Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Estatística

Implementação de um algoritmo evolucionário multiobjetivo para clustering de dados complexos

Mateus Carbone Ananias

Orientador: Dr. André Luiz Fernandes Cançado

Brasília, 25 de junho de 2019

Implementação de um algoritmo evolucionário multiobjetivo para clustering de dados complexos

Mateus Carbone Ananias

Dissertação apresentada ao Departamento de Estatística do Instituto de Ciências Exatas da Universidadede de Brasília como parte dos requisitos necessários à obtenção do título de Mestre em Estatística.

Brasília, 25 de junho de 2019

Implementação de um algoritmo evolucionário multiobjetivo para clustering de dados complexos

Dissertação elaborada pelo aluno Mateus Carbone Ananias submetida ao Programa de Pós-graduação em Estatística do Departamento de Estatística da Universidade de Brasília, como parte dos requisitos da etapa de qualificação para obtenção do grau de Mestre em Estatística. Texto aprovado por:

Comissão Julgadora:

- Prof. Dr. André Luiz Fernandes Cançado - orientador/presidente
- Prof. Dr. Luiz Henrique Duczmal - EST/UFMG
- Prof. Dr. Pedro Henrique Melo de Albuquerque - ADM/UnB
- Prof. Dr. Helton Saulo Bezerra dos Santos - EST/UnB (suplente)

Dedico este trabalho à minha esposa Claudia Edith e filhos Arhur e Giovanni, pela paciência, apoio e carinho durante a elaboração dessa dissertação.

Agradecimentos

Ao meu orientador Dr. André Luiz Fernandes Cançado, por todo o tempo e paciência que dedicou a me ajudar durante o processo de realização deste trabalho.

A minha família pela ajuda, apoio e força a mim dedicados.

Aos colegas pela amizade e suporte durante todo o mestrado.

E a todos que contribuíram para a realização deste trabalho, seja de forma direta ou indireta, fica registrado aqui, o meu muito obrigado!

Resumo

Devido ao grande volume e variedade de dados existentes atualmente, clusterizar dados tem sido um grande desafio. Este trabalho teve como objetivo implementar o MOCK (Multiobjective Clustering with Automatic k-determination) em linguagem R, estudar possíveis melhorias e aplicação em dados reais. Este algoritmo funciona através de um algoritmo genético multiobjetivo chamado de PESA-II, além de ter como funcionalidade a seleção automatizada de grupos. Os resultados apresentados por Handl and Knowles (2007) foram reproduzidos com sucesso para todos os métodos utilizados. Comparou-se o MOCK com outras técnicas de agrupamento em uma aplicação com dados reais. Estes dados constituem-se de dados genéticos com alta dimensionalidade. Mostrou-se que o MOCK conseguiu identificar os grupos desse conjunto de dados de maneira mais eficiente que os outros métodos. Conclui-se que o MOCK é uma ferramenta bastante eficaz para agrupar dados em uma grande diversidade de situações, competindo frente-a-frente com diversos métodos de agrupamento.

Palavras Chave: Agrupamento de dados, MOCK, PESA-II, Algoritmos genéticos, Algoritmos multiobjetivo, seleção automatizada de grupos, K-means, algoritmos hierárquicos.

Abstract

Because of the large volume and variety of data currently available, clustering data has been a major challenge. This work aimed to implement MOCK (Multiobjective Clustering with Automatic k-determination) in R, to study possible improvements and application in real data. This algorithm works through a multiobjective genetic algorithm called PESA-II, besides having as a function the automated selection of groups. The results presented by Handl and Knowles (2007) have been successfully reproduced for all methods used. MOCK was compared with other grouping techniques in an application with real data, this data consisted of genetic data with high dimensionality. It was shown that MOCK was able to identify the groups in this data set more efficiently than the other methods. It is concluded that MOCK is a very efficient tool for grouping data in a great diversity of situations, competing face-to-face with several grouping methods.

Keywords: Data clustering, MOCK, PESA-II, Genetic Algorithms, Multiobjective Algorithms, Automated Group Selection, K-means, Hierarchical Algorithms.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Imagine a seguinte situação: deseja-se particionar um conjunto formado por laranjas e maçãs. A princípio, observa-se cor, tamanho, formato e outras características intrínsecas dessas frutas. Inicialmente essas duas frutas parecem bastante diferentes entre si. Porém, imagine agora que além de laranjas e maçãs, há também batatas nesse conjunto. Neste caso é possível que a partir deste ponto, laranjas e maçãs já não sejam tão diferentes entre si, comparativamente às batatas, e possam compor o mesmo grupo (o grupo das frutas), enquanto um outro grupo poderia ser formado somente com as batatas. Partido deste simples exemplo, percebemos que processos de particionamento de conjuntos em grupos coesos podem, à medida que se adicionam novas informações, se tornar mais e mais complexos.

Considere agora que ao invés de frutas, deseja-se separar milhares de clientes de um supermercado quanto à preferência de compra, pois a nova campanha de marketing precisa de um direcionamento sobre um determinado produto. Neste caso, não se sabe a qual grupo cada um dos clientes pertence, nem mesmo se existem perfis de clientes tão diferentes que justifiquem o particionamento do conjunto de clientes. Ademais, têm-se apenas dados indiretos como dados sobre renda, valores comprados, quantidade de produtos comprados, produtos escolhidos, etc. Analisar cada indivíduo e agrupá-los pode ser muito difícil, e provavelmente não será possível para um ser humano analisar todas as possíveis relações entre as variáveis chegando a um resultado satisfatório. Dentro deste contexto, faz-se necessário o uso de técnicas com critérios matemáticos e computação intensiva chamadas de agrupamento de dados, ou *data clustering*.

Sendo muito importante na análise estatística ou mais comumente associado a aprendizado de máquina, técnicas de agrupamento de dados têm sido extensivamente estudadas.

Devido a crescente quantidade e complexidade dos dados disponibilizados, tem-se gerado a necessidade da criação de métodos que possam lidar com diversas característica dos mais variados conjuntos de dados.

Em linhas gerais, agrupar dados pode ser definido como uma tentativa de se particionar os elementos em grupos de forma que elementos dentro de um grupo tenham mais semelhança entre si do que elementos em grupos diferentes. O problema de particionamento de conjuntos é classificado como aprendizado não supervisionado, pois deve-se obter os grupos através de estruturas intrínsecas dos dados sem conhecimento prévio de a qual grupo cada elemento pertence, isto é, não está disponível um conjunto de treinamento do modelo.

Técnicas de agrupamento de dados podem ser aplicadas nas mais diversas áreas. Desde dados genéticos, passando por dados de mídia, como vídeos e fotos, até as técnicas mais avançadas de mineração de dados e aprendizado de máquina. Por via de regra, os dados mais comuns a serem agrupados são numéricos, mas também podem ser categóricos, com estruturas temporais e espaciais, entre outras formas de estrutura.

Existem diversos tipos de algoritmos para agrupar dados. Entre eles estão os métodos hierárquicos, por aglomeração e espaço de busca, para citar alguns exemplos. Nos próximos capítulos estes métodos serão melhor detalhados. De acordo com Jain (2010), trabalhos sobre técnicas de agrupamento podem ser encontradas desde a década de 1950, e desde então elas têm sido desenvolvidas de maneira multidisciplinar. Neste contexto, é possível afirmar que o K-means é o método mais conhecido e citado. Proposto há mais de 50 anos essa técnica tem sido utilizada como referência para desenvolvimento e comparação de vários outros métodos ao longo dos anos, e ainda se mostra muito eficiente em uma grande variedade de cenários.

De um ponto de vista evolutivo, a necessidade de reconhecer padrões parte da necessidade de sobrevivência sendo inerente à maioria do seres vivos. Segundo Pi et al. (2008) a grande facilidade de reconhecimento de padrões está atrelada diretamente à inteligência humana. Posteriormente, o reconhecimento de padrões em conjunto de dados está ligado a necessidades de gerar hipóteses, identificar características não conhecidas previamente, detectar anomalias, organizar dados, identificar similaridade entre objetos ou simplesmente conhecer mais sobre os dados em processos exploratórios.

Essa tarefa nem sempre é fácil pois os grupos podem diferir em tamanho, formato e densidade em dados com diversas dimensões. Mesmo em casos mais simples com a existência

de grupos não ambíguos, alguns algoritmos podem falhar, levando a conclusões equivocadas, causadas por diversos motivos. Isso evidencia que no processo de agrupamento de dados existe uma certa subjetividade, pois os algoritmos tendem a estimar a qualidade dos grupos encontrados por uma função de avaliação interna. Essa função também chamada de função-objetivo, tenta medir alguma propriedade específica, como a compactação de um grupo por exemplo. Então, para alguns conjuntos de dados, suposições da função-objetivo podem ser violadas, sendo necessário um conhecimento sobre em qual situação deve-se aplicar uma técnica. Outro elemento destes algoritmos que demonstra seu grau de subjetividade é a necessidade de, em alguns casos, ser necessário definir previamente o número de grupos, levando a possíveis enviesamentos.

Por exemplo, o K-means tende a ser muito eficiente para dados com grupos em formato elíptico. Já para dados com formatos alongados, um método aglomerativo hierárquico como a ligação simples pode obter resultados melhores. Quando há alguma informação sobre o processo de formação dos dados, então há a possibilidade de se usar misturas de distribuições. Dessa forma, a escolha do algoritmo deve ser sempre parcimoniosa.

De acordo com a literatura, sabe-se que métodos diferentes podem ter objetivos conflitantes. Para contornar essa dificuldade, utilizam-se métodos diferentes agregados, executa-se o mesmo método diversas vezes a partir de soluções iniciais e número de grupos diferentes, ou se aplicam diversos métodos complementares. Usualmente, métodos combinados ou *essemble* tendem a ser mais robustos que os algoritmos mais simples para casos em que os dados apresentem diferentes características simultaneamente.

Segundo Handl and Knowles (2007), a literatura deu pouca atenção a algoritmos multiobjetivo, ou seja, algoritmos que têm por meta tentar encontrar soluções otimizando mais de um objetivo simultaneamente. Até certo ponto, é a otimização de diversas funções objetivo que os métodos agregados tentam reproduzir. Neste contexto, Handl and Knowles (2007) desenvolveram o MOCK (Multiobjective clustering with automatic determination of the number of clusters), sendo este um algoritmo genético com duas características fundamentais. A primeira é a seleção automática do número de grupos (usualmente representado pela letra K), em geral um fator de conflito em muitos métodos. O MOCK possui uma metodologia robusta para determinação da quantidade de grupos. A segunda característica é o fato de utilizar dois objetivos conflitantes, ou seja, procura-se um método que em casos de existência de grupos bem definidos, um dos dois objetivos se sobressaia sobre

o outro, e para casos com dados em que a separação dos grupos não é clara, a utilização de dois objetivos se destaque sobre qualquer método com um único objetivo.

Como demonstrado por Handl and Knowles (2007), o MOCK se mostrou muito eficiente quando comparado a outros métodos com objetivos simples e métodos agregados. Para situações em que por conhecimento prévio, o k-mens ou os métodos hierárquicos eram sabidamente melhores, o MOCK se mostrou competitivo. Para casos multidimensionais ou com dados com grau de complexidade grande, o MOCK se destacou em relação aos outros métodos, inclusive aos métodos agregados. Partido de todos estes contextos, este trabalho teve como principal proposta a implementação do algoritmo MOCK em linguagem R e a reprodução de resultados descritos por (Handl and Knowles, 2007), e possíveis discussões e melhoria dos processo de seleção automatizada de grupos.

Neste texto será detalhada toda a teoria por trás do MOCK e os motivos que o levaram a ser objeto de estudo deste trabalho, além de outros métodos que foram utilizados para comparação.

Capítulo 2

Revisão de literatura

Neste capítulo será apresentada toda a instrumentação teórica para desenvolvimento de métodos de clusterização, e de maneira mais especifica os detalhes da construção metodológica do algoritmo MOCK.

2.1 Tipos de dados a serem agrupados

Como dito anteriormente, existe a necessidade de se agrupar toda a sorte de dados - numéricos, categóricos, temporais, dados não estruturados (textos) e etc. De fato, o tipo de dado que se deseja agrupar afeta diretamente na escolha do algoritmo e os primeiros algoritmos foram elaborados para agrupar dados com a suposição de que eram numéricos, e isso nem sempre ocorre. Entretanto, é possível afirmar que são poucos os algoritmos que conseguem lidar com mais de um tipo de dados simultaneamente.

Além de dados numéricos, atualmente, dois tipos de dados têm se destacado nos diversos tipos de análise: dados categóricos e textuais. apesar de não fazerem parte do escopo deste trabalho, vale ressaltar que estes tipos de dados antigamente não eram considerados em processos de clusterização pois em geral são mais difíceis de serem agrupados ou exigem uma grande capacidade computacional.

Dados categóricos se caracterizam por trazer algum atributo de uma observação, por exemplo, sexo, cor, grau de instrução ou qualquer outra característica, podendo ter ordem ou não. Desta forma, não se tem a noção usual de distância ou similaridade entre os dados, não sendo possível usar algoritmos tradicionais para dados numéricos. Para contornar esse problema é necessário utilizar outros tipos de funções de similaridade entre os dados que não se baseiam em distância simplesmente.

Textos geralmente são representados em vetores sem forma ou ordem, sendo tratados como um conjunto de palavras. Arquivos de texto usualmente são difíceis de trabalhar, pois contêm muitas dimensões e são muito esparsos, dado que em uma língua existe uma enorme quantidade de palavras. Comumente a característica atribuída para classificação é a frequência de palavras repetidas.

Existem ainda outros tipos de dados menos frequentes na literatura referente a agrupamento. Por exemplo, sequências discretas comuns em dados biológicos, séries temporais em que os dados dependem de uma estrutura de tempo e dados espaciais em que estes dependem de uma estrutura do local onde foi observado. Ainda existem outros tipos de dados menos comuns.

Neste trabalho por definição de como o MOCK foi construído, serão utilizados apenas dados numéricos para geração dos resultados.

2.2 Algoritmos Particionais

Tendo o K-means como principal representante desta classe de algoritmos, esses métodos se destacam devido à facilidade de implementação, simplicidade e eficácia. São muito citados na literatura da área pois podem ser aplicados em quase todos os tipos de dados. Uma característica muito importante dessa classe de métodos é a necessidade de predefinir um número fixo de grupos.

Consequentemente, um valor determinado de clusteres pode vir a ser uma dificuldade quando os grupos não apresentam separações claras ou possuem formas não regulares.

2.2.1 K-means

O K-means é um dos mais simples e mais eficientes método de agrupamento. Inicia-se escolhendo K pontos arbitrários representativos como centroides inicias. Cada observação dos dados é atribuída ao centroide mais próximo. Uma vez que os grupos são formados, os centroides para cada grupo são atualizados, calculados como o ponto médio do grupo. Então, o algoritmo repete interativamente os dois passos (ajuste dos centroides e formação dos grupos), até que os centroides não mudem de acordo com os critérios de convergência.

A primeira iteração é iniciada escolhendo centroides aleatoriamente, porém, existem outras técnicas de inicialização (e.g Hartigan e Wong, Milligan, K-Means++, entre outras).

Algoritmo 1: K-Means

Entrada: Matriz de dados

Saída: Classificação das obervações para o K definido

início

Criar os centroides;

enquanto *Não há convergência* **faça**

Associar cada observações ao centroide mais próximo;

Recalcular os centroides;

fim

fim

2.2.2 Definição matemática

O algoritmo K-means, em essência, busca minimizar a soma de quadrados dos erros (**SSE**) como função objetivo. Sejam $X = \{x_1, x_2, ..., x_N\}$, em que cada x_i é um vetor, e $C = \{C_1, C_2, ..., C_K\}$ uma partição dos N vetores em K grupos, sendo c_k o centroide do k-ésimo grupo, $k = 1, \ldots, K$. O processo iterativo em cada passo tem por objetivo minimizar o SSE dado por

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - c_k)^2, \tag{2.1}$$

em que o ponto que minimiza essa função é

$$c_k = \frac{\sum\limits_{x_i \in C_k} x_i}{|C_k|}, k = 1, \ldots, K.$$

Prova. Seja $SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - c_k)^2$ e derivando em relação a um c_j

$$\begin{aligned}
\frac{\partial}{\partial c_j} SEE &= \frac{\partial}{\partial c_j} \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - c_k)^2 \\
&= \sum_{k=1}^{K} \sum_{x_i \in C_k} \frac{\partial}{\partial c_j} (x_i - c_k)^2 \\
&= \sum_{x_i \in C_k} -2(x_i - c_j) = 0 \\
\sum_{x_i \in C_k} c_j &= \sum_{x_i \in C_k} x_i \\
\text{Como } \sum_{x_i \in C_k} c_j &= c_j |C_k|, \text{ tem-se que} \\
c_j &= \frac{\sum\limits_{x_i \in C_k} x_i}{|C_k|}.
\end{aligned}$$

□

Como a SSE é uma função estritamente convexa, existe apenas um ponto de mínimo. E como o melhor representante que minimiza o SSE é a média dos pontos do grupo, o SSE decresce monotonicamente a cada iteração, convergindo para um mínimo global.

Existem várias modificações e formulações do método, que não serão abordadas neste trabalho. Já o K-means original descrito neste capítulo faz parte integral da inicialização do MOCK, visto que além de extramente eficiente, a qualidade do K-means em lidar com dados com grupos em formato elíptico são de grande contribuição para geração da população inicial do MOCK. Essa inicialização através do k-means será detalhada mais adiante.

2.3 Algoritmos hierárquicos

Fugindo da limitação da pré-seleção de número de grupos, os métodos hierárquicos foram desenvolvidos para serem mais flexíveis e determinísticos, quando se comparado aos métodos particionais.

Há dois tipos de métodos hierárquicos, chamados de aglomerativos e divisivos. Os conceitos que determinam os dois métodos serão detalhando mais à frente, porém am-

bos podem ser representados por uma formação binária em forma de árvore chamada de dendrograma.

Um dendrograma é um tipo de diagrama que organiza determinados fatores ou variáveis. No caso de agrupamento de dados, os níveis do dendrograma são os grupos formados pelos critérios dos algoritmos utilizados, sendo estes chamados de nós. A cada nível da "árvore" cresce o valor de K e através de diferentes cortes horizontais nos ramos do dendrograma é possível obter grupos diferentes (Figura 2.1).

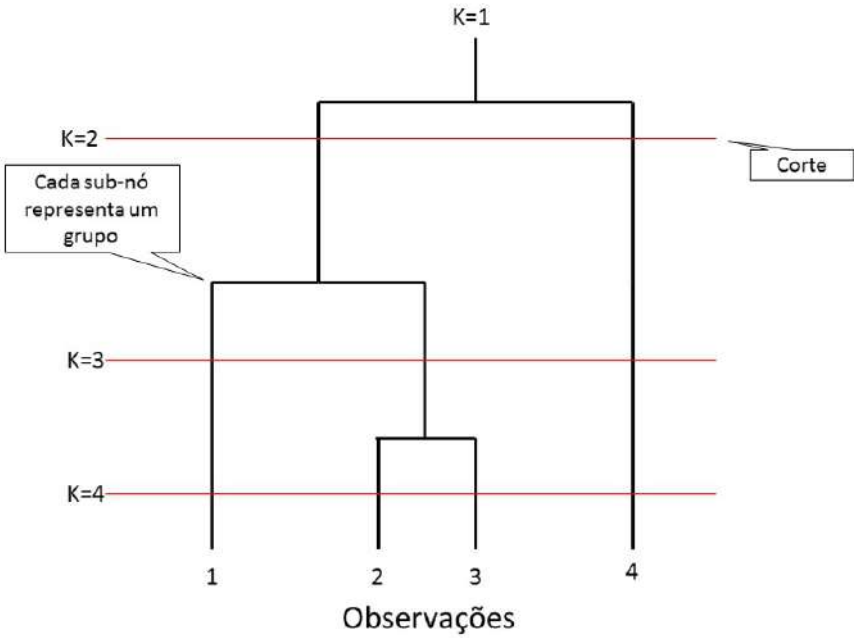


Figura 2.1: Exemplo de dendrograma para métodos hierárquicos

2.3.1 Métodos Aglomerativos

Os métodos aglomerativos são os mais comuns entre os métodos hierárquicos. Este grupo de algoritmos são conhecidos como agrupamento de *baixo para cima* pois, na fase inicial do algoritmo, cada observação é um grupo único, e portanto tem-se $K = N$. À medida que se avança, as observações são aglomeradas em grupos cada vez maiores, até que todas as observações formam um único grupo. Usando a Figura (2.1) como exemplo, na primeira etapa do algoritmo teria-se $K = 4$, na segunda etapa $K = 3$ e assim sucessivamente até $K = 1$.

Para que estas etapas do processo aglomerativo aconteçam, inicialmente é necessário definir a matriz de dissimilaridade e a medida de distância entre os pontos que será usada.

A matriz de dissimilaridade é formada a partir do cálculo da medida de distância escolhida (e.g euclidiana) para cada ponto dois a dois. Alguns métodos mais conhecidos do algoritmos aglomerativos são a ligação simples, ligação completa e critério de Ward.

2.3.1.1 Ligação simples

Com a capacidade de agrupar dados com formas alongadas e não elípticas, a ligação simples, *simple link* ou método do vizinho mais próximo, dá mais importância para regiões ou grupos estão mais próximos para os ligar. Assim sendo, não são levadas em consideração possíveis estruturas dos grupos.

As ligações entre as observações são feitas pela observação atual e seu vizinho mais próximo, figura 2.2. Define-se então que a similaridade entre os grupos é a similaridade entre a observação e seu vizinho mais próximo. Sejam a, b e c três grupos e d uma medida de distância qualquer. O agrupamento por ligação simples é dado por:

$$D(a+b,c) = \min\left\{d(a,c), d(b,c))\right\}.$$

2.3.1.2 Ligação completa

Na ligação completa a ligação entre dois grupos se dá considerando como medida de dissimilaridade a maior distância entre todos os elementos dos grupos. Ou seja, sejam a, b e c três grupos e d uma medida de distância qualquer. O agrupamento por ligação completa é dado por:

$$D(a+b,c) = \max\left\{d(a,c), d(b,c))\right\}.$$

Dessa forma, a ligação entre os grupos leva em consideração a estrutura dos dados. Porém, tanto o método da ligação simples quanto o da completa apresentam dificuldades na presença de outliers, transformado-os em grupos com uma única observação.

Na figura abaixo é possível visualizar como são formados os grupos na ligação simples e completa. Em cada etapa observa-se que cada métodos agrega grupos me maneira diferentes.

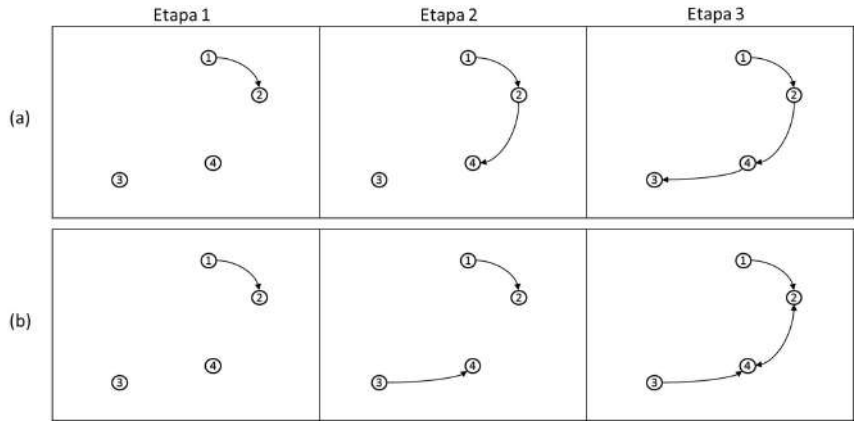


Figura 2.2: Ilustração dos tipos de algoritmos aglomerativos: (a)Ligação simples; (b)Ligação completa

2.3.1.3 Critério de Ward

Chamado também de critério de variância mínima, tem por objetivo ligar dois grupos considerando o incremento do SSE de forma que a ligação é feita entre os grupos que menos aumenta o SSE em cada passo. O método de Ward procura por partições que minimizem a perda associada a cada agrupamento.

Sejam a e b dois grupos. O critério de Ward é dado por:

$$\boldsymbol{W}(a,b) = \frac{N_a N_b}{N_a + N_b} \sum_{i=1}^{N_a} \sum_{j=1}^{N_b} (a_i - b_j)^2,$$

em que N_a é número de elementos no grupo a, grupos maior, N_b é número de elementos no grupos b e a_i e b_i são os elementos de a e b respectivamente. Então, a e b formarão um novo grupo se, e somente se, o incremento na SEE devido à agregação de a e b é o menor entre quaisquer outros dois grupos dois a dois.

Em geral, apresentam resultados insatisfatórios quando o tamanho dos grupos são parecidos e apresentam resultados melhores que o link simples e link completo.

Esses três métodos de agrupamento hierárquicos estão incorporados neste trabalho, participando do grupo de métodos que serão utilizados para comparação com o MOCK.

2.3.2 Métodos Divisivos

Conhecidos como métodos de agrupamento de *cima para baixo*, pois ao contrário do método aglomerativo, a inicialização se dá considerando todos os dados conectados em um único grupo. As conexões são separadas uma a uma formando grupos cada vez menores até

que cada observação seja um único grupo. Essas separações se dão através de critérios ou funções, como por exemplo o *Bisecting K-means*, que é uma método de separação baseado no critério de Ward. Assim sendo, a cada passo este critério escolhe a separação que mais diminui o SSE.

Como inicialmente os algoritmos divisivos usam toda a informação contida nos dados, estes métodos podem ser considerados como uma abordagem global, tendo na maioria das vezes melhores resultados que os métodos aglomerativos pois utilizam mais informação a partir da conexão entre todos os pontos. Ademais, são mais eficientes que os aglomerativos, pois não há a necessidade de que o agrupamento gere toda a cadeia hierárquica.

2.3.2.1 Árvore geradora mínima

Um fator importante dos métodos divisivos é a geração eficiente e correta das conexões entre todos os ponto na primeira etapa do algoritmo. Uma agregação enviesada pode levar a resultados não desejados, em que os critérios de separação dos grupos não consigam identificar os melhores pontos de corte.

Uma forma eficiente de gerar essas conexões são as árvores geradoras de custo mínimo ou *minimum spanning trees* (**MST**). Uma MST é um subgrafo T de um grafo G não dirigido com custo nas arestas, ligando todos os vértices com o menor custo possível, considerando a soma dos custos de todos os vértices de T. Para o caso de agrupamento de dados, a árvore geradora de custo mínimo seria a forma menos custosa de ligar todos os pontos em um único grupo, considerando a medida de distância como custo de cada aresta, figura 2.3. Os algoritmos mais conhecidos para encontrar a MST são os algoritmos de Prim e de Kruskal.

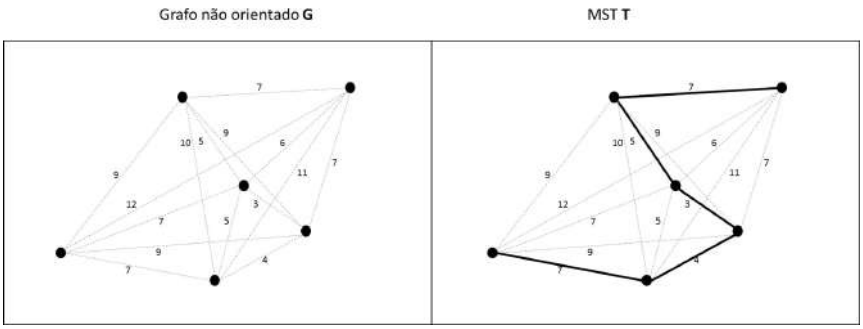


Figura 2.3: Exemplo de um grafo não orientado (esq.) e uma árvore geradora de custo mínimo (dir.).

2.3.2.2 Algoritmo de PRIM

Seguindo a descrição de uma MST, o algoritmo de Prim é um dos mais conhecidos e mais simples para a geração de árvores de custo mínimo. O algoritmo pode ser descrito pelos seguintes passos:

Algoritmo 2: PRIM

Entrada: Matriz de dados

Saída: Arvore geradora mínima conectando todos os dados

início

Inicie T com um vértice arbitrário e faça $\overline{T} = N - T$;

enquanto $T < N$ **faça**

Encontre a aresta $i \rightarrow j$ de menor custo tal que $i \in T$ e $j \in \overline{T}$;

Faça $T = T \cup \{j\}$ e $\overline{T} = \overline{T} - \{j\}$

fim

fim

Como este algoritmo em cada iteração procura apenas o vizinho mais próximo como a ligação simples, então, diz-se que este método tem caráter "guloso", dado que ignora qualquer estrutura que possa existir nos dados.

O detalhamento destes métodos é importante pois juntamente com o K-means, fazem parte da geração da inicialização do MOCK, na qual parte da população criada vem de uma adaptação de um método divisivo iniciado pelo algoritmo de Prim.

2.3.3 Métodos baseados em densidade

Outro grupo de métodos muito citado na literatura é o dos algoritmos de agrupamento baseados em densidade. Como não fazem suposições sobre o número de grupos nem sobre as distribuições dos dados, podem ser considerados como métodos não paramétricos.

Estes métodos se destacam quando aplicados a dados espaciais, pois se adaptam melhor a dados com formatos arbitrários que possam ser gerados por estruturas espaciais. Também se adequam bem a casos multidimensionais em que a suposição de que os dados tenham um único formato para várias dimensões geralmente não é válida.

De maneira superficial estes métodos se baseiam na ideia de que grupos em uma determinada região do espaço têm densidade de pontos grande, e entre os grupos há regiões

com densidade baixa de pontos.

O algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) parte ideia de separação espacial. Este algoritmo contabiliza a densidade considerando um raio fixo chamado de vizinhança, e conta a quantidade de pontos dentro dessa vizinhança. Dois pontos são considerados conectados se ambos fazem parte da mesma vizinhança ou se são conectados através da vizinhança de outros pontos. Logo os grupos são formados por pontos de desconexão entre as vizinhanças.

2.3.4 Outros métodos de agrupamento

Além das técnicas de agrupamento citadas acima, existem outras que merecem ser mencionadas, como por exemplo o agrupamento baseado em grid, fatoração de matriz não negativa e agrupamento espectral. Há também métodos aplicados a dados categóricos, ou mesmo a dados não estruturados como arquivos de textos. Pode-se dizer que a área de métodos de agrupamento tem crescido muito atualmente, principalmente em função da maior capacidade de processamento dos computadores e pela crescente demanda em análise de dados na chamada ciência de dados.

2.4 Otimização

Partimos agora para um conceito que define uma das características mais improntantes do MOCK, dado que um dos motivos do seu sucesso vem da capacidade de lidar com mais de um objetivo simultaneamente, paradigma conhecido como otimização multiobjetivo. Antes porém, é necessário um melhor entendimento do que computacionalmente significa otimização.

Dá-se o nome de otimização ao conjunto de ferramentas que têm por objetivo minimizar ou maximizar funções matemáticas (função-objetivo), encontrando pontos de mínimo ou máximo globais dessas funções. Na maioria das vezes encontrar estes pontos não é trivial, dependendo do comportamento da função-objetivo, pois, esta pode conter diversos mínimos locais (Figura 2.4) que atraem o algoritmo para soluções erradas. Assim sendo, é fundamental empregar o algoritmo mais adequado para cada situação.

Mais rigorosamente, considere a função $f : \mathbb{R}^n \to \mathbb{R}$. Um ponto x_0 pode ser chamado de mínimo global se, e somente se, $f(x_0) \leq f(x)\,, \forall\, x \in R^n$. Um ponto x_i dentro de um

conjunto aberto A pode ser chamado de mínimo local se, e somente se, $f(x_i) \leq f(x) \, \forall \, x \in A$.

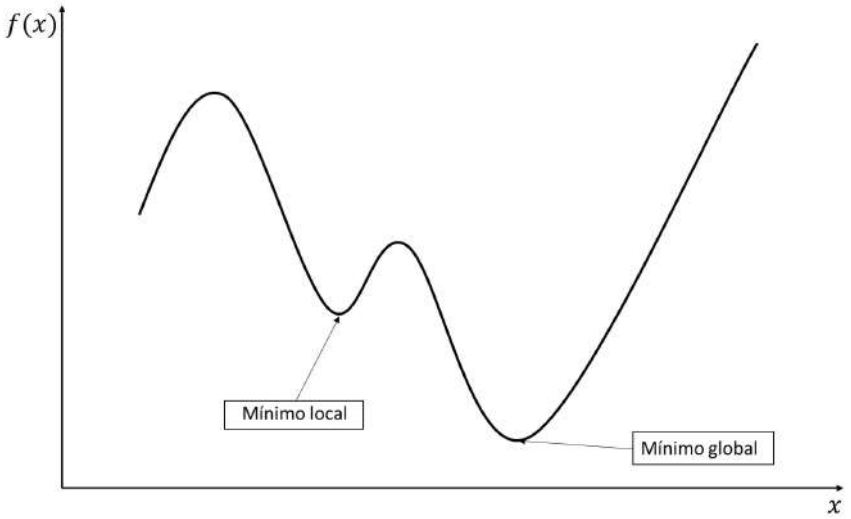


Figura 2.4: Exemplo de uma função com ponto de mínimo local e global

É evidente que o maior problema de um processo de otimização é garantia de existência de um ponto de mínimo global, ou mesmo, a garantia de que o foi atingido. Mesmo para funções estritamente convexas, dependendo da suavização da curvatura, os algoritmos usados podem não conseguir atingir pontos de mínimo.

Problemas de otimização são encontrados em quase todas as áreas de conhecimento como na economia, engenharia, computação e na estatística. Por exemplo, na estimação de parâmetros de um modelo de regressão, maximiza-se a função de verossimilhança.

De maneira geral, os problemas de otimização podem ser representados da seguinte forma. Seja $x \in \mathbb{R}^n$ um vetor de variáveis , $f : \mathbb{R}^n \to \mathbb{R}$ uma função-objetivo e $g_i : \mathbb{R}^n \to \mathbb{R}$ funções de restrição em relação a x, em que:

$$g_i(x) \leq b_i, \text{ com } b_i \in \mathbb{R} \text{ (Restrições de desigualdade)}$$

$$g_i(x) = d_i, \text{ com } d_i \in \mathbb{R} \text{ (Restrições de igualdade)}$$

Sem perda de generalidade g_i não necessariamente precisa existir. Definindo as condições do problema, então o processo de otimização passa por algum algoritmo para minimizar $f(x)$, dadas as restrições $g_i(x)$, caso existam. Se o objetivo do problema é maximizar a função, então troca-se o sinal, minimizando a função $-f(x)$.

Os métodos de otimização podem ser separados em determinísticos e estocásticos. Algoritmos determinísticos sempre levam à mesma resposta dado um ponto inicial igual,

diferentemente dos estocásticos, em que as escolhas feitas pelos algoritmos durante sua execução são aleatórias, levando a resultados diferentes mesmo quando partem do mesmo ponto inicial.

Os métodos determinísticos mais famosos são os do tipo gradiente descendentes. Estes métodos usam informação das derivadas de primeira e segunda ordem, sendo estas analíticas ou geradas numericamente. Os algoritmos de Newton e quase-Newton como o BFGS via de regra são os mais citados.

Métodos heurísticos são uma subclasse de algoritmos que podem ser determinísticos ou estocásticos. São chamados de heurísticos pois encontram soluções sub-ótimas, isto é, uma aproximação da solução ótima. Dessa maneira, espera-se que os algoritmos heurísticos funcionem bem na maioria das vezes mas não sempre. Comumente são muito eficazes pra encontrar boas soluções, mas não há a garantia de que atinjam a solução ótima. Um famoso exemplo para esta classe de métodos são os Algoritmos Genéticos, quem fazem parte integral do MOCK através do algoritmo PESA-II, que será detalhado mais adiante.

2.5 Otimização Multiobjetivo

Atualmente existe um dilema na sociedade entre o desenvolvimento econômico e a preservação do meio ambiente. É correto afirmar que o desenvolvimento, por mais sustentável que seja, causa algum prejuízo ambiental, sendo necessário conciliar o maior desenvolvimento econômico com o menor prejuízo ambiental possível. Essa forma de dilema é chamada *tradeoff* ou, em tradução livre, custo de oportunidade. É deste princípio de *tradeoff* que partem os métodos de otimização multiobjetivo.

Suponha que para um determinado problema existam duas ou mais funções-objetivo conflitantes e que são igualmente importantes, ou seja, não existe nenhuma informação adicional disponível para a solução do problema. Então, é necessária uma forma de maximizar ou minimizar todas as funções-objetivo simultaneamente. É neste tipo de contexto que os métodos multiobjetivo são necessários.

A otimização multiobjetivo pode ser definida de forma muito parecida a otimização simples. Seja $\mathbf{x} \in \mathbb{R}^n$, um vetor de variáveis , $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ um vetor de objetivos e

$\mathbf{g}_i : \mathbb{R}^n \to \mathbb{R}$ funções de restrições em relação a $\mathbf{x}$, em que:

$$\mathbf{g}_i(x) \leq \mathbf{b}_i, \text{ com } b_i \in \mathbb{R} \text{ (Restrições de desigualdade)}$$
$$\mathbf{g}_i(x) = \mathbf{d}_i, \text{ com } d_i \in \mathbb{R} \text{ (Restrições de igualdade)}$$

De maneira que, sem perda de generalização, g_i não necessariamente precisa existir. Diferentemente da otimização mono-objetivo, neste caso em geral não existe apenas uma solução ótima, mas uma coleção possivelmente infinita de soluções ótimas representando diferentes *tradeoffs*, chamadas de soluções Pareto-ótimas.

2.5.1 Soluções Pareto-ótimas

Para entender o que é uma solução Pareto-ótima é necessário definir o conceito de dominância. Uma solução $\mathbf{x_1}$ domina uma solução $\mathbf{x_2}$, se e somente se:

1. $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2)$, para todo $i = 1, ..., m$, ou seja, a solução $\mathbf{x}_1$ é melhor que ou igual a solução $\mathbf{x}_2$ em todos os objetivos.

2. $f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$, para pelo menos um $i \in \{1, ..., m\}$, ou seja, a solução $\mathbf{x}_1$ é estritamente melhor que a solução $\mathbf{x}_2$ para ao menos um objetivo.

Com a definição acima, é possível descrever uma solução Pareto-ótima como uma solução que não é dominada por nenhuma outra, e a fronteira de Pareto é formada por todas a soluções não dominadas (Figura 2.5). As fronteiras de Pareto posteriormente serão utilizadas no MOCK para definição da solução automatizada de grupos.

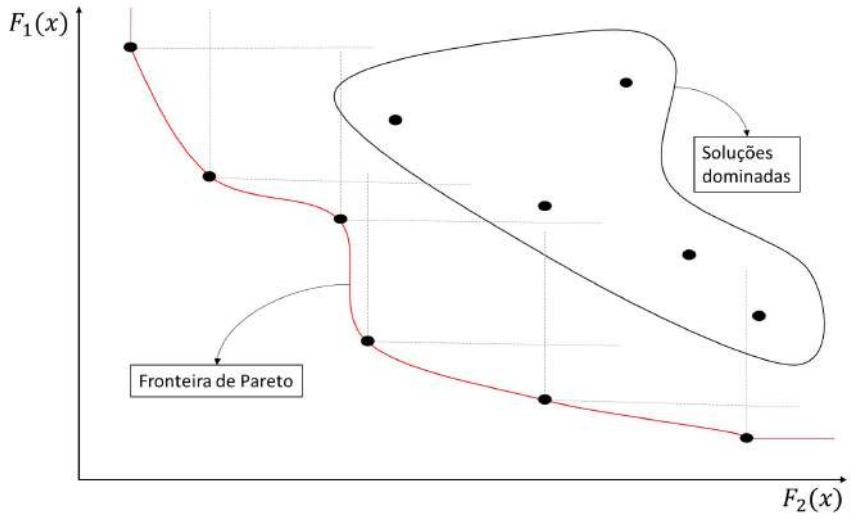


Figura 2.5: Exemplo de uma fronteira de Pareto para um problema com dois objetivos

2.6 Métodos evolutivos

De maneira mais abrangente, a computação natural (Castro, 2006) ou técnicas computacionais baseadas em fenômenos da natureza, tem se destacado em diversas áreas da computação, não apenas em otimização. Um bom exemplo são as redes neurais artificiais para modelos preditivos, muito utilizadas em inteligência artificial.

Os métodos evolucionários são um conjunto de algoritmos que têm inspiração em fenômenos naturais referentes à teoria sintética da evolução ou neodarwinismo. Conceitos como seleção natural, mutação e combinação gênica são os fundamentos clássicos deste grupo de algoritmos pertencentes à classe dos métodos heurísticos. O algoritmo principal do MOCK, chamado de PESA-II, é um método evolutivo que aceita otimização de mais de um objetivo simultaneamente.

Um algoritmo evolutivo (**AE**) básico pode ser visto no pseudocódigo abaixo, e cada etapa do algoritmo será detalhada mais a frente.

Algoritmo 3: AE básico

início

Cria-se uma população (conjunto de soluções, indivíduos) de forma aleatória;

Avalia-se a população inicial;

repita

Selecionar candidatos para cruzamento;

Realizar cruzamentos dos indivíduos selecionados;

executar o processo de mutação;

Avaliar os novos indivíduos gerados pelos cruzamentos e mutações

até *a convergência*;

fim

Algumas terminologias para a definição dos métodos evolutivos são:

- Cromossomo: Estrutura que codifica a estrutura de um indivíduo.
- Genótipo: Um conjunto de cromossomos que formam um ser vivo.
- Fenótipo: Caracterização dos indivíduos com base eu seu genótipo.
- Gene: Conjunto de símbolos que codificam um cromossomo.
- Alelos: Diferentes valores de um gene.
- Locus: posição de um gene.

O primeiro passo para um algoritmo evolutivo é a definição da representação genética ou cromossomo (Figura 2.6), que pode ser descrita como uma forma de traduzir a característica do problema atacado de uma maneira simplificada, permitindo a execução de processos de cruzamento e mutação. As codificações gênicas mais usadas são as binárias e de ponto flutuante ou representação real. Cada tipo de representação depende da característica do problema a ser solucionado.

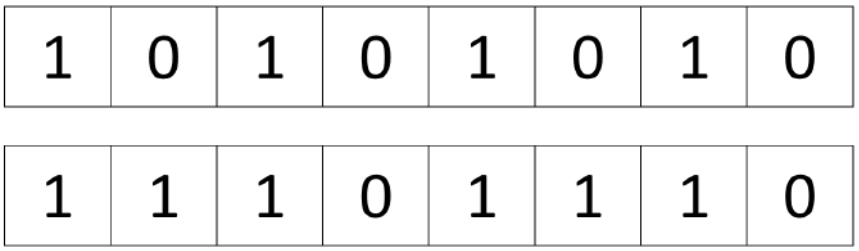


Figura 2.6: Exemplo de representação genética ou cromossomo através da dados binários

Uma segunda etapa do processo de um AE, é a avaliação e seleção dos indivíduos da população inicial (pais) para o processo de cruzamento, gerando uma nova geração de soluções (filhos), tal que cada iteração do algoritmo é considerada uma geração da população.

A nova geração então é agregada à população, mas antes passa por um processo de seleção, visto que, um parâmetro importante do algoritmo usado é o tamanho da população, impactando diretamente no desempenho do algoritmo. Populações muito grandes têm mais chance de obterem soluções melhores, porém acarretam em um custo computacional alto. o inverso vale para populações menores.

Para os processos de cruzamento ou recombinação, existem diversas estratégias na literatura da área. Este mecanismo é o responsável pela geração de novos indivíduos, que podem ser soluções melhores ou não. Os métodos de cruzamento mais comuns são:

- Cruzamento 1-ponto: um ponto de corte no cromossomo é escolhido ao acaso. Um filho é formado com a primeira parte do pai 1 e a segunda parte do pai 2; outro filho é formado com a primeira parte do pai 2 e a segunda parte do pai 1 (Figura 2.7).

- Cruzamento n-pontos: semelhante ao cruzamento 1-ponto, mas com n pontos de corte escolhidos ao acaso.

- Cruzamento uniforme: para cada posição do gene, associa-se uma probabilidade fixa de troca P (em geral $P = 0,5$). Então sorteiam-se n valores aleatórios $U \sim (0, 1)$, em que n é o comprimento do cromossomo. Para $i = 1, 2, \ldots, n$, se $u_i < P$, então o i-ésimo gene virá do pai 1; caso contrário do pai 2.

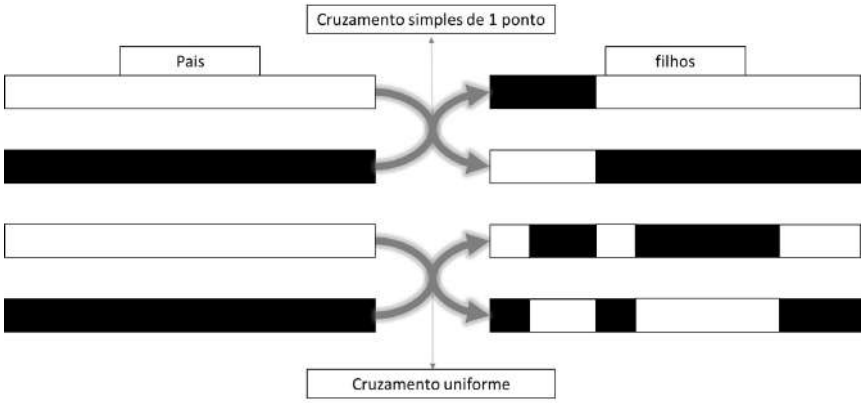


Figura 2.7: Exemplo de formas de cruzamento

Para o operador de mutação define-se uma taxa de mutação P_m, geralmente muito pequena, evitando alterações radicais. Gera-se um valor $U \sim (0, 1)$, e se esse valor for menor que P_m, então de forma aleatória um ou mais genes podem sofrer alguma alteração aleatória.

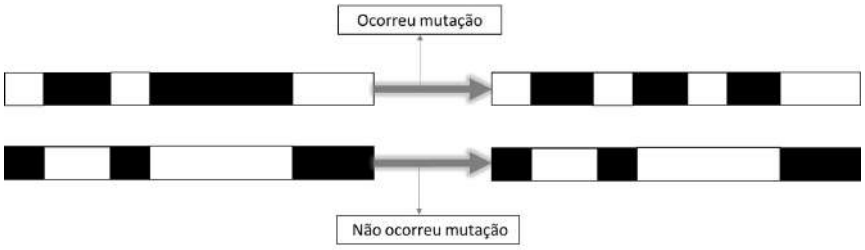


Figura 2.8: Exemplo um processo de mutação para dois filhos

Os três métodos evolutivos mais conhecidos são: programação evolutiva, estratégias evolutivas e os algoritmos genéticos, sendo que os algoritmos genéticos fazem parte do escopo deste trabalho. É importante frisar que os 3 métodos citados usam estratégias diferenciadas, por exemplo, a programação evolutiva não passa por um processo de cruzamento, ou seja, a reprodução dos indivíduos é assexuada.

2.7 PESA-II

Das estratégias evolutivas, os algoritmos genéticos (**AG**) passam por todas as etapas descritas no algoritmo 3, e seus conceitos podem ser expressos como um processo natural. Nos genes do indivíduo estão registradas suas características e cada geração é formada por combinações de genes da geração anterior. Qualquer indivíduo pode passar por um

processo de mutação que o leve a ser mais apto a sobrevivência e, dessa forma, ter maior probabilidade de gerar descendentes que devem herdar parte de suas características.

Analogamente, os AGs conseguem se aproveitar das melhores soluções encontradas e explorar melhor os espaços de busca através dos seus processos de cruzamento e mutação.

Um algoritmo usado neste trabalho e núcleo central do MOCK é o *Region-based selection in evolutionary multiobjective optimization* ou PESA-II, proposto por Corne et al. (2001). Sendo um algoritmo genético para otimização multiobjetivo, o PESA-II é caracterizado por duas estratégias principais. A primeira é a existência de duas populações chamadas de populações interna **IP** e externa **EP**, e a segunda estratégia é a separação das soluções em "nichos".

Mantém-se duas populações pela necessidade de garantir uma grande diversidade de soluções não dominadas. A EP tem por objetivo manter um grupo grande de soluções não dominadas, levando-as às próximas gerações e garantindo que boas soluções já encontradas possam ser mantidas. Esta estratégia é chamada de elitista, na medida que mantém apenas soluções não dominadas na EP. A IP por sua vez, utiliza a informação de EP para explorar novas soluções, e é atualizada a cada geração.

Os chamados nichos ou células são hipercubos criando um *grid* no espaço de objetivos, figura 2.9. A quantidade de soluções que ocupam cada nicho é utilizada para que o algoritmo selecione com maior probabilidade regiões menos povoadas, enviesando as soluções para que seja coberta toda a área do espaço de busca, evitando muitas soluções em uma mesma região.

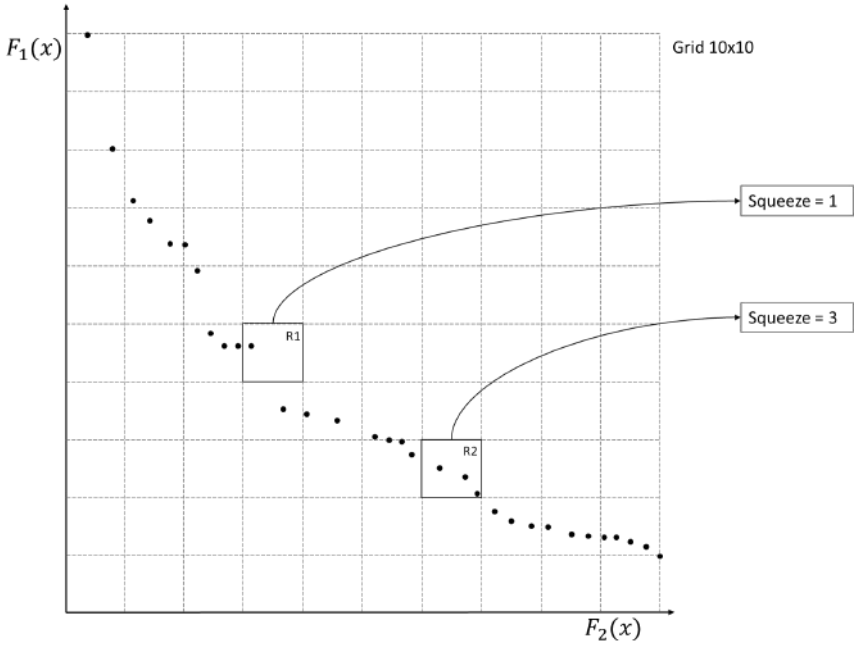


Figura 2.9: Exemplo de um grid criado no espaço de objetivos e a seleção de de dois nichos com seus respectivos *squezze factors*

O fator que define a escolha do nicho, chama-se *squezze factor*, que nada mais é que uma contagem de quantas soluções existem em um determinado nicho. Para o preenchimento da população interna é feito um sorteio de duas células, sendo escolhida a célula com o menor *squezze factor*. Na imagem 2.9, a célula escolhida seria o R1.

Selecionada um célula, uma solução dessa célula é sorteada para integrar a população interna. Este processo é repetido até que a IP seja completada. Deste ponto em diante, se iniciam os processo de cruzamento e mutação com qualquer critério que tenha sido escolhido, gerando novas soluções.

Por sua vez, as novas soluções são avaliadas e eventuais novas soluções não-dominadas são incorporadas à EP. O processo de seleção de células, cruzamento, mutação e seleção

são repetidos até a convergência.

Algoritmo 4: PESA-II

início
- Gerar população inicial $\boldsymbol{PS}$;
- Avaliar PS;
- Adicionar soluções não dominadas de PS à população externa $\boldsymbol{PE}$;
- **enquanto** *critério de convergência não for atingido* **faça**
  - Divida o espaço de busca em um grid de células;
  - Calcular o squeeze factor de todas as células;
  - **repita**
    - Selecionar duas células ao acaso;
    - Da célula menos povoada selecionar uma solução ao acaso;
  - **até** *Completar a população Interna* $\boldsymbol{PI}$;
  - **repita**
    - Seja $P_c \sim U_{[0,1]}$ a probabilidade de cruzamento;
    - **se** $P_c < p$ **então**
      - selecionar duas soluções de PI;
      - operar cruzamento;
      - operar mutação;
    - **senão**
      - selecionar uma solução de PI;
      - operar mutação;
    - **fim**
  - **até** *um número finito de vezes*;
  - Avaliar PI;
  - Incluir todas as soluções não dominadas de PI em PE;
- **fim**

fim

2.8 Agrupamento multiobjetivo com determinação automática do número de grupos

Finalmente chegamos à descrição do método utilizado neste trabalho. O algoritmo proposto por Handl and Knowles (2004), chamado de **MOCK** (Multiobjective clustering with automatic k-determination), é um método evolutivo para encontrar grupos de dados numéricos. De certa forma, o MOCK emprega a maioria dos conceitos descritos anteriormente, de maneira direta ou indiretamente.

Este algoritmo é baseado no PESA-II descrito anteriormente, ou seja, passa por todas as etapas descritas no Algoritmo 4. As fases mais importantes do MOCK são:

1. Criação da papulação inicial;
2. Aplicação do PESA-II para cruzamentos e mutação gerando melhores soluções;
3. Seleção automatizada de grupos;

Antes da descrição do algoritmos em si, é necessário o entendimento de dois conceitos. Primeiramente, a definição da codificação genética das soluções do algoritmo. Dado que o algoritmo de otimização do MOCK é um AG, então faz-se necessário que as possíveis soluções possam ser representadas com um cromossomo.

O segundo conceito é definido pelas funções-objetivo, pois são elas que definem o comportamento do algoritmo, fazendo com que se possa obter os melhores resultados possíveis.

2.8.1 Representação genética

Primeiramente, notemos que cada possível solução para o problema de clustering é uma partição do conjunto de dados e deve-se escolher uma forma de representar as soluções. A representação adotada é a *locus-based adjacency*. Nesta representação, cada solução é um vetor de genes $g_1, \ldots, g_N$, de comprimento igual ao número total de dados N. Cada gene pode receber um valor de alelo $j \in \{1, .., N\}$, que representa a posição de cada elemento na base de dados. Se o i-ésimo gene recebe o valor j, então o elemento i está *ligado* ao elemento j. Por essa lógica, portanto, i e j pertencem ao mesmo grupo, e qualquer outro elemento que esteja ligado a i ou j direta ou indiretamente também estará no mesmo grupo. Consequentemente, se dois elementos não se comunicam direta ou indiretamente, então ambos estão em grupos diferentes.

Uma grande vantagem deste método de representação é que o número e a composição dos grupos é obtida pelo processo de decodificação do gene. A figura 2.10 ajuda no esclarecimento do funcionamento da decodificação. Nesse exemplo, o conjunto de dados é formado por 10 elementos, cada um deles sendo um ponto no plano.

Na solução apresentada à esquerda o elemento 1 está ligado ao elemento 10, o elemento 2 está ligado a ele mesmo, o elemento 3 está ligado ao elemento 5, e assim por diante. Percebe-se que todos os pontos pertencem a um único grupo, pois todos os elementos se conectam, ou seja, a partir de um elemento i é possível chegar a qualquer outro elemento j precorrendo as ligações existentes. Essas conexões poderiam ser geradas por uma MST (seção 2.4.2.1).

A solução à direita foi obtida a partir de uma modificação no gene 5 da solução à esquerda. A ligação $5 \rightarrow 4$ foi substituída pela ligação $5 \rightarrow 5$, acarretando a geração de dois grupos distintos. Essa modificação poderia ser causada por uma mutação sofrida pela solução à esquerda, por exemplo.

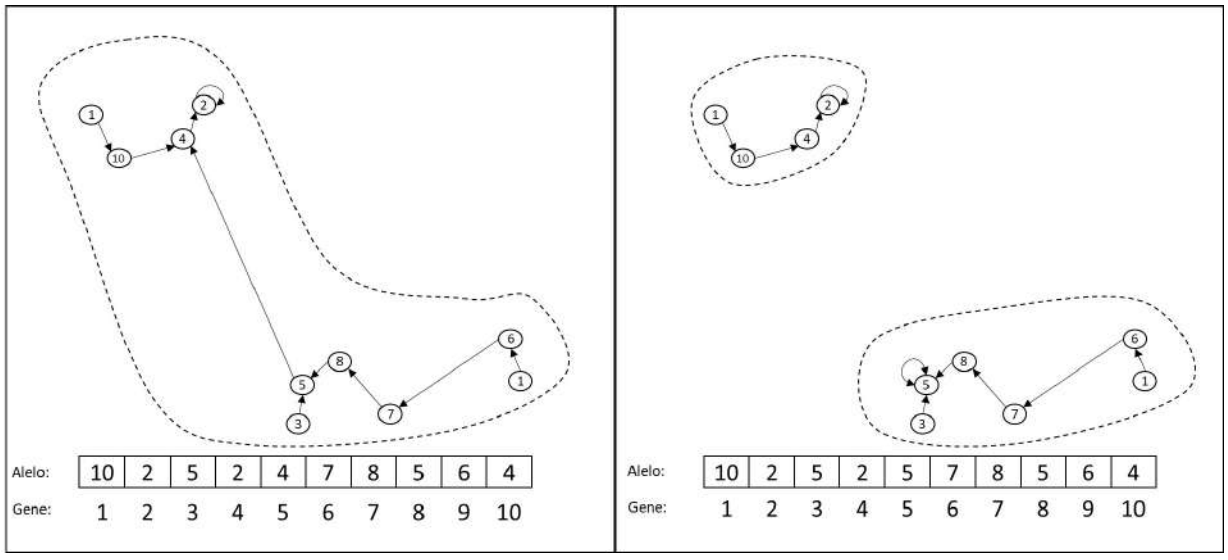


Figura 2.10: Representação de duas soluções e seus genótipos. Com a remoção de uma única ligação criam-se dois grupos.

2.8.2 Funções objetivo

As funções escolhidas foram a compactação ou desvio total e a conectividade. Individualmente cada função reflete aspectos diferentes de boas soluções. O desvio total reflete a qualidade de grupos mais próximos dos algoritmos particionais, como por exemplo o k-means. Dessa maneira tenta-se obter bons resultados para dados com formatos mais

regulares.

A conectividade tenta reproduzir uma medida de qualidade de algoritmos hierárquicos como a ligação simples por exemplo. A diferença é que existe um enviesamento dado pela quantidade de vizinhos mais próximos, obtendo uma relação simultânea entre hierarquia e separação espacial como no caso do algoritmos baseados em densidade de pontos.

A compactação ou desvio total de um grupo é dada por:

$$Dev(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k), \tag{2.2}$$

onde C é o conjunto de todos os grupos, μ_k é o centroide de um grupo $C_k \in C$ e δ uma medida de distância, no caso a Euclidiana.

A conectividade é dada por:

$$Conn(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} \gamma_{i,nn_j} \right), \tag{2.3}$$

em que

$$\gamma_{i,nn_j} = \begin{cases} \frac{1}{j} & , \text{ se } i \in C_k \text{ e } nn_j \notin C_k \\ 0 & , \text{ se } i \in C_k \text{ e } nn_j \in C_k \end{cases} \tag{2.4}$$

sendo nn_j o j-ésimo vizinho mais próximo de i, em que, $j \leq L$. L é um parâmetro que determina o número de vizinhos que contribuem para a conetividade e N é tamanho da base de dados. Assim, a conectividade é acrescida da quantidade $1/j$ caso o j-ésimo vizinho mais próximo do elemento i esteja em um grupo diferente do de i.

Note-se que ambas as funções-objetivo devem ser minimizadas.

2.8.3 População Inicial

Iniciando agora a descrição do algoritmo, partimos da criação da população inicial. O tamanho da população é controlado por um parâmetro de entrada chamado de k_{user}, que é uma definição de um número máximo de grupos que podem existir nos dados. Este parâmetro é fixado em 25 ou 50 grupos. Deste modo o tamanho máximo da população é $fsize = 2 \times k_{user}$, ou seja, $fsize = 50$ ou $fsize = 100$.

Para gerar as soluções iniciais, utilizam-se dois métodos distintos. O primeiro é um método hierárquico divisivo baseado em árvore geradora de custo mínimo, e o segundo é

o K-means. O motivo de usar dois métodos baseia-se no princípio de que estes métodos possuem características diferentes com relação à identificação das estrutura dos dados. O K-means, por exemplo, é muito eficiente para dados com formatos circulares e tende a gerar soluções com melhores valores de compactação. Já os métodos hierárquicos podem identificar grupos com formatos alongados e tendem a gerar soluções com melhores valores de conectividade. Utilizando-se destas estratégias, é possível gerar uma população com uma grande variabilidade de soluções preenchendo uma grande parte da fronteira de Pareto.

O método hierárquico, pode ser chamado de agrupamento por "ligações interessantes". De forma simplificada, aplica-se o algoritmo de Prim como descrito anteriormente para criar a MST dos dados completos. A partir da MST, procura-se por "ligações interessantes", definidas da seguinte maneira:

- Uma ligação $i \rightarrow j$ é considerada interessante se i e j não fazem parte dos grupos de L vizinhos mais próximos de j e i, respectivamente. De maneira mais formal, sejam $L_i = \{nn_{i1}, .., nn_{iL}\}$ o conjunto dos L vizinhos mais próximos de i, e $L_j = \{nn_{j1}, .., nn_{jL}\}$ o conjunto dos L vizinhos mais próximos de j. Se $i \notin L_j$ e $j \notin L_i$, então a ligação $i \rightarrow j$ é uma ligação interessante.

A partir da MST, os grupos são formados pela remoção das ligações consideradas interessantes. Na figura 2.10 é possível ver o efeito da remoção da ligação $5 \rightarrow 4$.

É possível obter $I + 1$ soluções distintas, em que I é o número de ligações interessantes. Essas soluções se dão pela ordenação das ligações interessantes pelo seu grau de interesse, dado pelo comprimento da ligação. De forma simples, ordenam-se as I ligações interessantes em ordem decrescente de comprimento, e removem-se as ligações interessantes uma-a-uma, gerando soluções com $\{0, 1, ..., I + 1\}$ grupos.

Com o objetivo de manter a maior variabilidade possível no conjunto da população inicial, o número máximo de soluções com que o método por ligações interessantes pode contribuir para a população inicial é dado por $\min\{I, [0,5 \times fsize - 1]\}$, ou seja, no máximo a metade da população inicial. As soluções geradas pelo K-means, ocorrem pela execução do algoritmo com apenas dez iterações e com diferentes números de grupos $\{2, ..., fsize - \min\{I, [0,5 \times fsize - 1]\}\}$, completando a população inicial.

2.8.4 Loop Principal

No loop principal do MOCK é finalmente utilizado o PESA-II. Gerada a população inicial iniciam-se as etapas descritas no Algoritmo 4.

Desta forma, as soluções inciais passam pelos processos de seleção, cruzamento e mutação. O método de cruzamento escolhido foi o uniforme, figura 2.7. Esse método tende a gerar uma variação maior nos indivíduos aumentando a probabilidade de gerar mais e melhores soluções, povoando toda a fronteira de soluções.

O operador de mutação utilizado é o chamado *nearest-neighbor mutation*, tem como objetivo enviesar o operador de mutação para que a mudança na ligação se dê apenas entre os L vizinhos mais próximos. Essa estratégia é usada, pois, se fosse possível que cada observação se ligasse a qualquer outra, as possibilidades de ligações seriam de N^N, gerando ligações que não fazem sentido. Limitando-se aos vizinhos mais próximos, a chance de gerar alguma solução espúria é reduzida. Assim, a probabilidade de mutação é dada por

$$P_m = \frac{1}{N} + \left(\frac{l}{N}\right)^2, \tag{2.5}$$

em que N é o número de observações e l é a posição do vizinho, sendo $l \leq L$.

Por último, o critério de parada do MOCK é o número de gerações. Os parâmetros escolhidos para o MOCK estão na tabela 2.1.

Tabela 2.1 - Parâmetros fixados para o MOCK

Descrição do Parâmetro	Parâmetro
Número de gerações	1000
Tamanho máximo da população externa EP	1000
Tamanho da população interna IP	10
Resolução do hipergrid por dimensão	10
Número máximo de grupos K_{user}	25 ou 50
Tamanho da população inicial $fsize$	$2 \times K_{user}$
Inicialização	Ligações interessantes e K-means
Tipo de mutação	nearest-neighbor mutation
Probabilidade de mutação	$P_m = \frac{1}{N} + \left(\frac{l}{N}\right)^2$
Tipo de cruzamento	Cruzamento uniforme
Probabilidade de cruzamento	0,7
Funções objetivo	Conectividade e Compactação

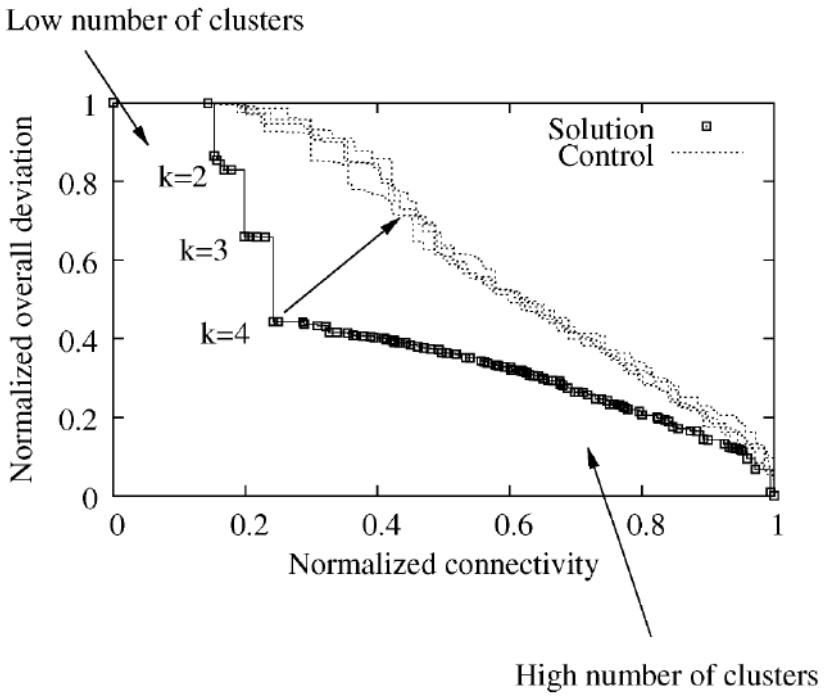
2.8.5 Seleção Automática de grupos

Como já citado anteriormente o PESA-II tem a característica de procurar obter a fronteira de Pareto mais povoada possível, explorando um maior número de soluções. Com estas soluções definidas, então procura-se reduzir o grupo de soluções para apenas uma única.

As duas funções-objetivo são conflitantes, ou seja, a diminuição da compactação causa um aumento na conectividade, e quando as soluções são colocadas em ordem crescente de número de grupos, observa-se que os comportamentos das soluções são diferentes antes e depois do número verdadeiro de grupos.

Para um número de grupos menor que o verdadeiro, espera-se que a razão entre a compactação e conectividade seja grande, e para soluções com mais grupos que o verdadeiro essa razão tende a diminuir. Isso ocorre pois a compactação tende a cair muito quando se separa um grupo em dois grupos verdadeiros, porém, seu valor diminui pouco se dividido um grupo real. Desta forma, espera-se que o gráfico da relação entre as soluções tenda a ter "picos" até o número real de grupos, e se mantenha relativamente estável após este número.

Para identificar a melhor solução na fronteira gera-se uma série de dados simulados baseados nos dados observados, sob a suposição de que não há grupos. Dessa forma é possível estimar uma relação de comparação com soluções referentes a existência de grupos ou não nos dados. Se o comportamento da solução é parecido com o comportamento dos dados simulados, então não há indícios da existência de grupos. Desta maneira, é possível afirmar que existem dois conjuntos de fronteiras para os dados: a fronteira de soluções formadas pelas soluções não dominadas para os dados reais, e as fronteiras de controle geradas pelas soluções não dominadas em cada conjunto de dados simulados, figura 2.11.



Fonte: Handl and Knowles (2007)

Figura 2.11: Representação de uma solução do MOCK, com fronteira de soluções, fronteiras de controle com sua respectivas *attainment surface*

A geração dos dados simulados é feita a partir de uma rotação ortogonal dos dados, partindo do mesmo princípio de uma análise de componentes principais. Os autovetores são gerados da matriz de covariância dos dados, gerando um hipercubo no autoespaço, em que os limites de cada hipercubo são usados como parâmetros para gerar N elementos uniformemente. Os valores uniformes gerados neste autoespaço são rotacionados para voltar à região original dos dados. É possível obter uma visão mais clara deste procedimento no esquema da figura 2.12.

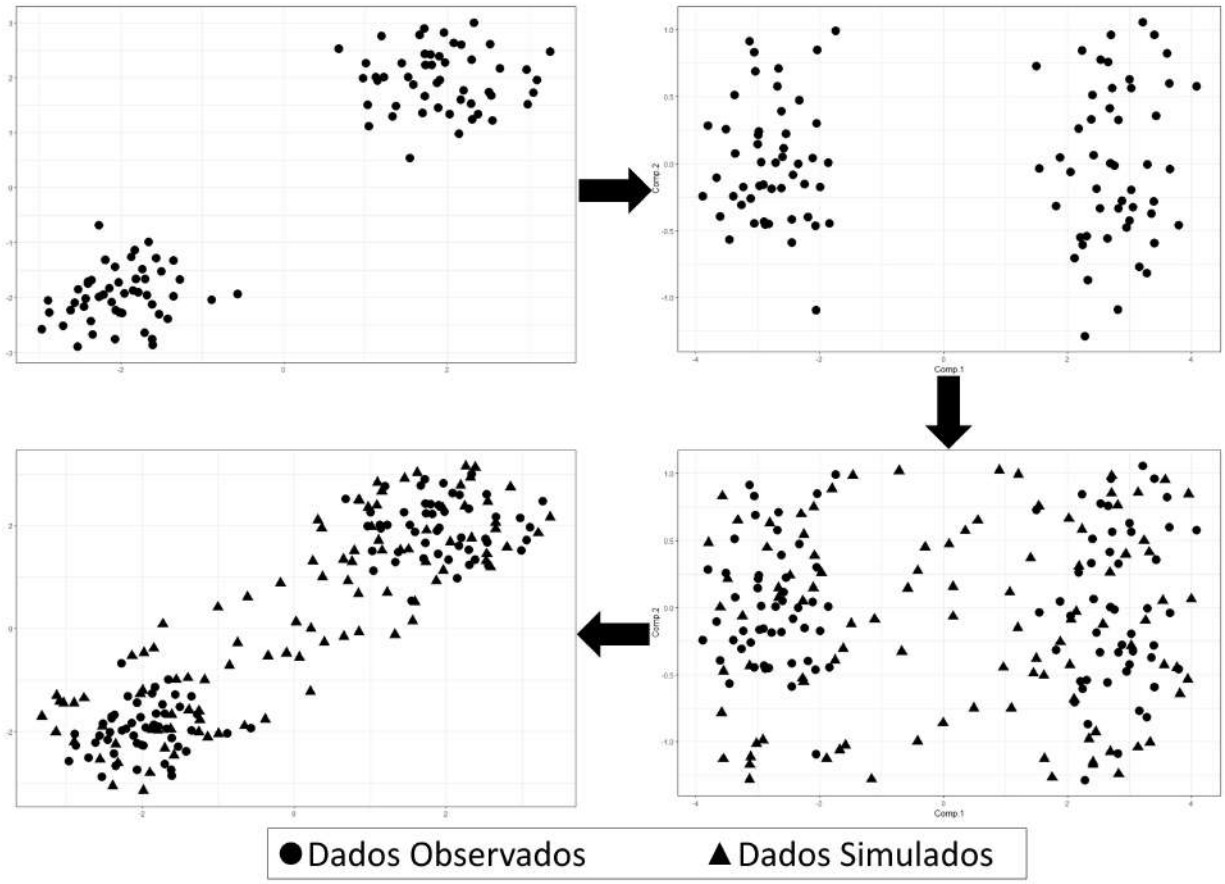


Figura 2.12: Esquema sequencial da geração de dados aleatórios. Imagem superior à esquerda mostra os dados observados; Imagem superior à direita mostra os dados rotacionados; Imagem inferior à direita mostrar a geração de dados uniformes na mesma região dos dados observados; Imagem inferior à esquerda mostra os dados gerados sofrendo transformação inversa.

Para cada conjunto de dados de controle o algoritmo MOCK é executado, gerando a respectiva fronteira de Pareto. Geradas as fronteiras de soluções para os dados reais e de controle, inicia-se o procedimento para identificação do número correto de grupos.

Primeiramente é limitado o espaço de busca pelo número máximo de grupos encontrados tanto para os dados reais quanto de controle. Este limite de grupos é determinado pelo parâmetro K_{user}. Em seguida, para cada fronteira são identificados os valores máximos e mínimos da conectividade e compactação, e todos os valores são então padronizados, limitando-se os resultados entre $[0, 1]$. Além disso, os resultados das funções-objetivo são novamente padronizados pela sua raiz quadrada, melhorando a precisão na identificação das melhores soluções, pois, segundo Handl and Knowles (2007) as duas funções-objetivo utilizadas têm uma relação quadrática.

Por fim, é calculada a *attainment surface* para a fronteira de soluções e de controle.

Essa superfície é definida como a fronteira que separa o espaço de objetivos em uma região em que qualquer ponto é dominado por ao menos um ponto do conjunto de soluções, e uma região em que qualquer ponto não é dominado por nenhuma solução obtida, como pode ser visto na figura 2.11.

Essa abordagem se faz necessária, pois não há informação entre as soluções, logo, qualquer suposição de que a fronteira de Pareto seja suave não é verdadeira. Para mais detalhes consultar Fonseca and Fleming (1996).

Com as superfícies estimadas, calcula-se o *attainment score*, que é a menor distância euclidiana entre um ponto na superfície das soluções para um ponto na superfície de controle. A solução com o maior attainment score é considerada a melhor solução.

2.8.6 Avarege silhouette width

Para comparar o MOCK com outros métodos, foi necessário escolher algoritmos que pudessem selecionar automaticamente os grupos. O principal método escolhido foi o *Avarege silhouette width*.

Um dos métodos mais citados para a definição de número de grupos, tem a característica de medir o quão bem uma observação pertence a um determinado grupo. Para cada observação i, associa-se um valor s_i chamado de *silhouette width*. Cada s_i é dado pela seguinte fórmula:

$$s_i = \frac{(b_i - a_i)}{max(a_i, b_i)}$$

sendo

- b_i = menor distancia entre a observação i e qualquer outra observação de qualquer outro grupo que não seja o mesmo de i, e
- a_i = média das distancias entre i e as observações do grupo ao qual pertence i.

Assim, quanto mais próximo o valor de s_i está de 1 melhor o ponto i está ajustado ao seu grupo, ao passo que, um valor próximo de 0 indica que provavelmente esse ponto está na fronteira entre dois grupos, e, se $s_i < 0$ provavelmente o ponto está associado ao grupo errado.

Para a seleção automática de grupo é utilizada a média das somas de s_i de cada grupo, sendo o maior valor aquele que indica o melhor número de grupos.

Vale ressaltar que o *Avarege silhouette width* não elimina a necessidade de escolha adequada do algoritmo de clusterização, pois esse método apenas indica o melhor número de grupos possível dentro de pré-seleções feitas pelo algoritmo escolhido (e.g k-means, dbscan e etc.).

2.8.7 Índice de Rand Ajustado

O objetivo do índice de Rand IRA ajustado é medir a qualidade da clusterização obtida. Para isso, leva em consideração a interseção entre os grupos encontrados e os pré-definidos, em que $IRA \in (-\infty, 1]$, e quanto mais próximo de 1 melhor o ajuste de grupos.

Seja um conjunto de dados, separado em grupos com $\boldsymbol{K} = K_1, ..., K_r$ e uma série de grupos encontrados por método de clusterização $\boldsymbol{C} = C_1, ..., C_s$ para $r, s \leq n$. Seja também $n_{i,j}$ a contagem do número de observações na interseções $C_i \cap K_j$. Temos, portanto, a seguinte tabela de contingência:

	K_1	K_2	...	K_s	Soma Linha
C_1	n_{11}	n_{12}	...	n_{1s}	a_1
C_2	n_{21}	n_{22}	...	n_{2s}	a_2
...	...	...	...	...	...
C_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Soma Coluna	b_1	b_2	...	b_s	

O IRA é então calculado como

$$IRA = \frac{\sum_{ij}\left(\frac{n_{ij}}{2}\right) - \left[\sum_i \frac{a_i}{2} \sum_j \frac{b_j}{2}\right] / \left(\frac{n}{2}\right)}{\frac{1}{2}\left[\sum_i \frac{a_i}{2} + \sum_j \frac{b_j}{2}\right] - \left[\sum_i \frac{a_i}{2} \sum_j \frac{b_j}{2}\right] / \left(\frac{n}{2}\right)}.$$

Capítulo 3

Análises e Resultados

3.1 Reprodução dos resultados do artigo

Um dos objetivos propostos era a reprodução do resultados obtidos no artigo Handl and Knowles (2007). Obteve-se então um conjunto de 80 base de dados com diversas características, sendo estas aplicadas a vários métodos de classificação. Para cada configuração de base de dados, foram classificados 10 conjuntos de dados gerados ao acaso. As bases de dados podem ser obtidas no link `https://personalpages.manchester.ac.uk/staff/Julia.Handl/generators.html`

Os métodos de classificação utilizados para comparação com o MOCK foram o K-means e três algoritmos hierárquicos, com diferentes métodos de separação de grupos. Entre os métodos hierárquicos foram contemplados a ligação simples, ligação complete e critério de Ward, sendo estes métodos já descritos anteriormente neste texto. Estas técnicas foram escolhidas devido a boas propriedades para determinadas estruturas de dados e por serem alguns dos tipos mais utilizados de algoritmos para classificação de dados.

Nenhum dos algoritmos empregados dispõe de métodos de seleção automática de grupos, portanto, aplicou-se 3 métodos distintos de seleção automatizada de grupos, também muito utilizados em análises de dados. Este métodos são o *Avarege silhouette width*, estatística GAP e uma classificação por método *ensemble* ou agregado.

Métodos *ensemble* nada mais são que a utilização de dois ou mais métodos para seleção de número de grupos. Esses métodos são utilizados simultaneamente e, na maioria das vezes é feita uma "votação", em que a maioria define o número de grupos a ser escolhido, porém, é possível calcular outras estatísticas para definição além do critério da maioria.

O pacote NbClust do R foi o pacote que possibilitou mais facilmente a utilização destes

diversos algoritmos, dado que este tem suporte para ao menos nove algoritmos de clusterização e contém aproximadamente 30 índices que podem ser utilizados para seleção de número de grupos. O método ensemble utilizado considera um votação vencida pela maioria deste índices.

A medida de avaliação de qualidade de ajuste empregado foi o índice de *Rand* ajustado. Para cada configuração dos conjuntos de dados e métodos empregado foi obtido um K e um IRA, sendo respectivamente a média de número de grupos e a média dos índice de Rand de 10 conjuntos com a mesma dimensão e número de grupos. Os resultados obtidos estão resumidos na Tabela 3.1 e na Figura 3.1.

Tabela 3.1 - Resultados obtidos para oito configurações de bases de dados diferentes

Dimensões	Número de grupos	MOCK		link completo_Sil		K-Means_Gap		K-means_Sil		link simplies_Sil		Ward_sil		K-means_all	
		K	IRA	K	IRA	K_Means	IRA	K_means	IRA	K	IRA	K	IRA	K	IRA
10d	10c	12,30	0,97	8,60	0,57	2,00	0,19	8,20	0,86	2,00	0,00	9,00	0,95	4,50	0,53
10d	20c	19,90	1,00	19,60	0,99	2,00	0,11	15,90	0,86	18,20	0,80	19,80	1,00	11,90	0,49
10d	40c	36,80	0,94	39,10	0,98	2,00	0,05	35,40	0,89	35,90	0,68	39,50	0,99	4,70	0,15
10d	4c	4,20	0,99	4,70	0,81	2,00	0,53	3,40	0,90	2,00	0,00	3,60	0,92	3,30	0,91
2d	10c	9,00	0,87	13,50	0,65	2,20	0,24	11,20	0,78	8,00	0,15	10,20	0,81	2,80	0,33
2d	20c	14,30	0,81	14,30	0,74	2,00	0,11	15,90	0,81	16,30	0,66	17,10	0,92	3,80	0,25
2d	40c	18,70	0,58	34,00	0,69	2,00	0,06	32,40	0,76	27,70	0,37	34,70	0,86	3,10	0,12
2d	4c	4,30	0,93	3,70	0,61	2,50	0,59	3,20	0,76	4,40	0,50	3,60	0,91	2,80	0,69

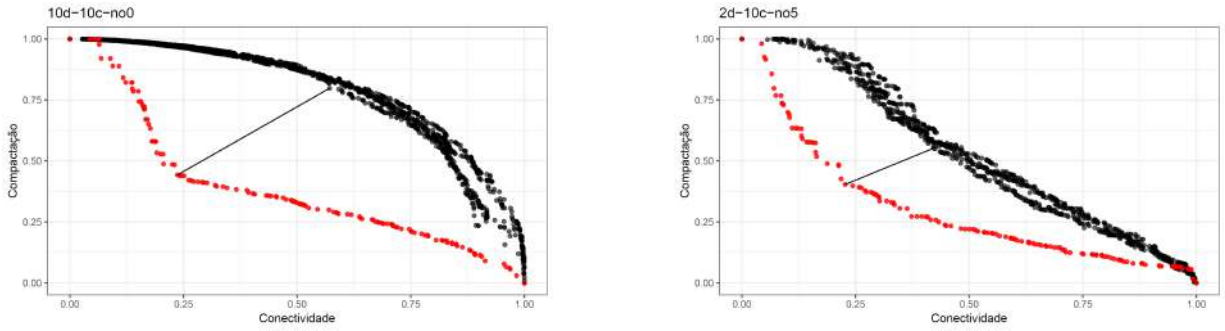


Figura 3.1: Exemplos de resultados obtidos através do MOCK para dois diferentes bancos de dados.

Observa-se na tabela 3.1 que os valores são semelhantes aos obtidos por Handl and Knowles (2007), mostrando que a implementação do MOCK em código R pode ser considerada um sucesso.

3.2 Aplicação em dados reais

Os dados utilizados (TCGA-PANCAN-HiSeq-801x20531), foram obtidos no *UCI Machine Learning Repository* `https://archive.ics.uci.edu/ml/datasets/gene+expression+`

`cancer+RNA-Seq`, e são parte do *The Cancer Genome Atlas Pan-Cancer analysis project*. Este projeto agrupa diversos tipos de dados sobre ao menos doze tipos diferentes de câncer, sendo extraídos através de diversas técnicas de ativações de DNA e RNA, Cancer Genome Atlas Research Network et al. (2013).

Especificamente, os dados aplicados ao MOCK, são um seleção aleatória de 801 observações de expressões genéticas de cinco tipo diferentes de câncer, extraídos por uma técnica de ativação de RNA chamada de HiSeq. Os tipos de câncer existentes na base de dados são:

- LUAD: Adenocarcinoma pulmonar
- BRCA: Carcinoma invasivo da mama
- KIRC: Carcinoma de células claras renais renais
- PRAD: Adenocarcinoma da próstata
- COAD: Adenocarcinoma do cólon

A uso deste conjunto de dados se deu principalmente por sua alta dimensionalidade. Neste caso são aproximadamente 25 mil colunas, em que cada coluna é um único gene e cada linha um paciente diagnosticado com um dos tipos de câncer descrito. Cabe ressaltar que o objetivo desta análise não é agregar valor quanto à classificação sobre os tipos de câncer, mesmo que isso possa ser feito, mas sim analisar o comportamento dos algoritmos de classificação junto às características destes dados.

Supôs-se que o algoritmo do MOCK não seria muito influenciado pela alta dimensionalidade da base de dados, deste modo, o tempo de execução não seria efetivamente afetado pela quantidade de colunas. Entretanto, o resultado não foi como esperado, pois a grande quantidade de colunas impactou significativamente o tempo de execução. Porém, foi constatado que este problema foi causado em maior parte pela não otimização do código desenvolvidos.

Outro fator importante foi a limitação da metodologia proposta para geração dos dados simulados para as fronteiras de controle. Esta por sua vez utiliza uma rotação ortogonal para que o valores gerados possam estar no mesmo formato dos valores observados (e.g Figura 2.12). Como o número de colunas é maior que o número de linhas não foi possível

executar a metodologia proposta para o MOCK. Assim sendo, pode-se considerar que essa seria uma limitação deste algoritmo a ser atacada, gerando um algoritmo mais abrangente e com maior precisão.

Para a execução do MOCK com o conjunto de dados genéticos foi utilizados uma abordagem mais simples. Gerar valores uniformes entre o mínimo e o máximo para cada gene, gerando assim um hipercubo multidimensional. Na Tabela 3.2 nota-se que ao menos três métodos podem ser comparados quanto ao IRA, tal que, o K-means e o critério de Ward, utilizam o silhoute width para seleção de grupos. Observa-se uma leve vantagem para MOCK, entretanto, o tempo de execução do MOCK é maior que dos outros métodos, valendo-se do fato que os códigos estão escritos em R e não otimizados, competiam com métodos já extremamente otimizados, e mesmo assim, estes por sua vez levaram um tempo considerável para sua execução pelo pacote NbClust.

Tabela 3.2 - Tabela de resultados obtidos a partir dados géticos RNA-Seq (HiSeq) PANCAN

Algoritmo	MOCK	K-means_sil	link simples_sil	link completo_sil	Ward_Sil
Número de grupos	6	5	2	5	6
IRA	0,9	0,89	0	0,85	0,89
Tempo de execução em horas	30	5,24	5,27	5,24	5,27

Apesar dos métodos citados poderem ser comparados com relação ao IRA, olhando paras as Tabelas 3.3 e 3.4, percebe-se que o K-means obteve um resultado geral bastante diferente do MOCK. Ainda que o K-means tenha encontra cinco grupos, as classes de tumores COAD e LUAD foram associados ao mesmo grupo, já o MOCK separou todos os outros grupos de maneira mais precisa mesmo que tenha criado um grupo a mais. Isso provavelmente ocorreu por vantagem do MOCK da utilização de mais de um objetivo com deferentes características, possibilitando a identificação de estruturas as quais o K-means não foi capaz de reconhecer. Ademais, é possível supor que o MOCK pudesse ter um resultado melhor caso a geração dos dados simulados tivessem sido gerados de maneira mais precisa. Na imagem 3.2, observam-se as fronteiras de Pareto e de controle da base de dados utilizada.

Tabela 3.3 - Tabela dos grupos obtidos a partir dados géticos RNA-Seq (HiSeq) PANCAN para o K-means com o Avarage Silhoutte Width

Classe	1	2	3	4	5	Total
BRCA			50		250	300
COAD		78				78
KIRC	145	1				146
LUAD		139	2			141
PRAD				136		136
Total	145	218	52	136	250	801

Tabela 3.4 - Tabela dos grupos obtidos a partir dados géticos RNA-Seq (HiSeq) PANCAN para o MOCK

Classe	1	2	3	4	5	6	Total
BRCA			253		47		300
COAD						78	78
KIRC				146			146
LUAD		139			2		141
PRAD	136						136
Total	136	139	253	146	49	78	801

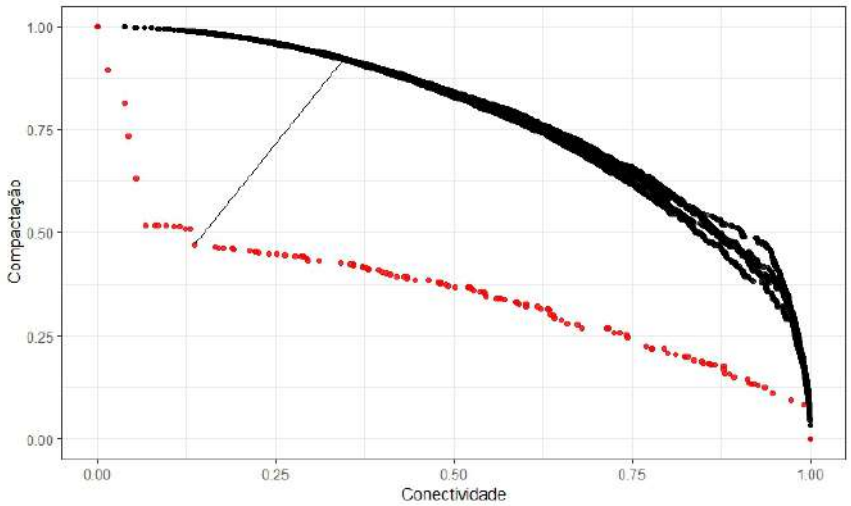


Figura 3.2: Resultado do MOCK para base de dados genéticos

3.3 *Códigos do MOCK*

Mesmo que não se tenha criado um pacote no sentido estrito da palavra, os código e bases de dados deste trabalho podem ser encontrados no github no repositório: `https:`

//github.com/mateuscarbone/MOCK_R, estando aberto para contribuições e possíveis melhorias.

Capítulo 4

Conclusões

Este trabalho assumiu o objetivo de reproduzir e compreender o funcionamento do algoritmo MOCK para classificação de dados, frente a grande quantidade de conceitos utilizados para o seu desenvolvimento. Outro objetivo foi a implementação do algoritmo em linguagem R para que possa ser utilizado, entendido e aprimorado por outras pessoas. Pode-se dizer que nestes dois aspectos foi obtido sucesso, devido à complexidade do projeto proposto.

O MOCK possui em seu cerne diversos conceitos absorvidos de vários outros métodos de agrupamento. Partindo, por exemplo, de árvores geradoras mínimas, que são base de alguns métodos hierárquicos. Passando pela geração de populações com o uso do K-means, até a utilização de duas funções-objetivo que possuem a finalidade de traduzir aspectos conflitantes de alguns algoritmos de clusterização. Sendo estes, proximidade entre as observações do mesmo grupo e entre grupos através da compactação, distância espacial e proximidade entre vizinho através da conectividade.

Outro ponto importante é o emprego de um algoritmo genético multiobjetivo que pode ser considerado um grande diferencial para o MOCK, dada sua versatilidade nas mais diversas situações, além da seleção automatizada de grupos que se mostrou bastante eficiente.

Um objetivo que infelizmente não foi possível ser alcançado foi a melhoria da seleção automática de grupos. Fica claro na Imagem 3.2, que a despeito de ter selecionado a melhor solução entre todos os algoritmos comparados, não foi selecionada a melhor solução dentre todas as geradas pelo algoritmo. Quando se observa os saltos na fronteira de soluções geradas, identifica-se que existe a possibilidade de que uma solução melhor possa ter sido encontrada. Porém, esta não foi selecionada devido à limitação da metodologia utilizando

as *attainment surfaces*. O MOCK pode ser considerados como "estado da arte"com relação métodos de agrupamento, mas se beneficiaria enormemente de critérios para seleção de grupos mais aprimorados.

Outra limitação encontrada foi com relação a alta dimensionalidade, independentemente da solução utilizada neste trabalho, para a criação de dados simulados. Quando este problema ocorre, seria possível a proposição de alguma solução que pudesse ser eficiente e mais precisa quando o número de dimensões ultrapassa o número de observações. Além disso, com o auxílio de um grupo maior de pessoas utilizando a implementação em R, pode-se se obter um algoritmo mais eficiente que o escrito até então, pois, mesmo que tenha havido um esforço para a melhor performance do MOCK, não estava no escopo deste trabalho a criação de um algoritmo otimizado.

De acordo com os resultados já obtidos pelos autores do MOCK e reproduzidos nessa dissertação, o MOCK pode ser usado em uma grande variedade de situações, como em dados genéticos, utilizado neste trabalho, ou em segmentação de mercado, detendo-se em apenas dois exemplos. Estes dois tipos de problemas geralmente necessitam de uma precisão elevada, e como o MOCK se mostrou versátil, de forma que, conseguiu competir com uma enorme variedade de métodos, mesmo em situações em que estes sabidamente já eram muito eficazes, logo, para exploração e mineração de dados o MOCK pode ser uma ferramenta bastante útil.

Por fim este projeto contribuiu para o conhecimento e disseminação do MOCK como método de agrupamento de dados. Dada sua importância e sua capacidade, neste trabalho, tocou-se apenas superficialmente em aspectos metodológicos, levando a crer que este possa ainda ser alvo de estudos mais aprofundados agregando ainda mais valor a este método.

Referências Bibliográficas

C. C. Aggarwal and C. K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC, 1st edition, 2013. ISBN 1466558210, 9781466558212.

N. Andreasson, A. Evgrafov, M. Patriksson, E. Gustavsson, and M. Onnheim. *Introduction to Continuous Optimization*. Edition 2 : 1. Studentlitteratur AB, 2013. ISBN 9789144060774. URL `https://books.google.com.br/books?id=AHqYngEACAAJ`.

S. Bechikh, R. Datta, and A. Gupta, editors. *Recent Advances in Evolutionary Multi-objective Optimization*, volume 20 of *Adaptation, Learning, and Optimization*. Springer, 2017. ISBN 978-3-319-42977-9. doi: 10.1007/978-3-319-42978-6. URL `https://doi.org/10.1007/978-3-319-42978-6`.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

Cancer Genome Atlas Research Network, J. N. Weinstein, E. A. Collisson, G. B. Mills, K. R. Shaw, B. A. Ozenberger, K. Ellrott, I. Shmulevich, C. Sander, and J. M. Stuart. The cancer genome atlas pan-cancer analysis project. *Nat Genet*, 45(10):1113–1120, Oct. 2013. doi: 10.1038/ng.2764. URL `https://www.ncbi.nlm.nih.gov/pubmed/24071849?dopt=Abstract`. 37

L. N. d. Castro. *Fundamentals of Natural Computing (Chapman & Hall/Crc Computer and Information Sciences)*. Chapman & Hall/CRC, 2006. ISBN 1584886439.

M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs. Nbclust: Determining the best number of clusters in a data set, 2015. URL `https://cran.r-project.org/web/packages/NbClust/index.html`.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844, 9780262033848.

D. W. Corne, J. D. Knowles, and M. J. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848. Springer, 2000.

D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, and M. J. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001*, pages 283–290. Morgan Kaufmann Publishers, 2001. 22

K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001. ISBN 047187339X.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996. URL `http://dl.acm.org/citation.cfm?id=3001460.3001507`.

C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, PPSN IV, pages 584–593, London, UK, UK, 1996. Springer-Verlag. ISBN 3-540-61723-X. URL `http://dl.acm.org/citation.cfm?id=645823.670856`. 33

W. Forst and D. Hoffmann. *Optimization—Theory and Practice*. Springer Undergraduate Texts in Mathematics and Technology. Springer New York, 2010. ISBN 9780387789774. URL `https://books.google.com.br/books?id=vkLR9w7ro2QC`.

P. Gabriel and A. Delbem. *Fundamentos de algoritmos evolutivos*. ICMC-USP, 2008. URL `https://books.google.com.br/books?id=XGj_PgAACAAJ`.

J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. In *Techinical report TR-COMPSYSBIO-2004-02*, UMIST, Manchester, England, 2004. 25

J. Handl and J. Knowles. Improvements to the scalability of multiobjective clustering. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2372–2379 Vol. 3, Sept 2005. doi: 10.1109/CEC.2005.1554990.

J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *Trans. Evol. Comp*, 11(1):56–76, Feb. 2007. ISSN 1089-778X. doi: 10.1109/TEVC.2006.877146. URL `http://dx.doi.org/10.1109/TEVC.2006.877146`. 9, 11, 3, 4, 32, 35, 36

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 02 1985. doi: 10.1007/BF01908075.

A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8): 651–666, June 2010. ISSN 0167-8655. doi: 10.1016/j.patrec.2009.09.011. URL `http://dx.doi.org/10.1016/j.patrec.2009.09.011`. 2

Y. Jin, editor. *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, 2006. ISBN 978-3-540-30676-4.

M. Li, S. Yang, X. Liu, and K. Wang. Ipesa-ii: Improved pareto envelope-based selection algorithm ii. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-Criterion Optimization*, pages 143–155, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Y. Pi, W. Liao, M. Liu, and J. Lu. Theory of cognitive pattern recognition. In P.-Y. Yin, editor, *Pattern recognition techniques, technology and applications*, pages 433–463. InTech Journals, 2008. ISBN 9789537619244. 2

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL `http://www.sciencedirect.com/science/article/pii/0377042787901257`.

R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. 63:411–423, 2000.

E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization, 01 2003.