágios do processo de seleção, compara dois indivíduos p_i e p_j. Para isso, considera dois atributos dos indivíduos, o *rank* de não-dominação e a *crowding distance*, e é definido como $p_i \prec_n p_2$ se $rank(p_i) < rank(p_j)$ ou $(rank(p_i) = rank(p_2)$ e $crd(p_i) > crd(p_2))$. Isto é, entre duas soluções com diferentes *ranks* de dominação, a selecionada é aquela com o menor *rank*, ou seja, a solução que domina a outra. Porém, se as soluções encontram-se no mesmo fronte, a selecionada é aquela que se encontra numa região com menor densidade, ou seja, menor *crowding distance*.

Espelhando-se no procedimento adotado para ordenação com base na não-dominação, na *crowding distance* e no operador de comparação descritos acima, o NSGA-II funciona da seguinte maneira (DEB et al., 2002):

1. são gerados aleatoriamente n^P indivíduos para compor a população inicial P_0;
2. distribui-se P_0 em frontes com base na não-dominação;
3. atribui-se o valor de aptidão de cada indivíduo da população. O valor de aptidão corresponde ao *rank* do fronte ao qual pertence o indivíduo;
4. usa-se seleção por torneio binário, recombinação e mutação para criar uma população-prole Q_0, de tamanho n^P. Para a seleção, nessa primeira geração, leva-se em conta apenas o valor de aptidão;
5. inicia-se o número da geração: $t = 0$;
6. forma-se uma população combinada $R_t = P_t \cup Q_t$, de tamanho $2n^P$;
7. ordena-se R_t em frontes, de acordo com a não-dominação;
8. selecionam-se os indivíduos para a nova geração, da seguinte maneira:

- enquanto $|P_{t+1}| + |F_i| \leq n^P$ (ou seja, o número de indivíduos já adicionados à nova geração, mais o tamanho do i-ésimo fronte, não exceder o tamanho que a população deve ter):
  - calcula-se a *crowding distance* para as soluções de F_i;
  - incluem-se os membros de F_i na população P_{t+1}, $(P_{t+1} = P_{t+1} \cup F_i)$;
  - $i = i + 1$;
- ordena-se F_i (último fronte, que, juntamente com os indivíduos que já estão na nova geração, excede o tamanho máximo da população), de acordo com $\prec_n$, em ordem crescente;
- selecionam-se os primeiros $n^P - |P_{t+1}|$ elementos de F_i, sendo incluídos na população P_{t+1};

9. usa-se seleção por torneio binário, recombinação e mutação para criar uma população-prole Q_{t+1}, de tamanho n^P. Para criar cada indivíduo da população Q_{t+1} (n^P serão criados):

   - selecionam-se dois indivíduos da população P_{t+1}, utilizando torneio binário e o conceito de *crowding distance*;
   - aplicam-se os operadores de recombinação e mutação;

10. $t = t + 1$; e

11. se a quantidade de gerações informada pelo usuário não tiver sido alcançada, retorna-se ao passo 6.

Os operadores de recombinação e mutação podem ser quaisquer operadores apropriados que satisfaçam as necessidades do problema.

2.2. Programação Genética

Programação Genética (PG) (KOZA, 1992) é um dos ramos da Computação Evolutiva, tendo sido concebida como uma tentativa de lidar com uma das questões centrais em ciência da computação: Como os computadores podem aprender a resolver problemas sem serem expressamente programados? (KOZA, 1995).

Todos os programas de computador podem ser vistos como uma sequência de funções (operações) aplicadas a argumentos (valores). Com base nessa lógica, primeiramente os compiladores traduzem um programa para uma árvore sintática (*parse tree*), para em seguida convertê-la para linguagem de máquina. Assim, qualquer programa de computador pode ser graficamente estruturado como uma árvore (KOZA, 1995).

Programação Genética é uma extensão dos algoritmos genéticos, em que cada indivíduo da população é um programa de computador sendo representado na forma de árvore sintática. O âmbito de busca da PG é o espaço de todos os possíveis programas de computador, composto pelas funções e terminais apropriadamente criados para o domínio do problema. As funções podem ser operações aritméticas, rotinas de programação, funções matemáticas, funções lógicas ou funções específicas do domínio do problema (KOZA, 1995). A Figura 2.3 ilustra a função $\left(2.2 - \left(\frac{X}{11}\right)\right) + (7 * \cos(Y))$ sendo representada na forma de uma árvore sintática.

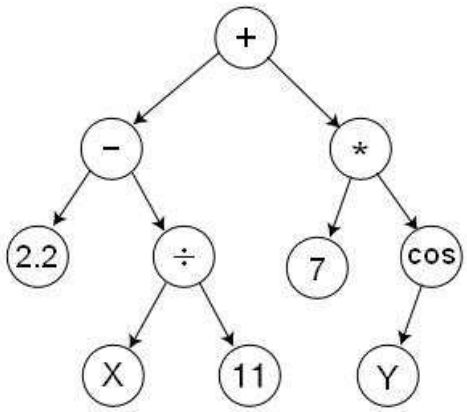


Figura 2.3 – Uma função representada na forma de árvore.

Para se aplicar PG a um problema, faz-se necessário implementar cinco passos preparatórios, de modo a determinar (KOZA, 1995; WILLIS et al., 1997):

1. o conjunto de terminais *T*: as entradas do programa que representa cada indivíduo na população;

2. o conjunto das funções *F*: funções utilizadas, juntamente com os terminais, para construir as possíveis soluções para um dado problema;

3. a forma de calcular a aptidão do indivíduo: a aptidão é um valor numérico atribuído a cada membro da população, e serve para medir o grau de desempenho do indivíduo em relação ao problema que está sendo resolvido;

4. os parâmetros para controlar a execução: por exemplo, o tamanho da população e a taxa de mutação; e

5. o critério de parada: geralmente definido como a quantidade de gerações ou uma taxa de tolerância do erro para o valor de aptidão.

Deve-se observar que os três primeiros passos preparatórios determinam o espaço de busca do algoritmo, enquanto os outros dois afetam a qualidade e a velocidade da busca (WILLIS et al., 1997).

Um fato importante é que todos os elementos do conjunto de funções devem aceitar, como um de seus argumentos, quaisquer valores e tipos de dados que porventura possam constar no conjunto de terminais ou que possam ser retornados por qualquer função do conjunto de funções (KOZA, 1992; 1995). Isto é, o conjunto de funções e o conjunto de terminais devem possuir a propriedade fechamento (*closure*).

O algoritmo de PG é simples, podendo ser implementado por meio dos seguintes passos:

1. cria-se aleatoriamente uma população de indivíduos (programas);

2. executam-se os seguintes passos até que o critério de parada seja satisfeito (cada execução desse laço significa a criação de uma nova geração):

    - avalia-se cada indivíduo com relação à função de aptidão estabelecida, atribuindo-lhe o valor correspondente;
    - cria-se uma nova população mediante a aplicação dos operadores genéticos (reprodução, cruzamento e mutação) aos indivíduos da população selecionados com probabilidade baseada no valor de aptidão; e

3. retorna-se com o melhor programa encontrado, ou, no caso de problemas multiobjetivos, com um conjunto deles.

2.2.1. População Inicial

A população inicial é constituída por um conjunto de estruturas, criadas aleatoriamente e compostas por funções e terminais, representando uma "busca cega" inicial sobre o espaço de soluções do problema.

Koza (1992) especificou três técnicas de iniciação de uma população em PG:

- *Grow*: cria árvores de profundidade variável, possivelmente bastante desbalanceadas. O nós são escolhidos de forma aleatória, entre o conjunto de funções e o conjunto de terminais. Uma árvore está concluída quando atinge um tamanho máximo predefinido ou quando todos os nós-folha forem terminais.
- *Full*: cria árvores totalmente balanceadas. Os nós intermediários são escolhidos aleatoriamente do conjunto de funções até que a árvore atinja o tamanho predefinido, quando então são escolhidos os terminais para os nós-folha; e
- *Ramped-half-and-half*: o método aumenta significativamente a diversidade na população inicial, já que são criados indivíduos para cada nível de profundidade entre "1" (um) e a profundidade máxima preestabelecida. Além disso, metade da população é criada por meio do método *grow*, enquanto a criação da outra metade se dá por meio do método *full.*

2.2.2. Função de Aptidão

Por aptidão entende-se a capacidade de um indivíduo para resolver determinado problema. Esse fator é bastante significativo na evolução da população, pois serve de parâmetro para classificar os indivíduos e, assim, determinar aqueles que devem passar para a próxima geração, os que irão gerar parte dessa nova geração e os que serão descartados.

Koza (1992) explica que a aptidão pode ser medida de diferentes formas, tanto explícita como implicitamente. Entretanto, a abordagem mais comum consiste em definir uma medida explícita para cada indivíduo da população, atribuindo-lhe um valor escalar de aptidão, calculado por meio de um processo bem definido de avaliação.

Na Subseção 2.1.2 foram apresentados métodos para o cálculo do valor de aptidão, comuns a todos os algoritmos evolutivos, incluindo a Programação Genética.

2.2.3. Métodos de Seleção

O método de seleção tem por objetivo escolher os programas que deverão sofrer a ação dos operadores genéticos e compor a nova geração. Para isso, ele geralmente opera com base no valor de aptidão.

Atualmente, os métodos de seleção mais utilizados são (BLICKLE; THIELE, 1995; RODRIGUES, 2002):

- Seleção Proporcional: usa a aptidão normalizada, disposta em uma "roleta", sendo que cada indivíduo possui uma porção proporcional ao seu valor de aptidão normalizado. Em seguida, um número aleatório, entre 0 (zero) e 1 (um), é produzido para simular a posição em que a "roleta" parou, a qual indicará o indivíduo selecionado.
- *Stochastic Universal Sampling* (SUS): semelhante ao método de seleção proporcional; porém, todos os indivíduos ocupam fração igual, e a roleta pode apontar para mais de um elemento ao mesmo tempo, representando a quantidade de elementos a serem selecionados.
- Seleção por Truncamento: a seleção é realizada aleatoriamente entre os melhores indivíduos do conjunto. O tamanho desse conjunto é passado como parâmetro para o método.
- Seleção por Torneio: alguns indivíduos são escolhidos aleatoriamente, selecionando-se em seguida aquele que detiver o maior valor de aptidão.
- Torneio Léxico: semelhante à seleção por torneio. A principal diferença é que, se mais de um indivíduo detém o maior valor de aptidão, um novo teste é realizado entre os melhores, podendo basear-se no tamanho da árvore (LUKE; PANAIT, 2002).

2.2.4. Operadores Genéticos

Os indivíduos da nova população são formados por meio de três operadores principais: reprodução, cruzamento e mutação. Tão logo a nova população esteja completa, a antiga é descartada (KOZA, 1992).

Reprodução

Essa é uma operação muito importante no processo de desenvolvimento genético, por garantir a sobrevivência dos indivíduos mais aptos. Trata-se de uma operação assexuada, pois trabalha com apenas um indivíduo. Compreende os dois seguintes passos:

1. um indivíduo é selecionado da população de acordo com algum método baseado no valor de aptidão; e

2. é copiado, sem nenhuma alteração, para a nova população.

Cruzamento

Conhecido como uma operação de recombinação sexual, opera em um par de indivíduos, para produzir um par de filhos, formados por partes de ambos os pais. Funciona da seguinte maneira:

1. são selecionados dois programas;

2. um ponto de cruzamento é escolhido em cada programa-pai, utilizando uma distribuição probabilística uniforme; e

3. as subárvores abaixo desses pontos são trocadas.

A Figura 2.4 ilustra o processo de cruzamento entre dois programas $((2 * (x + x)) + 1)$ e $(((x + 1) * x) - 2)$, sendo que, nesta figura, os nós intercambiados são indicados com aro em negrito. As subárvores são trocadas, gerando $((x + 1) + 1)$ e $(2 * ((x + x) * x) - 2)$ como filhos (RODRIGUES, 2002).

Mutação

Como discutido anteriormente, a mutação é a operação genética que pretende introduzir alterações aleatórias na estrutura da população. No contexto da PG, pode ser descrita por meio dos seguintes passos:

1. um indivíduo é selecionado;

2. um ponto de mutação é escolhido aleatoriamente, podendo ser interno (função) ou externo (terminal); e

3. o ponto de mutação é substituído por uma nova subárvore, gerada de forma aleatória.

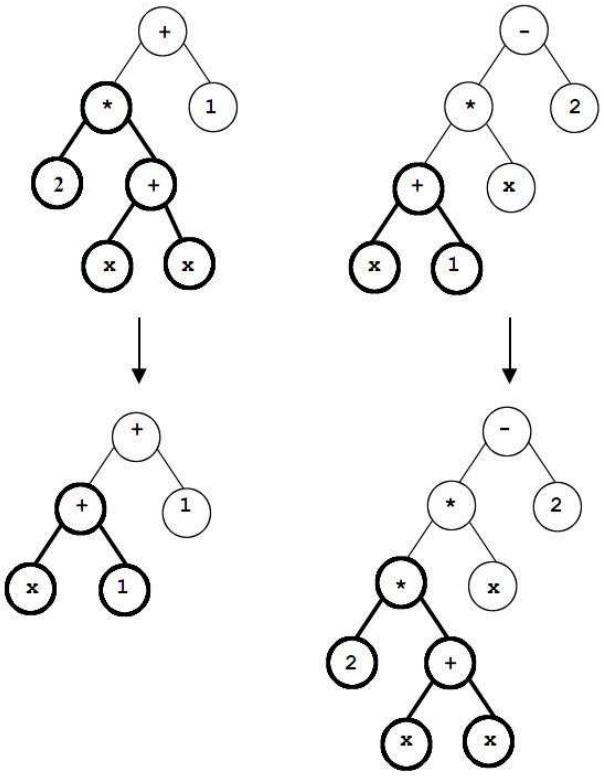


Figura 2.4 – Exemplo de cruzamento entre dois programas.

2.3. Considerações Finais

Grande parte dos problemas do mundo real envolve o controle de várias medidas de qualidade que devem ser melhoradas simultaneamente. Trata-se dos problemas multiobjetivos. Neste capítulo, foram apresentados os algoritmos evolutivos, que representam uma classe de métodos de otimização bastante adequados para tratar essa categoria de problemas. Nesse contexto, foram descritos NSGA-II, um algoritmo evolutivo multiobjetivo bastante popular, e a Programação Genética, uma classe de algoritmos evolutivos que opera com soluções representadas em árvores, sendo que ambos serviram de base para o desenvolvimento da abordagem proposta neste estudo.

No próximo capítulo, são apresentados conceitos básicos e avançados sobre agrupamento de dados, fusão de partições e técnicas de validação de agrupamentos.

3. AGRUPAMENTO DE DADOS, FUSÃO DE PARTIÇÕES E TÉCNICAS DE VALIDAÇÃO

Na primeira parte deste capítulo, são apresentados conceitos relativos à tarefa de agrupamento de dados bem como os algoritmos básicos de agrupamento utilizados no desenvolvimento desta dissertação. Na segunda seção, a abordagem de fusão de partições é introduzida, dando-se ênfase aos operadores de fusão de maior interesse para este trabalho. Já a terceira seção apresenta técnicas de validação de agrupamentos de dados, ao passo que a quarta apresenta duas abordagens de agrupamento de dados de cunho evolutivo multiobjetivo, sendo que uma delas também envolvendo noções de fusão de partições.

3.1. Agrupamento de Dados

Em geral, o objetivo dos algoritmos de agrupamento de dados é particionar uma coleção de objetos em diferentes grupos, sendo que os itens contidos em um mesmo grupo deverão ser similares, enquanto aqueles que estão em grupos diferentes deverão ser dissimilares (XU; WUNSCH, 2005; HANDL; KNOWLES, 2006).

A formação de um agrupamento baseia-se em um princípio indutivo, também conhecido como critério de agrupamento, segundo o qual é selecionado o modelo (ou estrutura) que melhor se ajuste a um determinado conjunto de dados para representar os seus grupos (ESTIVILL-CASTRO, 2002). Ou seja, o critério de agrupamento expressa o objetivo intrínseco que está por trás do processo de agrupamento a ser realizado.

Hruschka e Ebecken (2003) idealizaram a seguinte definição formal do problema de agrupamento de dados. Seja um conjunto de *n* objetos $X = \{X_1, X_2, ..., X_n\}$ a serem agrupados, onde cada $X_i \in \mathbb{R}^d$ é um vetor de *d* medidas

reais que dimensionam as características do objeto. Os objetos devem ser agrupados em conjuntos disjuntos $C = \{C_1, C_2, ..., C_k\}$, onde *k* é o número de grupos, de forma que tenhamos as seguintes condições respeitadas:

1. $C_1 \cup C_2 \cup ... \cup C_k = X$;
2. $C_i \neq \emptyset, \forall\, i, 1 \leq i \leq k$; e
3. $C_i \cap C_j = \emptyset, \forall\, i \neq j, 1 \leq i \leq k\, e\, 1 \leq j \leq k$.

Obedecendo a essas condições, verifica-se que todos os objetos devem pertencer a algum grupo. Além disso, um objeto não pode pertencer a mais de um grupo, ou seja, os conjuntos são disjuntos e cada grupo deve ter pelo menos um elemento.

O valor de *k* pode ser conhecido ou não. Caso o valor de *k* seja fornecido como parâmetro de entrada para o algoritmo, o problema é referenciado na literatura como "problema de *k-clustering*", do verbo "*to cluster*" ou agrupar (FASULO, 1999). Caso contrário, ou seja, se o *k* for desconhecido, o problema é referenciado como "problema de agrupamento automático", e a obtenção do valor de *k* faz parte do processo de solução do problema, conforme explicado por Doval et al. (1999).

De acordo com Jain et al. (1999), a utilização de qualquer técnica de agrupamento de dados requer a implementação de cinco passos, a saber:

1. definição da forma de representação dos dados a serem agrupados;
2. definição de uma forma adequada para medir a proximidade (similaridade) entre os dados. A medida de proximidade mais comum para objetos cujos atributos são todos contínuos é a distância euclidiana, dada pela Equação 3.1;

$$d(x_i, x_y) = \sqrt{\sum_{l=1}^{d} (x_{il} - x_{jl})^2}$$

Equação 3.1

3. definir uma técnica de agrupamento e aplicá-la aos dados, para construção dos grupos;

4. avaliar os grupos construídos. O que se procura neste passo é, basicamente, avaliar a partição final em relação aos objetivos esperados e verificar o quão bom foi o agrupamento; e

5. interpretar os resultados obtidos. Refere-se ao processo de examinar cada grupo com relação a seus objetos, para rotulá-los, descrevendo a natureza do grupo.

O método ideal de agrupamento de dados deveria atender aos seguintes requisitos (NG; HAN, 1994; ESTER et al., 1996; AGRAWAL et al., 1998; HAN; KAMBER, 2001):

- descobrir grupos com forma arbitrária: considerando o espaço euclidiano, a forma de cada grupo pode ser esférica, linear, alongada, elíptica, etc.;

- identificar grupos de tamanhos variados: alguns métodos tendem a formar partições com tamanhos homogêneos, muito embora os grupos sejam, na verdade, heterogêneos;

- aceitar os diversos tipos de dados possíveis e a combinação desses tipos;

- ser insensível à ordem de apresentação dos objetos. Quando um conjunto de objetos é apresentado com diferentes ordenamentos, o método deve fornecer os mesmos resultados;

- trabalhar com objetos de qualquer dimensão (i.e., com qualquer número de atributos);

- trabalhar com qualquer quantidade de dados: uma base de dados pode conter milhões de objetos (instâncias, exemplos), e o método de agrupamento de dados deve ser rápido e escalável com a quantidade de objetos a ser agrupada;

- fornecer resultados interpretáveis e utilizáveis;

- ser robusto com relação a ruídos na base de dados;

- exigir o mínimo de conhecimento para se determinar os parâmetros de entrada: os valores apropriados são geralmente desconhecidos e difíceis de determinar;

- aceitar restrições: aplicações no mundo real podem precisar agrupar objetos de acordo com vários tipos de restrição; e

- encontrar um número adequado de partições: quando se tratar de problema de agrupamento automático.

Nenhuma técnica atual de agrupamento de dados atende a todos esses requisitos adequadamente, embora um trabalho considerável esteja sendo feito para que sejam atendidos pontos específicos ou conjuntos com alguns destes (AGRAWAL et al., 1998). Além disso, como mencionado por Hartigan (1985), “diferentes agrupamentos são adequados a diferentes propósitos”, o que dificulta a escolha do melhor algoritmo, já que essa tarefa depende da especificação do problema.

Segundo Fasulo (1999), os métodos heurísticos existentes para a solução de problemas de agrupamentos de dados podem, em geral, ser classificados em métodos hierárquicos e métodos de particionamento. Nos métodos hierárquicos, os grupos vão sendo formados gradativamente, por meio de aglomerações ou divisões de elementos/grupos, gerando uma hierarquia de partições, geralmente representadas por meio de uma estrutura em árvore.

Nos algoritmos particionais de agrupamento, o conjunto de elementos é dividido em *k* subconjuntos – podendo *k* ser conhecido ou não –, sendo que cada configuração (partição) obtida é avaliada por meio de uma função-objetivo. Caso a avaliação do agrupamento indique que a configuração não atende ao problema em questão, nova configuração é obtida por meio da migração de elementos entre os grupos. O processo continua nesse curso iterativo até que algum critério de parada seja satisfeito (BERKHIN, 2002).

Nas seções seguintes, são descritos os quatro algoritmos de agrupamento de dados utilizados nos experimentos descritos neste documento, sendo eles os algoritmos hierárquicos com Ligação Simples (LS) e Ligação Média (LM), o algoritmo *k*-médias (KM) e o algoritmo *Shared Nearest Neighbor* (SNN) (ERTÖZ et al., 2002).

3.1.1. Algoritmos Hierárquicos – Ligação Simples e Ligação Média

Os algoritmos de Ligação Simples (LS) e de Ligação Média (LM) são algoritmos hierárquicos do tipo aglomerativo (BARBARA, 2000; JAIN; DUBES, 1988; XU; WUNSCH, 2005); ou seja, iniciam com a quantidade de grupos igual à quantidade de elementos da base de dados, cada grupo contendo apenas um único elemento, e constroem a hierarquia de partições reunindo sucessivamente os grupos mais similares (mais próximos). A diferença entre os algoritmos LS e LM reside na forma como é calculada a proximidade entre os grupos.

No algoritmo LS, a medida de proximidade é dada pela menor distância entre todos os possíveis pares de objetos x_i e x_j, com $x_i \in c_a$ e $x_j \in c_b$, onde c_a e c_b são duas partições candidatas a se unir (Equação 3.2) (BARBARA, 2000).

$$d(c_a, c_b) = \min_{x_i \in c_a, x_j \in c_b} d(x_i, x_j)$$

Equação 3.2

Já no algoritmo LM, a proximidade é calculada pela média aritmética das distâncias entre todos os possíveis pares de objetos x_i e x_j, com $x_i \in c_a$ e $x_j \in c_b$, onde c_a e c_b são duas partições candidatas a se unir (Equação 3.3) (BARBARA, 2000).

$$d(c_a, c_b) = \frac{1}{|c_a||c_b|} \sum_{x_i \in c_a} \sum_{x_j \in c_b} d(x_i, x_j)$$

Equação 3.3

Dado um conjunto de objetos *X*, esses algoritmos funcionam da seguinte forma:

1. constrói-se a matriz de proximidade entre os grupos. Inicialmente, cada grupo possui apenas um elemento. No caso do algoritmo LS, a proximidade entre dois grupos é dada pela Equação 3.2; no caso do algoritmo LM, pela Equação 3.3;

2. verifica-se quais são os grupos mais próximos, reunindo-os em um único grupo;

3. atualiza-se a matriz de proximidade. Eliminam-se as linhas e colunas referentes aos grupos que foram reunidos e incluem-se uma linha e uma coluna para o novo

grupo formado por essa união. Calculam-se novamente as proximidades pela Equação 3.2 ou pela Equação 3.3; e

4. retorna-se ao passo 2, caso a condição de parada não tenha sido alcançada. A condição de parada pode ser, por exemplo, atingir um determinado número de grupos ou chegar a um único grupo.

No agrupamento hierárquico, as soluções são tipicamente representadas na forma de dendograma, que consiste em um tipo especial de estrutura de árvore. Cortando-se o dendograma em um certo nível, é possível transformar a hierarquia em uma partição, conforme ilustrado na Figura 3.1. As duas linhas tracejadas representam cortes com três e quatro grupos.

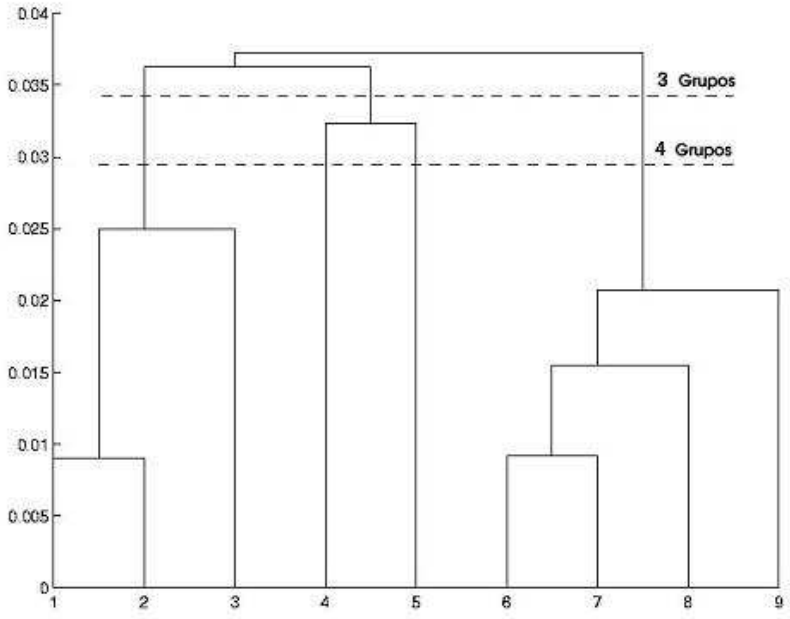


Figura 3.1 – Dendograma.

Algoritmos de agrupamento hierárquico resultam em grupos de formas convexas, apresentando complexidade $O(n^2)$ (XU; WUNSCH, 2005). Eles se baseiam em decisões locais, e, após um objeto ter sido associado a um grupo, não volta a ser considerado, significando que esses algoritmos não possuem um mecanismo para corrigir associações realizadas incorretamente.

3.1.2. Algoritmo k-médias (KM)

O algoritmo *k*-médias (HARTIGAN, 1975; HARTIGAN; WONG, 1979) é, de longe, a mais popular ferramenta de agrupamento de dados utilizada em aplicações científicas e industriais (BERKIN, 2002). O nome vem da forma como os grupos são representados; ou seja, cada grupo da partição resultante é representado por um centróide, que pode ser visto como o centro de gravidade do grupo e é calculado a partir da média dos vetores dos objetos pertencentes ao grupo.

O algoritmo *k*-médias particiona um conjunto de *n* objetos em *k* grupos, onde o *k* é fornecido como entrada para o algoritmo. Os grupos são formados de maneira a maximizar a similaridade intra-grupo, que é medida em relação ao centróide.

Inicialmente, são selecionados *k* objetos, geralmente de forma aleatória, cada um representando o centróide de um grupo. Em seguida, a medida de similaridade é utilizada para calcular a proximidade dos elementos remanescentes em relação a cada um dos centróides, e eles são inseridos no grupo de maior similaridade.

Na sequência, os centróides são calculados pela média dos objetos do grupo, e novamente são calculadas as distâncias entre os objetos e os centróides. Os objetos podem ser realocados em outros grupos, e o processo se repete até que os centróides não mudem, ou até que a execução alcance determinado número de iterações. O objetivo do *k*-médias consiste em minimizar a distância entre cada ponto e o centróide do grupo ao qual pertence o ponto.

Segue uma definição mais formal do algoritmo, conforme Jain et al. (1999). Dado um conjunto de dados *X* e um valor *k*, que representa a quantidade de grupos em que *X* deve ser particionado, o *k*-médias funciona da seguinte maneira:

1. determinam-se *k* centróides iniciais dos grupos. Esses centróides iniciais são selecionados aleatoriamente ou seguindo alguma heurística;

2. calcula-se a distância entre cada objeto remanescente e o centróide de cada grupo;

3. associa-se cada objeto ao grupo com o centróide mais próximo;

4. recalculam-se os centróides considerando os objetos alocados; e

5. retorna-se ao passo 2, caso o critério de parada não tiver sido satisfeito.

A complexidade computacional do algoritmo *k*-médias é *O*(*nkd*) (XU; WUNSCH, 2005), sendo ele sensível à escolha dos centróides iniciais e da sua forma de atualização. Dependendo da escolha dos centróides, o algoritmo converge para um ótimo local. O *k*-médias tende a encontrar grupos de formato esférico de igual tamanho e bem separados.

3.1.3. Algoritmo Shared Nearest Neighbor (SNN)

O algoritmo *Shared Nearest Neighbor* (SNN) tem como critério de agrupamento o encadeamento ou ligação entre os objetos a serem agrupados (ERTÖZ et al., 2002).

SNN encontra os vizinhos mais próximos de cada objeto, utilizando como medida de proximidade o número de vizinhos que dois objetos compartilham no espaço dos dados. Com essa medida de proximidade, o algoritmo trabalha para encontrar os pontos mais representativos, e constrói os grupos ao redor desses pontos. Além da quantidade de vizinhos mais próximos a ser considerada, o algoritmo trabalha ainda com dois outros tipos de parâmetros (ERTÖZ et al., 2002):

1. parâmetros relacionados ao peso das conexões do grafo dos vizinhos mais próximos compartilhados ou "similaridade SNN" (*strong*, *merge* e *label*); e

2. parâmetros relacionados ao número de conexões fortes ou "densidade SNN" (*topic* e *noise*). Esses parâmetros são limiares (*thresholds*) nos quais se baseia cada passo do algoritmo.

A seguir, são detalhados os passos do SNN, segundo (ERTÖZ et al., 2002):

1. dado um conjunto de objetos *X*, calcula-se uma matriz de proximidade que contenha a proximidade entre cada par de objetos de *X*;

2. encontram-se os n_v vizinhos mais próximos de cada objeto;

3. constrói-se o grafo dos vizinhos mais próximos compartilhados (grafo SNN), compreendendo os seguintes subpassos:

- calcula-se o número de vizinhos compartilhados para todo par de objetos;
- acrescenta-se uma conexão entre os objetos x_i e x_j, se ambos têm o outro em sua lista de vizinhos mais próximos; e
- torna-se o peso da conexão igual ao número de vizinhos mais próximos compartilhados por x_i e x_j. Esse peso captura a "similaridade SNN" entre os dois objetos, $sim(x_i, x_j)$.

4. calcula-se a "densidade SNN" (também chamada de conectividade) de cada objeto. Se o peso de uma conexão do grafo SNN (ou seja, a "similaridade SNN") é maior do que o limiar *strong*, então essa conexão é considerada forte. A "densidade SNN" de um objeto x_i, $den(x_i)$, é dada pelo número de conexões fortes que o objeto x_i possui.

5. para cada objeto x_i:

   - se $den(x_i) <$ *noise*, então descarta-se o objeto x_i;
   - se $den(x_i) >$ *topic*, então x_i é similar à maioria dos seus vizinhos e é escolhido como objeto representativo de sua vizinhança.

6. para cada par de objetos de *X*, (x_i, x_j), se $sim(x_i, x_j) >$ *merge* e se pelo menos um deles é representativo, então, x_i e x_j são inseridos no mesmo grupo; e

7. associam-se a grupos os objetos que não são ruídos nem representativos de algum grupo. Para isso, percorre-se a lista dos vizinhos mais próximos compartilhados de todos os objetos já associados a algum grupo, procurando identificar conexões com objetos ainda não pertencentes a nenhum grupo. Os objetos não associados a um grupo, e cujo peso da conexão com um objeto representativo é maior do que *label*, são associados ao mesmo grupo do objeto representativo.

A complexidade computacional do algoritmo SNN é $O(n^2)$. Esse algoritmo é ideal para encontrar grupos com formas, tamanhos e densidades diferentes. Ele também lida bem com ruídos e *outliers*. SNN também é apropriado para lidar com dados de alta dimensionalidade (ERTÖZ et al., 2002).

3.2. Fusão de Partições

A natureza exploratória do processo de agrupamento de dados requer métodos eficazes e eficientes, sendo que essa tarefa poderia se beneficiar da força combinada de vários algoritmos individuais de agrupamento de dados. Este é justamente o foco de pesquisa por trás da abordagem conhecida como fusão de partições (ou *clustering ensemble*) (STREHL; GHOSH, 2002; FERN; BRODLEY, 2003, 2004; TOPCHY et al., 2005): encontrar uma combinação de múltiplas partições que proporcione melhorias em termos de qualidade à solução final do processo de agrupamento de dados.

Em vários aspectos, a fusão de partições vai além do que é tipicamente alcançado por algoritmos individuais de agrupamento de dados (TOPCHY et al., 2005). Dentre os objetivos da área de fusão de partições, seguem três de particular interesse para este estudo:

- Robustez: desempenho médio melhor para os diversos domínios e conjuntos de dados;
- Novidade: identificação de uma partição final que não pode ser obtida com nenhum algoritmo aplicado individualmente; e
- Estabilidade: obtenção de soluções de agrupamento com menor sensibilidade a ruídos, *outliers*, variações de amostragem ou à variabilidade dos algoritmos.

Geralmente, um sistema de fusão de partições resolve o problema de agrupamento de dados em dois passos. No primeiro, é gerado um conjunto de possíveis soluções para o problema, cada solução apresentando informações sobre os dados segundo a perspectiva (critério de agrupamento) do algoritmo que a gerou. Em um segundo passo, essas soluções são combinadas para produzir uma partição-consenso. Assume-se nesse processo que as informações essenciais sobre os dados que se encontram nas partições iniciais sejam transferidas para a partição-consenso.

Dois aspectos fundamentais para o sucesso da fusão de partições encerram a diversidade das partições iniciais e a forma de determinação da partição-consenso. Na Subseção 3.2.1, são resumidas as abordagens mais comuns descritas na literatura para se gerar a diversidade necessária nas partições iniciais; já na Subseção 3.2.2 são discutidas

as abordagens para se gerar a partição-consenso. Na Subseção 3.2.3, são apresentadas as duas técnicas de fusão de partições de maior interesse para o presente estudo.

3.2.1. Geração de Partições Iniciais

A diversidade das partições iniciais serve de base para a construção da partição final, uma vez que fornecem informações sobre a estrutura de partição dos dados de acordo com diferentes perspectivas. Assim, para se escolher os algoritmos de agrupamento, ou as formas de aplicação de um mesmo algoritmo, faz-se necessário compreender o objetivo esperado da fusão, atentando-se para as informações que se deseja extrair dos dados.

Ghaemi et al. (2009) descrevem algumas abordagens atualmente utilizadas para se gerar as partições iniciais com a diversidade necessária:

- aplicar vários algoritmos de agrupamento de dados (STREHL; GHOSH, 2002);
- executar várias vezes um mesmo algoritmo com diferentes iniciações e parâmetros (FRED, 2001; FRED; JAIN, 2002; LUO et al., 2006);
- realizar o agrupamento utilizando diferentes subconjuntos de objetos (reamostragem), com todas as características originais (STREHL; GHOSH, 2002; FERN; BRODLEY, 2004);
- realizar o agrupamento com todas as amostras, porém utilizando diferentes subconjuntos dos atributos originais (STREHL; GHOSH, 2002) ou uma projeção das amostras em um espaço de dimensão menor (FERN; BRODLEY, 2004); e
- empregar algoritmos mais simples do que os convencionais, chamados de algoritmos de agrupamento fracos (TOPCHY et al., 2005).

O presente trabalho contempla tanto fusões homogêneas quanto heterogêneas. Isso porque as partições iniciais são obtidas pela aplicação de mais de um algoritmo de agrupamento à base de dados. Além disso, para se incrementar mais ainda os níveis de diversidade, também são criadas partições utilizando diferentes parâmetros e diferentes formas de iniciação de um mesmo algoritmo.

3.2.2. Determinação da Função-Consenso

Uma função-consenso recebe um conjunto de partições-base como entrada e as agrega em uma única partição final. Essa função ou operador realiza o passo essencial da fusão de partições, pois determina como as partições serão realmente combinadas.

É importante observar que a combinação de partições constitui uma tarefa complexa, equivalente ao problema de encontrar uma partição mediana em relação às partições-base, que já é um problema comprovadamente NP-Completo (TOPCHY et al., 2005). Os diferentes tipos de funções-consenso são descritos a seguir (TOPCHY et al., 2004).

Métodos baseados em coassociação: a similaridade entre os objetos é estimada pelo número de grupos que um par de objetos compartilha em todas as partições-base. Uma matriz de similaridade é criada, cujas células expressam a força de coassociação entre pares de objetos. Para tanto, consideram-se as co-ocorrências do par de objetos no mesmo grupo como votos para a sua coassociação. A partição final é obtida aplicando-se a essa matriz algum método de agrupamento de dados baseado em similaridade.

Alguns problemas são enfrentados na aplicação dessa classe de métodos, dentre os quais: a falta de uma abordagem para definir qual algoritmo de agrupamento deve ser usado para gerar a partição final; e a baixa confiabilidade na estimativa de coassociações quando é baixo o número de partições a serem combinadas.

Métodos baseados em grafo/hipergrafo: os grupos são representados como hiperarestas, cujos vértices correspondem aos objetos que estão sendo agrupados, de maneira que cada hiperaresta descreve um conjunto de objetos pertencentes ao mesmo grupo. Uma vez que os grupos são representados dessa forma, pode-se utilizar um algoritmo de particionamento de grafos para se gerar a partição-consenso.

Fern e Brodley (2004) explicam que a entrada para um problema de particionamento de grafo é um grafo composto por vértices e arestas valoradas, representado por $G = (V, W)$, em que *V* é um conjunto de vértices e *W* é uma matriz de similaridade simétrica e não-negativa, $|V| \times |V|$, que representa a similaridade entre cada par de objetos. Nesse contexto, particionar um grafo em *k* partes significa encontrar *k* grupos disjuntos de vértices $\pi = \{c_1, c_2, ..., c_k\}$, em que $\bigcup_{j=1}^{k} c_j = V$. A menos que um

grafo tenha *k*, ou mais de *k*, componentes fortemente conectados, qualquer partição de *k* componentes cruzará algumas das arestas do grafo. A soma dos pesos das arestas cortadas é definida como o corte de uma partição $\pi : Corte(\pi, W) = \sum W(i, j)$, sendo que nesse caso os vértices *i* e *j* não pertencem ao mesmo grupo. Em outras palavras, dado um grafo valorado, o objetivo do particionamento de grafo consiste em encontrar uma partição de *k* componentes que reduza o número de cortes necessários, sendo que cada componente deve conter aproximadamente o mesmo número de vértices.

Métodos baseados em informação mútua: Informação Mútua (MI, do inglês *Mutual Information*) é uma medida simétrica que quantifica a informação estatística compartilhada entre duas distribuições, podendo fornecer uma indicação segura da informação compartilhada entre duas partições de dados (STREHL; GHOSH, 2002). Sob esta perspectiva, uma função-consenso numérica é formulada em termos da informação mútua entre os rótulos na partição-consenso e os rótulos nas partições-base, sendo que a partição-consenso seria obtida pela maximização dessa função, considerando o número de grupos desejados. Para solucionar o problema de otimização dessa função numérica, heurísticas baseadas em coassociação (FRED; JAIN, 2003) ou hipergrafo (STREHL; GHOSH, 2002) são utilizadas.

Topchy et al. (2005) definem uma função-consenso com base na informação mútua generalizada, otimizando-a mediante emprego do algoritmo *k*-médias.

Métodos baseados em votação: se a correspondência dos rótulos entre as partições estiver correta, um procedimento simples de votação pode ser adotado para determinar a qual grupo pertence cada objeto na partição-consenso. Entretanto, a correspondência dos rótulos é o que dificulta a combinação não-supervisionada. A ideia principal é fazer uma rerrotulagem de todas as partições com base na melhor correspondência com uma partição de referência, que pode ser uma das partições-base ou um novo agrupamento do conjunto de dados. Além disso, durante a votação, pressupõe-se que o número de grupos em todas as partições-base seja o mesmo, assim como na partição-consenso. Isso requer o prévio conhecimento do número de grupos na partição-consenso.

Modelo Misto Finito: neste caso, especifica-se a função-consenso com base na máxima verossimilhança de um modelo de mistura finita sobre o conjunto de partições-base (TOPCHY et al. 2004). Esse conjunto de partições é modelado com uma mistura de

distribuições multinominais no espaço dos rótulos dos grupos. O problema de máxima verossimilhança pode ser resolvido por meio do algoritmo *Expectation-Maximization*.

3.2.3. Funções-Consenso de Interesse

De acordo com Topchy et al. (2005), os algoritmos de fusão de partições que utilizam uma abordagem baseada em grafos/hipergrafos são mais robustos que aqueles baseados em matriz de coassociações. Dessa forma, quatro algoritmos se destacam na literatura: três deles propostos por Strehl e Ghosh (2002) e um proposto por Fern e Brodley (2004). Por isso, esses algoritmos foram escolhidos para fazer parte deste estudo, servindo de alvo de comparação para os resultados alcançados pela nova abordagem proposta baseada em Programação Genética. Além disso, o algoritmo de Fern e Brodley (2004) é utilizado como parte integrante do método apresentado.

Para a operação dos quatro algoritmos, são desnecessários os atributos originais dos objetos, requerendo-se apenas o rótulo do grupo ao qual pertence cada objeto em cada uma das partições-base. Os quatro algoritmos são detalhados a seguir.

3.2.3.1. Métodos Propostos por Strehl e Ghosh (2002)

Strehl e Ghosh (2002) formalizam o problema de fusão de partições como um problema de otimização combinatória em termos de compartilhamento de informações mútuas. Como a otimização de uma função-consenso com base na informação mútua é um problema combinatório de natureza difícil, os autores propõem três métodos baseados em heurísticas: *Cluster-based Similarity Partitioning Algorithm* (CSPA), *HiperGraph-Partitioning Algorithm* (HGPA) e *Meta-Clustering Algorithm* (MCLA). Os três algoritmos têm um custo computacional baixo; assim sendo, os três podem ser executados em sequência, e uma função supra-consenso, baseada na informação mútua compartilhada, passa a escolher, dentre as três respostas geradas, aquela que será a partição-consenso final.

Conforme explicado por Strehl e Ghosh (2002), a Informação Mútua é uma medida simétrica que quantifica, estatisticamente, as informações compartilhadas por duas distribuições, podendo, portanto, fornecer uma indicação sólida da informação compartilhada por um par de partições. Assim, a Informação Mútua Normalizada (NMI) estimada entre duas partições π^a e π^b é dada pela Equação 3.4, em que $|.|$ indica o

número de objetos de um grupo, K^a e K^b denotam o número de grupos nas partições π^a e π^b, respectivamente, c_h^a é o *h*-ésimo grupo de π^a, c_l^b é o *l*-ésimo grupo de π^b e *n* é o número de objetos do conjunto de dados (STREHL; GHOSH, 2002).

$$\phi^{(NMI)}(\pi^a, \pi^b) = \frac{\sum_{h=1}^{K^a}\sum_{l=1}^{K^b} |c_h^a \cap c_l^b| \log\left(\frac{n|c_h^a \cap c_l^b|}{|c_h^a||c_l^b|}\right)}{\sqrt{\left(\sum_{h=1}^{K^a} |c_h^a| \log\left(\frac{|c_h^a|}{n}\right)\right)\left(\sum_{l=1}^{K^b} |c_l^b| \log\left(\frac{|c_l^b|}{n}\right)\right)}}$$

Equação 3.4

Com base nessa medida de informação mútua, Strehl e Ghosh (2002) definem uma nova medida, a Informação Mútua Normalizada Média (ANMI), entre uma partição π^i e um conjunto Π de *r* partições, sendo esta dada pela Equação 3.5.

$$\phi^{(ANMI)}(\pi^i, \Pi) = \frac{1}{r}\sum_{q=i}^{r} \phi^{(NMI)}\left(\pi^i, \pi^q\right)$$

Equação 3.5

A partição consenso $\pi^{F(k-opt)}$ é aquela com valor máximo de $\phi^{(ANMI)}$ em relação às partições individuais em Π, dado que *k* é o número de grupos desejado para a partição-consenso. Ou seja, $\phi^{(ANMI)}$ é a função-objetivo, ao passo que $\pi^{F(k-opt)}$ passa a ser obtida a partir da Equação 3.6:

$$\pi^F = \arg\max_{\pi^i} \sum_{q=1}^{r} \phi^{(NMI)}\left(\pi^i, \pi^q\right).$$

Equação 3.6

Como já afirmado, os três métodos heurísticos propostos por Strehl e Ghosh (2002) podem ser executados em sequência, gerando três partições-consenso, sendo que a medida $\phi^{(ANMI)}$ pode ser utilizada para escolher a melhor partição dentre as geradas. A primeira atividade consiste em fazer uma representação das partições-base em forma de hipergrafo.

Para cada partição-base é construída uma matriz binária de pertinência, com uma coluna para cada grupo e as linhas representando os objetos do conjunto de dados. Todas as células da matriz são preenchidas com “1” (um), se o objeto pertencer ao

grupo referente à coluna, ou “0” (zero), no caso contrário. A concatenação dessas matrizes forma uma matriz de adjacências que representa o hipergrafo. A Tabela 3.1 exemplifica como um conjunto de partições-base (à esquerda) é representado como a matriz de adjacência (à direita) que representa o hipergrafo.

	π^1	π^2	π^3	π^4
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

$\Longleftrightarrow$

Vert	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}
	(c_1^1)	(c_2^1)	(c_3^1)	(c_1^2)	(c_2^2)	(c_2^3)	(c_1^3)	(c_2^3)	(c_3^3)	(c_1^4)	(c_2^4)
v_1	1	0	0	0	1	0	1	0	0	1	0
v_2	1	0	0	0	1	0	1	0	0	0	1
v_3	1	0	0	0	1	0	0	1	0	0	0
v_4	0	1	0	0	0	1	0	1	0	1	0
v_5	0	1	0	0	0	1	0	0	1	0	1
v_6	0	0	1	1	0	0	0	0	1	0	0
v_7	0	0	1	1	0	0	0	0	1	0	0

Tabela 3.1 – Exemplo de um conjunto de partições sendo representadas na forma de um hipergrafo

Os três métodos propostos por Strehl e Ghosh (2002) são apresentados a seguir.

CSPA (*Cluster-based Similarity Partitioning Algorithm*)

Esse algoritmo baseia-se em coassociação. Uma matriz binária $n \times n$, onde *n* é a quantidade de objetos no conjunto de dados, é criada para cada partição-base. Cada célula, representada por um par de objetos (número da linha, número da coluna), recebe “1” (um), se pertencer ao mesmo grupo, ou “0” (zero), no caso contrário. Esse conjunto de pares serve de entrada para a criação de uma matriz de similaridade geral, cujas entradas denotam as frações correspondentes às partições nas quais dois objetos pertencem ao mesmo grupo.

A matriz de similaridade geral é então utilizada para reagrupar os objetos, utilizando-se para isso um algoritmo de agrupamento baseado em similaridade. Strehl e Ghosh (2002) utilizam o algoritmo METIS (KARYPIS; KUMAR, 1998).

A Figura 3.2 ilustra a criação das matrizes de similaridade iniciais e a sua consolidação na matriz de similaridade geral, utilizando o exemplo apresentado na Tabela 3.1. Toda partição-base contribui com uma matriz de similaridade, sendo cada par de objetos indicado pela cor preta, quando pertencer ao mesmo grupo, ou pela cor

branca, no caso contrário. A média de cada célula é então utilizada para criar a matriz de similaridade geral, que apresenta valores na escala de cinza.

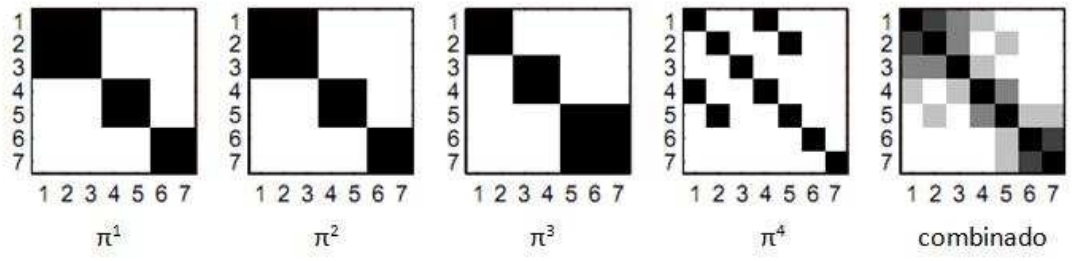


Figura 3.2 – Ilustração da matriz de similaridade produzida por CSPA.

O algoritmo CSPA utiliza a mais simples e óbvia heurística, porém apresenta complexidade computacional quadrática em *n*, sendo *n* o número de instâncias do conjunto de dados.

HGPA (*HiperGraph-Partitioning Algorithm*)

Neste método, a fusão de partições é formulada como um processo de particionamento de um hipergrafo. A ideia é cortar o número mínimo de hiperarestas que particionam o hipergrafo em *k* componentes desconectados de tamanhos aproximados, sendo que o valor de *k* é informado pelo usuário como entrada do algoritmo. Para particionar o hipergrafo, Strehl e Ghosh (2002) utilizaram o algoritmo HMETIS (KARYPIS et al., 1997).

Strehl e Ghosh (2002) explicam que a obtenção de grupos com tamanhos parecidos constitui uma restrição-padrão no particionamento de grafos, que evita a criação de grupos triviais. Porém, isso implica que se os grupos reais têm tamanhos bem diferentes, o método torna-se inapropriado.

A Figura 3.3 ilustra o funcionamento de HGPA, tomando como base as partições-base e o hipergrafo apresentados na Tabela 3.1. Cada hiperaresta é representada por uma elipse, circundando os objetos que ela conecta. A partição-consenso $\{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_6, x_7\}\}$ tem o menor número de cortes de hiperarestas (quatro), e é tão balanceada quanto possível para uma quantidade solicitada de três grupos em uma base de dados com sete objetos.

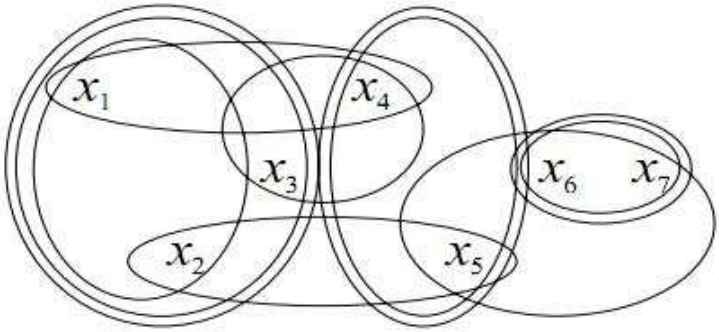


Figura 3.3 – Ilustração da fusão de partições obtida via HGPA.

MCLA (*Meta-Clustering Algorithm*)

Em cada partição-base, os grupos podem conter os mesmos objetos ou um grande número de objetos compartilhados. Assim, tais grupos podem ser considerados similares entre si. Com base nisso, o MCLA constrói um grafo que modela o relacionamento (similaridade) entre os grupos pertencentes às diversas partições-base. Em seguida, o grafo é particionado de tal maneira que os grupos que permanecerem no mesmo conjunto são semelhantes. Os objetos são, então, atribuídos aos grupos aos quais estão mais fortemente associados.

Segundo Strehl e Ghosh (2002), os principais passos do MCLA são:

- Construir um metagrafo: cada grupo de partições-base é considerado um vértice de outro grafo regular não direcionado. O peso da aresta entre os vértices correspondentes aos grupos c_j^i e c_t^s é definido utilizando-se a medida de Jaccard (BOUTIN; HASCOËT, 2004), de acordo com a Equação 3.7.

$$w\left(c_j^i, c_t^s\right) = \frac{|c_j^i \cap c_t^s|}{|c_j^i \cup c_t^s|}$$

Equação 3.7

- Agrupar as hiperarestas: o agrupamento se dá pelo particionamento do metagrafo em *k* metagrupos balanceados, sendo *k* uma entrada do algoritmo fornecido pelo usuário. Para realizar esse particionamento, Strehl e Ghosh (2002) utilizam o algoritmo METIS (KARYPIS; KUMAR, 1998).

- Unir os grupos de cada metagrupo: para cada metagrupo, transformam-se as hiperarestas em uma única meta-hiperaresta. Cada objeto pode pertencer a mais de um metagrupo. Assim, para cada meta-hiperaresta, calcula-se um vetor descrevendo o nível de associação de cada objeto com o metagrupo. O nível de associação de um objeto com um metagrupo é dado pela média do número de grupos desse metagrupo que contêm o objeto.

- Determinar o metagrupo final de cada objeto: cada objeto é associado ao metagrupo com o qual possui maior valor de associação. Os desempates são decididos aleatoriamente. A partição-consenso é a partição dos objetos indicada pelos metagrupos.

Deve-se notar que não há a garantia de que todo metagrupo contenha pelo menos um objeto. Assim, a partição-consenso contém, no máximo (e não exatamente), *k* grupos.

3.2.3.2. Hybrid Bipartite Graph Formulation (HBGF)

Fern e Brodley (2004) propõem a construção de um grafo bipartido a partir do conjunto de partições-base que serão combinadas, modelando, simultaneamente, tanto objetos quanto classes como vértices do grafo, e, na sequência, particionando o grafo por meio de uma técnica tradicional. Dessa forma, HBGF retém todas as informações fornecidas pelo conjunto de partições-base, possibilitando que as similaridades entre os objetos e entre os grupos sejam consideradas na formação da partição-consenso.

Na sequência, apresenta-se uma descrição mais formal do HBGF, segundo (FERN; BRODLEY, 2004):

- Dado um conjunto de partições-base, HBGF constrói um grafo $G = (V, W)$ da seguinte maneira:

  - $V = V^C \cup V^O$, onde V^C contém os vértices referentes aos grupos das partições-base Π, ao passo que V^O contém os vértices referentes aos objetos do conjunto de dados *X*.

- O peso entre dois vértices *i* e *j* é definido da seguinte maneira: se $i, j \in V^C$ ou $i, j \in V^O$ (ou seja, ambos representam grupos ou ambos representam objetos), então $w(i, j) = 0$; caso contrário, se o objeto x_i pertence ao grupo c_j, então $w(i, j) = w(j, i) = 1$; senão, $w(i, j) = 0$.

Note que a matriz *W* pode ser definida da seguinte maneira:

$$W = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix},$$

onde *A* é uma matriz de conectividade na qual as linhas correspondem aos objetos e as colunas correspondem aos grupos. Se a célula $A(i, j)$ tem o valor "1" (um), isso ocorre porque o objeto *i* pertence ao *j*-ésimo grupo; caso contrário, o valor de $A(i, j)$ é "0" (zero).

A Figura 3.4 mostra um exemplo de aplicação do HBGF. Os itens (a) e (b) dessa figura descrevem duas diferentes partições de nove objetos cada, enquanto o item (c) mostra o grafo construído pelo HBGF. Os vértices dispostos em forma de losango representam os grupos, e os vértices dispostos em forma de círculo representam os objetos do conjunto de dados. Uma aresta entre um vértice "objeto" e um vértice "grupo" indica que o grupo contém o objeto. Todas as arestas têm peso "1" (um), já que as arestas com peso "0" (zero) foram omitidas do grafo. Nesse grafo, os vértices "grupo" estão conectados somente com os vértices "objeto" e vice-versa, formando um grafo bipartido. Se uma nova partição for adicionada ao conjunto de partições-base, um novo conjunto de vértices "grupo" será adicionado ao grafo, e cada um deles ficará conectado a um vértice "objeto" correspondente.

A linha tracejada, mostrada na Figura 3.4(c), indica uma partição desse grafo em duas partes, obtida por meio de algum algoritmo de particionamento de grafos tradicional. Para esse fim, Fern e Brodley (2004) utilizam duas técnicas distintas: *Spectral Graph Partitioning* (NG et al., 2002) e METIS (KARYPIS; KUMAR, 1998).

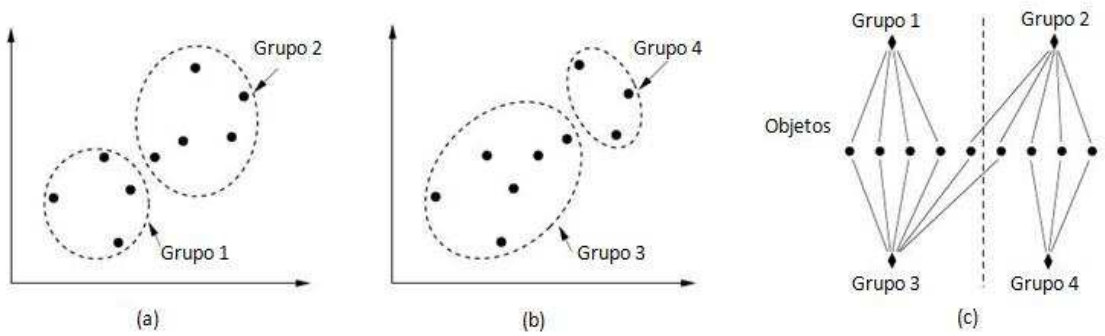


Figura 3.4 – Ilustração de uma fusão de partições por HBGF.

Para gerar as partições-base, Fern e Brodley (2004) utilizaram duas abordagens: a reamostragem (DUDOIT; FRIDLYAND, 2003) e a projeção randômica (FERN; BRODLEY, 2003). Nos dois métodos, *k*-médias foi utilizado como algoritmo-base de agrupamento de dados, e o número de grupos *k* foi pré-especificado para cada conjunto de dados, permanecendo o mesmo para todas as execuções de agrupamento e fusão.

Nos experimentos apresentados por Fern e Brodley (2004), os autores observaram que o HBGF apresentou um desempenho equivalente ou superior àqueles obtidos pelas abordagens de Strehl e Ghosh (2002). Por esse motivo, HBGF é o método de fusão adotado para integrar as hierarquias de fusão geradas por Programação Genética, no contexto da abordagem proposta neste estudo.

3.3. Validação de Agrupamentos

O resultado de um algoritmo de agrupamento de dados ou de uma fusão de partições pode ser avaliado por meio de índices estatísticos. Esses índices revelam de maneira quantitativa e objetiva a adequação da estrutura encontrada. A adequação de um agrupamento de dados refere-se ao modo como a estrutura obtida fornece informações verdadeiras sobre os dados agrupados (JAIN; DUBES, 1988).

A validade de um agrupamento de dados pode ser expressa por meio de três tipos de critério: externo, interno e relativo. Um critério expressa a estratégia pela qual o agrupamento será avaliado, enquanto o índice representa a estatística pela qual é testada a validade (JAIN; DUBES, 1988).

Um critério externo avalia o agrupamento com base em um conhecimento prévio dos dados. Esse conhecimento pode ser uma partição previamente conhecida ou um

agrupamento construído por um especialista da área com base em conhecimento prévio (FACELI, 2006). Já o critério interno avalia um agrupamento sem esse tipo de informação; sua avaliação se dá com base apenas nos dados originais. O critério interno tenta medir a qualidade do agrupamento por meio das informações intrínsecas dos dados.

Segundo Jain e Dubes (1988), os critérios relativos indicam, dentre algumas estruturas, quais são melhores em um ou mais aspectos, qual delas é mais estável, ou, ainda, qual delas apresenta o melhor valor para determinado aspecto dos dados. Há vários índices que podem ser empregados com critérios relativos, e que, em geral, também podem ser aplicados com critérios internos. A diferença reside na sua forma de aplicação.

Os resultados apresentados neste estudo serão avaliados segundo o índice *Rand* Corrigido (CR, do inglês *Corrected Rand*), que deve ser empregado consoante algum critério de validação externo (HUBERT; ARABIE, 1985). Essa escolha se deve ao fato de o índice não ser sensível ao número de grupos do agrupamento, além de ser o mais utilizado nas avaliações e comparações de algoritmos de agrupamento de dados (FACELI, 2006).

O índice CR faz a comparação de duas partições, π^a e π^b, de maneira que, para ser utilizado no contexto de um critério de validação externo, uma das partições de entrada é a estrutura dos dados previamente conhecida, enquanto a outra é a própria partição sob avaliação. Esse índice apresenta valor em torno de “0” (zero), podendo ser negativo caso a partição-alvo, que está sendo avaliada, seja totalmente diferente da partição de referência. Por outro lado, assume valor próximo a “1” (um) quando as partições são idênticas (JAIN; DUBES, 1988). Por exemplo, se dois objetos x_1 e x_2 são associados ao mesmo grupo em π^a e a grupos diferentes em π^b, isso resulta em um decréscimo do valor do índice. O CR corresponde a uma normalização do índice *Rand*, proposto por Hubert e Arabie (1985), sendo calculado por meio da Equação 3.8.

$$CR = \frac{\sum_{i=1}^{K^a}\sum_{j=1}^{K^b}\binom{|c_i^a \cap c_j^b|}{2} - \left[\sum_{i=1}^{K^a}\binom{|c_i^a|}{2}\sum_{j=1}^{K^b}\binom{|c_j^b|}{2}\right]/\binom{n}{2}}{\left[\sum_{i=1}^{K^a}\binom{|c_i^a|}{2} + \sum_{j=1}^{K^b}\binom{|c_j^b|}{2}\right]/2 - \left[\sum_{i=1}^{K^a}\binom{|c_i^a|}{2}\sum_{j=1}^{K^b}\binom{|c_j^b|}{2}\right]/\binom{n}{2}}$$

Equação 3.8

Outros dois índices de interesse para este estudo, que não dependem de informações externas, já que avaliam as partições segundo critérios de validação internos, são a variância intra-*cluster* e a conectividade (HANDL; KNOWLES, 2005).

A variância intra-*cluster* (var) de uma partição é obtida pelo somatório das distâncias entre os objetos e o centróide do grupo ao qual eles pertencem. Grupos mais compactos indicam uma maior qualidade da partição. Dessa forma, a partição tem uma melhor avaliação quando o seu índice é baixo. Esse índice tende a valorizar os grupos de formato esférico (HANDL; KNOWLES, 2005). Formalmente, var é dada pela Equação 3.9, em que μ_k é o centróide do grupo c_k e $d(.,.)$ denota a função de distância (no *k*-médias, a distância euclidiana).

$$var(\pi) = \sum_{c_k \in \pi} \sum_{x_i \in c_k} d(x_i, \mu_k)$$

Equação 3.9

A conectividade (con) relaciona-se à noção de encadeamento ou ligação. Esse índice mede o grau com que objetos vizinhos no espaço dos dados são colocados no mesmo grupo, sendo medido de acordo com a Equação 3.10 e Equação 3.11, em que nn_{ij} é o *j*-ésimo vizinho mais próximo do objeto x_i e v é o número de vizinhos que contribuem para a conectividade, parâmetro este informado pelo usuário.

$$con(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{v} a(x_i, nn_{ij})$$

Equação 3.10

$$a(x_i, nn_{ij}) = \begin{cases} \frac{1}{j}, & \text{se } \nexists_{c_k} : x_i, nn_{ij} \in c_k \\ 0, & \text{caso contrário} \end{cases}$$

Equação 3.11

Quanto menor for o valor do índice con, melhor será o agrupamento. Esse índice captura a densidade local, podendo, portanto, detectar formas arbitrárias dos grupos (HANDL; KNOWLES, 2005).

3.4. Algoritmos de Agrupamento de Dados Multiobjetivo de Interesse

Os algoritmos *Multi-Objective Clustering with automatic K-determination* (MOCK) (HANDL; KNOWLES, 2005; HANDL; KNOWLES, 2006) e *Multi-Objective Clustering Ensemble* (MOCLE) (FACELI, 2006; FACELI et al., 2008) são algoritmos de agrupamento de dados que adotam internamente algum método evolutivo multiobjetivo. Como servirão de alvo de comparação com a abordagem proposta neste trabalho (Capítulo 5), esses algoritmos serão brevemente descritos a seguir.

3.4.1. MOCK (Multi-Objective Clustering with automatic K-determination)

MOCK (HANDL; KNOWLES, 2005; HANDL; KNOWLES, 2006) é um algoritmo de agrupamento de dados multiobjetivo que usa o algoritmo evolutivo multiobjetivo PESA-II (CORNER et al., 2001) para gerar uma aproximação do conjunto ótimo de Pareto. MOCK otimiza dois objetivos simultaneamente, gerando um conjunto de partições que representa diferentes compromissos com esses objetivos variando-se o números de grupos *k*. Além disso, com base na forma do fronte de Pareto, MOCK indica automaticamente o subconjunto com as melhores soluções.

A representação dos indivíduos utilizada no MOCK é uma cadeia de caracteres (cromossomo) de tamanho *n*, sendo *n* a quantidade de objetos na base de dados. Cada elemento do cromossomo (gene) representa um objeto da base de dados e pode assumir valor entre 1 (um) e *n*, significando que existe uma ligação entre o objeto indicado pela posição do gene no cromossomo e o objeto indicado pelo valor assumido pelo gene, ou seja, os dois objetos pertencem ao mesmo grupo. A decodificação dessa representação requer a identificação de todos os sub-grafos. Todos os objetos pertencentes ao mesmo sub-grafo são associados ao mesmo grupo.

A iniciação da população baseia-se numa mistura de soluções geradas via o algoritmo *k*-médias e via Árvore Geradora Mínima (MST, do inglês *Minimum Spanning Tree*). Para a recombinação, é utilizado cruzamento uniforme; para mutação, usa-se um operador especializado; e para diminuir o espaço de busca, adota-se a mutação dos vizinhos mais próximos.

Como funções-objetivo, são adotados dois índices de validação, a variância intra-*cluster* (var) (Equação 3.9) e a conectividade (con) (Equação 3.10). Segundo Handl e Knowles (2006), um importante aspecto na escolha dessas funções é o seu potencial para balancear a tendência de cada uma em aumentar (variância intra-*cluster*) ou a diminuir (conectividade) o número de grupos. Porquanto, a interação das duas é importante a fim de se explorar as diversas regiões do espaço de soluções e não se convergir para uma solução trivial (*n* grupos com apenas um objeto, no caso da variância intra-*cluster*, e um grupo com *n* objetos, no caso da conectividade).

O procedimento para escolha das melhores soluções baseia-se na suposição de que a estrutura dos dados está refletida na forma da fronteira de Pareto. Das tendências observadas referente aos dois objetivos empregados, é possível afirmar que, incrementando-se o número de grupos *k*, obtém-se uma melhora na variância intra-*cluster* (var), σV, ao custo de uma degradação na conectividade (con), σC, indicando certa organização na fronteira de Pareto baseada na quantidade de grupos que as soluções apresentam. Para um número de grupos *k* menor que o real, espera-se que a razão $R = \frac{\sigma V}{\sigma C}$ seja elevada, já que a separação de dois grupos causa significativa diminuição em var, com um pequeno aumento em con. Quando se ultrapassa o número correto de grupos, essa razão diminui, passando a degradação de var a ser menos significativa. Contudo, causa-se um grande aumento em con, pois um grupo correto está sendo dividido.

Assim, a mudança evidente em *R* que ocorre no número correto de grupos pode ser observada como um ponto de inflexão no gráfico da fronte de Pareto. Para auxiliar na determinação desse ponto, Handl e Knowles (2006) sugerem adotar como referência uma distribuição aleatória de dados. Esses dados são agrupados por meio do MOCK, gerando um conjunto de frontes de referência. Para cada solução no fronte-solução normalizado, é calculado um *attainment score*, dado pela distância até as *attainment surfaces* dos frontes de referência. Em seguida, é traçado um gráfico dos *attainment score* em relação a *k*. A solução correspondente ao máximo dessa curva é selecionada como a melhor solução.

3.4.2. MOCLE (Multi-Objective Clustering Ensemble)

MOCLE (FACELI, 2006; FACELI et al., 2008) é um algoritmo multiobjetivo de fusão de partições, que emprega um algoritmo genético multiobjetivo para otimizar simultaneamente duas ou mais medidas de validação de agrupamentos, sendo capaz de encontrar um conjunto de partições de alta qualidade em relação a essas diversas medidas. Como todo método de fusão, MOCLE gera um conjunto de partições-base, que constituirão a população inicial empregada pelo algoritmo genético. Por sua vez, esse algoritmo encontrará um conjunto de partições-consenso realizando a fusão das partições iterativamente, durante o processo de otimização, por meio do operador de recombinação.

A população inicial é construída utilizando-se diversos algoritmos de agrupamento de dados, que extraem informações da base de dados segundo diferentes visões. Além disso, também são aplicadas diferentes configurações de parâmetros para cada algoritmo, de modo a se gerar partições com diferentes níveis de refinamento. Nos experimentos descritos em (FACELI, 2006; FACELI et al., 2008), os algoritmos de agrupamento utilizados foram os algoritmos hierárquicos com ligação média e com ligação simples, o *k*-médias e o algoritmo *Shared Nearest Neighbor* descritos nas Subseções 3.1.1, 3.1.2 e 3.1.3, respectivamente.

A fusão de partições se dá por meio de um operador de recombinação especial, que resulta em uma partição que é o consenso entre duas partições-pais; ou seja, durante o processo de evolução, as partições são combinadas duas a duas. Esse operador restringe a busca às soluções que estão no conjunto de partições iniciais e às suas combinações. Além disso, durante o processo de evolução as partições iniciais de baixa qualidade são gradativamente eliminadas, enquanto as melhores partições e as boas combinações são preservadas, contribuindo para novas combinações.

No processo de seleção das partições, são empregadas funções-objetivo que correspondem a medidas de qualidade de uma partição. Assim como MOCK, MOCLE utiliza duas medidas de validação internas, sendo uma delas baseada na compacidade das soluções no espaço de busca, a variância intra-*cluster* (var) (Equação 3.9), e a outra a conectividade (con) (Equação 3.10), ambas apresentadas na Seção 3.3.

MOCLE apresenta também uma configuração que utiliza uma medida de validação externa, provendo um aspecto semissupervisionado ao algoritmo. Porém, essa configuração não é descrita aqui, por não se relacionar com a temática de interesse deste estudo.

3.5. Considerações Finais

No presente capítulo, foram apresentados conceitos básicos e avançados sobre agrupamento de dados, fusão de partições e validação de agrupamentos. Também foram descritas técnicas referentes a esses temas que contribuíram para o desenvolvimento da abordagem proposta nesta dissertação. Isso inclui os algoritmos de agrupamento de dados *k*-médias, LS, LM e SNN, os métodos de fusão de partições CSPA, HGPA, MCLA e HBGF e os índices de validação interna *Rand* Corrigido, variância intra-*cluster* e conectividade.

Além disso, a aplicação de técnicas de algoritmos evolutivos multiobjetivos em problemas de agrupamento de dados foi exemplificada pela descrição de dois algoritmos, MOCK e MOCLE.

No próximo capítulo, a abordagem proposta é apresentada em detalhes, seguida de uma descrição dos conjuntos de dados e das diferentes configurações do novo algoritmo empregados nos experimentos.

4. MCHPF – Multiobjective Clustering with Hierarchical Partition Fusion

A primeira parte deste capítulo apresenta uma descrição detalhada de todos os componentes do método proposto, destacando como eles se integram para solucionar problemas de fusão de agrupamentos de dados. Na segunda parte, são expostas as configurações de instâncias alternativas desenvolvidas para o método proposto.

4.1. Abordagem Proposta

Grosso modo, o método proposto, designado MCHPF (*Multiobjective Clustering with Hierarchical Partition Fusion*), baseia-se em aplicar, conjuntamente, as estratégias de fusão de partições e agrupamento multiobjetivo para evoluir estruturas de agrupamento relativas a um dado conjunto de dados via Programação Genética (PG). Ao final do processo evolutivo, MCHPF retorna um time de fusores diversos contendo informações relevantes e complementares para os especialistas no domínio dos dados.

Mais especificamente, no contexto do MCHPF, adota-se PG para evoluir hierarquias de fusões como programas. Para tanto, inicialmente, geram-se várias partições a partir de diferentes algoritmos de agrupamento de dados. Daí, cria-se uma população aleatória inicial de hierarquias de fusões, aplicando um ou mais operadores de fusão já disponíveis na literatura a essas partições-base. Assim, cada indivíduo da população representa uma nova forma de fundir um subconjunto das partições iniciais.

Em seguida, essa população é evoluída utilizando, como funções de aptidão, índices de validação de partições disponíveis na literatura. Ao final do processo, obtém-se um conjunto de indivíduos representando, cada um, uma hierarquia de fusões atuando sobre uma ou mais das partições originais. O objetivo esperado é o de que as

partições resultantes do processo evolutivo sejam de melhor qualidade que as iniciais, segundo os critérios de qualidade pré-estabelecidos.

A Figura 4.1Figura 4.1 apresenta um fluxograma relativo ao MCHPF. Observe que a população inicial, após a sua geração, é organizada em frontes, com base nos índices de qualidade de partições adotados. Para todos os indivíduos, é calculada a *crowding distance*. Em seguida, a seleção dos indivíduos e a aplicação dos operadores genéticos são realizadas e um conjunto de novas soluções é gerado. Os indivíduos recém-criados e os já existentes na população são reorganizados em frontes e o processo evolutivo é repetido até que uma quantidade de gerações seja alcançada.

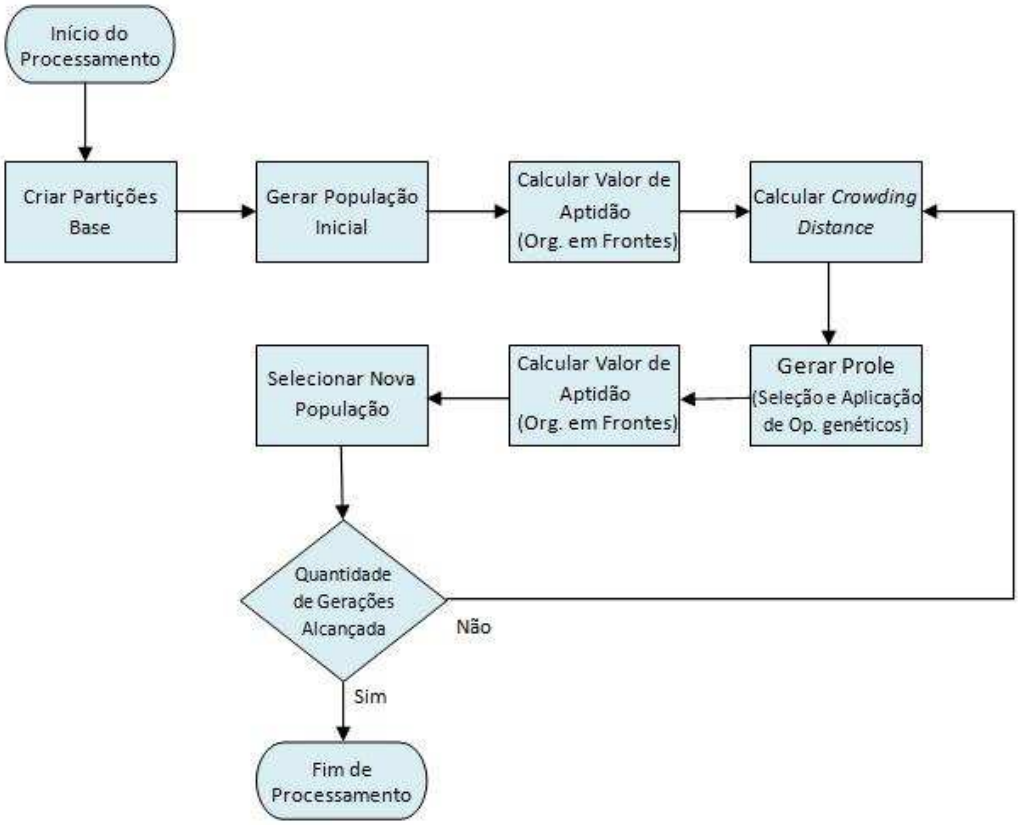


Figura 4.1 – Fluxograma do MCHPF.

Os componentes do MCHPF padrão são descritos nas subseções seguintes. Primeiramente, serão descritas a representação das soluções e a forma de geração da população inicial (4.1.1), seguidas do cálculo de valor de aptidão de cada indivíduo (4.1.2), da forma de seleção utilizada (4.1.3), da estratégia para a geração da nova população (4.1.4) e do critério utilizado para definir o término da execução (4.1.5).

4.1.1. Representação das Soluções e Geração da População Inicial

Como em todo método de fusão de partições, inicialmente é gerado um conjunto de diferentes partições iniciais, $X = \{X_1, X_2, ..., X_n\}$, que serão combinadas. No caso do MCHPF, esse conjunto deve ser obtido a partir de algoritmos de agrupamento de dados que processam diferentes critérios. Assim, as informações estruturais da massa de dados são extraídas de acordo com diferentes perspectivas, contribuindo para aumentar a diversidade da população inicial. Também são aplicadas diferentes configurações de parâmetros (tais como número e densidade dos grupos) para cada algoritmo, gerando-se partições com diferentes níveis de refinamento.

Neste trabalho, para fins de comparação, foram utilizadas as mesmas partições-base adotadas em (FACELI, 2006; FACELI et al., 2008), produzidas com os algoritmos *k*-médias (KM), hierárquico com ligação média (LM), hierárquico com ligação simples (LS) e *Shared Nearest Neighbor* (SNN), descritos no Capítulo 3. Esses algoritmos foram selecionados devido à sua capacidade de encontrar grupos com diversos formatos e por apresentar diferentes graus de separação no espaço de busca. Os algoritmos LS e SNN não têm limitações quanto à forma dos grupos, porém falham quando os grupos ficam muito próximos. Por sua vez, grupos de forma esférica e com pouca separação espacial são facilmente identificados pelos algoritmos KM e LM.

As partições produzidas com esses algoritmos foram representadas como cadeias de rótulos de tamanho igual à quantidade de amostras do conjunto de dados, sendo que cada rótulo indica a qual grupo pertence a amostra correspondente. Por exemplo, para um conjunto de dados com sete amostras, uma possível cadeia de rótulos seria $X_1 = (1, 1, 1, 2, 2, 3, 3)$. Isso indica que a primeira, a segunda e a terceira amostras do conjunto de dados encontram-se no mesmo grupo, rotulado como grupo ‘1’. A quarta e a quinta estão no grupo ‘2’, enquanto a sexta e a sétima ficam no grupo ‘3’.

Na configuração-padrão do MCHPF, as hierarquias de fusão que compõem a população inicial foram criadas aplicando-se, como operador de fusão, o algoritmo HBGF, descrito na Subseção 3.2.3.2, às partições iniciais. Esse algoritmo foi escolhido porque, segundo Topchy et al. (2005), os algoritmos de fusão de partições baseados em grafos/hipergrafos são mais robustos que aqueles baseados em matriz de coassociações.

Além disso, HBGF não utiliza os atributos originais dos objetos, mas apenas o rótulo do grupo ao qual pertence cada objeto, o que ajuda a diminuir a complexidade do MCHPF.

No contexto do MCHPF, HBGF é aplicado para a fusão de duas ou três partições em uma partição-consenso. As partições a serem combinadas podem estar entre as partições-base ou podem ser resultantes de outras fusões. A quantidade de grupos da partição resultante é indicada para o HBGF como a média aritmética das quantidades de grupos das partições originais. Por exemplo, considere duas partições, uma com cinco e outra com sete grupos. Aplicando-se o operador de fusão HBGF a essas partições, é gerada uma partição-consenso contendo seis grupos.

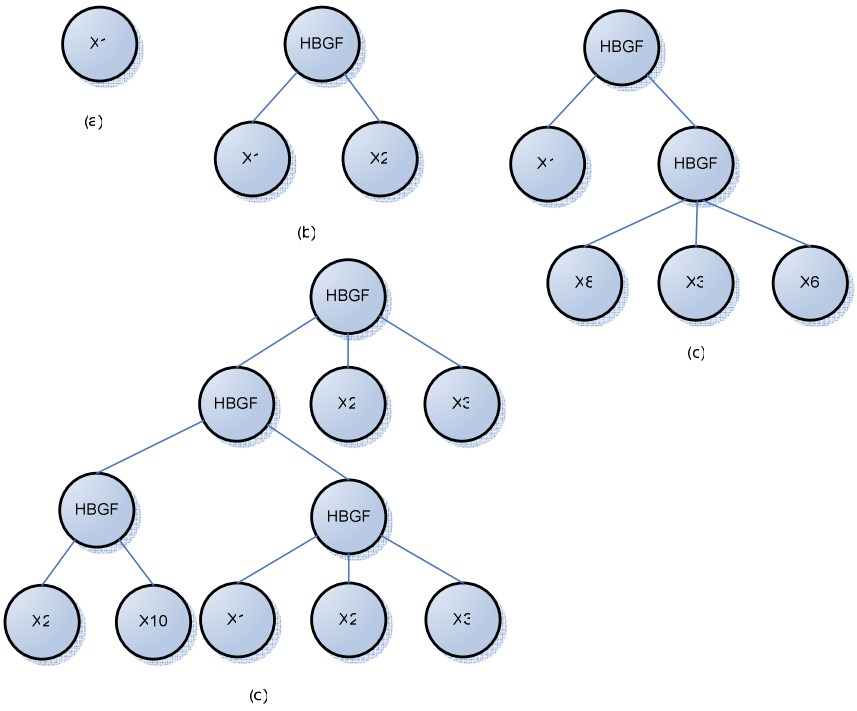


Figura 4.2 – Indivíduos representando possíveis fusões de partições.

Para o uso da PG, uma população de tamanho pré-definido e fixo é criada. Cada indivíduo pertencente a essa população representa uma estrutura hierárquica de fusões. Os nós-folha (símbolos terminais) são representados pelas partições originalmente criadas por meio dos algoritmos de agrupamentos de dados (partições iniciais), enquanto os nós intermediários (símbolos funcionais) são representados pelo operador de fusão utilizado (HBGF). Isso garante a propriedade de fechamento (*closure*), descrita na Seção 2.2 e necessária para a aplicação da técnica de PG. A Figura 4.2 ilustra uma

possível representação de alguns indivíduos pertencentes à população de uma dada geração durante o processo evolucionário.

Deve-se notar que o resultado da aplicação das fusões estabelecidas em uma hierarquia (um indivíduo) é uma partição. Por exemplo, considere a seção (c) da Figura 4.2 e as partições-base:

- $X_1 = (1, 1, 1, 2, 2, 3, 3)$;
- $X_3 = (1, 1, 2, 2, 3, 3, 3)$;
- $X_6 = (2, 2, 2, 3, 3, 1, 1)$; e
- $X_8 = (3, 3, 2, 2, 4, 4, 1)$.

Inicialmente, é realizada a fusão das partições X_8, X_3, X_6, retornando a partição $X_i = (3, 3, 1, 1, 2, 2, 2)$, que, então, participará de uma nova fusão com a partição X_1. O resultado final da "execução do programa" relativo a esse indivíduo será a partição resultante dessa fusão, ou seja, $X_j = (2, 2, 1, 1, 1, 3, 3)$.

O método utilizado para gerar a população inicial indica o formato das primeiras árvores. Na configuração-padrão do MCHPF, foi utilizado o método *full*, que cria árvores balanceadas, conforme descrito na Subseção 2.2.1. Porém, até mesmo populações iniciadas por meio do método *full* podem, ao longo das gerações, apresentar árvores desbalanceadas inseridas por meio dos operadores genéticos. Isso indica que uma partição inicial de boa qualidade segundo os critérios de validação pode entrar na população na forma de uma árvore de apenas um nó.

4.1.2. Cálculo do Valor de Aptidão

No MCHPF-padrão, foram empregados os mesmos índices adotados por MOCK e MOCLE: a variância intra-*cluster* (var) (Equação 3.9) e a conectividade (con) (Equação 3.10). Em ambos os casos, o objetivo é a minimização do índice calculado. Esses índices são medidas de validação interna; ou seja, a partição é avaliada com base apenas nos dados originais. Ressalta-se, assim, que MCHPF é uma abordagem não-supervisionada, já que nenhum conhecimento prévio dos dados é utilizado para indicar que uma partição está mais próxima que outra da solução ideal.

Um importante aspecto da escolha desses dois critérios é o potencial que eles têm para balancear a quantidade dos grupos nas partições. Enquanto o objetivo baseado em compacidade favorece o aumento de grupos, partições com uma quantidade menor de grupos indicam uma melhora no índice conectividade. Assim, como já discutido no âmbito de MOCK e MOCLE, a utilização das duas funções é importante para explorar as diversas regiões do espaço de soluções e não convergir para uma solução trivial (*n* grupos com apenas um objeto, no caso da variância intra-*cluster*, ou apenas um grupo com *n* objetos, no caso da conectividade). O valor de aptidão é calculado conforme descrito no algoritmo NSGA-II, utilizando esses dois índices como funções-objetivo.

Com base no método NSGA-II, descrito na Subseção 2.1.3.1, em cada geração, a população é distribuída em frontes. O primeiro fronte reúne os indivíduos não-dominados por nenhum outro indivíduo, ou seja, não há nenhum outro indivíduo que tenha melhor avaliação em todos os critérios de validação utilizados (var e con). Dessa forma, o primeiro fronte corresponde ao fronte de Pareto.

O segundo fronte abriga os indivíduos que são dominados apenas pelos indivíduos do primeiro fronte, enquanto o terceiro aglomera os indivíduos dominados apenas pelos do primeiro e do segundo fronte, e assim por diante. Cada indivíduo recebe o valor de aptidão correspondente ao fronte ao qual pertence. Por exemplo, os indivíduos do fronte de Pareto recebem valor de aptidão igual a um.

4.1.3. Seleção

Efetuado o cálculo de aptidão, parte da população é selecionada para ser submetida à aplicação dos operadores genéticos. Assim como o cálculo do valor de aptidão, a seleção é feita com base no método NSGA-II, que utiliza o valor de aptidão e uma estimativa da densidade da região em que se encontra o indivíduo no espaço de soluções, a *crowding distance*, descrita na Subseção 2.1.3.1.

O método consiste em um torneio entre dois indivíduos escolhidos aleatoriamente. O indivíduo vencedor é aquele com menor valor de aptidão. Caso os dois tenham o mesmo valor de aptidão, o selecionado será aquele com menor densidade local, ou seja, menor valor correspondente à *crowding distance*.

Esse método ajuda a manter um conjunto diverso de soluções, já que se diminuem as chances de um indivíduo ser selecionado caso a região do espaço de soluções em que ele se encontra apresente alta densidade.

4.1.4. Operadores Genéticos

Novas soluções (partições) são geradas aplicando operadores genéticos aos indivíduos selecionados. Espera-se que os indivíduos gerados nesse processo apresentem a combinação das características que tornaram os pais mais aptos, ou seja, que os indivíduos gerados nesse processo sejam uma evolução dos pais. São utilizados pelo MCHPF os operadores genéticos de reprodução, cruzamento e mutação, descritos na Subseção 3.2.4.

Os operadores de cruzamento e mutação envolvem o processo de seleção aleatória de uma subárvore nos indivíduos originais. No operador de cruzamento, são utilizados dois indivíduos, sendo escolhida aleatoriamente uma subárvore de cada indivíduo. Essas subárvores são trocadas, gerando dois novos indivíduos. Já o operador de mutação, que utiliza apenas um indivíduo-pai, cria um novo indivíduo, trocando uma subárvore selecionada por outra criada de forma aleatória.

Observe que, devido à aleatoriedade da seleção das subárvores, os novos indivíduos podem ser representados por árvores com formatos variados, como árvores não-balanceadas e apresentando diferentes níveis de profundidade.

Por meio desses dois operadores, é gerada uma prole de tamanho igual ao da população inicial. A escolha do operador a ser utilizado para gerar o novo indivíduo obedece a uma taxa de probabilidade, arbitrada no MCHPF padrão em 50% para cada operador.

O operador de reprodução transfere alguns indivíduos para a próxima geração, sem qualquer tipo de alteração. No MCHPF, essa operação se dá pelo processo elitista de seleção dos indivíduos para a nova geração, conforme descrito na seção a seguir.

4.1.5. Elitismo e Critério de Parada

Os indivíduos recém-criados são avaliados conforme os critérios de validação utilizados (var e con) e inseridos na população atual. A população atual, agora contendo

os indivíduos recém-criados e os antigos, é novamente ordenada em frontes e os valores de aptidão de seus indivíduos são atualizados.

A população que passará para a próxima geração é formada por aqueles com melhor índice de aptidão da população atual. Isso significa que os primeiros a entrar na nova população serão os que pertencem ao primeiro fronte; em seguida, virão os do segundo fronte, e assim por diante, até que o tamanho máximo pré-estabelecido da população tenha sido alcançado.

Esse processo de selecionar os indivíduos, aplicar os operadores de cruzamento e mutação, avaliar os recém-criados, re-organizar a população em frontes e eleger a nova população é repetido até que uma quantidade de iterações pré-estabelecida tenha sido completada, representando as gerações do processo evolutivo.

Ao término do processamento, é apresentado como resposta o conjunto de indivíduos pertencentes à primeira fronte da população final. Ou seja, aquelas partições que não foram dominadas por nenhuma outra, em termos dos critérios de validação empregados, durante todo o processamento do método.

4.2. Configurações Alternativas do MCHPF

Além da configuração-padrão do MCHPF (denotada como MCHPF1), foram desenvolvidas outras 10 configurações. Elas se diferenciam em cinco pontos principais: o método utilizado para gerar a população inicial; a quantidade de níveis das árvores iniciais; o tamanho da população; os métodos de fusão de partições utilizados; e as soluções consideradas como resultado final.

Com relação à iniciação da população, as diversas configurações do MCHPF podem utilizar o método *full* ou o método *ramped-half-and-half*, descritos na Subseção 2.2.1. Além desses dois, foi desenvolvido um método alternativo, com base no método *full*. Esse método alternativo foi utilizado em apenas uma configuração, e consiste em gerar 90% da população inicial por meio do método *full*, enquanto os outros 10% são formados pelas partições iniciais com melhor índice de não-dominação em relação às duas funções de avaliação de agrupamento utilizadas, a variância intra-*cluster* (var) e conectividade (con). Isto é, as partições iniciais são avaliadas segundo os critérios de avaliação de agrupamento var e con, e as soluções não-dominadas são inseridas na

população inicial até completar o tamanho predefinido. Neste caso, esses indivíduos correspondem a árvores com apenas um nó.

Configuração MCHPF	Método de Iniciação	Quantidade de Níveis da População Inicial	Tamanho da População	Método de Fusão	Soluções Consideradas
1	*full*	2	50	HBGF	Fronteira de Pareto
2	*full*	2	50	HBGF	População total
3	Alternativo – 90% *full* e 10% melhores indivíduos das partições iniciais	2	50	HBGF	População total
4	*ramped-half-and-half*	5	100	HBGF	Fronteira de Pareto
5	*full*	2	50	MCLA	Fronteira de Pareto
6	*full*	2	50	MCLA	População total
7	*full*	2	50	HBGF e MCLA	Fronteira de Pareto
8	*full*	2	50	HBGF e MCLA	População total
9	*full* – reamostragem	2	50	HBGF	Fronteira de Pareto
10	*ramped-half-and-half*	5	100	MCLA	Fronteira de Pareto
11	*ramped-half-and-half*	5	100	HBGF e MCLA	Fronteira de Pareto

Tabela 4.1 – Configurações alternativas do MCHPF

Todas as configurações, com exceção de MCHPF9, geram a população inicial utilizando um conjunto de partições-base construído por meio dos algoritmos de agrupamento de dados KM, LM, LS e SNN. MCHPF9 utiliza a mesma técnica empregada por Fern e Brodley (2004) em seus experimentos; ou seja, as partições-base são geradas pela aplicação do algoritmo KM sobre conjuntos de dados derivados da base original a partir de reamostragens aleatórias (DUDOIT; FRIDLYAND, 2003).

Para o tamanho da população, as configurações assumem as quantidades de 50 ou 100 indivíduos. Em estudos preliminares, foi observado que populações superiores a 100 indivíduos aumentam substancialmente o tempo computacional, sem propiciar ganhos significativos aos resultados.

Os métodos de fusão utilizados como operadores da PG foram HBGF e MCLA. As configurações variam entre utilizar apenas um desses métodos e utilizar os dois simultaneamente. Quando trabalha com os dois métodos simultaneamente, uma mesma árvore pode apresentar fusões sendo realizadas por meio do método HBGF e do MCLA.

Por fim, as configurações podem divergir com relação à quantidade de soluções a serem consideradas como resultado final, podendo utilizar apenas os indivíduos que se encontram no primeiro fronte de Pareto ou toda a população final. A Tabela 4.1 apresenta um resumo das 11 configurações desenvolvidas para MCHPF.

4.3. Considerações Finais

Neste capítulo, foi apresentada a abordagem MCHPF (de *Multiobjective Clustering with Hierarchical Partition Fusion*). A descrição de todos os seus componentes e métodos foi organizada para indicar a seqüência de passos que o algoritmo deve realizar. Além disso, foi apresentado o fluxograma do método na Figura 4.1. Por fim, diferentes configurações do método proposto foram descritas.

No próximo capítulo, serão discutidos os resultados alcançados pelo MCHPF padrão, em termos de eficiência e eficácia, nos diferentes experimentos realizados sobre as diferentes bases de dados, provendo um contraste com as abordagens correlatas. Nesse âmbito, também são avaliados os eventuais ganhos incorridos com as demais configurações do método proposto.

5. EXPERIMENTOS E RESULTADOS

Na primeira seção deste capítulo, são expostos os aspectos de configuração dos experimentos computacionais realizados, compreendendo uma descrição dos conjuntos de dados utilizados e dos diferentes métodos empregados para fins de comparação. Na segunda seção, a versão padrão do método proposto é avaliada, sob o critério de eficiência, de acordo com o custo computacional incorrido. Ainda nesta seção, é feita uma análise de eficácia da abordagem mediante o cálculo do CR das partições obtidas pela configuração padrão. Na Seção 5.3, realiza-se uma comparação entre o método proposto e outros métodos de agrupamento de dados e fusão de partições. Finalmente, na quarta parte (Seção 5.4), são avaliadas as configurações alternativas do MCHPF que apresentaram os melhores resultados.

5.1. Configuração dos Experimentos

Nos experimentos, foram utilizados os mesmos 10 conjuntos de dados empregados por Faceli (2006) nos experimentos com o MOCLE. Cada um deles possui um conjunto de n^E estruturas conhecidas, $\Pi_E = \{\pi^{E1}, \pi^{E2}, ..., \pi^{En^E}\}$. Quatro desses conjuntos são artificiais, tendo sido construídos para conter estruturas heterogêneas e em diferentes níveis de refinamento. Os conjuntos artificiais são *ds2c2sc13*, *ds3c3sc6*, *ds4c2sc8* e *spiralsquare*. Dois outros conjuntos, *iris* e *glass*, são *benchmarks* comumente utilizados para testar métodos de aprendizado de máquina, encontrando-se disponíveis no repositório de dados da UCI (Newman et al., 1998). Por já terem sido amplamente empregados na literatura, torna-se fácil comparar os resultados obtidos com um novo método qualquer com aqueles produzidos por pesquisas correlatas, como é o caso aqui. Os outros quatro conjuntos contêm dados reais relativos a problemas de bioinformática: *golub*, *proteínas*, *leukemia* e *lung*.

Grande parte desses conjuntos contém pelo menos duas estruturas distintas. Nos conjuntos-*benchmark* e reais, essas estruturas correspondem a diferentes classificações dos dados. Somente dois deles apresentam uma única estrutura conhecida: o conjunto *iris*, que será utilizado por ser o conjunto de dados mais comumente empregado em testes de agrupamento de dados; e o conjunto *lung*, cujos grupos apresentam subdivisões, apesar de não possuir uma segunda estrutura (FACELI, 2006).

A Tabela 5.1 apresenta um resumo das características dos conjuntos de dados utilizados. Nessa tabela, *n* representa o número de objetos no conjunto de dados, *d* é a dimensão ou número de atributos dos dados, n^E representa o número de estruturas conhecidas e K^{Ej} o número de grupos na *j*-ésima estrutura conhecida. Considerando os 10 conjuntos de dados, totalizam-se 22 estruturas conhecidas. Uma maior descrição dos conjuntos de dados trabalhados pode ser obtida no Apêndice II.

Tipo	Conjunto de Dados	*n*	*d*	n^E	K^{E1}	K^{E2}	K^{E3}	K^{E4}
Artificiais	ds2c2sc13	588	2	3	2	5	13	-
	ds3c3sc6	905	2	2	3	6	-	-
	ds4c2sc8	485	2	2	2	8	-	-
	spiralsquare	2000	2	2	2	6	-	-
Benchmarks	glass	214	9	3	2	5	6	-
	iris	150	4	1	3	-	-	-
Reais	golub	72	3571	4	2	3	4	2
	proteinas	698	125	2	4	27	-	-
	leukemia	327	271	2	3	7	-	-
	lung	197	1000	1	4	-	-	-

Tabela 5.1 – Características dos conjuntos de dados

Como já mencionado, as partições iniciais utilizadas foram as mesmas adotadas em (Faceli, 2006), que consistiam em partições com $k \in [K^{min}, K^{max}]$, onde $K^{\min} = \min_{\pi^{Ej} \in \Pi_E} K^{Ej}$ e $K^{\max} = 2 \max_{\pi^{Ej} \in \Pi_E} K^{Ej}$; ou seja, o número de grupos dessas partições varia do menor número de classes nas estruturas conhecidas até duas vezes o maior número de classes nas estruturas. Por exemplo, para o conjunto de dados *golub*, o menor número conhecido de classes que uma estrutura apresenta é dois. Já a estrutura conhecida mais refinada apresenta quatro classes. Sendo assim, foram utilizadas partições que variam de duas até oito classes.

Foi estabelecido o limite de 10 gerações para cada execução do MCHPF, independentemente do conjunto de dados trabalhado. Essa decisão deve-se a experimentos preliminares, nos quais foi observado que um número maior de gerações não produzia resultados muito diferentes. Para o parâmetro v (número de vizinhos mais próximos) utilizado no cálculo do índice conectividade (con), adotou-se um valor diferente para cada conjunto de dados: 5% de *n* (número de amostras do conjunto de dados). A Tabela 5.2 mostra os valores de v, bem como o número de grupos mínimo e máximo ($K^{\min}$, $K^{\max}$) utilizados para gerar as partições iniciais e a quantidade dessas partições iniciais n^I geradas para cada conjunto de dados.

Conjunto de Dados	$K^{\min}$	$K^{\max}$	n^I	v
ds2c2sc13	2	26	114	30
ds3c3sc6	3	12	42	46
ds4c2sc8	2	16	66	25
spiralsquare	2	12	73	100
glass	2	12	48	11
iris	3	6	20	8
golub	2	8	46	4
proteinas	4	54	170	35
leukemia	3	14	38	17
lung	4	8	22	10

Tabela 5.2 – Valores de parâmetros e partições iniciais geradas

Os resultados obtidos pelo MCHPF foram comparados com os resultados publicados por Faceli (2006) para seguintes algoritmos:

- os algoritmos de agrupamento de dados utilizados para gerar as partições iniciais (KM, LM, LS e SNN);
- o método supraconsenso de fusão de partições de Strehl e Ghosh (2002), daqui em diante identificado pelo acrônimo ES, que seleciona a melhor fusão gerada pelas funções-consenso CSPA, HGPA e MCLA;
- o algoritmo de fusão de partições HBGF, de Fern e Brodley (2004);
- o método de agrupamento multiobjetivo MOCK, de Handl e Knowles (2006); e
- o algoritmo multiobjetivo de fusão de partições MOCLE, de Faceli (2006).

Tanto MOCK quanto MOCLE geram um fronte de Pareto, com soluções que correspondem a diferentes compromissos entre os objetivos otimizados (con e var).

Para o parâmetro v, Faceli (2006) utilizou 5% do total de amostras do conjunto de dados, e a quantidade de gerações para esses dois algoritmos foi fixada em 50.

Os métodos ES e HBGF geram uma única partição, com *k* grupos, sendo *k* fornecido pelo usuário. Assim, para gerar um conjunto de soluções, os algoritmos foram executados várias vezes, com os mesmos valores de *k* usados para gerar as partições-base ($k \in [K^{\min}, K^{\max}]$). As mesmas partições-base empregadas no MOCLE foram utilizadas no ES e no HBGF.

Faceli (2006) utilizou o índice de validação externa *Rand* Corrigido (CR), descrito na Seção 3.3, para avaliar a qualidade das soluções.

Para os algoritmos LM, LS e SNN, que são determinísticos, foi calculado o CR entre cada partição π^{Si} do conjunto de soluções $\Pi_S = \{\pi^{S1}, \pi^{S2}, ..., \pi^{Sn^S}\}$ e cada estrutura π^{Ej} do conjunto de estruturas conhecidas $\Pi_E = \{\pi^{E1}, \pi^{E2}, ..., \pi^{En^E}\}$. Para cada estrutura conhecida, foi selecionada a melhor partição do conjunto de soluções em relação a essa estrutura, ou seja, a partição π^{Si} que apresenta maior CR quando comparada com π^{Ej}.

Como os algoritmos MOCK e MOCLE não são determinísticos, foram realizadas 30 execuções para cada conjunto de dados, utilizando-se as mesmas configurações iniciais. Como os métodos ES e HBGF dependem da ordem de apresentação das partições iniciais, foram realizadas 30 execuções, apresentando as partições de forma aleatória. A mesma quantidade de execuções foi realizada para o algoritmo KM, com iniciação aleatória dos centróides.

Para obtenção dos valores do CR desses algoritmos, foi adotado o mesmo procedimento observado para os algoritmos determinísticos. Assim, para cada estrutura conhecida, π^{Ej}, foi selecionada a partição π^{Si} mais próxima de π^{Ej} para cada uma das 30 execuções. Em seguida, para cada π^{Ej}, foram calculados a média e o desvio-padrão do CR das 30 partições π^{Si} selecionadas. Como MCHPF não é determinístico, o mesmo procedimento utilizado nos algoritmos não-determinísticos foi utilizado para calcular o seu CR correspondente.

Para verificar a significância estatística das diferenças de desempenho (CR) entre os algoritmos, foram empregados o teste Friedman e o pós-teste Nemenyi,

indicados para comparar vários algoritmos aplicados em múltiplos conjuntos de dados (HOLLANDER; WOLFE, 1999; DEMŠAR, 2006). Segundo Demšar (2006), o teste Friedman é mais apropriado que o teste ANOVA para analisar algoritmos de aprendizado de máquina.

Para aplicar o teste Friedman, inicialmente os algoritmos são ordenados para cada conjunto de dados, separadamente. O algoritmo que obtiver o melhor desempenho recebe o *rank* um; o que tiver o segundo melhor recebe *rank* dois; e assim por diante. Em caso de empate, a média dos *ranks* é associada a todos os algoritmos que apresentaram igual desempenho.

Dados n^A algoritmos e n^D conjuntos de dados, r_j^i é o *rank* do *j*-ésimo algoritmo no *i*-ésimo conjunto de dados. No teste Friedman, a hipótese nula é a de que todos os algoritmos sejam equivalentes, razão pela qual a média de seus *ranks*, $R_j = \frac{1}{n^D}\sum_{i=i}^{n^D} r_i^j$, deve ser igual. A estatística de Friedman é dada pela Equação 5.1. Como χ_F^2 é indesejavelmente conservadora, Demšar (2006) informa que Iman e Davenport (1980) derivaram uma estatística melhor, dada pela Equação 5.2, que é distribuída de acordo com a distribuição-F com $(n^A - 1)$ e $(n^A - 1)$ $(n^D - 1)$ graus de liberdade.

$$\chi_F^2 = \frac{12n^D}{n^A(n^A+1)}\left[\sum_{j=1}^{n^A} R_j^2 - \frac{n^A(n^A+1)^2}{4}\right]$$

Equação 5.1

$$F_F = \frac{(n^D-1)\chi_F^2}{n^D(n^A-1)-\chi_F^2}$$

Equação 5.2

Se a hipótese nula for rejeitada, Demšar (2006) advoga o uso do pós-teste de Nemenyi para comparar cada algoritmo com todos os demais. Segundo esse teste, dois algoritmos são significativamente diferentes se a média dos *ranks*, R_j, difere de pelo menos uma diferença crítica, *CD*, dada pela Equação 5.3, em que os valores críticos, q_α, baseiam-se na *Studentized range statistic*, divididos por $\sqrt{2}$ (DEMŠAR, 2006).

$$CD = q_\alpha \sqrt{\frac{n^A(n^A+1)}{6n^D}}$$

Equação 5.3

5.2. Eficiência e Análise das Hierarquias Obtidas

Uma medida imprescindível para se avaliar a eficiência do método proposto consiste em se verificar o tempo computacional médio necessário para a sua execução. Dessa forma, foram realizadas 10 execuções do MCHPF com a configuração-padrão descrita na Seção 4.1[1], cujos resultados encontram-se na Tabela 5.3. Nessa tabela, são apresentados o tempo médio de execução do método (em segundos), seu desvio-padrão, além da quantidade de amostras (*n*) e a dimensão (*d*) do conjunto de dados.

Conjunto de Dados	*n*	*d*	Tempo Médio (s)	Desvio-padrão
iris	150	4	16,4316	1,0932
lung	197	1000	28,2877	1,9507
golub	72	3571	29,3906	0,6214
glass	214	9	30,3984	1,0416
leukemia	327	271	44,5398	2,4028
ds4c2sc8	485	2	75,9319	2,4365
ds2c2sc13	588	2	122,5029	5,0152
ds3c3sc6	905	2	207,9509	7,5114
proteínas	698	125	330,6472	20,6861
spiralsquare	2000	2	1.053,5	43,6357

Tabela 5.3 – Tempo médio de execução do MCHPF

Como se pode observar na Tabela 5.3, quanto menor for o número de amostras a serem agrupadas, mais eficiente tende a ser o método proposto. Isso se deve, principalmente, ao elevado custo da fusão de partições com muitas amostras. Na configuração-padrão do MCHPF, a fusão de partições se dá por meio do método HBGF, de Fern e Brodley (2004), descrito na Subseção 3.2.3.2. Outro ponto que interfere na eficiência do método é a quantidade de atributos da base de dados. Isso se deve às técnicas de validação de agrupamentos (variância intra-*cluster* e conectividade) utilizadas para avaliar os indivíduos da população. Elas fazem uso dos atributos dos

[1] Execuções realizadas em um computador com a seguinte configuração: processador T2250 Intel com 1.73GHz, 1 GB de Memória RAM e Windows Vista como sistema operacional.

dados para calcular seus respectivos índices, conforme descrito na Seção 3.3. Assim, quanto maior for a dimensão dos dados, maior será o tempo computacional. Em geral, pode-se afirmar que a versão padrão do MCHPF apresenta uma boa escalabilidade com relação aos fatores *n* e *d*.

Dados	Máx. Níveis	*k*	1 Nível	2 Níveis	3 Níveis	4 Níveis	5 Níveis	6 Níveis
ds2c2sc13	3	2	0,2708	**0,3796**	0,2735	-	-	-
		5	0,6149	**0,7632**	0,6649	-	-	-
		13	0,5568	0,5329	**0,6405**	-	-	-
ds3c3sc6	5	3	0,4275	0,6010	0,5747	0,6313	**0,8058**	-
		6	0,3624	0,4800	0,4865	0,4363	**0,5075**	-
ds4c2sc8	4	2	0,1337	0,1770	**0,1954**	0,1895	-	-
		8	0,5565	0,6892	0,7085	**0,7529**	-	-
spiralsquare	5	2	0,5131	**0,6311**	0,5399	0,5056	0,3191	-
		6	0,5007	0,5355	0,5757	0,5918	**0,5976**	-
glass	5	2	0,2568	0,4020	0,3899	0,3283	**0,4990**	-
		5	0,2045	0,3301	0,3261	0,2844	**0,4146**	-
		6	0,1071	0,1810	0,1903	0,1844	**0,2339**	-
iris	6	3	**0,6930**	0,6455	0,6345	0,6226	0,6326	0,6310
golub	5	2	0,3168	0,4282	0,4713	0,4853	**0,6035**	-
		3	0,3564	0,4811	0,5615	0,5136	**0,6118**	-
		4	0,2723	0,3286	0,3568	0,3873	**0,5101**	-
		2	0,0096	0,0118	**0,0134**	0,0019	0,0085	-
proteinas	4	4	0,1415	0,2266	**0,2511**	0,2090	-	-
		27	0,0740	0,0797	**0,0799**	0,0632	-	-
leukemia	5	3	0,1968	0,2305	**0,2337**	0,2185	0,2299	-
		7	0,5681	0,6362	0,6385	0,5729	**0,6768**	-
lung	5	4	0,3775	0,4419	0,4655	0,3568	**0,4754**	-

Tabela 5.4 – Média de CR dos indivíduos com diferentes níveis de hierarquia

Para cada uma das bases de dados, todos os indivíduos da população final das 30 execuções da PG foram agrupados de acordo com a quantidade de níveis das suas respectivas árvores de fusão. Daí, para cada grupo, calculou-se a média dos valores do índice CR dos indivíduos, comparando as partições produzidas por eles com cada uma das estruturas conhecidas daquela base. Esses resultados estão presentes na Tabela 5.4.

A coluna “Máx. Níveis” dessa tabela registra a quantidade máxima de níveis encontrada nas populações finais das 30 execuções, ao passo que *k* indica a quantidade de grupos em cada uma das estruturas conhecidas da base de dados. Já as colunas que indicam as quantidades de níveis (1 Nível, 2 Níveis, etc.) consignam as médias de CR encontradas para as correspondentes quantidades de níveis.

Por exemplo, para a base *ds3c3sc6*, MCHPF apresentou, na população final, hierarquias com até cinco níveis (coluna “Máx. Níveis”). Assim, a média de CR dos indivíduos com cinco níveis para a estrutura E1 da base *ds3c3sc6* é 0,8058 (coluna “5 Níveis”), enquanto para a estrutura E2 com seis grupos é 0,5075.

Observando-se a Tabela 5.4, é fácil verificar que as médias de CR mais elevadas encontram-se nos indivíduos da população final com dois ou mais níveis de hierarquia, ou seja, que apresentam pelo menos uma fusão de partições. A exceção fica por conta da base *iris*, cujos melhores resultados foram encontrados via indivíduos simples, representando uma das partições-base geradas pelos algoritmos de agrupamento. Esse resultado ratifica o fato de que os métodos de fusão de partições tendem a apresentar ganhos de robustez para tratar bases de dados com estruturas de diferentes formatos quando comparados com os algoritmos de agrupamento simples (STREHL; GHOSH, 2002; FERN; BRODLEY, 2004; FACELI et al., 2008).

Ainda com base na Tabela 5.4, é interessante observar que a maioria das médias de CR mais elevadas (14 estruturas) está associada a indivíduos da população final com maiores quantidades de níveis de hierarquia, ou seja, indivíduos em que resultados de fusões serviram como entrada para outras fusões. Esse fato demonstra experimentalmente que uma hierarquia de fusores, aliada à correta seleção das partições que farão parte das fusões, é capaz de produzir melhores resultados que os algoritmos de agrupamentos de dados individuais.

A Tabela 5.5 e a Tabela 5.6 contêm os valores do CR para os conjuntos de dados *ds2c2sc13*, *ds3c3sc6*, *ds4c2sc8*, *spiralsquare*, *glass*, *iris*, *golub*, *proteínas*, *leukemia* e *lung*. Conforme descrito na Seção 5.1, para os algoritmos LM, LS e SNN, tais valores correspondem à melhor solução, considerando o índice CR, em relação a cada estrutura conhecida. Para as demais técnicas (KM, ES, HBGF, MOCK e MOCLE), os valores

correspondem à média e desvio-padrão (DP) em 30 execuções. Em cada estrutura, destaca-se em negrito o maior valor médio do índice CR.

Dados		MCHPF		LM		LS		SNN		KM	
Base	*k*	M	DP	M	DP	M	DP	M	DP	M	DP
ds2c2sc13	2	**1**	0	**1**	-	**1**	-	**1**	-	**1**	0
	5	**1**	0	**1**	-	**1**	-	**1**	-	0,7890	0,040
	13	0,7777	0,009	0,6170	-	0,8720	-	**1**	-	0,6510	0,008
ds3c3sc6	3	**0,9365**	0,020	0,7890	-	0,5250	-	0,9270	-	0,9000	0
	6	**0,6484**	0,021	0,4901	-	0,2496	-	0,6181	-	0,6014	0,030
ds4c2sc8	2	0,3364	0,037	0,2807	-	0,0974	-	0,1606	-	**0,7943**	0,050
	8	**0,9076**	0,013	0,8310	-	0,0190	-	0,5860	-	0,8160	0,019
spiralsquare	2	**1**	0	**1**	-	**1**	-	**1**	-	**1**	0
	6	0,6667	0	0,6608	-	0,5579	-	**0,9943**	-	0,6633	0,008
glass	2	0,6938	0,024	0,6720	-	0,1710	-	**0,7120**	-	0,6320	0,018
	5	**0,5729**	0,022	0,5599	-	0,1270	-	0,5152	-	0,4752	0,036
	6	**0,2808**	0,017	0,2640	-	0,0400	-	0,2500	-	0,2350	0,015
iris	3	0,7592	0	0,5860	-	0,5640	-	**0,8230**	-	0,7230	0,007
golub	2	**0,8815**	0,082	0,8760	-	0,0780	-	0,8550	-	0,5070	0,211
	3	**0,8773**	0,024	0,7980	-	0,0030	-	0,8550	-	0,5020	0,127
	4	0,6778	0,050	**0,6930**	-	0,1080	-	0,6770	-	0,4020	0,147
	2	0,0491	0,013	0,0570	-	**0,3150**	-	0,1120	-	0,0240	0,015
proteinas	4	**0,3366**	0,012	0,3130	-	0,0660	-	0,3000	-	0,3040	0,023
	27	0,1403	0,005	0,0620	-	0,0070	-	0,0550	-	**0,1460**	0,005
leukemia	3	0,2881	0,008	0,3250	-	0,0200	-	0,0440	-	**0,6770**	0,062
	7	**0,7808**	0,005	0,5430	-	0,0050	-	0,0040	-	0,7480	0,021
lung	4	**0,7928**	0,043	0,5240	-	0,1170	-	0,6450	-	0,3500	0,043

Tabela 5.5 – Média de CR dos algoritmos individuais e do MCHPF

Antes de se analisar a Tabela 5.5 e a Tabela 5.6, foi aplicado um teste de hipóteses para se avaliar a significância estatística das diferenças de desempenho observadas entre os métodos comparados. Primeiramente, foi aplicado o teste de Friedman (HOLLANDER; WOLFE, 1999; DEMŠAR, 2006) para comparar todas as técnicas. Com isso, identificou-se que os métodos podem ser considerados estatisticamente diferentes, a um nível de significância de 0,05.

Aplicando-se o pós-teste Nemenyi (HOLLANDER; WOLFE, 1999; DEMŠAR, 2006), foi possível observar que MCHPF apresentou desempenho significativamente superior ao dos algoritmos LM, LS, KM, ES e HBGF, a um nível de significância 0,05. Com o nível de significância alterado para 0,10, MOCK passa a fazer parte desse grupo.

Na análise que segue, a designação “algoritmos individuais” diz respeito aos algoritmos KM, LM, LS e SNN, enquanto a denominação “técnicas de combinação” relaciona-se aos algoritmos ES, HBGF, MOCK e MOCLE. MOCLE apresenta duas configurações: uma (denominada MOCLE M) utiliza o método de fusão MCLA, de Strehl e Ghosh (2002), e a outra (MOCLE H) utiliza o método HBGF, de Fern e Brodley (2004).

5.3. Comparação do MCHPF com os Algoritmos Individuais

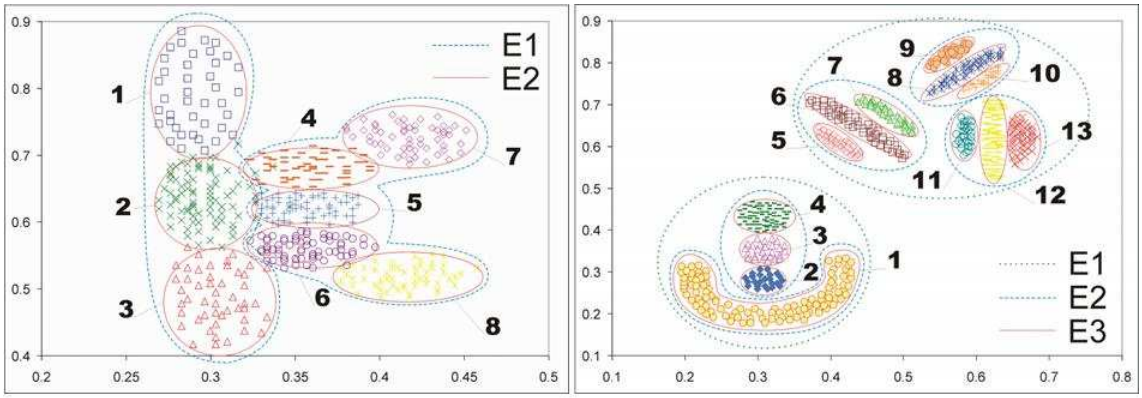
Conforme discutido na Seção 5.1, os algoritmos individuais, ou seja, aqueles utilizados para gerar as partições-base, foram selecionados por apresentarem capacidade de encontrar grupos com diversos formatos e apresentando diferentes graus de separação no espaço de busca. Os algoritmos baseados no conceito de compacidade – caso do KM e do LM – encontram grupos de formas esféricas. Já aqueles baseados no conceito de encadeamento – LS e SNN –, embora não tenham limitações quanto à forma, podem falhar quando há uma pequena separação espacial entre os grupos.

Analisando-se os resultados da Tabela 5.5, esses aspectos tornam-se evidentes, já que os algoritmos individuais tendem a produzir resultados muito bons nos conjuntos de dados que, supostamente, estão em conformidade com as suposições feitas pelos critérios de agrupamento que utilizam. No entanto, eles não mostram um desempenho consistente ao longo dos diferentes conjuntos de dados.

Dados		MCHPF		ES		HBGF		MOCK		MOCLE H		MOCLE M	
Base	*k*	M	DP	M	DP	M	DP	M	DP	M	DP	M	DP
ds2c2sc13	2	**1**	0	0,6910	0,058	0,8874	0,142	**1**	0	**1**	0	**1**	0
	5	**1**	0	0,9920	0,022	0,9111	0	**1**	0	**1**	0	**1**	0
	13	0,7777	0,009	**0,7860**	0,042	0,7703	0,002	0,7080	0,024	0,7770	0	0,7770	0
ds3c3sc6	3	0,9365	0,020	0,8900	0,023	0,8900	0	0,8610	0,072	0,9410	0,010	**0,9800**	0,014
	6	0,6484	0,021	0,5969	0,044	0,5662	0	0,6482	0,059	0,6737	0,017	**0,6740**	0,013
ds4c2sc8	2	0,3364	0,037	0,2958	0,078	**0,3756**	0,056	0,2359	0,010	0,3486	0,002	0,2876	0,047
	8	**0,9076**	0,013	0,8460	0,057	0,8940	0	0,8850	0,022	0,8560	0,025	0,8530	0,025
spiralsquare	2	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0
	6	0,6667	0	0,6381	0,062	**0,7312**	0,022	0,6672	0	0,6665	0	0,6670	0
glass	2	**0,6938**	0,024	0,6340	0,048	0,6605	0	0,5560	0,012	0,6720	0	0,6720	0
	5	**0,5729**	0,022	0,4764	0,039	0,5044	0	0,4405	0,010	0,5599	0	0,5599	0
	6	**0,2808**	0,017	0,2240	0,016	0,2394	0	0,2040	0,013	0,2640	0	0,2640	0
iris	3	0,7592	0	0,7590	0,017	0,7592	0	**0,8160**	0,088	0,7650	0,018	0,7790	0,028
golub	2	**0,8815**	0,082	0,7430	0,175	0,8421	0	0,6840	0,059	0,8760	0	0,8760	0
	3	**0,8773**	0,024	0,6370	0,129	0,7903	0,053	0,7950	0,059	0,8550	0	0,8550	0
	4	0,6778	0,050	0,5890	0,124	0,6506	0,017	0,6220	0,055	**0,6930**	0	**0,6930**	0
	2	0,0491	0,013	0,1100	0,074	0,0067	0,009	0,0570	0,008	**0,3150**	0	**0,3150**	0
proteínas	4	**0,3366**	0,012	0,3150	0,020	0,3311	0	0,2770	0,010	0,3130	0	0,3130	0
	27	0,1403	0,005	0,1330	0,006	0,1230	0,005	0,0710	0,009	**0,1430**	0	**0,1430**	0
leukemia	3	0,2881	0,008	0,3150	0,046	0,3625	0,035	**0,4130**	0	0,2760	0	0,2790	0,003
	7	0,7808	0,005	0,6590	0,053	0,5603	0	**0,7820**	0,014	0,7700	0	0,7740	0,005
lung	4	**0,7928**	0,043	0,4380	0,071	0,5174	0,053	0,7600	0,023	0,6450	0	0,7670	0,079

Tabela 5.6 – Média e desvio-padrão do índice CR obtidos pelas técnicas de combinação e por MCHPF

A Figura 5.1 apresenta dois dos conjuntos de dados artificiais utilizados nos experimentos e descritos na Seção 5.1. A Figura 5.1 (a) apresenta *ds4c2sc8*, que contém duas estruturas conhecidas: E1, com 2 grupos (linha tracejada) e E2, com 8 grupos (linha contínua). Já a Figura 5.1 (b) apresenta o conjunto de dados *ds2c2sc13*, que contém três estruturas conhecidas: E1, com 2 grupos (linha pontilhada), E2, com 5 grupos (linha tracejada), e E3, com 13 grupos (linha contínua).



(a) ds4c2sc8 (b) ds2c2sc13

Figura 5.1 – Conjuntos de dados artificiais *ds4c2sc8* e *ds2c2sc13* (FACELI, 2006)

Segundo a estrutura E1 do conjunto de dados *ds4c2sc8*, apesar de os grupos não possuírem forma esférica, eles estão muito próximos entre si, o que prejudica o desempenho dos algoritmos baseados em conectividade (LS e SNN). Segundo a estrutura E2, além de os grupos possuírem forma esférica, eles não estão tão próximos, razão pela qual os resultados dos algoritmos baseados em compacidade (KM e LM) se sobressaírem em relação aos baseados em conectividade nas duas estruturas conhecidas, ou seja, obtêm maior valor de CR, como pode ser visto na Tabela 5.5 na página 79.

Já para o conjunto de dados *ds2c2sc13*, o mesmo comportamento não acontece. Para E1 e E2, os algoritmos conseguem encontrar a estrutura correta, com exceção do KM para E2, que apresenta um resultado inferior aos demais, devido ao formato de alguns grupos. Porém, para a estrutura E3, a maioria dos grupos tem forma elíptica, o que leva os algoritmos LS e SNN a apresentar melhores resultados, já que para ambos não há restrições quanto ao formato do grupo.

Por outro lado, os resultados obtidos via MCHPF comprovam a robustez esperada da conciliação dos vários algoritmos de agrupamento, por meio da fusão hierárquica das

partições, com a simultânea otimização das duas funções de avaliação de agrupamento consideradas, a variância intra-*cluster* (var) e a conectividade (con).

Enquanto MCHPF pode ser superado pelos algoritmos individuais nos conjuntos de dados que se apresentam em perfeita conformidade com as suposições feitas pelos critérios de agrupamento subjacentes (estrutura E3 do conjunto *ds2c2sc13*, por exemplo), o novo método apresenta resultados melhores ou semelhantes em 13 das 22 estruturas conhecidas dos 10 conjuntos de dados empregados.

Além disso, em nenhuma das estruturas, MCHPF apresentou o pior resultado dentre todos os algoritmos. Isso indica que as partições-base de baixa qualidade ou as fusões inapropriadas são descartadas durante o processo de evolução das hierarquias e que, por meio de novas fusões, foram geradas novas partições de qualidade superior.

5.4. Comparação do MCHPF com as Técnicas de Combinação

Os resultados reunidos na Tabela 5.6 (página 81) indicam que, em tese, quando comparado com as técnicas de combinação, MCHPF mantém a robustez observada na análise da seção anterior. De fato, considerando-se todas as técnicas de combinação, o método proposto alcançou o melhor resultado em 11 das 22 estruturas investigadas nos 10 conjuntos de dados, sendo que em apenas três houve empate. Além disso, em nenhuma estrutura o método proposto apresentou o pior resultado.

Na comparação isolada do MCHPF com os métodos de fusão de partições ES e HBGF, o método proposto apresentou desempenho notável. Na comparação com ES, MCHPF obteve melhor resultado em 17 estruturas, enquanto ES assinalou o melhor resultado em apenas duas, registrando-se empate em três. Em comparação com HBGF, o número de estruturas em que MCHPF foi melhor se repete, contra três do HBGF e dois empates.

Esses resultados fortalecem a indicação anteriormente mencionada de que uma hierarquia de fusores, aliada à correta seleção das partições que farão parte dessas fusões, pode apresentar ganhos significativos em comparação a uma fusão simples de várias partições ao mesmo tempo, ou seja, sem qualquer tipo de seleção.

Na comparação com MOCK, MCHPF também apresentou desempenho bastante significativo, retornando partições mais próximas das reais em 14 estruturas. Já MOCK

aproximou-se mais das reais em cinco estruturas, enquanto em outras três coincidiram os índices CR alcançados pelas duas técnicas.

Em geral, MOCLE apresentou os resultados mais competitivos com MCHPF. Ainda assim, MCHPF foi bem-sucedido na maioria dos casos. Comparado com a configuração do MOCLE que utiliza o método de fusão HBGF (MOCLE H), o método proposto sobressaiu-se em 10 resultados, enquanto apenas sete do MOCLE H se destacaram, havendo cinco empates.

Com a configuração do MOCLE que utiliza o fusor MCLA (MOCLE M), MCHPF mostrou superioridade em 11 estruturas, enquanto apenas seis do MOCLE M conseguiram destacar-se, ocorrendo também cinco empates.

Vale ainda ressaltar que, segundo as configurações do MOCLE H, do MOCLE M e do MOCK descritas na Seção 5.1, os métodos chegam aos resultados apresentados em 50 gerações. Enquanto isso, MCHPF realiza a tarefa em apenas 10 gerações.

5.5. Comparação entre as Configurações do MCHPF

Conforme descrito na Seção 4.2, para análise mais avançada das potencialidades da nova abordagem, foram implementadas 11 configurações diferentes de MCHPF. Nas tabelas do Apêndice I, podem ser visualizados, para cada uma das 22 estruturas das bases utilizadas nos experimentos, os valores de média e desvio-padrão do índice de CR obtidos em 30 execuções das 11 configurações de MCHPF.

A Tabela 5.7 apresenta as cinco configurações que se sobressaíram por apresentarem as maiores quantidades de melhores médias de CR, ou por manterem um padrão elevado dessas médias dentre as 22 estruturas trabalhadas. São elas: MCHPF1 (configuração-padrão), MCHPF2, MCHPF3, MCHPF4 e MCHPF8.

Analisando-se os resultados exibidos por essas cinco configurações, percebe-se que MCHPF4 obteve o melhor desempenho geral, já que prevaleceu em 11 dos casos, incluindo-se as situações de empate. Porém, por trabalhar com um valor elevado para o tamanho da população (100) e com um maior número de níveis nas árvores geradas na população inicial (cinco), MCHPF4 apresenta tempo computacional elevado, chegando, em média, a ser seis vezes mais demorada do que a configuração-padrão.

Dados		MCHPF1		MCHPF2		MCHPF3		MCHPF4		MCHPF8	
Base	*K*	M	DP	M	M	M	DP	M	DP	M	DP
ds2c2sc13	2	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0
	5	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0
	13	0,7777	0,0097	0,7893	0,0297	0,7833	0,0095	0,7784	0,0040	**0,8007**	0,0423
ds3c3sc6	3	0,9365	0,0200	0,9365	0,0200	0,9352	0,0171	**0,9418**	0,0176	0,9404	0,0222
	6	0,6484	0,0212	0,6485	0,0211	0,6455	0,0185	**0,6554**	0,0188	0,6489	0,0229
ds4c2sc8	2	0,3364	0,0371	0,3470	0,0365	0,3415	0,0350	0,3275	0,0356	**0,3526**	0,0324
	8	0,9076	0,0136	0,9080	0,0137	0,9062	0,0139	**0,9164**	0,0102	0,9017	0,0173
spiralsquare	2	**1**	0	**1**	0	**1**	0	**1**	0	**1**	0
	6	0,6667	0,0004	0,6667	0,0004	0,6669	0,0005	**0,6670**	0,0005	0,6666	0,0003
glass	2	0,6938	0,0249	0,7025	0,0228	**0,7051**	0,0200	0,6929	0,0248	0,6902	0,0212
	5	0,5729	0,0226	0,5810	0,0190	**0,5820**	0,0179	0,5677	0,0233	0,5663	0,0191
	6	0,2808	0,0179	0,2859	0,0169	**0,2874**	0,0148	0,2777	0,0185	0,2746	0,0156
iris	3	0,7592	0	**0,7640**	0,0185	0,7620	0,0153	0,7592	0	0,7632	0,0153
golub	2	0,8815	0,0822	0,9136	0,0211	0,9107	0,0202	0,9098	0,0841	**0,9314**	0,0350
	3	0,8773	0,0242	0,8818	0,0247	0,8865	0,0270	0,8706	0,0300	**0,8882**	0,0276
	4	0,6778	0,0506	0,6970	0,0103	0,6970	0,0160	0,7031	0,0432	**0,7161**	0,0199
	2	0,0491	0,0139	0,0503	0,0140	0,0516	0,0144	**0,0612**	0,0257	0,0556	0,0195
proteinas	4	0,3366	0,0123	0,3368	0,0122	0,3324	0,0103	**0,3399**	0,0114	0,3332	0,0119
	27	0,1403	0,0056	0,1408	0,0051	0,1407	0,0069	**0,1426**	0,0047	0,1412	0,0055
leukemia	3	0,2881	0,0088	0,2897	0,0124	0,2945	0,0232	**0,2969**	0,0299	0,2937	0,0239
	7	0,7808	0,0059	**0,7822**	0,0041	0,7821	0,0039	0,7809	0,0054	0,7810	0,0032
lung	4	0,7928	0,0433	**0,7966**	0,0391	0,7742	0,0558	0,7801	0,0624	0,7653	0,0615

Tabela 5.7 – Médias de CR das cinco melhores configurações do MCHPF

Na segunda posição, vem a configuração MCHPF8, que apresenta oito melhores resultados, também incluindo-se os empates. Essa configuração é muita similar à segunda configuração (MCHPF2), diferenciando-se apenas na metodologia de fusão de partições utilizada, já que MCHPF2 utiliza apenas HBGF, enquanto MCHPF8 utiliza os dois métodos (HBGF e MCLA). MCHPF8 apresenta um tempo computacional médio 1,22 vezes maior que o da configuração-padrão.

A segunda configuração ocupa o terceiro posto em termos de eficácia, registrando o mesmo tempo computacional da configuração-padrão. Além disso, ambas apresentam alta similaridade, diferenciando-se apenas na quantidade de soluções consideradas como resultado

final, já que a configuração-padrão utiliza apenas o primeiro fronte de Pareto, enquanto MCHPF2 considera todos os 50 indivíduos da população final.

Ocupando a quarta colocação, MCHPF3 também apresentou basicamente o mesmo tempo computacional médio da configuração-padrão. A única diferença entre MCHPF2 e MCHPF3 é que esta utiliza uma forma alternativa de iniciar a população: 90% da população são produzidos pelo método *full* e os outros 10% correspondem às melhores partições-base, conforme descrito na Seção 4.2.

A quantidade de soluções retornadas é um ponto importante a se considerar. Por exemplo, utilizando a configuração-padrão para a base de dados *iris*, a média do número de indivíduos (hierarquias) ocupando o primeiro fronte de Pareto é de 10,43, número este bem reduzido se comparado com o das segunda, terceira e oitava configurações (50 indivíduos).

Finalmente, outros fatores que podem influenciar bastante na escolha da configuração do MCHPF são a quantidade de indivíduos na população e a quantidade de níveis das árvores da população inicial, já que, apesar de apresentar os melhores resultados dentre todas as configurações, MCHPF4 apresenta tempo computacional bastante elevado.

5.6. Considerações Finais

Na primeira seção do capítulo foram, descritas as configurações dos experimentos computacionais realizados, envolvendo os conjuntos de dados utilizados e os métodos correlatos que serviram de alvo de comparação. Também foram descritas as técnicas estatísticas utilizadas na verificação da significância das diferenças de desempenho entre os algoritmos.

Na Seção 5.2 deste capítulo, a eficiência do método proposto foi avaliada por meio do tempo computacional médio expedido em sua execução para cada uma das bases de dados consideradas. Além disso, foi realizado um comparativo das médias do índice CR incorridas variando-se como parâmetro a quantidade de níveis das árvores da população final da PG.

Após a esta seção, foi conduzida uma comparação em termos de eficácia entre o método proposto e outras técnicas representativas de agrupamento de dados e fusão de partições. Por fim, foram discutidos os resultados obtidos pelas configurações de MCHPF que obtiveram melhor desempenho em termos do índice CR.

No capítulo seguinte, são oferecidas as conclusões, ressaltando-se as contribuições do estudo e indicando-se novos trabalhos que poderão ser desenvolvidos a partir desta dissertação.

6. CONCLUSÃO E TRABALHOS FUTUROS

A principal contribuição deste trabalho está em oferecer uma nova abordagem baseada em Programação Genética para a construção multiobjetiva de estruturas hierárquicas de fusão de partições, estas obtidas a partir de diferentes técnicas de agrupamento de dados.

Mais especificamente, integrando as estratégias de agrupamento multiobjetivo e fusão de partições, o MCHPF realiza automaticamente as etapas mais importantes da análise de agrupamento: aplica vários algoritmos que visualizam diferentes aspectos dos dados; combina e recombina as partições resultantes desses algoritmos; e seleciona aquelas com melhores compromissos de qualidade, consoante diferentes medidas de validação.

As estruturas em árvore, características da Programação Genética, proporcionaram uma novidade no contexto de *clustering ensembles*, a hierarquia de fusores. Essas “fusões de fusões”, aliadas a uma correta seleção de partições, são capazes de promover ganhos significativos em relação a fusões simples de várias partições, sem qualquer tipo de seleção.

Um grupo de 10 bases de dados, cada uma com um conjunto de estruturas conhecidas, totalizando 22 estruturas, foi utilizado nos experimentos. Em geral, a abordagem MCHPF conseguiu produzir soluções de alta qualidade, chegando, em alguns casos, a encontrar a estrutura correta. E, nas situações em que a estrutura correta não foi encontrada, o método proposto foi capaz de encontrar várias partições que representam a massa de dados sob diferentes perspectivas e com variados níveis de refinamento.

Além disso, MCHPF foi comparado com outras abordagens existentes na literatura, sendo que cada uma delas foi executada várias vezes e teve calculadas suas métricas de qualidade. A aplicação de testes estatísticos aos valores dessas métricas mostrou um desempenho satisfatório do método proposto em relação às outras abordagens.

Neste trabalho, foram desenvolvidas e avaliadas 11 configurações diferentes da abordagem proposta. Porém, os algoritmos de agrupamento de dados utilizados na geração das partições-base e as funções-objetivo (medidas de validação interna) adotadas na Programação Genética multiobjetivo foram sempre os mesmos. Isso constitui a primeira possibilidade de trabalho futuro: aumentar o escopo de investigação, considerando outros algoritmos de agrupamento de dados, tais como os mapas auto-organizáveis de Kohonen (XU; WUNSCH, 2005; JIANG; ZHOU, 2004) e outros métodos de cunho bio-inspirado (CASTRO, 2006), juntamente com a otimização de outras medidas de qualidade como funções-objetivo. Nesse contexto, a implementação de uma versão semi-supervisionada do MCHPF (envolvendo algum critério de validação externo) parece ser uma frente também interessante de ser explorada, assim como já foi feito para MOCLE (FACELI, 2006).

Outra importante linha de pesquisa futura diz respeito a investigar uma forma mais eficiente de efetuar as fusões das partições, visto que essa parte do processo constitui uma alta parcela do tempo computacional gasto pelo MCHPF. Já existem alguns trabalhos na literatura nesse sentido, propondo abordagens mais escaláveis de se realizar a fusão de partições em grandes bases de dados (VISWANATH; JAYASURYA, 2006; HOREA et al., 2009).

O emprego do MCHPF no âmbito de outras aplicações computacionais, como aquelas envolvendo segmentação de imagens (JIANG; ZHOU, 2004) e mineração de dados multimídia (GLLAVATA et al., 2006), também é uma linha interessante de ser explorada futuramente.

Finalmente, outra linha de pesquisa futura seria a de contemplar a possibilidade de se incorporar técnicas de seleção/extração de atributos em uma nova configuração do MCHPF. Desse modo, diferentes visões/projeções dos mesmos dados poderiam ser atribuídas aos diferentes algoritmos de agrupamento responsáveis pela geração das partições-base. Nesse contexto, um exemplo de abordagem bem-sucedida para lidar com dados de alta dimensão é aquela proposta por Fern e Brodley (2003), a qual se baseia no conceito de projeção aleatória.

BIBLIOGRAFIA

AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., RAGHAVAN, P. (1998). Automatic subspace clustering on high dimensional data for data mining applications, *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Seattle, Washington, USA, pp. 94–105.

BÄCK, T., HAMMEL, U., e H.-P. SCHWEFEL (1997). Evolutionary computation: Comments on the history and current state, *IEEE Transactions on Evolutionary Computation* 1(1): 3–17.

BARBARA, D. (2000). An introduction to cluster analysis for data mining. http://www-users.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf [Acessado em 10/02/2009].

BERKHIN, P. (2002) Survey of clustering data mining techniques, Technical report, Accrue Software, São Jose, Califórnia.

BHATTACHARJEE, A., RICHARDS, W. G., STAUNTON, J., LI, C., MONTI, S., VASA, P., LADD, C., BEHESHTI, J., BUENO, R., GILLETTE, M., LODA, M., WEBER, G., MARK, E. J., LANDER, E. S., WONG, W., JOHNSON, B. E., GOLUB, T. R., SUGARBAKER, D. J. e MEYERSON, M. (2001). Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma sub-classes. *Proceedings of Natl. Acad. Sci.* USA. 98: 13790-13795.

BLICKLE, T. e THIELE, L. (1995). A comparison of selection schemes used in genetic algorithms, *Evolutionary Computation* 4(4): 361–394.

BOUTIN, F., e HASCOËT, M. (2004). Cluster validity indices for graph partitioning, *Proceedings of the 8th International Conference on Information Visualization (IV'04)*, London, England, UK, pp. 376–381.

CASTRO, L. N. (2006). *Fundamentals of Natural Computing*. Chapman Hall.

COELLO COELLO, C. A. (1999). An updated survey of evolutionary multiobjective optimization techniques : State of the art and future trends, *Proceedings of 1999 Congress on Evolutionary Computation*, Washington D.C., pp. 3–13.

CORNE, D. W., JERRAM, N.R., KNOWLES, J. D., e OATES, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, São Francisco, Califórnia, USA, pp. 283-290.

DEB, K., PRATAP, A., AGARWAL, S., e MEYRIVAN, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation* 6(2): 182–197.

DEMŠAR, J. (2006). Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* 7: 1–30.

DOVAL, D., MANCORIDIS, S. e MITCHELL, B. S. (1999). Automatic clustering of software systems using a genetic algorithm, *Proceedings of the International Conference on Software Tools and Engineering Practice (STEP '99)*, p.73

DUDOIT, S., FRIDLYAND, J. e SPEED, T. P. (2000). Comparison of discrimination methods for the classification of tumors using gene expression data. Technical Report 576, Department of Statistics, UC Berkeley.

DUDOIT, S., e FRIDLYAND, J. (2003). Bagging to improve the accuracy of a clustering procedure, *Bioinformatics* 19: 1090–1099.

ERTÖZ, L., STEINBACH, M., e KUMAR, V. (2002). A new shared nearest neighbor clustering algorithm and its applications, *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications, 2nd SIAM International Conference on Data Mining (SDM'2002)*, pp. 105–115.

ESTIVILL-CASTRO, V. (2002). Why so many clustering algorithms – A position paper. *SIGKDD Explorations* 4(1): 65–75.

ESTER, M., KRIEGEL, H.-P., SANDER, J., e X. XU. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining (KDD-96)*, Portland, Oregon, USA, pp. 226–231.

FACELI, K. (2006) Um framework para análise de agrupamento baseado na combinação multi-objetivo de algoritmos de agrupamento, Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação – ICMC- USP.

FACELI, K., CARVALHO, A.C.P.L.F., SOUTO, M.C.P. (2005). Validação de algoritmos de agrupamento, Relatório técnico 254, Instituto de Ciencias Matemáticas e de Computação – ICMC- USP. ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/rel_tec/RT_254.pdf [acessado em 08/07/2009].

FACELI, K., CARVALHO, A. C. P. L. F. e SOUTO, M. C. P. (2008). Multi-objective clustering ensemble, *International Journal of Hybrid Intelligent Systems* 4(3): pp. 145–156.

FASULO, D. (1999). An analysis of recent work on clustering algorithms, Technical Report, Department of Computer Science and Engineering, University of Washington.

FERN, X. Z., e BRODLEY, C. E. (2003). Random projection for high dimensional data clustering: A cluster ensemble approach, *Proceedings of the Twentieth International Conference on Machine Learning(ICML'2003)*, Washington, DC, USA, pp. 186–193.

FERN, X. Z. e BRODLEY, C. E. (2004). Solving cluster ensemble problems by bipartite graph partitioning, *Proceedings of the Twenty-first International Conference on Machine Learning (ICML'2004)*, New York, NY, USA, pp. 36–43.

FONSECA, C. M. e FLEMING, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation* 3(1): 1–16.

FRED, A. L. N. (2001). Finding consistent cluster in data partitions, *Proceedings of the Second International Workshop on Multiple Classifier Systems (MCS'2001)*, Volume 2096 of *Lecture Notes in Computer Science*, Cambridge, UK, pp. 309–318.

FRED, A. L. N. e JAIN, A. K. (2002). Data clustering using evidence accumulation, *Proceedings of the 16th International Conference on Pattern Recognition*, Lisbon, Portugal, pp. 276–280.

FRED, A. e JAIN A. K. (2003). Robust data clustering, *Proceedings of IEEE computer Society Conference of Computer Vision and Pattern Recognition, (CVPR'2003)*, Vol. 2, Madison – Wisconsin, USA, pp. 128–133.

GHAEMI , R., SULAIMAN, N., IBRAHIM, H. e MUSTAPHA, N. (2009). A survey: Clustering ensembles techniques, *World Academy of Science, Engineering and Technology*, 50: 636–645.

GLLAVATA, J., QELI, E. e FREISLEBEN, B. (2006). Detecting text in videos using fuzzy clustering ensembles, *Proceedings of the Eighth IEEE international Symposium on Multimedia*, IEEE Computer Society, Washington, DC, pp. 283–290.

GOLUB, T., SLONIM, P. T. D. K., HUARD, C., GAASENBEEK, M., MESIROV, J., COLLER, H., LOH, M., DOWNING, J., CALIGIURI, M., BLOOMFIELD, C. e LANDER, E. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring, *Science* 286(5439), 531-537.

HANDL, J. e KNOWLES J. (2005). Exploiting the trade-off – the benefits of multiple objectives in data clustering, *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO'2005*), Volume 3410 of *Lecture Notes in Computer Science*, Guanajuato, Mexico, pp. 547–560.

HAN, J., e KAMBER, M. (2001). Cluster analysis, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 1. ed., Cap. 8, New York, USA, Academic Press.

HANDL, J. e KNOWLES J. (2006). Multiobjective clustering and cluster validation, *Multi-Objective Machine Learning*, Springer Berlin/Heidelberg, USA, pp. 21–47.

HARTIGAN, J. (1975). *Clustering Algorithms*. John Wiley & Sons, New York, NY.

HARTIGAN, J. e WONG, M. (1979). Algorithm AS136: A k-means clustering algorithm, *Applied Statistics* 28: 100–108.

HARTIGAN, J.A. (1985). Statistical theory in clustering, *Journal of Classification* 2(1): 63–76.

HOLLANDER, M. e WOLFE, D. A. (1999). *Nonparametric Statistical Methods*. Wiley-Interscience.

HOREA, P., HALL, L. O., e GOLDGOFA, D. B. (2009). A scalable framework for cluster ensembles, *Pattern Recognition* 42(5): 676–688.

HRUSCHKA, E. R. e EBECKEN, N. F. F. (2003), A Genetic algorithm for cluster analysis, *Intelligent Data Analysis*, Berlin, Germany, pp. 15–25.

HUBERT, L.J. e ARABIE, P. (1985). Comparing partitions, *Journal of Classification* 2: 193–218.

IMAN, R. L. e DAVENPORT J. M. (1980). Approximations of the critical region of the Friedman statistic, *Comm. Stat. Theor. Meth.* A9(6): 571–595.

JAIN, A. e DUBES R. (1988). *Algorithms for Clustering Data.* Prentice Hall.

JAIN, A., M. MURTY, e FLYNN, P. (1999). Data clustering: A review, *ACM computing Surveys* 31(3): 264–323.

JIANG, Y. e ZHOU, Z. (2004). SOM ensemble-based image segmentation, *Neural Processing Letters*20(3): 171–178.

KARYPIS, G., AGGARWAL, R., KUMAR, V., e SHEKHAR, S. (1997). Multilevel hypergraph partitioning: Applications in VLSI domain, *Proceedings of the Design and Automation Conference*, pp. 526–529.

KARYPIS, G. e KUMAR, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM Journal on Scientific Computing* 20(1): 359–392.

KONAK, A., COIT, D. W. e SMITH, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial, *Reliability Engineering and System Safety* 91(9): 992–1007.

KOZA, J. R. (1992). *Genetic Programming: On the Programming of Computers Means of Natural Selection*. MIT Press.

KOZA, J. R. (1995). Survey of genetic algorithms and genetic programming, *Proceedings of the Wescon 95 - Conference Record: Microelectronics, Communications Technology, Producing Quality Products, Mobile and Portable Power, Emerging Technologies*. pp. 589–594.

LUKE, S. e PANAIT, L. (2002). Lexicographic parsimony pressure, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 829–836.

LUO, H., JING, F. e XIE, X. (2006). Combining multiple clusterings using information theory based genetic algorithm, *Proceedings of the IEEE International Conference on Computational Intelligence and Security*, Guangzhou, China, pp. 84–89.

MAIA Jr, L. C. e BIANCHI, R. A. C. (2001). Usando programação genética para evoluir agentes jogadores de futebol de robôs, *Revista Pesquisa e Tecnologia FEI*, São Bernardo do Campo – SP, 21: 3–11.

MONTI, S., TAMAYO, P., MESIROV, J. e GOLUB, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning* 52(1-2): 91-118.

NEWMAN, D., HETTICH, S., BLAKE, C. e MERZ, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html [Acessado em 22/11/2008]. University of California, Irvine, Dept. of Information and Computer Sciences.

NG, R. T. e HAN, J. (1994). Efficient and effective clustering methods for spatial data mining, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Santiago, Chile, pp. 144–155.

NG, R. T., JORDAN, M., e WEISS, Y. (2002). On spectral clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems*, vol. 14.

RODRIGUES, E. L. M. (2002). Evolução de funções em programação genética orientada a gramáticas, Dissertação de Mestrado, Universidade Federal do Paraná, Curitiba.

SRINIVAS, N. e DEB, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation* 2(3): 221–248.

STREHL, A. e GHOSH, J. (2002).Cluster ensembles – A knowledge reuse framework for combining multiple partitions, *Journal of Machine Learning Research* 3: 583–617.

TAN, A. C., GILBERT, D., e DEVILLE, Y. (2003). Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics* 14: 206-217.

TOPCHY, A., JAIN, A. e PUNCH , W. (2004). A mixture model for clustering ensembles, *Proceedings of the SIAM international Conference on Data Mining (SDM'2004)*, Lake Buena Vista, Florida, USA, pp. 331–338.

TOPCHY, A., JAIN, A. e PUNCH, W. (2005). Clustering ensembles: Models of consensus and weak partitions, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(12): 1866–1881.

WILLIS, M.-J., HIDEN, H.G., MARENBACH, P., MCKAY, B. e MONTAGUE, G.A. (1997). Genetic programming: An introduction and survey of applications, *Proceedings of Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA'97*, pp. 314–319, IEEE Press.

VISWANATH, P. e JAYASURYA, K. (2006). A fast and efficient ensemble clustering method, *Proceedings of 18th International Conference on Pattern Recognition (ICPR'06),* vol. 2, pp.720–723.

XU, R. e WUNSCH, D. (2005). Survey of clustering algorithms, *IEEE transactions on Neural Networks* 16(3): 645–678.

YEOH, E.-J., ROSS, M. E., SHURTLEFF, S. A., WILLIAMS, W. K., PATEL, D., MAHFOUZ, R., BEHM, F. G., RAIMONDI, S. C., RELLING, M. V., PATEL, A., CHENG, C., CAMPANA, D., WILKINS, D., ZHOU, X., LI, J., LIU, H., PUI, C.-H., EVANS, W. E., NAEVE, C., WONG, L., e DOWNING, J. R. (2002). Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell* 1(2): 133-143.

ZITZLER, E., LAUMANNS, M. e BLEULER, S. (2004). A tutorial on evolutionary multiobjective optimization. Em: K. S. Xavier Gandibleux, Marc Sevaux and V. T'kindt (Eds.), *Metaheuristics for Multiobjective Optimization*, Volume 535 of *Lecture Notes in Economics and Mathematical Systems*, Berlin, pp. 3–37. Springer-Verlag.

ZITZLER, E., LAUMANNS, M. e THIELE, L. (2001). SPEA2: Improving the performance of the strength pareto evolutionary algorithm, Technical Report 103, Computer Engineering and Communication Networks Lat (TIK), Swiss Federal Institute of Technology (ETH) Zurich.

APÊNDICE I – ÍNDICE CR DAS DIFERENTES VERSÕES DE MCHPF

Na tabela que segue, são apresentados os valores da média (M) e do desvio-padrão (DP) do índice CR obtidos em 30 execuções das 11 configurações do MCHPF para cada uma das 22 estruturas conhecidas das bases de dados utilizadas nos experimentos:

Dados		MCHPF1		MCHPF2		MCHPF3		MCHPF4		MCHPF5	
Base	*k*	M	DP	M	DP	M	DP	M	DP	M	DP
ds2c2sc13	2	1	0	1	0	1	0	1	0	1	0
	5	1	0	1	0	1	0	1	0	1	0
	13	0,7777	0,0097	0,7893	0,0297	0,7833	0,0095	0,7784	0,0040	0,7802	0,0221
ds3c3sc6	3	0,9365	0,0200	0,9365	0,0200	0,9352	0,0171	0,9418	0,0176	0,9297	0,0076
	6	0,6484	0,0212	0,6485	0,0211	0,6455	0,0185	0,6554	0,0188	0,6304	0,0157
ds4c2sc8	2	0,3364	0,0371	0,3470	0,0365	0,3415	0,0350	0,3275	0,0356	0,3410	0,0171
	8	0,9076	0,0136	0,9080	0,0137	0,9062	0,0139	0,9164	0,0102	0,8672	0,0242
spiralsquare	2	1	0	1	0	1	0	1	0	1	0
	6	0,6667	0,0004	0,6667	0,0004	0,6669	0,0005	0,6670	0,0005	0,6663	0,0010
glass	2	0,6938	0,0249	0,7025	0,0228	0,7051	0,0200	0,6929	0,0248	0,6684	0,0139
	5	0,5729	0,0226	0,5810	0,0190	0,5820	0,0179	0,5677	0,0233	0,5463	0,0155
	6	0,2808	0,0179	0,2859	0,0169	0,2874	0,0148	0,2777	0,0185	0,2577	0,0075
iris	3	0,7592	0	0,7640	0,0185	0,7620	0,0153	0,7592	0	0,7592	0
golub	2	0,8815	0,0822	0,9136	0,0211	0,9107	0,0202	0,9098	0,0841	0,8979	0,0670
	3	0,8773	0,0242	0,8818	0,0247	0,8865	0,0270	0,8706	0,0300	0,8882	0,0372
	4	0,6778	0,0506	0,6970	0,0103	0,6970	0,0160	0,7031	0,0432	0,6934	0,0370
	2	0,0491	0,0139	0,0503	0,0140	0,0516	0,0144	0,0612	0,0257	0,0513	0,0075
proteinas	4	0,3366	0,0123	0,3368	0,0122	0,3324	0,0103	0,3399	0,0114	0,3151	0,0133
	27	0,1403	0,0056	0,1408	0,0051	0,1407	0,0069	0,1426	0,0047	0,1407	0,0053
leukemia	3	0,2881	0,0088	0,2897	0,0124	0,2945	0,0232	0,2969	0,0299	0,2934	0,0226
	7	0,7808	0,0059	0,7822	0,0041	0,7821	0,0039	0,7809	0,0054	0,7783	0,0090
lung	4	0,7928	0,0433	0,7966	0,0391	0,7742	0,0558	0,7801	0,0624	0,7343	0,0661

Dados		MCHPF6		MCHPF7		MCHPF8		MCHPF9		MCHPF10		MCHPF11	
Base	*k*	M	DP	M	DP	M	DP	M	DP	M	DP	M	DP
ds2c2sc13	2	1	0	1	0	1	0	1	0	1	0	1	0
	5	1	0	1	0	1	0	0,9221	0,0417	1	0	1	0
	13	0,7975	0,0364	0,7816	0,0301	0,8007	0,0423	0,6459	0,0151	0,7802	0,0091	0,7767	0,0061
ds3c3sc6	3	0,9299	0,0076	0,9404	0,0222	0,9404	0,0222	0,9603	0,0222	0,9295	0,0063	0,9331	0,0134
	6	0,6310	0,0153	0,6486	0,0231	0,6489	0,0229	0,6590	0,0140	0,6274	0,0152	0,6426	0,0201
ds4c2sc8	2	0,3454	0,0179	0,3467	0,0271	0,3526	0,0324	0,2711	0,0556	0,3268	0,0495	0,3404	0,0172
	8	0,8700	0,0245	0,8989	0,0201	0,9017	0,0173	0,8892	0,0146	0,8694	0,0282	0,9089	0,0154
spiralsquare	2	1	0	1	0	1	0	1	0	1	0	1	0
	6	0,6663	0,0010	0,6666	0,0003	0,6666	0,0003	0,6668	0,0003	0,6665	0	0,6667	0,0003
glass	2	0,6740	0,0129	0,6820	0,0202	0,6902	0,0212	0,6772	0,0240	0,6690	0,0112	0,6846	0,0225
	5	0,5504	0,0101	0,5616	0,0185	0,5663	0,0191	0,5402	0,0203	0,5486	0,0082	0,5624	0,0225
	6	0,2602	0,0067	0,2708	0,0155	0,2746	0,0156	0,2626	0,0136	0,2582	0,0076	0,2714	0,0167
iris	3	0,7972	0,0348	0,7592	0	0,7632	0,0153	0,7557	0,0145	0,7612	0,0110	0,7592	0
golub	2	0,9210	0,0344	0,8980	0,0623	0,9314	0,0350	0,7382	0,1598	0,9106	0,0949	0,8870	0,1019
	3	0,8937	0,0330	0,8761	0,0450	0,8882	0,0276	0,8135	0,0964	0,8946	0,0371	0,9014	0,0344
	4	0,7066	0,0244	0,6974	0,0334	0,7161	0,0199	0,5606	0,1270	0,7077	0,0464	0,6831	0,0610
	2	0,0533	0,0079	0,0507	0,0156	0,0556	0,0195	0,0277	0,0134	0,0600	0,0394	0,0566	0,0245
proteinas	4	0,3161	0,0137	0,3295	0,0141	0,3332	0,0119	0,3281	0,0097	0,3170	0,0108	0,3274	0,0078
	27	0,1415	0,0053	0,1404	0,0069	0,1412	0,0055	0,1517	0,0079	0,1419	0,0038	0,1440	0,0046
leukemia	3	0,3041	0,0379	0,2934	0,0238	0,2937	0,0239	0,2803	0,0093	0,2976	0,0238	0,2909	0,0095
	7	0,7795	0,0069	0,7806	0,0028	0,7810	0,0032	0,7738	0,0123	0,7798	0,0027	0,7802	0,0043
lung	4	0,7483	0,0640	0,7638	0,0618	0,7653	0,0615	0,5045	0,0340	0,7058	0,0698	0,7139	0,0730

APÊNDICE II – CONJUNTO DE DADOS

O conjunto de dados *ds2c2sc13* foi especialmente projetado para conter três estruturas diferentes: E1, E2 e E3. A Figura II. 1 (FACELI, 2006) mostra a E1 como sendo a estrutura mais geral, contendo dois grupos. A estrutura E2 é um refinamento de E1, contendo cinco grupos e E3 é um refinamento de E2, com 13 grupos. Os 588 objetos desse conjunto de dados estão divididos nos grupos conforme observado na Tabela II. 1. Na estrutura E1, os grupos têm formato esférico e estão bem separados uns dos outros, o que facilita a tarefa de agrupamento. Já nas estruturas E2 e E3, os grupos são bastante heterogêneos, sendo que os grupos esféricos de E2 são vistos como três grupos alongados de E3.

Estrutura	Número de objetos nos grupos												
	1	2	3	4	5	6	7	8	9	10	11	12	13
E3	108	24	27	73	32	49	33	40	28	24	32	71	47
E2	108	124			114			92			150		
E1	232				356								

Tabela II. 1 Tamanho dos grupos - ds2c2sc13

O conjunto de dados *ds3c3sc6*, Figura II. 1 (FACELI, 2006), contém duas estruturas: E1, com três grupos e E2, com seis grupos, um refinamento de E1. Os grupos têm formatos e tamanhos variados e, nas duas estruturas, eles não estão bem separados uns dos outros. Os 905 objetos desse conjunto de dados estão divididos conforme a Tabela II. 2.

Estrutura	Número de objetos nos grupos					
	1	2	3	4	5	6
E2	79	197	287	56	135	151
E1	276		287	342		

Tabela II. 2 Tamanho dos grupos - ds3c3sc6

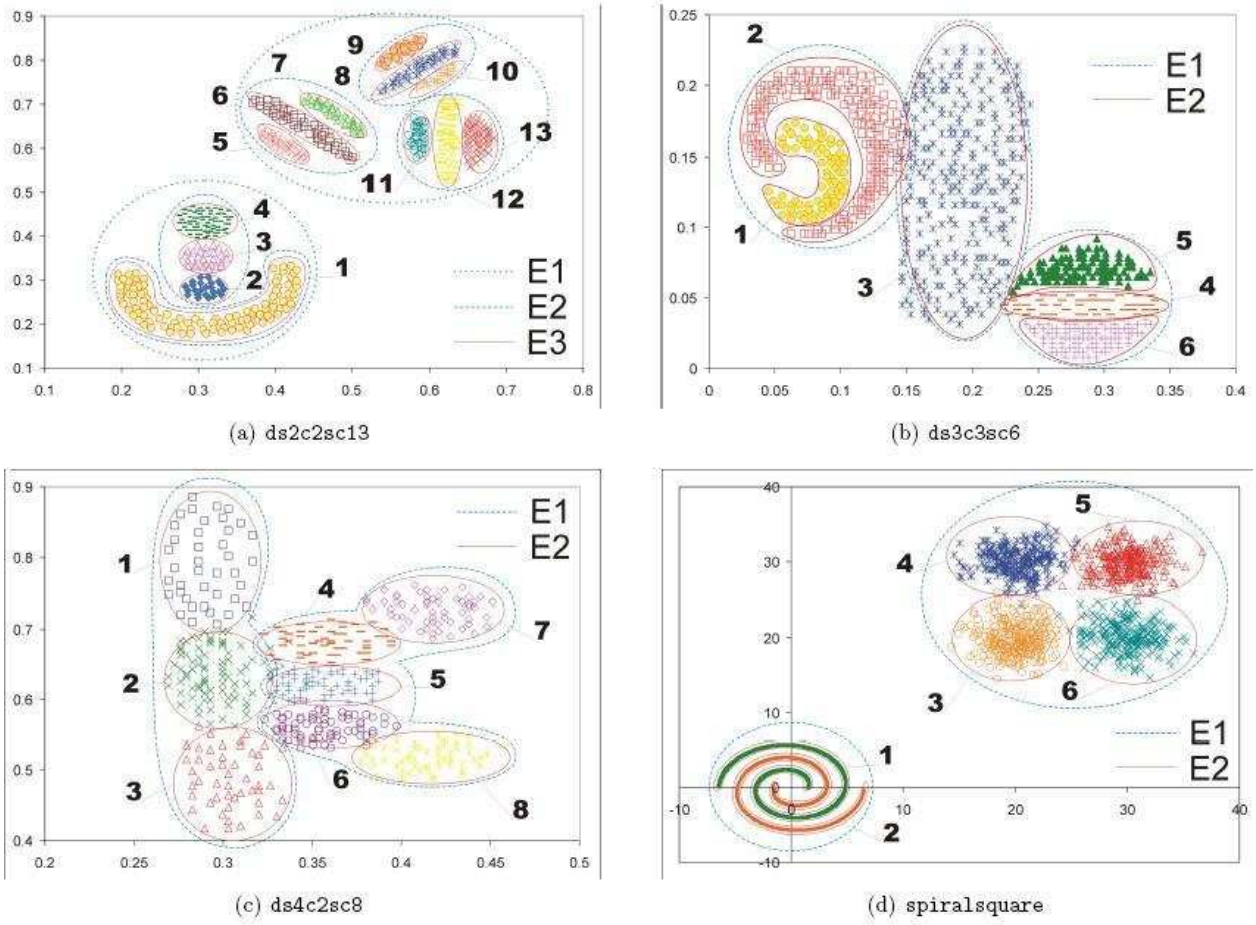


Figura II. 1 Conjunto de dados artificiais

O conjunto de dados *ds4c2sc8*, representado na Figura II. 1 (FACELI, 2006), também contém duas estruturas conhecidas, E1 e E2. A estrutura E2, com oito grupos, é o refinamento da E1, com dois grupos. As duas estruturas apresentam grupos de formas e tamanhos variados e que não apresentam grande separação espacial. Os 485 objetos desse conjunto de dados estão divididos conforme a Tabela II. 3.

Estrutura	Número de objetos nos grupos							
	1	2	3	4	5	6	7	8
E2	40	82	52	53	69	70	62	57
E1	174			311				

Tabela II. 3 Tamanho dos grupos - ds4c2sc8

O conjunto de dados *spiralsquare* foi construído conforme conjunto de dados descrito em HANDL e KNOWLES (2004). Esse conjunto de dados contém duas estruturas: E1, com dois grupos, e E2, um refinamento de E1, com 6 grupos, como pode ser visto na Figura II. 1

(FACELI, 2006). A estrutura E1 contém grupos esféricos e eles estão bem separados uns dos outros. Na estrutura E2, um dos grupos da estrutura E1 foi dividido em dois grupos em espiral e o outro em quatro novos grupos esféricos e com baixa separação uns dos outros. Os 2000 objetos desse conjunto de dados estão divididos conforme a Tabela II. 4.

Estrutura	Número de objetos nos grupos					
	1	2	3	4	5	6
E2	500	500	250	250	250	250
E1	1000			1000		

Tabela II. 4 Tamanho dos grupos - spiralsquare

O conjunto de dados *glass* refere-se à classificação de vidros e foi obtido do repositório de dados UCI (Newman et al., 1998). Ele contém três estruturas conhecidas, E1, E2, E3. A estrutura E3 é a mais refinada, contendo seis grupos: BWFP – vidro processado de janela de edifício (*building windows float processed*), BWNFP – vidro não-processado de janela de edifício (*building windows non float processed*), VWFP – vidro processado de janela de carro (*vehicle windows float processed*), C – vidro de recipientes (*containers*), T – vidro de louças (*tableware*) e vidro de farol (*headlamp*). A estrutura E2, um pouco menos refinada, contém cinco grupos, que se distinguem entre vidro de edifício (BWFP e BWNFP), vidro de carro (VWFP), recipiente (C), louça (T) e farol (H). Por fim, a estrutura E1, contendo dois grupos, agrupa os objetos em vidro de janela (BWFP, BWNFP e VWFP) e vidro que não é de janela (C, T e H). Os 214 objetos desse conjunto de dados estão divididos conforme a Tabela II. 5. Os atributos correspondem ao índice de refração e à porcentagem de diversos óxidos no vidro. Como os atributos correspondem a medidas em unidades diferentes, eles foram normalizados para o intervalo [0, 1] (FACELI, 2006).

Estrutura	Número de objetos nos grupos					
	BWFP	BWNFP	VWFP	C	T	H
E3	70	76	17	13	9	29
E2	146		17	13	9	29
E2	163			51		

Tabela II. 5 Tamanho dos grupos - glass

O conjunto de dados *iris*, também obtido do repositório de dados UCI (NEWMAN et al., 1998), contém apenas uma estrutura conhecida com três grupos que correspondem a três

espécies da planta iris: *Iris Setosa*, *Iris Versicolor* e *Iris Verginica*. Cada grupo contém 50 objetos. Os atributos correspondem ao comprimento e largura das pétalas e sépalas, em centímetros.

O conjunto de dados *golub* representam dados de expressão gênica de pacientes com leucemia aguda (GOLUB et al., 1999). Para esses dados foram considerados quatro estruturas conhecidas. As estruturas de dados E1, com dois grupos, e E2, com três, constituem as duas principais estruturas. A E1 representa a divisão das amostras em leucemia aguda linfoblástica, ou ALL (*Acute Lymphoblastic Leukemia*) e leucemia mielóide aguda, ou AML (*Acute Myeloid Leukemia*). A E2, um refinamento da E1, subdivide o grupo ALL em dois: um com as amostras de células de linhagem T (T-ALL) e outro com as amostras de células de linhagem B (B-ALL). Na estrutura E3, com quatro grupos, os objetos estão agrupados pela instituição de origem das amostras: DFCI (*Dana-Farber Cancer Institute*), CALGB (*Cancer and Leukemia Group* B), SJCRH (*Jude Children´s Research Hospital*) e CCG (*Children´s Cancer Group*). A estrutura E4, dois grupos, indica se as amostras são provenientes de medula óssea, BM (*Bone Marrow*), ou de sangue periférico, PB (*Peripheral Blood*). Os 72 objetos desse conjunto de dados estão divididos conforme a Tabela II. 6. Para essa base foi aplicado o seguinte pré-processamento, como em (DUDOIT et al., 2000): todos os valores menores que 100 foram convertidos para 100 e todos os valores maiores que 16000 foram convertidos para 16000. Dos 7129 genes originais, foram selecionados apenas os genes que possuíam um valor de expressão tal que *minimo/maximo >5* e (*maximo – mínimo*) > 500, o que resulta em um total de 3571 genes. Em seguida, foi aplicado o logaritmo na base 10.

Estrutura	Número de objetos nos grupos			
	B-ALL	T-ALL	AML	
E2	38	9	25	
E1	47		25	
Estrutura	Número de objetos nos grupos			
	DFCI	CALGB	SJCRH	CCG
E3	44	15	8	5
Estrutura	Número de objetos nos grupos			
	BM		PB	
E4	62		10	

Tabela II. 6 Tamanho dos grupos - golub

O conjunto de dados *proteinas* se refere a dobras de proteínas (TAN et al., 2003), foram consideradas duas estruturas conhecidas: E1, com quatro grupos, e E2, um refinamento de E1, com 27 grupos. Os atributos correspondem a características extraídas a partir das seqüências das proteínas. Os 698 objetos desse conjunto de dados estão divididos conforme a Tabela II. 7.

Estrutura	Número de objetos nos grupos														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E2	19	16	32	15	18	16	74	21	29	13	16	32	12	13	16
E1	116						226								
Estrutura	Número de objetos nos grupos														
	16	17	18	19	20	21	22	23	24	25		26		27	
E2	77	23	24	40	22	17	24	18	15	15		40		41	
E1	260									96					

Tabela II. 7 Tamanho dos grupos - proteinas

O conjunto de dados *leukemia* (YEOH et al., 2002) contém dados de expressão gênica obtidos com *microarrays*. Os dados se referem a subtipos de leucemia acuda linfoblástica pediátrica e estão divididos em seis grupos diagnosticáveis (BCR-ABL, E2A-PBX1, *Hyperdiploid* >50, MLL, TEL-AML1 e T-ALL) e outro grupo que contém amostras que não se enquadram em nenhum dos grupos anteriores (OTHERS). Outra estrutura também foi considerada em que os dados são divididos em três grupos mais gerais: as leucemias linhagem B ou B-ALL, as de linhagem T ou T-ALL e as amostras rotuladas como OTHERS. Os 327 objetos desse conjunto de dados estão divididos conforme a Tabela II. 8.

Estrutura	Número de objetos nos grupos						
	BCR	E2A	Hyperdiploid>50	MLL	TEL	T-ALL	OTHERS
E2	15	27	64	20	79	43	79
E1	205					43	79

Tabela II. 8 Tamanho dos grupos - leukemia

O conjunto de dados *lung*, originalmente empregado em (Bhattacharjee et al. 2001), contém apenas uma estrutura com 4 grupos. Ela consiste de amostras relacionadas a câncer de pulmão, englobando amostras de tecido normal (NL) e três tipos de câncer de pulmão:

adenocarcinomas (AD), carcinomas de células escamosas (*squamous cell carcinomas*) (SQ) e 20 carcinóides (COID). Os 197 objetos desse conjunto de dados estão divididos conforme a Tabela II. 9. Para esse conjunto de dados, foi empregada a mesma versão utilizada por Monti et al. (2003).

Estrutura	Número de objetos nos grupos			
	AD	SQ	COID	NL
E1	139	21	20	17

Tabela II. 9 Tamanho dos grupos - lung

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo