UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE COMPUTAÇÃO E TECNOLOGIA DA INFORMAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EDUARDO ANDREETTA FONTANA

ALGORITMOS DE CLUSTERIZAÇÃO APLICADOS NA ANÁLISE GENÔMICA DA BACTÉRIA *Escherichia coli*

CAXIAS DO SUL

2013

EDUARDO ANDREETTA FONTANA

ALGORITMOS DE CLUSTERIZAÇÃO APLICADOS NA ANÁLISE GENÔMICA DA BACTÉRIA *Escherichia coli*

Trabalho de conclusão de curso para graduação em Ciência da Computação pela Universidade de Caxias do Sul.
Área de concentração: biotecnologia
Orientador Prof. ª Dr.ª Scheila de Avila e Silva

CAXIAS DO SUL

2013

EDUARDO ANDREETTA FONTANA

ALGORITMOS DE CLUSTERIZAÇÃO APLICADOS NA ANÁLISE GENÔMICA DA BACTÉRIA *Escherichia coli*

Trabalho de conclusão de curso para graduação em Ciência da Computação pela Universidade de Caxias do Sul.
Área de concentração: biotecnologia
Orientador Prof. ª Dr.ª Scheila de Avila e Silva

Aprovado em 28/11/2013

Banca Examinadora

_____
Prof. Dr.ª Scheila de Avila e Silva
UNIVERSIDADE DE CAXIAS DO SUL – UCS

_____
Prof. Dr. André Gustavo Adami
UNIVERSIDADE DE CAXIAS DO SUL – UCS

_____
Prof. Dr. André Luis Martinotto
UNIVERSIDADE DE CAXIAS DO SUL – UCS

RESUMO

Um volume crescente de dados de sequências genômicas está disponível em banco de dados públicos e privados. Além desse armazenamento ocorre, paralelamente, a necessidade de análise desses dados através de plataformas computacionais. A exploração destes dados por um processo mediado por computador caracteriza-se uma tarefa de mineração de dados e pode ser realizada por meio de diferentes abordagens, dentre as quais a clusterização (*clustering*, agrupamento, ou análise de *cluster*). Este trabalho consiste em efetuar uma análise genômica do *DNA* da bactéria *E. coli (Escherichia coli)* através da aplicação dos métodos de clusterização como os algoritmos *K-Means* e *CURE (Clustering Using Representatives* – Clusterização Utilizando Representantes*)*. Estes algoritmos foram implementados em uma linguagem de programação de modo a processar os segmentos de promotores, genes e terminadores do *DNA* da bactéria, através do uso de ferramentas diferenciadas e uma metodologia própria, com base nas etapas da análise de *cluster*. Os resultados obtidos foram analisados e comparados de modo a demonstrar a eficácia do reconhecimento de padrões por vias computacionais, contribuindo assim nos estudos de pesquisas com padrões de genes, promotores e terminadores desta bactéria. Sendo que, os *clusters* gerados apresentaram aglomerações pertinentes ao contexto biológico, isto é, *cluster* com muitas sequências pertencentes a uma mesma classificação. Considerando que ambos os algoritmos apresentaram grupos de objetos bem definidos, em pelo menos um contexto de configuração, a técnica mostra-se válida e aberta a aplicações futuras.

Palavras-Chave: Bactéria. *Escherichia coli*. *DNA*. Algoritmo. Clusterização.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

SUMÁRIO

1 INTRODUÇÃO E JUSTIFICATIVA

A quantidade de dados, da biologia e demais ciências, encontra-se em amplificação e crescimento acelerado (CALLEBAUT, 2012). Este volume crescente de dados no mundo, principalmente os dados de sequências genômicas, está disponível em banco de dados públicos e privados. Além desse armazenamento ocorre, paralelamente, a necessidade de análise desses dados, o que torna indispensável à utilização de plataformas computacionais eficientes para a interpretação, oportunizando assim o surgimento da Bioinformática. Esta, por sua vez, ocasiona a unificação de diversas linhas de conhecimento como a matemática, a estatística, a ciência da computação e a biologia molecular (PROSDOCIMI *et al*., 2002).

Devido ao elevado crescimento de dados relacionados a sequenciamentos de *DNA* por vários projetos em andamento, observou-se que as transformações e as manipulações destes dados através de operações manuais são, muitas vezes, impraticáveis. Desta forma, tornou-se necessária à exploração destes dados por um processo mediado por computador. Com esta necessidade, a mineração de dados agregou-se como um campo de pesquisa a ser explorado, permitindo que muitos cientistas realizem pesquisas sem nem mesmo utilizar um laboratório ou uma máquina de sequenciamento (BERGERON, 2003).

As tarefas de mineração de dados comumente aplicadas nos dados de sequências de *DNA* são agrupamentos (*clustering*, clusterização, ou análise de *cluster*) e classificação. *Clustering* é uma atividade puramente orientada a dados que utiliza apenas os dados do estudo ou experiência para agrupar as medições. A classificação, em contrapartida, utiliza dados adicionais, incluindo heurísticas e medidas para atribuir aos grupos (BERGERON, 2003). A tarefa de clusterização aborda a análise de dados através de atributos, com o objetivo de identificar padrões para a organização de conjuntos físicos ou abstratos de objetos similares. Consiste em criar agrupamentos de objetos que identificam uma classe de objetos similares entre si e dissimilares entre as classes (DE AMO, 2004; MAIMON e ROKACH, 2010; JAIN *et al*., 1999; SHAMIR e SHARAN, 2001). Estas tarefas de clusterização são compostas por algoritmos que são utilizados para a identificação de padrões relevantes em conjuntos de genes, com o objetivo de encontrar características importantes que compõem cada organismo vivo e, a partir delas, obter um estudo

mais eficaz em relação a estes organismos (SHAMIR e SHARAN, 2001). Além disso, é possível fornecer percepções sobre a função de um determinado segmento de *DNA*, como o gene. Por exemplo, se dois genes possuem características semelhantes observadas através de padrões da estrutura do segmento, eles podem estar funcionalmente relacionados (BERGERON, 2003). Dentre os organismos vivos, alguns são considerados modelos biológicos, por ser de fácil manipulação em laboratório. Um exemplo é a bactéria *Escherichia coli (E. coli)*, que vive no intestino do ser humano (TORTORA, 2005).

Este trabalho consiste em efetuar uma análise genômica do *DNA* da bactéria *E. coli* através da aplicação dos algoritmos de clusterização *K-Means* e *CURE (Clustering Using Representatives* – Clusterização Utilizando Representantes*)*. Deste modo, procurou-se identificar quais algoritmos de clusterização encontram melhores resultados na busca de padrões nas sequências genômicas da bactéria *E. coli*. Esta aplicação é realizada através da construção de um programa para a plataforma *desktop* que implementa os algoritmos citados por uma linguagem de programação. Os resultados obtidos são comparados entre si de modo a identificar a eficácia do reconhecimento de padrões. Além disso, os resultados são apresentados à área de biologia e são analisados de forma a contribuir nos estudos e pesquisas com padrões de genes, terminadores e promotores da bactéria *E. coli*.

1.1 ESTRUTURA DO TRABALHO

O estudo deste trabalho inicia-se pelo Capítulo 2, onde são abordados os fundamentos biológicos necessários para a clusterização de dados genéticos, no qual é descrita a bactéria *E. coli* e os fundamentos genéticos envolvidos na expressão gênica da mesma. No Capítulo 3, são descritos os algoritmos de clusterização (*K-Means* e *CURE*) utilizados na análise genômica, assim como as etapas necessárias para a realização do processo de análise de *cluster*, da qual possui a etapa de extração de atributos. Na descrição da extração de atributos são demonstrados os métodos de extração de atributos em escala linear e não linear e o método da coocorrência, complementares no processo de determinação de características para a formatação dos objetos do algoritmo. Ainda no Capítulo 3, são apresentados trabalhos de clusterização semelhantes, aplicados na área da biologia, biomedicina e genética, que sustentam a viabilidade de realização deste trabalho. No Capítulo 4 é apresentada a metodologia utilizada na solução desenvolvida. Neste capítulo, encontra-se o fluxo de trabalho realizado, fonte de dados, ferramentas utilizadas, entre outros. Finalizando o estudo, tem-se o Capítulo 5, onde são apresentados os resultados da solução aplicada e descritas as simulações de clusterização. Por último, conclui-se o estudo através do Capítulo 6, que envolve uma discussão dos fatos ocorridos no decorrer do processo, assim como sugestões de trabalhos futuros nesta área de pesquisa.

2 FUNDAMENTOS BIOLÓGICOS

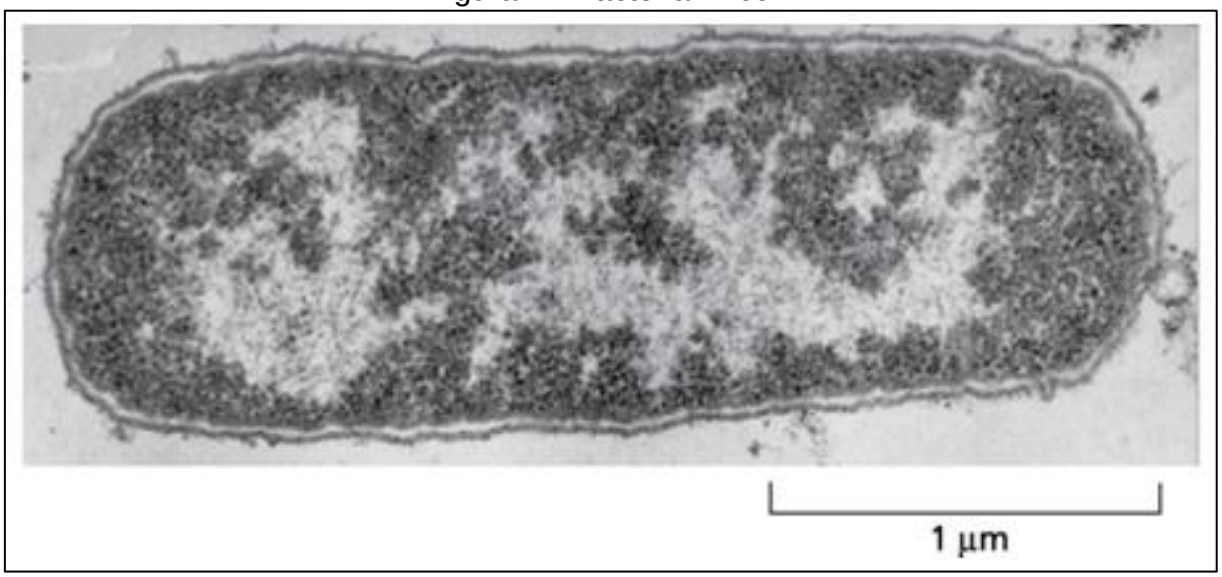
Neste capítulo são abordados os conceitos biológicos fundamentais para a compreensão dos objetivos do trabalho, bem como o melhor entendimento das implicações biológicas presentes nos resultados obtidos. Deste modo, não serão apresentados todos os detalhes biológicos relacionados com as sequências analisadas, uma vez que estas não são necessárias ao escopo deste trabalho.

2.1 A BACTÉRIA *ESCHERICHIA COLI*

A bactéria *Escherichia coli* (*E. coli*) é uma espécie de bactéria de fácil manipulação em laboratório, pois se reproduz com bastante facilidade, obtendo-se uma nova cópia em cerca de vinte minutos. Por este e outros motivos, é utilizada por biólogos como um organismo modelo (padrão biológico) na realização de pesquisas científicas (TORTORA, 2005). Assim como as algas cianofíceas, as bactérias, de um modo geral, são procariontes, ou seja, organismos de uma única célula em sua maioria e que não contém o *DNA* delimitado por uma membrana, ficando este disposto diretamente no citoplasma da célula (MURRAY *et al*., 2010).

A *E. coli* foi descrita pela primeira vez em 1885 pelo pediatra alemão *Theodore Escherich* e inicialmente sob a denominação de *Bacillus coli comune*. Porém, após uma revisão, foi renomeada para *Escherichia coli* em referência ao pesquisador que a descobriu (CHEN, 2005). De um modo geral, é classificada como uma bactéria que faz parte da família *Enterobacteriaceae*. Possui formato alongado e semelhante a um bastonete, designando-se assim um bacilo, conforme observa-se na Figura 1 (HOLT, 1993). São bactérias que vivem no intestino humano, sendo que a parceria de relação, neste caso, se constrói através da troca de vitaminas K e B criadas pela bactéria, e os nutrientes fornecidos pelo hospedeiro (TORTORA, 2005). Muitas cepas (variantes) da *E. coli* não são patogênicas, ou seja, não provocam doenças. Porém, as que provocam, podem resultar em infecções urinárias, septicemias, meningites e outros tipos de infecção como as intestinais e extraintestinais (TRABULSI, 1999).

Figura 1 - Bactéria *E. coli*



Fonte: ALBERTS *et al*. (2011)

A partir dos estudos genéticos realizados com esta bactéria, foram criadas diversas variantes a partir de mutações em seu genoma, sendo essas geradas espontaneamente ou a partir ações antrópicas (modificado pela ação humana). Dentre as cepas existentes está a *Escherichia coli K12 K12 MG1655*, a qual será analisada neste trabalho. Por se tratar de um organismo padrão, as inferências realizadas a partir dela podem ser aplicadas em outras bactérias (MURRAY *et al*., 2010).

2.2 FUNDAMENTOS GENÉTICOS DA EXPRESSÃO GÊNICA DE *E. coli*

Nesta seção são apresentados os conceitos genéticos elementares para compreensão de como os genes de uma bactéria são expressos durante os seus processos vitais.

A estrutura do *DNA (deoxyribonucleic acid – ácido desoxirribonucleico)* foi descrita originariamente por *James Watson e Francis Crick*, no ano de 1953. Através de estudos com base em vários trabalhos da época, publicaram na revista *Nature* um trabalho denominado *Molecular Structure of Nucleic Acids - A Structure for Desoxyribose Nucleic Acid* (Estrutura Molecular dos Ácidos Nucleicos - Uma Estrutura para o Ácido Desoxirribonucleico) (DE ROBERTIS, 1993 *apud* DE AVILA E SILVA, 2006).

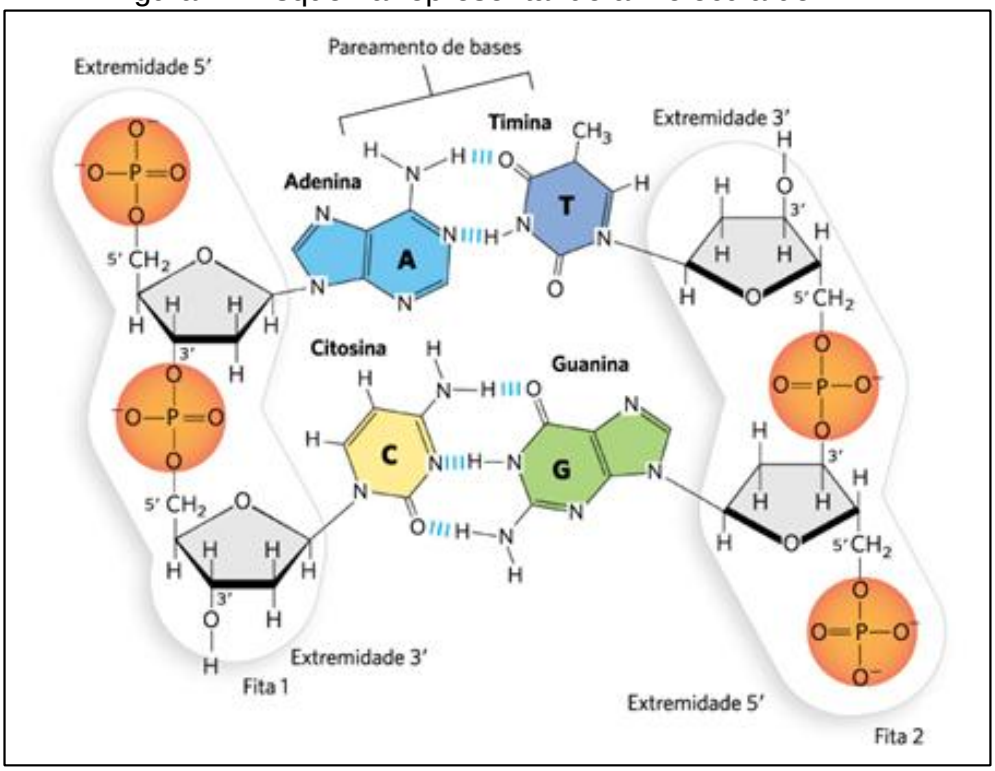
> O *DNA* ou ácido desoxirribonucleico é uma molécula que existe praticamente em todas as células dos organismos vivos e de alguns vírus, com a função de armazenar a informação genética (DE ROBERTIS, 1993 *apud* DE AVILA E SILVA, 2006, p. 19).

O *DNA* é um polímero composto de unidades químicas individuais chamadas de nucleotídeos. Essas bases se ligam em uma estrutura de forma alternada e repetitiva, formando assim uma cadeia, chamada de cadeia de *DNA*. As quatro bases nitrogenadas que compõem as sequências de *DNA* são: adenina (A), guanina (G), citosina (C) e timina (T) (BERG *et al*., 2002).

A cadeia de *DNA* é composta por duas fitas de nucleotídeos que se ligam entre si paralelamente. Em uma das fitas, na extremidade de cada um dos nucleotídeos, está uma base nitrogenada, que se liga com seu par alternadamente correspondente na fita oposta. As quatro bases nitrogenadas configuram ligações par a par, denominadas pares de bases (pb), entre si através das pontes de hidrogênio que, conforme sua variação de número de pontes, determina que A liga-se sempre com T e C liga-se sempre com G. Ou seja, A sempre estabelece ligação com um nucleotídeo de T e um nucleotídeo de C sempre estabelece ligação com G (ALBERTS *et al*., 2002).

Estes pares de fitas, unificados através de pontes de hidrogênio, enrolam-se em sentido vertical obtendo um formato helicoidal. Desta forma, a molécula é composta por duas fitas complementares e antiparalelas, conhecida como dupla-hélice (ALBERTS *et al*., 2002). Na Figura 2 observa-se parte do polímero e os seus elementos. A sequência representada é A-C na fita da esquerda (5’ - 3’) e sua sequência complementar T-G na fita da direita (3’ - 5’). As bases nitrogenadas estão no centro da molécula. As pontes de hidrogênio estão representadas pelo grupo destacado (DE AVILA E SILVA, 2006).

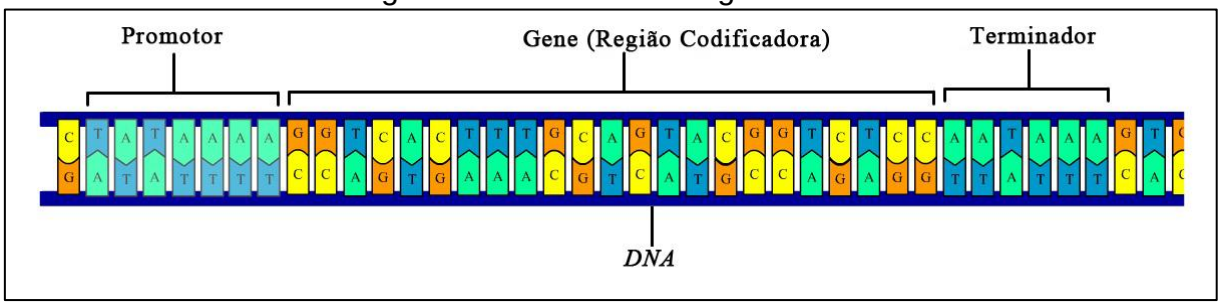
Figura 2 - Esquema representando a molécula de *DNA*



Fonte: COX *et al*. (2012)

Ao longo da extensão da fita de *DNA* encontra-se partes de segmentos de nucleotídeos responsáveis por diferentes funções biológicas. Deste modo, a molécula torna-se uma estrutura segmentada, na qual cada segmento possui um papel específico para a célula: promotores, região codificante (também chamada de gene) e terminadores, por exemplo, conforme observa-se na Figura 3.

Figura 3 - Fita de *DNA* segmentada



Fonte: GIANNINI (2013)

A expressão gênica é formada por dois processos: transcrição e tradução. A transcrição é a criação do *RNA* mensageiro (*RNAm, Ribonucleic Acid* - Ácido Ribonucleico) e a tradução se caracteriza pela síntese de proteínas.

Considerando a estrutura exposta, percebe-se que o *DNA* assume a função de armazenar as informações genéticas dos organismos, sendo estas informações decodificadas em ações por estruturas biológicas específicas. Assim, os promotores e terminadores possuem a informação necessária para controlar a expressão dos elementos codificadores. Desta forma, estes elementos possuem função regulatória, pois controlam a quantidade de produtos gênicos disponíveis na célula em um dado momento (LEWIN, 2001 *apud* DE AVILA E SILVA, 2006; SIVARAMAN *et al*., 2005).

> O estudo da expressão gênica, bem como de sua regulação, é um ponto importante de estudo, já que a transcrição (primeiro passo da expressão gênica) é regulada pela interação entre sequências específicas da molécula de *DNA* e proteínas que aí se ligam (DE AVILA E SILVA, 2006, p. 15).

Os promotores, ou a região promotora, são responsáveis pelo início do processo de transcrição, sendo necessário o seu reconhecimento pela enzima do *RNA* polimerase (*RNAp*). A localização dos promotores é ligeiramente anterior à região codificante (*Open reading frame - ORF*) (SIVARAMAN *et al*., 2005 *apud* DE AVILA E SILVA, 2006). Esta última por sua vez, é também chamada de gene, e pode ser definida como qualquer trecho de *DNA* que codifica um produto com função biológica potencialmente. A identificação de um gene é a primeira indicação de que um segmento de *DNA* pode ser funcional. Por fim, há os terminadores, os quais são responsáveis por finalizar o processo de transcrição desencadeado pela enzina *RNAp* (LEWIN, 2001).

Fazendo uma analogia com a computação, é possível associar o *DNA* a uma *string* de dados. Ou seja, uma sequência linear de informações que podem ser representadas por caracteres. Em termos de estrutura, o *DNA* se comporta como informação estática, semelhante a um banco de dados e, conforme a necessidade de utilização destas informações pelos demais sistemas celulares, as mesmas são consultadas de uma forma controlada pela enzima *RNAp*, semelhante a um programa de computador, escrito por uma linguagem específica, em consulta a uma base de dados (HOWARD e BENSON, 2002).

Neste trabalho, a composição estrutural apresentada pelos promotores, genes e terminadores, será de grande relevância, pois a partir dela serão analisados padrões estruturais ocultos, por meio de metodologia computacional (ou seja, uma metodologia *in silico)*.

2.3 CONSIDERAÇÕES FINAIS

Os fenômenos biológicos são complexos e requerem integração de muitas áreas do conhecimento para ampla compreensão de suas implicações (BARRERA *et al*, 2004). Neste capítulo, procurou-se abranger apenas os conceitos relacionados aos processos que envolvem genes, promotores e terminadores, uma vez que estes foram os elementos escolhidos para a análise deste trabalho. Além disso, considerando a ampla gama de seres vivos, alguns são considerados modelos como a bactéria *E. coli*. Deste modo, o conhecimento gerado a partir dela pode ser expandido para outras bactérias.

3 ALGORITMOS DE CLUSTERIZAÇÃO E APLICAÇÕES NA BIOLOGIA

Este capítulo apresenta a fundamentação teórica sobre os algoritmos de clusterização, procurando detalhar os aspectos computacionais e matemáticos, bem como, estabelecendo relação com trabalho de cunhos biológicos.

A análise de *cluster*, conhecida também como clusterização (*clustering*), é uma tarefa de mineração de dados que consiste em um processo de analisar dados, através de seus atributos, objetivando a identificação de padrões para a organização de conjuntos físicos ou abstratos de objetos similares (também denominados elementos ou componentes). Através dela, é possível criar agrupamentos que identificam uma classe de objetos similares entre si e dissimilares entre as classes. Estas classes de objetos, ou grupos de objetos similares, são denominadas *cluster*. A clusterização é uma tarefa não supervisionada, ou seja, não necessita de um aprendizado prévio para a identificação de padrões, diferentemente da classificação convencional utilizada em redes neurais (inteligência artificial), na qual é necessário treinar a rede com um escopo resumido de dados para que a esta possa identificar padrões em dados em escopos diferentes. Esta forma, auto-organizável de selecionar os dados, determina uma aprendizagem *por observação*, na qual o algoritmo aprende observando um único escopo dados, ao mesmo tempo em que o classifica. Em consequência disso, ao iniciar o processo de análise, não é necessário determinar etiquetas (classes) previamente conhecidas (DE AMO, 2004; MAIMON e ROKACH, 2010; JAIN *et al*., 1999; SHAMIR e SHARAN, 2001).

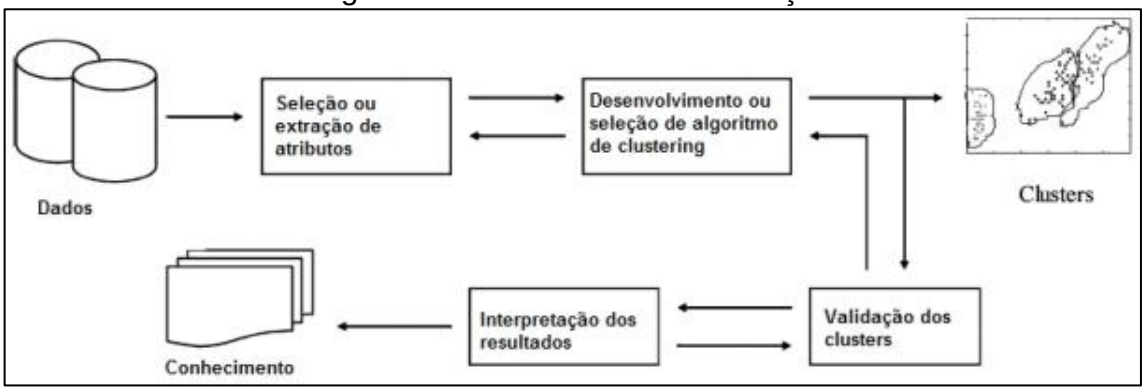
O processo de análise dos dados é realizado através da comparação de seus atributos. Estes atributos são elencados através da extração das características relevantes de cada tipo de dados. A realização desta seleção e extração de atributos é tão fundamental quanto à execução do algoritmo em si. Com a comparação dos atributos, é definida a similaridade entre os elementos do *cluster*, representada pela distância entre estes, que é calculada a partir de equações específicas de cada algoritmo. Desta forma, quanto maior a distância entre um elemento e outro, maior a probabilidade destes elementos pertencerem a *clusters* diferentes. O inverso também é verdadeiro, ou seja, quanto menor a distância entre os elementos, maior é a probabilidade de pertencerem ao mesmo *cluster* e fazer parte da mesma categoria (DE AMO, 2004). A fundamentação teórica da seleção dos atributos pode ser encontrada na Seção 3.2**.**

3.1 ETAPAS DA ANÁLISE DE *CLUSTER*

A aplicação dos algoritmos somente se torna possível após algumas etapas essenciais, de preparação dos dados de entrada, conforme o algoritmo escolhido. Após a aplicação do algoritmo, a validação e interpretação dos resultados são igualmente importantes para a obtenção de dados significativos na questão em análise.

A clusterização possui fases que segmentam o processo de analisar os dados. Conforme Xu e Wunsch (2005), descreve-se uma sequência de quatro principais estágios de operações que devem ser realizadas. Trata-se da seleção ou extração de atributos, desenvolvimento ou seleção do algoritmo de clusterização, validação dos *clusters* e interpretação dos resultados, conforme observa-se na Figura 4. Estes passos são retroalimentáveis, desta forma, estão intimamente relacionados e afetam os *clusters* resultantes. Consequentemente, é um processo que deve ser repetido múltiplas vezes, até atingir um resultado significativo para o objetivo proposto.

Figura 4 - Processo de clusterização



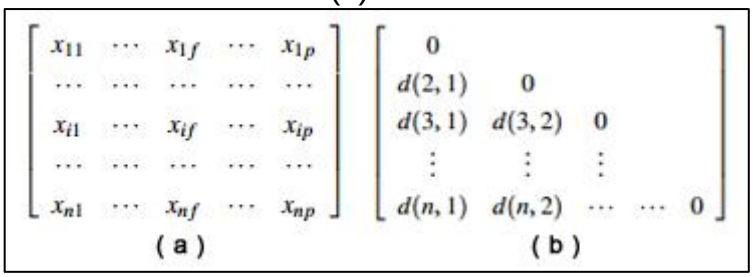
Fonte: XU e WUNSCH (2005), p. 646

O primeiro estágio é a seleção ou extração de atributos. Esta fase tem por objetivo a organização dos dados através de suas características de forma a definir os atributos que são utilizados pelo algoritmo no cálculo de distância entre os objetos do *cluster*. Esta operação pode, conforme a necessidade, efetuar transformações nos dados originais, modificando estes dados a fim de qualificar a extração, diminuindo a carga de trabalho dos algoritmos (JAIN e DUBES, 1998; JAIN *et al*., 1999; XU e WUNSCH, 2005).

Os dados são arranjados em uma matriz de objetos, disposta em duas estruturas diferenciadas, escolhida conforme a abordagem utilizada na extração dos atributos. A primeira estrutura contém, em cada linha, um objeto e, em cada coluna, um atributo que caracterizam estes objetos, definindo assim uma matriz de dados (HAN e KAMBER, 2001; DE AMO, 2004), como mostra a Figura 5 (a). A segunda estrutura contém as distâncias entre dois objetos previamente calculadas, no qual o elemento da coluna *j* e linha *i* da matriz é o número $d(i,j)$ que representa a distância (dissimilaridade) entre os objetos *i* e *j*, ver Figura 5 (b), desta forma é denominada matriz de dissimilaridade (DE AMO, 2004). O cálculo de obtenção da dissimilaridade, a extração dos valores da matriz de dados e as possíveis transformações dos dados originais para a obtenção destes atributos, serão detalhados na Seção 3.2.

A geração da matriz de dados e de dissimilaridade é realizada normalmente durante a fase de seleção ou extração de atributos. No entanto, podem ser alteradas e/ou recriadas durante a fase de execução (desenvolvimento ou seleção do algoritmo de clusterização), para realimentar o sistema (HAN e KAMBER, 2001; DE AMO, 2004).

Figura 5 - Matriz de dados (a) e matriz de dissimilaridade (b)



Fonte: HAN e KAMBER (2001)

Para que uma distância existente na matriz de dissimilaridade seja válida, é necessário e suficiente que as seguintes condições sejam satisfeitas, para quaisquer objetos i, j, k, indiferentemente da equação utilizada.

1. $d(i,j) \geq 0$;
2. $d(i,i) = 0$;
3. $d(i,j) = d(j,i) \rightarrow simetria$;
4. $d(i,j) \leq d(i,k) + d(k,j) \rightarrow desigualdade\ triangular$.

A condição 1 implica que todos os elementos da matriz de dissimilaridade são positivos (incluindo zero). A condição 2 implica que a diagonal da matriz de

dissimilaridade é formada por zeros. A condição 3, por sua vez, implica que a matriz de dissimilaridade é simétrica com relação à diagonal e por isso, registra-se nela somente os elementos abaixo da diagonal. A condição 4 trata da desigualdade triangular, no qual o comprimento de um dos lados é sempre inferior a soma do comprimento dos outros dois lados. Assim, qualquer equação que satisfaz às quatro condições pode gerar uma distância válida (DE AMO, 2004; JAIN e DUBES, 1998). O detalhamento matemático para a obtenção dos valores de distância, analisados nas condições de 1 a 4, são detalhados na Seção 3.2.

O segundo estágio é o desenvolvimento ou seleção de um algoritmo de *clustering*, devidamente apropriado para o conjunto de dados. Uma vez escolhidos os atributos, a construção ou reutilização de um algoritmo de clusterização torna o reconhecimento dos *clusters* um problema de otimização, o qual é definido matematicamente e traz consigo várias soluções disponíveis na literatura. É importante investigar o problema em questão para selecionar um algoritmo de clusterização adequado, uma vez que não existe um algoritmo suficientemente abstrato que abrange todas as áreas, sendo todos muito específicos ao contexto do problema (KLEINBERG, 2003). A descrição detalhada dos algoritmos de clusterização encontra-se na Seção 3.3.

O terceiro estágio é a validação dos *clusters*. Nesta fase, verificam-se os agrupamentos gerados e identifica-se a coerência entre os agrupamentos. É preciso utilizar critérios de avaliação eficientes para identificar os padrões, de modo que o resultado obtido a partir do algoritmo seja confiável. Deve-se, para isso, responder questões como: quantos *clusters* estão escondidos dentre os dados? Outra questão é: por que se escolhe um algoritmo ou método ao invés de outro? A validação matemática é diferente para cada algoritmo empregado (JAIN *et al*., 1999). À medida que os resultados são obtidos, pode-se discernir entre modificar ou readaptar o método de clusterização escolhido. Caso não seja alterado, pode-se continuar reaplicando o mesmo método, prosseguindo assim com o objetivo de conseguir um resultado que realize a distribuição dos grupos da melhor maneira possível (JAIN e DUBES, 1998; XU e WUNSCH, 2005).

O quarto estágio é a interpretação dos resultados. Através desta interpretação busca-se gerar conhecimento para o problema em questão. Geralmente, esta fase é realizada com o acompanhamento de especialistas de

domínio do contexto do problema, para garantir confiabilidade no conhecimento extraído (XU e WUNSCH, 2005).

3.2 EXTRAÇÃO DE ATRIBUTOS

Nesta seção são apresentadas algumas formas de seleção e extração de atributos. Através delas, é possível mensurar as características de cada elemento, as quais assumem o formato de atributo (DE AMO, 2004). As formas aqui apresentadas são: extração por atributos contínuos lineares e não lineares e a extração pelo método de coocorrência. A descrição matemática para estes métodos é de contexto genérico e não específicas para o problema tratado neste trabalho. Outras formas de organização de dados, como extração por atributos binários, extração por atributos nominais e ordinais, e extração de atributos mistos, não serão descritos, visto que não são pertinentes ao escopo do trabalho. Mais detalhes sobre estes outros métodos de extração podem ser encontrados em Han e Kamber (2001), Kaufman e Rousseeuw (2005), Maimon e Rokach (2010) e Jain e Dubes (1998). Salienta-se a extração de atributos é ampla e dinâmica e deve ser trabalhada de forma a organizar os dados realçando as características mais importantes dos mesmos, de acordo com o contexto do problema.

3.2.1 Extração de atributos em escala linear e não linear

Os atributos em escala linear são utilizados nos cenários em que os dados estão concentrados em uma escala de valores contínuos, ou seja, com uma alternância constante de valores dentro de uma ordem de grandeza representada. Como por exemplo: peso, altura, latitude, longitude, temperatura, entre outros. A extração deste tipo de atributo, na maioria dos casos, pode-se ser realizada diretamente a partir dos valores originais (utilizando ou não formas de padronização), originando diretamente a matriz de dados, na qual cada coluna é um atributo de valor linear. Alguns algoritmos, como o *K-Means* e *CURE*, utilizam-se apenas desta matriz como parâmetro de entrada de dados, sem a necessidade de utilizar as equações para cálculo da distância preliminar (DE AMO, 2004). Entretanto, as unidades de medida utilizadas para medir estes valores (kg, g, metro,

cm, entre outros) podem afetar a análise de *clusters*, de forma que, se a unidade possuir uma escala elevada, têm-se poucos *clusters*. Porém, se a escala for refinada ou diminuta, têm-se agrupamentos em demasia. Apenas nesta circunstância, antes de realizar a extração dos atributos para geração da matriz de dados, deve-se padronizar os mesmos (DE AMO, 2004; HAN e KAMBER, 2001).

A padronização dos dados tem como objetivo dar um peso igual a cada um dos atributos. Para isso, aplica-se o seguinte procedimento de padronização (HAN e KAMBER, 2001):

1. Calcula-se o desvio médio absoluto para cada atributo A_f:

$$s_f = \frac{1}{n}\sum_{i=1}^{n}\left|x_{if} - m_f\right|, \tag{1}$$

sendo m_f = valor médio do atributo A_f, x_{if} = o valor de A_f para um objeto *i* e *n* = quantidade de objetos. Observa-se que s_f é um valor associado à coluna *f* da matriz de dados (Figura 5), onde são operados com os valores x_{if} da coordenada *f* de cada objeto x_i.

2. Calcula-se a medida padrão ou *z-score* para o atributo *f* de cada objeto *i*:

$$z_{if} = \frac{x_{if} - m_f}{s_f}, \tag{2}$$

sendo z_{if} = valor padronizado do elemento x_{if}.

O desvio médio absoluto s_f é mais robusto no que diz respeito a ruídos (*outliers*) do que o desvio médio padrão σ_f:

$$\sigma_f = \frac{1}{n}\sum_{i=1}^{n}\left(x_{if} - m_f\right)^2. \tag{3}$$

Se um dos valores, pertencentes a coluna *f*, está distante da média dos valores (tratando-se, portanto, de um ruído), seu efeito é amenizado no cálculo do desvio padrão (muito mais do que no cálculo do desvio absoluto), tornando-se imperceptível. O que não ocorre no desvio médio absoluto. A vantagem de utilizar o

desvio médio absoluto é que os *z-score* de anomalias não se tornam muito pequenos, portanto as anomalias podem ser detectadas (DE AMO, 2004).

Na extração dos atributos lineares, geralmente são utilizadas as seguintes equações para o cálculo da medida de distância: *Euclidean (4), Manhatan (city-block) (5) e Minkowski (6)*:

$$d(i,j) = \sqrt{\sum_{l=1}^{n} \left(x_{il} - x_{jl}\right)^2}, \tag{4}$$

$$d(i,j) = \sum_{l=1}^{n} \left|x_{il} - x_{jl}\right|, \tag{5}$$

$$d(i,j) = \sqrt[q]{\sum_{l=1}^{n} \left(\left|x_{il} - x_{jl}\right|\right)^q}, \tag{6}$$

onde *i* e *j* são os objetos dos quais está sendo calculada a distância, *n* = quantidade de atributos, x_{il} ou x_{jl} = valor do atributo *l* para o objeto *i* ou *j* respectivamente, sendo $q \geq 1$. Considerando que, a distância de *Minkowski* generaliza tanto a distância *Euclidean*, que é um caso especial sendo $q = 2$, quanto à distância de *Manhattan*, que é um caso especial sendo $q = 1$.

Para ressaltar a importância de certos atributos no cálculo da distância, considera-se uma distância ponderada, que consiste em se associar pesos a cada uma das coordenadas do objeto. Por exemplo, a distância *Euclidean* ponderada é dada por:

$$d(i,j) = \sqrt{\sum_{l=1}^{n} w_l \left(x_{il} - x_{jl}\right)^2}, \quad (7)$$

sendo $w_1, \ldots, w_n$ os pesos de cada um dos atributos envolvidos na descrição dos objetos (HAN e KAMBER, 2001; KAUFMAN e ROUSSEEUW, 2005).

Os atributos escalonados não lineares estão configurados em uma escala não linear, ou seja, com uma alternância variável de valores dentro de uma ordem de grandeza representada. Como por exemplo, uma escala exponencial, que é representada através da equação Ae^{Bt} ou Ae^{-Bt}, sendo *A* e *B* constantes positivas. Outro exemplo de aplicação deste tipo de atributo é o crescimento de uma população de bactérias ou a desintegração de um elemento radioativo, que também possuem conceitos medidos de acordo com uma escala exponencial.

Para extração deste tipo de atributo, são apresentadas três abordagens para o cálculo da dissimilaridade $d(i,j)$ entre dois objetos *i* e *j*, onde todos os atributos escalonados não lineares, sendo a segunda e a terceira abordagem mais eficazes. A primeira trata os atributos escalonados não lineares da mesma forma como se trata os atributos em escala linear. No entanto, não é utilizada em demasia, sendo que desta forma a escala linear provoca uma distorção nos dados, resultando em uma incoerência em relação aos dados iniciais (DE AMO, 2004; HAN e KAMBER, 2001).

Na segunda abordagem, aplica-se uma transformação logarítmica ao valor x_{if} de cada atributo A_f de um objeto *i*, obtendo $y_{if} = log\,(x_{if})$. Desta forma, os valores y_{if} podem ser tratados como se fossem valores em uma escala linear. Repare que, dependendo de como for escalonado o valor, outras transformações poderão ser empregadas. Neste exemplo, utilizou-se a função *log*, já que é a inversa da função exponencial (JAIN e DUBES, 1998).

A terceira abordagem trata os valores x_{if} atribuindo características de valores ordinais contínuos. Associa-se a cada valor um número entre 0 e $M_f - 1$, sendo M_f o número total de valores assumidos pelo atributo A_f. Sendo que este número M_f, de modo contrário do que acontece com os atributos ordinais, pode ser de grandeza elevada. Uma vez realizada esta associação, trata-se os valores associados da mesma forma como tratou-se os atributos em escala linear (KAUFMAN e ROUSSEEUW, 2005; MAIMON e ROKACH, 2010).

3.2.2 Seleção ou extração por coocorrência

Coocorrência é o princípio distribucional que diz respeito à possibilidade de unidades, de algum recurso, ocorrerem umas em combinação com outras. Uma quantificação em coocorrência é caracterizada pela contagem de combinações conjuntas de determinados recursos em um escopo finito de dados, sendo armazenado, geralmente em uma estrutura de vetor ou matriz de dimensão variável (*M*-dimensional). Esta abordagem é comumente aplicada em dados categóricos, nos quais cada recurso, definido como atributo, representa uma característica específica do objeto em análise (BERKHIN, 2006). A aplicação desta forma de extração de atributos é flexível em grande parte dos algoritmos de clusterização, como o *K-Means* e o *CURE*, pois uma vez gerada a matriz de dados a partir do vetor *M*-dimensional, é possível efetuar a entrada destes dados nestes algoritmos. Além destes, existem algoritmos que são especializados neste tipo de organização de dados, como o *CACTUS (Clustering Categorical Data Using Summaries)* e o *STIRR (Sieving Through Iterated Reinforcement)*. Desta forma, maiores informações sobre estes algoritmos, podem ser encontradas em Berkhin (2006).

Para representação de atributos de dados biológicos, tem-se o trabalho de Ji (2008), no qual identifica-se a aplicação da coocorrência na descrição de funções de genes utilizando um vocabulário controlado. Neste artigo, foi aprimorada a representação de características da linguagem natural de genes através de anotações funcionais e utilizando expansão de termo automática. Como ponto de partida o autor apresenta o método de representação de características com base na frequência de termos (*Term Frequency – Inverse Document Frequency - TF * IDF*), que utiliza coocorrência estruturada em um modelo de espaço vetorial (*Vector Space Model - VSM*). No entanto, o autor cita que este método possui limitações que foram observadas em estudos anteriores de anotação genômica e como melhoria propõem dois métodos para expansão de termo automática (*Automatic Term Expansion - ATE*), o primeiro com base em expansão de consulta (*Query Expansion - QE*) e o segundo com base em recuperação de informação (*Information Retrieval - IR*).

Para fins de desenvolvimento deste trabalho, descreve-se apenas o método *TF * IDF*, sendo que a informação para os demais métodos podem ser encontrados

em Ji (2008). A construção do espaço vetorial inicia-se com a seleção de um grupo de características (atributos) de acordo com algum critério estatístico ou heurístico. Este grupo é denotado por $T = \{t_1, t_2, \dots, t_M\}$, representando o conjunto de termos (*tokens* de textos ou palavras) para quantificação, estruturados em um vetor de espaço *M*-dimensional que compõem as características de cada objeto e pode ser representado em um histograma de peso de cada característica referente ao conjunto de dados. Dado um objeto *X,* utiliza-se a representação *TF * IDF* para formatar o conteúdo do objeto em um vetor, formulado por:

$$\begin{aligned} X &= (x_1, x_2, \dots, x_M) \\ &= (tf_1 \,.\, idf_1, tf_2 \,.\, idf_2, \dots, tf_M \,.\, idf_M), \end{aligned} \tag{8}$$

onde tf_i é a frequência de termos (*TF*) (i.e. a ocorrência) do termo t_i no objeto *X*, e idf_i é o inverso da frequência de objeto (*document*) (*IDF*) de t_i, em relação a todo conjunto de dados, definido por:

$$idf_i = -\log \frac{df_i}{N}, \tag{9}$$

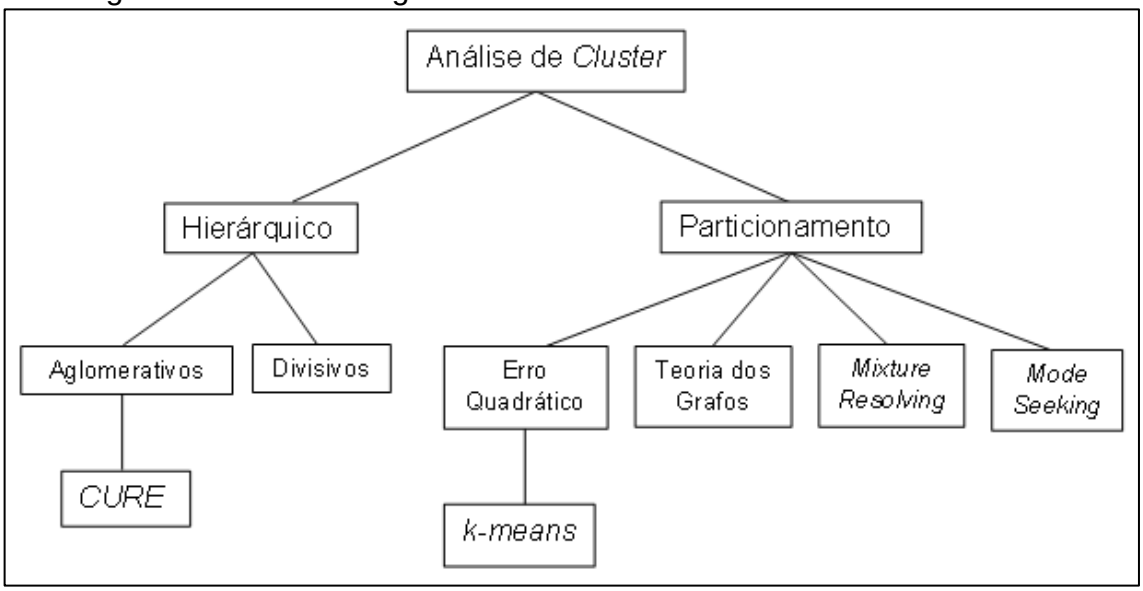
na qual df_i é a frequência de objeto de t_i, ou seja, o número de objetos que contém t_i, e *N* é o total de objetos no conjunto de dados.

Outra aplicação de coocorrência em domínios biológicos é encontrada em Li *et al*. (2011), que implementa o *framework* para generalização de acoplamento de características (*Feature Coupling Generalization - FCG*). Neste trabalho os autores mostram que a técnica de coocorrência mostra-se eficiente como parâmetro de entrada para esses algoritmos, pois enriquece a rotulação de dados esparsos e melhora o desempenho de características localizadas. A metodologia descrita foi aplicada em três diferentes questões: reconhecimento da nomenclatura da entidade gênica (*Gene Named Entity Recognition* - NER), extração de interação proteína-proteína (Protein-protein Interaction Extraction - *PPIE*) e classificação de texto para anotação de ontologia gênica (*Text Classification for Gene Ontology Annotation – TC-GO*). Outros detalhes sobre o *framework* podem ser obtidos no trabalho de Li *et al.* (2011).

3.3 ALGORITMOS DE CLUSTERIZAÇÃO

O elemento central da clusterização é o algoritmo que realiza a comparação dos dados através dos atributos, definindo distâncias entre os objetos e unificando ou particionando grupos de objetos de forma a classificar os mesmos. Estes algoritmos, conhecidos como métodos de clusterização, são definidos conforme a necessidade do contexto e a adequação aos dados disponíveis, a partir dos atributos escolhidos (MAIMON e ROKACH, 2010, p. 278). Dentre as diversas classificações, Fraley e Raftery (1998) e Jain *et al*. (1999), dividem os métodos em dois grandes grupos: métodos hierárquicos e métodos de partição, conforme Figura 6. Por outro lado, Han e Kamber (2001) sugeriram categorizar os métodos em três categorias principais: métodos com base em densidade, *clustering* com base em modelos e métodos com base em grade. É importante salientar que, devido ao grande volume de nomenclaturas que definem clusterização, neste trabalho define-se que os termos: clusterização, *clustering,* análise de *cluster* e tarefa de agrupamento, sempre significarão o processo de clusterizar (agrupar) dados de um modo geral.

Figura 6 - Estrutura organizacional dos métodos de análise de *cluster*

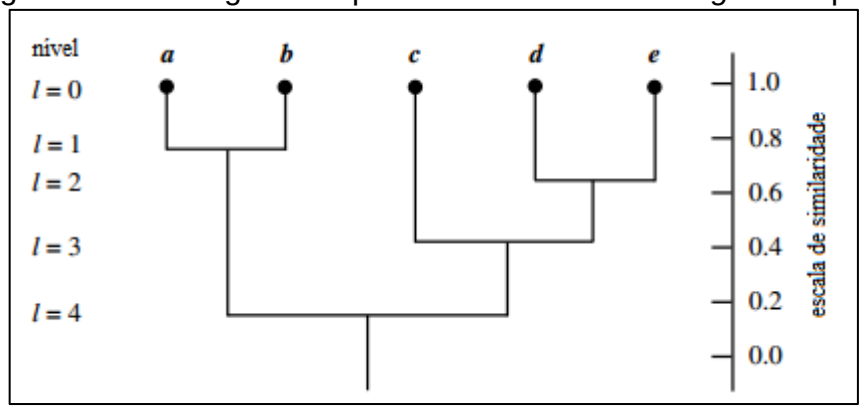


Fonte: adaptação de JAIN *et al*. (1999)

Os agrupamentos criados pelos algoritmos de *clustering* hierárquicos podem ser apresentandos em uma estrutura de árvore binária, em um formato denominado

dendrograma (Figura 7). Esta árvore representa a hierarquia de proximidade entre os objetos. A raiz da árvore é o conjunto de dados inteiros e os nodos folhas são os objetos. Os nodos intermediários representam a magnitude da proximidade entre os objetos. A altura do dendrograma indica a distância entre um par de objetos ou entre um par de subgrupos, ou ainda entre um objeto e um *cluster*. Para analisar os resultados, deve-se particionar o dendrograma em diferentes níveis de abstração, sendo assim possível visualizar estruturas e grupos em potencial, como por exemplo, dados de pesquisas sobre evolução das espécies (HAN e KAMBER, 2001).

Figura 7 - Dendrograma representando um *clustering* hierárquico



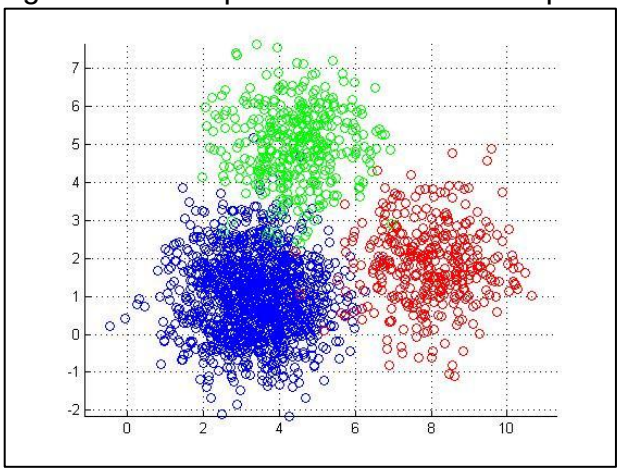
Fonte: HAN e KAMBER (2001)

Os algoritmos hierárquicos são divididos em aglomerativos e divisivos. Nos aglomerativos, é utilizada a forma *bottom-up*, na qual inicialmente cada objeto é um *cluster* e, à medida que o algoritmo é executado, ocorrem combinações entre esses *clusters*, de modo que no final todos os objetos pertençam ao mesmo *cluster*. Nos divisivos, é utilizada a forma *top-down*, na qual acontece a operação oposta, tendo inicialmente o conjunto de dados inteiro como um único *cluster* e, à medida que o algoritmo é executado, ocorrem divisões destes grupos, de modo que no final cada objeto torna-se um único *cluster* (JAIN *et al*., 1999; HAN e KAMBER, 2001).

Os agrupamentos criados pelos algoritmos de particionamento podem ser apresentados em um formato de plano cartesiano bidimensional, quando as coordenadas dos pontos (objetos) possuírem apenas duas dimensões. No entanto, podem ser trabalhados objetos com pontos de coordenadas *n*-dimensionais. Quando bidimensional, possibilita a visualização de conjuntos de pontos próximos uns aos outros, formando grandes blocos (*clusters*) que representam a similaridade dos

objetos, conforme observa-se na Figura 8. Inicialmente, o conjunto de dados é divido em *k* grupos de objetos. No decorrer da execução do algoritmo, os elementos são realocados entre os *clusters*, reorganizando estes grupos. O processo de recálculo e realocação é finalizado quando os objetos não alteram mais entre os grupos e o sistema se estabiliza. Esta é uma abordagem exaustiva e que no final das iterações, pode não gerar agrupamentos significativos. Por este motivo, o uso de heurísticas torna-se um mecanismo para aumentar a relevância da solução encontrada (HAN e KAMBER, 200; JAIN *et al*., 1999).

Figura 8 - Exemplo de *clusters* em um plano



Fonte: CHEN (2012)

Considerando todos os algoritmos aplicáveis aos problemas de clusterização, optou-se pela aplicação de dois deles: um hierárquico e outro de particionamento. Deste modo, conforme o objetivo deste trabalho, os algoritmos K-*Means* e *CURE* são os que apresentam maior aplicabilidade e, portanto são apresentados nas Seções 3.3.1 e 3.3.2, respectivamente.

3.3.1 K-Means

O *K-Means* é um algoritmo de clusterização pertencente à categoria de métodos particionais, na subcategoria de métodos que utilizam o erro quadrático para a mensuração da convergência da partição, em relação ao centro do *cluster*. Esse algoritmo busca realocar os elementos de forma iterativa, através da menor distância em relação a um conjunto de pontos centrais (centroides), representados pela média destes elementos. Outra característica é a criação de conjuntos de

dados que são aglomerados compactados, ou seja, bem dissimilares entre si, mas com dados muito parecidos no interior. O *K-Means* possui como entrada, dados modelados geralmente através de atributos em escala linear, alocados na matriz de dados, que compõem as variáveis das equações: *Euclidean (4), Manhatan (city-block) (5) e Minkowski (6)*, utilizadas no cálculo da distância. Além da matriz de dados, necessita como entrada de dados um parâmetro de valor numérico inteiro denominado *k*, que tem a função de identificar o número de grupos previamente conhecidos que se deseja trabalhar durante a execução. Além disso, opcionalmente, pode-se informar um parâmetro de valor numérico inteiro denominado *T*, que representa o número máximo de iterações desejadas (DE AMO, 2004; JAIN *et al*., 1999; JONES e PEVZNER, 2004; MAIMON e ROKACH, 2010).

O funcionamento do *K-Means* é apresentado, de forma abrangente, pelos passos 1 a 4. O algoritmo com detalhamento matemático é encontrado no Anexo A deste documento.

1. Seleciona-se *k* objetos para serem os centroides iniciais. Estes objetos são o centro de gravidade inicial de cada *k cluster* gerado. Esta seleção pode ser arbitrária, ou com base em alguma técnica, ou até mesmo heurística, no entanto o método de seleção é facultativo (DE AMO, 2004; HAN e KAMBER, 2001).
2. Para cara objeto do conjunto de objetos disponível, calcula-se o valor de distância em relação aos *k* centroides existentes, através das Equações 4, 5 e 6. Na sequência, aloca-se o objeto ao *cluster* do qual este tem menor proximidade em relação ao centroide, ou seja, a menor distância calculada (DE AMO, 2004; HAN e KAMBER, 2001).
3. Para cada *k cluster*, calcula-se um novo centroide a partir da média dos valores de cada atributo dos elementos pertencentes ao *cluster*. Este será o novo centro de gravidade do *cluster*, para balancear o mesmo. No caso de atributos lineares, o cálculo do novo centroide é definido por:

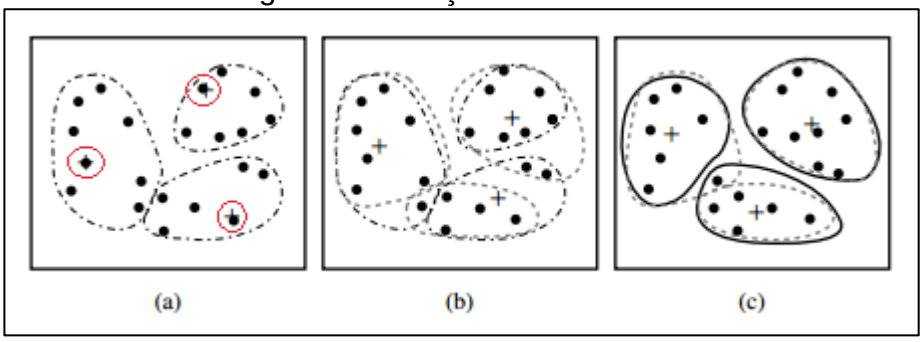
$$\mu_k = \frac{1}{n_k} \sum_{q=1}^{n_k} x_q, \tag{10}$$

sendo μ_k o valor médio dos objetos do *cluster k*, n_k a quantidade de objetos no *cluster k* e x_q o valor do objeto *n* para o atributos *q*, envolvidos no somatório, pertencentes a *k* (MAIMON e ROKACH, 2010).

4. Verifica-se se um ou mais objetos foram realocados para algum *cluster*. Neste caso, retorna-se ao passo 2 e repete-se o processo. Caso nenhum objeto tenha sido realocado, finaliza-se o processo, pois os *clusters* estabilizaram. Além deste critério de parada, pode-se interromper o processo ao atingir um número *T* de iterações esperadas. (DE AMO, 2004; HAN e KAMBER, 2001).

A Figura 9 ilustra o funcionamento do método *K-Means* para $k = 3$. Observa-se nas iterações (a), (b) e (c) que o os centroides, representados pelos pontos circundados e cruz (+), alteram de posição dentro do grupo, de forma a estar sempre no centro e balancear o mesmo. Observa-se também, através dos pontilhados, a realocação dos objetos (pontos em negrito) que tende a diminuir na medida em que acontecem as iterações, demonstrando que os agrupamentos convergem para a estabilização (DE AMO, 2004; HAN e KAMBER, 2001).

Figura 9 - Iterações do *K-Means*



Fonte: HAN e KAMBER (2001)

O *K-Means* possui complexidade de execução $O(T * K * m * N)$, onde *T* representa o número de iterações, *m* é o número de instâncias do conjunto de objetos disponíveis que são caracterizadas por *N* atributos. Esta é uma de suas principais vantagens, pois garante eficiência ao tratar grande quantidade de dados. Além disso, possui vantagem de possuir baixa complexidade de desenvolvimento. Em virtude destas vantagens, é largamente utilizado por pesquisadores e é aplicado na resolução de problemas de ordem geral de agrupamento e classificação de

informações que não necessitam de um aprendizado prévio (DE AMO, 2004; HAN e KAMBER, 2001).

Dentre as desvantagens, o algoritmo possui um valor pré-definido de números de agrupamentos (número *k*), obrigando o conhecimento da quantidade de padrões que se deseja reconhecer. Outra desvantagem é a sensibilidade a ruídos, na qual objetos com valores altos podem causar uma grande alteração no centro de gravidade dos *clusters* e assim, distorcer a distribuição dos dados nos mesmos (DE AMO, 2004; JONES e PEVZNER, 2004; MAIMON e ROKACH, 2010).

3.3.2 Clustering Using Representatives

O algoritmo *Clustering Using Representatives (CURE)* pertence à categoria de métodos hierárquicos, na subcategoria de métodos aglomerativos. Este algoritmo busca unificar, de forma iterativa, os objetos conforme o grau de semelhança, com os representantes de cada *cluster*. Possui como entrada uma estrutura idêntica ao *K-Means*, composta por dados em escala linear, modelados através de atributos e alocados na matriz de dados. Além desta matriz, o algoritmo necessita como entrada de dados: um parâmetro de número *k* de *clusters* finais esperados, um parâmetro de número *c* de representantes que são alocados em cada grupo e um parâmetro denominado fator *alpha* (α) que é utilizado na retração dos representantes dentro do *cluster* (DE AMO, 2004; HAN e KAMBER, 2001; JAIN *et al*., 1999).

No início da aplicação do algoritmo cada objeto é considerado um *cluster*, ou seja, cada objeto representa solitariamente um grupo. A cada iteração, os objetos, ou grupo de objetos, são analisados e aglutinados conforme a proximidade entre os mesmos, determinada pela distância entre os representantes de cada grupo, gerando *clusters* maiores. Conforme o crescimento dos *clusters* repete-se o processo, elegendo-se o número *c* de representantes, deslocando-os com o fator *alpha,* calculando a proximidade e determinando novos reagrupamentos. A aglutinação a cada iteração é realizada em apenas um par de *clusters*, ou objetos, de menor distância. O processo finaliza quando os agrupamentos convergem em um único *cluster* final. No entanto, para obter resultados pertinentes, interrompe-se o processo ao atingir um número *k* de *clusters* finais esperados, sendo este número previamente conhecido (HAN e KAMBER, 2001; XU e WUNSCH, 2005).

Através dos passos 1 a 7, verifica-se com detalhes o funcionamento do *CURE*, assim como a retração dos representantes e a medida de distância entre estes. O algoritmo com o detalhamento matemático é encontrado no Anexo B deste documento.

1. Identifica-se cada objeto da matriz de dados como sendo um *cluster* isolado, formando inicialmente um número de *clusters* igual à quantidade de objetos disponíveis (DE AMO, 2004).
2. Calculam-se as distâncias (Equações 4, 5 ou 6) entre todos os objetos, de forma a obter uma matriz de dissimilaridade. Porém, armazena-se somente a menor distância encontrada para cada par de objetos. Para este armazenamento pode ser utilizada uma estrutura do tipo *heap*, onde é possível incluir os *clusters* de forma ordenada, e uma estrutura *k-tree*, onde é possível armazenar os representantes obtidos no passo 4. Mais detalhes sobre estas estruturas são encontrados em Guha *et al*. (1998) e Leiserson *et al*. (2001).
3. Calcula-se o centroide do *cluster* a partir da média dos objetos dos *clusters* envolvidos, através da Equação 11. Salientando-se que inicialmente, quando o *cluster* possuir apenas um objeto, este será o centroide.

$$w_c = \frac{n_u * u_c + n_v * v_c}{n_u + n_v}, \tag{11}$$

   sendo w_c é o centroide do *cluster* que foi originado pela união do *cluster* u com o *cluster* v, n_u a quantidade de objetos no *cluster* u, u_c o centroide do *cluster* u, n_v a quantidade de objetos no *cluster* v e v_c o centroide do *cluster* v (GUHA, 1998).
4. Com o objetivo de calcular a distância entre dois *clusters*, são selecionados *c* representantes destes através dos passos 4.1 a 4.4. Estes *c* pontos são escolhidos de forma a representar regiões bem distintas dentro do grupo, conforme observa-se os pontos destacados com um círculo na Figura 10. Se *c* for menor que a quantidade objetos no *cluster*, então estes objetos serão os próprios representantes, seguindo diretamente para o passo 5.

4.1 Calcula-se a distância (Equações 4, 5 ou 6) entre os objetos do grupo em relação ao centroide, calculado no passo 3. Seleciona-se como primeiro representante r_1 sendo o objeto com maior distância, ou seja, mais afastado do centro, podendo ser até mesmo um ruído (*outlier*).

4.2 Calcula-se a distância, (Equações 4, 5 ou 6) entre os objetos restantes do *cluster* em relação à r_1. Seleciona-se como segundo representante r_2 aquele que estiver mais afastado de r_1.

4.3 O enésimo representante r_n será aquele cuja distância a $\{r_1, r_2, \dots, r_{n-1}\}$ é a maior possível. Desta forma, a distância de r_n a $\{r_1, r_2, \dots, r_{n-1}\}$ é o mínimo entre $\{d(r_3, r_1), d(r_3, r_2), \dots, d(r_{n-1}, r_{n-2})\}$.

4.4 Repete-se o processo 4.3 até atingir até o número *c* de representantes para o *cluster* em questão.

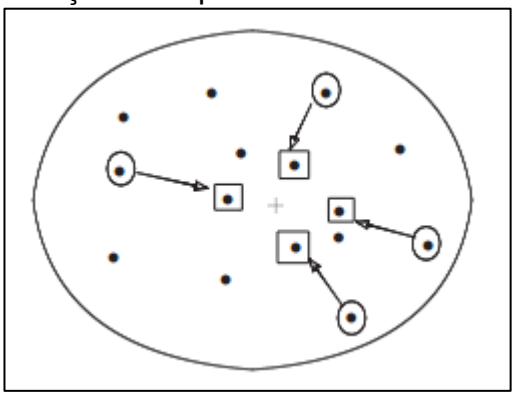
5. Na sequência, efetua-se o cálculo (Equação 12) da retração dos representantes pelo fator *alpha* (sendo $0 \leq \alpha \leq 1$), conforme apresentado na Figura 10, na qual observa-se o movimento dos representantes através da retração por um fator $\alpha = \frac{1}{2}$, por exemplo. O ponto de origem é identificado pelo círculo e o movimento do representante apontado pela seta, em direção ao centro. Este deslocamento mantém o formato inicial do *cluster* e transporta os representantes ao centro do conjunto, aumentando a eficácia do cálculo de distância, tornando o algoritmo menos sensível a ruídos. Ou seja, caso um representante escolhido no passo 4 seja um ruído, após a retração este se aproximará do centro mantendo-se equilibrado em relação aos demais representantes, que não são ruídos, e o centro (DE AMO, 2004; XU e WUNSCH, 2005).

   O cálculo da retração dos representantes é dado por:

$$r = r + \alpha * (c - r), \tag{12}$$

   sendo r o representante ao qual se está calculando a retração e c o centroide do *cluster a*o qual r pertence.

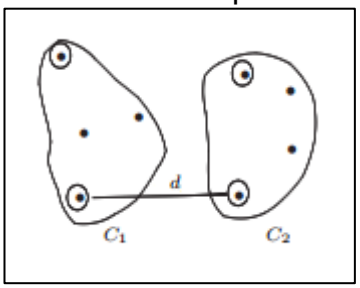
Figura 10 - Retração de representantes em um *cluster CURE*



Fonte: DE AMO (2004)

6. Conforme observa-se na Figura 11, a distância entre dois *clusters* é dada pela distância mínima entre dois pontos, um em cada *cluster*, estando estes na fronteira mais próxima entre os *clusters*. Desta forma, após a definição dos representantes para os *clusters*, calcula-se a distâncias (Equações 4, 5 e 6) entre estes, de modo a encontrar um par de *clusters* com menor distância em relação aos demais pares, e posteriormente aglutina-los. A busca pelo par de *clusters* não é definida por um processo de comparação entre todos os representantes de todos os *clusters* disponíveis. Este seria um processo demorado e oneroso (DE AMO, 2004; JONES e PEVZNER, 2004). Então, ao invés deste, utiliza-se uma abordagem de busca através de uma árvore geradora mínima (*MST – Minimum Spanning Tree*). Também conhecida por árvore geradora de custo mínimo, esta estrutura é uma árvore geradora de um grafo ponderado cuja soma dos pesos das arestas é mínima (SZWARCFITER, 1986). No algoritmo *CURE*, este grafo é mantido através da estrutura citada no passo 2 e é utilizado, a cada iteração, no cálculo da distância entre dois *clusters*, configurando assim a *política MST*.

Figura 11 - Distância entre os representantes dos grupos



Fonte: DE AMO (2004)

7. Verifica-se se a quantidade de grupos é menor ou igual a *k clusters* esperados. Se não for, repete-se o processo a partir do passo 3. Todavia, pode-se utilizar como critério de parada o cenário em que a distância mínima entre os *clusters* seja menor do que um limite fornecido. Porém esta abordagem é complexa e custosa, não sendo muito utilizada (DE AMO, 2004; XU e WUNSCH, 2005).

Uma das grandes vantagens da aplicação do *CURE* é poder detectar formas arbitrárias de *clusters*. Este comportamento normalmente não acontece com métodos por particionamento. Outra vantagem é que este método é consideravelmente robusto quanto a ruídos, pois através dos *c* representantes, a retração em direção ao centro de gravidade do *cluster* tem o efeito de diminuir a influência destes ruídos (DE AMO, 2004; HAN e KAMBER, 2001; JAIN *et al*., 1999).

Dentre as desvantagens, *CURE* possui complexidade $O(n^2)$, onde *n* é a quantidade de objetos do conjunto de dados disponíveis. Esta complexidade é encontrada no passo 2, onde tem-se o cálculo das distâncias iniciais. No entanto, esta complexidade é encontrada novamente no passo 6, no cálculo das distâncias entre os representantes de dois grupos, onde é necessário realizar uma comparação de distância entre todos os represente destes dois grupos. Outra desvantagem de *CURE, e dos* métodos hierárquicos no geral, é que estes possuem determinada incapacidade de realizar ajustes quando uma decisão de fusão ou cisão foi executada. Ou seja, ao realizar uma aglutinação ou divisão que não tende ao resultado esperado, o algoritmo fica incapacitado de retroagir a esta operação e realinhar a execução (HAN e KAMBER, 2001; JAIN *et al*., 1999 XU e WUNSCH, 2005).

3.4 APLICAÇÕES EM BIOLOGIA

Os métodos de clusterização vêm sendo aplicados na análise de vetores de *DNA,* alterando o cenário de pesquisas biomédicas e genéticas. Suas aplicações possibilitam uma visão simultânea da transcrição de centenas de genes, de diferentes origens. A informação obtida por este monitoramento de sequências gênicas em diferentes estágios celulares, tipos de tecidos, condições clínicas e organismos pode ajudar no entendimento de suas funções e como essas se influenciam mutuamente. Deste modo, contribuindo para a orientação de diagnósticos e efeitos em tratamentos medicinais (SHAMIR e SHARAN, 2001; XU e WUNSCH, 2010).

Exemplos de aplicações de algoritmos de clusterização e análise de *cluster* a partir de sequências de *DNA* podem ser encontrados em diversos artigos. Shamir e Sharan (2001) descrevem algumas das principais abordagens algorítmicas para clusterização de dados de expressão gênica, assim como as tecnologias do cenário biológico para geração de dados para clusterização (*cDNA Microarrays*, *Oligonucleotide Microarrays* e *Oligonucleotide Fingerprinting* - *ONF*) e algoritmos recorrentemente utilizados na clusterização gênica: *HSC* (Highly Connected Subgraph), *CLICK* (*Cluster Identification via Connectivity Kernels*), *CAST* (*Cluster Affinity Search Technique*), *SOM* (*Self Organizing Maps*), *K-Means* e Clusterização Hierárquica. Estes algoritmos foram aplicados em um conjunto de dados da expressão gênica de uma levedura. Os resultados foram avaliados de modo a encontrar qual algoritmo apresenta melhor abordagem de clusterização, porém os autores concluem que, através de todas as disciplinas matemáticas e aplicações realizadas, todos os algoritmos descritos apresentam pontos positivos e negativos. Desta forma, é impossível escolher um único algoritmo para resolver todas as classes de problemas. A decisão eventual de qual solução e de qual algoritmo funciona melhor depende de quem está analisando e da questão específica de clusterização que se está tentando responder.

Na biomedicina, exemplos de aplicação de algoritmos de clusterização são encontrados em áreas como a análise da expressão gênica de dados, análise de sequência genômica, mineração de documentos biomédicos e análise de imagens de ressonância magnética. No entanto, devido à diversidade de análises de *clusters*, diferentes terminologias, objetivos e pressupostos subjacentes de diferentes

algoritmos, determinar a combinação correta do algoritmo na aplicação biomédica é uma decisão complexa (XU e WUNSCH, 2010). Xu e Wunsch (2010) apresentam uma revisão das aplicações dos algoritmos de clusterização envolvendo as pesquisas biomédicas, de modo a auxiliar na seleção do algoritmo mais adequado. Nesta proposta de Xu e Wunsch (2010), são discutidos os problemas relativos à análise de *cluster*, utilizando para isto os mais importantes e influentes algoritmos de agrupamento, como *K-Means*, *CURE*, entre outros. Os autores concluem que os algoritmos hierárquicos são uma opção imprópria para agrupamento de dados em larga escala, devido ao grau de complexidade computacional. Além disso, a falta de robustez também restringe suas aplicações em um ambiente ruidoso. Porém, *CURE* e alguns outros algoritmos hierárquicos, apresentam eficácia com reconhecimento progressivo pelos pesquisadores biomédicos. Em suma, as principais dificuldades são a falta de orientação efetiva para implementação dos algoritmos e ajuste de parâmetros, ou apenas a falta de uma boa comunicação entre os campos da biologia e computação.

A análise de *cluster* por métodos hierárquicos é comumente abordada na análise de dados de expressão gênica (TORONEN, 2004). Os agrupamentos criados pelos algoritmos de *clustering* hierárquicos podem ser apresentandos em uma estrutura de árvore binária, em um formato denominado dendrograma, conforme descrito na Seção 3.3. Para que esta análise possa ser realizada, é necessário selecionar os *clusters* através do corte no dendrograma, em um nível adequado e/ou através da extração de uma lista de genes ordenada. No entanto este corte no nível adequado resulta na perda de informações em outros níveis. Já a extração da lista de genes ordenada depende do método de ordenação utilizado na geração dos agrupamentos (TORONEN, 2004). Com o propósito de facilitar a análise e melhorar a qualidade da seleção de *clusters*, Toronen (2004) aborda a seleção de *clusters* significativos a partir de árvores de *clusters* hierárquicos através da classificação gênica. As categorias de genes utilizadas estão enquadradas no processo biológico, função molecular e localização celular. O método compara todos os *clusters* do dendrograma com todas as classes de genes realizando uma mensuração de distribuição hipergeométrica, que busca a melhor correlação entre a classe de gene a ramificação do dendrograma. Este método foi demonstrado com aplicação nos dados da expressão gênica de uma levedura. Os resultados obtidos mostraram que o método encontrou *clusters* com informações relevantes a partir de

muitos níveis do dendrograma e indicou que estes mesmos *clusters* não poderiam ser obtidos por um corte simples na árvore. O método foi aplicado em diversas árvores geradas por diferentes algoritmos hierárquicos e a comparação aplicada demonstrou que o método selecionou os mesmos *clusters* para estes diferentes dendrogramas.

Pesquisas em análise de clusterização de sequências de *DNA*, detecção e reconhecimento de promotores têm sido realizadas, como por exemplo, Radjiman *et al*. (2006). O objetivo dos autores foi analisar e extrair o comportamento correlacionado de sequências gênicas com base em seus padrões de expressão, através de um algoritmo de clusterização não supervisionado, denominado Clusterização Super Paramagnética (*Super Paramagnetic Clustering - SPC*). Este algoritmo utiliza propriedades físicas de um ferromagneto granular heterogêneo com base em um modelo estatístico de ferromagnetos desordenados (BLATT *et al*., 1996). O método de análise utiliza simulações *Swendsen-Wang cluster Monte Carlo* (EDWARDS e SOKAL, 1988) para distinguir *clusters* através da mensuração de pares de funções correlacionadas a partir de diferentes resoluções. A clusterização foi aplicada em 4541 sequências de *DNA* contendo regiões de promotores ativos, com classes de vertebrados e artrópodes, incluindo os genes virais. As informações foram mensuradas a partir de frequências de tri-nucleotídeos e tetra-nucleotídeos. Os resultados separaram as sequências de artrópodes e vertebrados com sucesso em duas diferentes classes com apenas 9,25% de sequências de artrópodes erroneamente classificadas. Isto indica que, através de uma perspectiva funcional, estas sequências têm funções gênicas elevadas e se correlacionam com as sequências dos *clusters* de vertebrados.

Com o desenvolvimento de tecnologias, são criados atualmente muitos dados biológicos de *DNA*. Como consequência disso, os bancos de dados genômicos estão crescendo exponencialmente em poucos anos. Como forma de estudar estes dados de forma abrangente, muitos métodos novos de análises estão sendo propostos. Com base neste cenário, Liu *et al.* (2006) descreve um trabalho de clusterização de sequências de *DNA* através de vetores de características. Deste modo, as sequências de *DNA* são representadas como pontos de um espaço de doze dimensões (*twelve-dimensional*) permitindo o agrupamento sequências homólogas, abrindo espaço para a comparação de sequências em âmbito global para milhões de genes simultaneamente. A composição destes vetores é dada a

partir da definição de atributos que representam características comuns das sequências. O primeiro atributo do vetor é obtido através ocorrência dos termos A, C, T e G, semelhante à teoria de coocorrência descrita na Seção 3.2.2, porém de forma mais simplificada. Sendo que estes termos podem estar distribuídos em combinações de AA, AC, AT e AG, ou AAA, AAC, AAT, entre outras. Os demais atributos do vetor são determinados por: total da distância de cada nucleotídeo base até o primeiro nucleotídeo e total da distribuição de cada nucleotídeo único. Através destes vetores, os dados são clusterizados pelo método *BLAST (Basic Local Alignment Search Tool)*. A aplicação desta forma global de estruturação de genes foi testada em famílias de proteínas e constatou-se que os membros de cada família foram apresentados em distâncias menores do que os membros de famílias diferentes. Além disso, foi possível distinguir sequências aleatórias com base na mesma base de composição. Através desta constatação, o autor relata que o método de comparação mostrou-se consistente e que novos genes descobertos podem ser verificados através desta abordagem.

3.5 CONSIDERAÇÕES FINAIS

A análise de *cluster* parte da preparação dos dados e extração dos atributos até a validação dos resultados, abrangendo a aplicação dos algoritmos. Dentre todas as fases, a mais complexa consiste na organização e extração dos atributos para a entrada de dados. Esta dificuldade se agrava no contexto da genética, uma vez que a quantidade de informações para diversos organismos aumenta diariamente. Consequentemente, uma abordagem computacional é interessante, pois apesar das limitações, apresenta inúmeras possibilidades de descobertas. Desta forma, a fundamentação da análise de *cluster* pode ser explorada em trabalhos futuros, como o desenvolvimento da extração de atributos utilizado *ATE* (Seção 3.2.2), por exemplo.

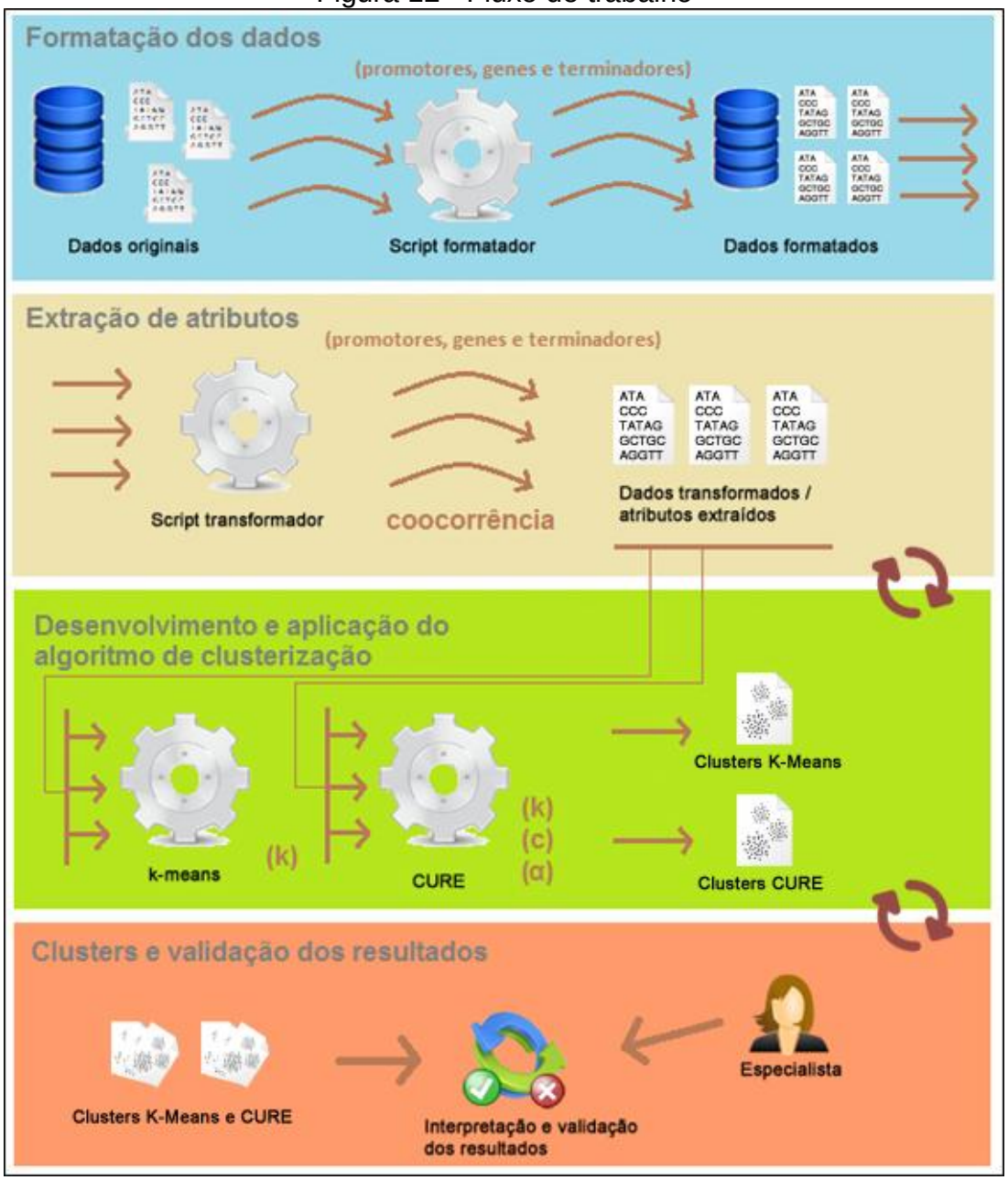
4 METODOLOGIA

Neste capítulo, os elementos da metodologia são apresentados individualmente, juntamente com o fluxo de trabalho executado no desenvolvimento e aplicação dos algoritmos de clusterização.

4.1 FLUXO DE TRABALHO

Observa-se através da Figura 12 o fluxo de trabalho executado. Na fase de formatação de dados realizou-se a seleção dos dados originais da fonte de dados (Seção 4.3) assim como a formatação dos mesmos através da preparação dos dados (Seção 4.5), tendo como saída os segmentos de *DNA* formatados em arquivos de promotores genes e terminadores. Estes arquivos serviram de entrada para a fase de extração de atributos (Capítulo 3.2 e Seção 4.5) resultando nos arquivos com atributos e valores para serem processados pelos algoritmos. Na fase de desenvolvimento e aplicação do algoritmo de clusterização foi desenvolvido um aplicativo de plataforma *desktop* (Seção 4.6) e processados os arquivos de saída da fase anterior. Os resultados obtidos deste processamento foram os *clusters*, dos quais foram interpretados na fase de validação dos resultados (Capítulo 5).

Figura 12 - Fluxo de trabalho



4.2 ORGANISMO

O organismo utilizado como elemento de estudo através dos métodos de clusterização é a bactéria modelo *E. coli* da cepa *K12-MG1655*, cuja descrição encontra-se na Seção 2.1.

4.3 FONTE DE DADOS

Os dados utilizados na solução referem-se ao sequenciamento de *DNA* da bactéria *E. coli*. Trata-se de um conjunto de arquivos em formatos textuais obtidos da base de dados do portal de acesso público *RegulonDB*. Para este trabalho, optou-se por utilizar os arquivos referentes às segmentações de promotores, genes (região codificadora) e terminadores, sendo que para cada uma destas, analisou-se uma determinada quantidade de sequências, conforme Tabela 1. Informações adicionais podem ser encontradas em: *http://regulondb.ccg.unam.mx*.

Tabela 1 - Quantidade de sequências

Promotores	8452
Genes	4632
Terminadores	252
Total	13336

Fonte: autor (2013)

4.4 FERRAMENTAS

Para desenvolvimento da solução e obtenção dos resultados foram utilizadas as seguintes ferramentas: plataforma de desenvolvimento *Microsoft .NET Framework* versão 4, ambiente integrado de desenvolvimento (*Integrated Development Environment* - IDE) *Microsoft Visual Studio 2010 Professional* e linguagem de desenvolvimento orientada a objetos *Microsoft C#.* Ambas as ferramentas constituem parte do conjunto de ferramentas de desenvolvimento de *software* da *Microsoft*, sendo o *framework* requisito básico para execução da solução, no entanto nativo ao sistema operacional *Windows* (versões recentes com os pacotes de atualizações). Optou-se por utilizar estas tecnologias por serem largamente utilizadas e consolidadas no meio comercial e educacional. Além disso, por possuírem confiabilidade, robustez e segurança no processamento e gerenciamento de grandes quantidades de dados, conforme citado pelo fabricante (VISUAL STUDIO 2010, 2013; MICROSOFT C#, 2013; FRAMEWORK, 2013).

4.5 PREPARAÇÃO DOS DADOS

Os arquivos provenientes do *RegulonDB* possuem formato específico com informações não relevantes na aplicação dos algoritmos de clusterização. Deste modo, a primeira etapa do desenvolvimento da solução consistiu na formatação dos dados, que tratou de selecionar as informações adequadas destes arquivos, através dos recursos de análise de expressão regular e identificação de padrões de espaçamento e tabulação. As informações extraídas estão destacadas nas áreas A, B, C e D da Figura 13, sendo:

A. Identificador do gene;
B. Nome do gene;
C. Sigmas relacionados, quando promotor;
D. Segmento de *DNA*.

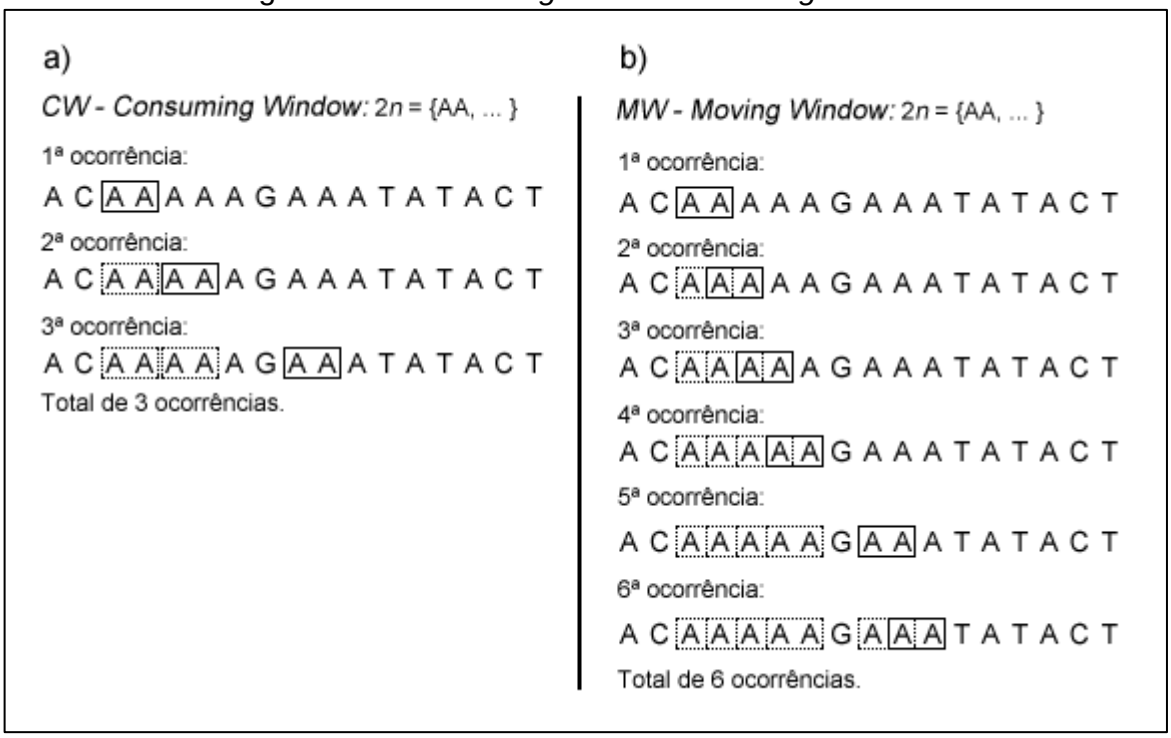
Figura 13 - Arquivo fonte de dados de promotores

Fonte: autor (2013)

A segunda etapa do desenvolvimento da solução consistiu na extração dos atributos a partir do *DNA* selecionado na fase de formatação. O método de extração de atributos escolhido foi à coocorrência (Seção 3.2.2), por ser um método já utilizado na literatura e com resultados relevantes para a área em questão. Os termos utilizados para montar o vetor *M*-dimensional, que representa a frequência de termos, foram gerados a partir da combinação dos quatro nucleotídeos A, C, T e G

em um conjunto de tamanho *X*. Para isso, identificou-se por *X*n o vetor gerado a partir de todas as combinações possíveis entre os nucleotídeos (n), para um conjunto de tamanho *X*, por exemplo: 2n = {AA, AC, AT, AG, CA, CC, CT, CG, TA, TC, TT, TG, GA, GC, GT, GG}, 3n = {AAA, AAC, AAT, AAG, ACA, ACC, ACT, ACG, ATA, ATC, ATT, ATG, AGA, AGC, ...}. A partir destes vetores, verificou-se a coocorrência destes termos utilizando a teoria *VSM* (Equação 9), extraindo assim os atributos. De forma complementar, para a realização da quantificação da frequência de cada conjunto de termos do vetor *X*n, foram utilizadas duas abordagens de combinações dos termos, conforme Figura 14.

Figura 14 - *Consuming Window x Moving Window*



Fonte: autor (2013)

A primeira abordagem (a) (*Consuming Window - CW*) consome o termo depois de lido, fazendo com que a janela deslizante do conjunto não reutilize o mesmo. A segunda abordagem (b) (*Moving Window - MW*) não consome o termo após a leitura, permitindo que o mesmo alimente o próximo conjunto conforme o deslizamento da janela.

4.6 ALGORITMOS DE CLUSTERIZAÇÃO

A terceira etapa do desenvolvimento da solução consistiu na implementação dos algoritmos de clusterização *K-Means* e *CURE*, conforme descritos nas Seções 3.3.1 e 3.3.2, respectivamente. Ambos os algoritmos foram construídos em um aplicativo de plataforma *desktop*, cujo desenvolvimento ocorreu por meio da utilização das ferramentas descritas na Seção 4.4. O aplicativo foi arquitetado utilizando-se *threads*, de forma a desvincular a interface do sistema ao módulo de processamento, permitindo assim modificação ou portabilidade da interface para outros ambientes, sem necessidade de alteração nos algoritmos. Maiores detalhes sobre o aplicativo podem ser encontrados no Anexo C deste trabalho.

4.7 CONSIDERAÇÕES FINAIS

O fluxo de trabalho proposto concretizou-se durante o desenvolvimento. Foram utilizadas as ferramentas propostas, sem a necessidade de alteração por outras ferramentas. Algumas dificuldades foram encontradas nas etapas de formatação, extração de atributos e desenvolvimento dos algoritmos. Por exemplo, na fase de formatação dos dados, houve dificuldade em extrair os dados dos arquivos, pois os mesmos não possuem um padrão de divisão das colunas de informações. Desta forma foi necessário utilizar mais de uma técnica para extrair as informações, conforme citado na Seção 4.5. Já, na fase de extração de atributos, a principal dificuldade consistiu em selecionar um método de extração adequado para os algoritmos escolhidos, o qual deve contemplar a matriz de dados (Seção 3.1). Considerando esta premissa, foi selecionado o método de coocorrência, conforme citado na Seção 4.5.

Na fase de desenvolvimento dos algoritmos, o desenvolvimento de *CURE*, por possuir ordem de complexidade quadrática, sub-rotinas para seleção de representante e cálculo de fator *alpha*, exigiu o carregamento do arquivo de atributos na íntegra em memória principal (*RAM)*.

5 RESULTADOS E DISCUSSÃO

Os resultados foram obtidos através da realização de simulações para cada um dos algoritmos selecionados, sendo a configuração de máquina utilizada: Processador: *Intel(R) Core(TM)2 Duo CPU P8600 @2.40GHz*; Memória: *DDR2, 6 GBytes, Dual Channel*. Os valores de tempos de execução são referentes a esta configuração. A parametrização da extração de atributos e dos algoritmos foram inicialmente estabelecidas a partir dos dados de entrada. Os resultados esperados são *clusters* que caracterizam distintamente os segmentos de promotores, genes e terminadores.

5.1 SIMULAÇÕES UTILIZANDO *K-MEANS*

As simulações com o algoritmo *K-Means* utilizaram como dados de entrada os atributos extraídos dos segmentos de promotores, genes e terminadores. Os parâmetros e configurações gerais utilizados estão descritos na Tabela 2.

Tabela 2 - Configurações gerais para *K-Means*

	Parâmetros	**Valores**	**Observação**
Atributos	Vetor *M*-dimensional	De 2n até 7n	Conjunto de nucleotídeos para os vetores *M*-dimensionais.
	Window	*CW* e *MW*	Seleção da frequência dos termos (Seção 4.3).
K-Means	*Clusters*	C1, C2 e C3	Sendo cada *cluster* correspondendo a um segmento, contido em um arquivo específico.
	Centroide inicial	Aleatório	Para cada um dos três arquivos.
	Cálculo de distâncias	*Euclidean*	*Euclidean* (4) (Seção 3.2.1).

Na primeira simulação aplicaram-se os vetores de 2n até 7n, abrangendo *CW* e *MW*. Analisando os resultados da distribuição da quantidade de objetos

(sequências) para cada um dos segmentos, constatou-se que *K-Means* não separou os terminadores, genes e promotores nos *clusters* gerados, conforme apresentado na Tabela 3. No entanto, para os atributos de 2n, *K-Means* distinguiu quase que totalmente um *cluster* (C3) contendo terminadores.

Tabela 3 - *K-Means*: resultados 1ª simulação

Configuração		**Tempo de execução (aprox.)**	**Quantidade de objetos por *cluster***			
Nucleotídeos	***Window***		***Cluster***	**Promotor**	**Gene**	**Terminador**
2n	*MW*	3 min	C1	5463	262	4
			C2	2989	4369	17
			C3	0	1	231
2n	*CW*	3 min	C1	2632	4295	15
			C2	5820	336	7
			C3	0	1	230
3n	*MW*	15 min	C1	3	2676	20
			C2	3955	558	141
			C3	4494	1398	91
3n	*CW*	15 min	C1	3	2656	25
			C2	4356	1310	90
			C3	4093	666	137
4n	*MW*	25 min	C1	0	991	2
			C2	3980	1843	48
			C3	4472	1798	202
5n	*MW*	1 h	C1	0	345	0
			C2	0	2198	2
			C3	8452	2089	250
6n	*MW*	1 h	C1	8451	2434	252
			C2	1	0	0
			C3	0	2198	0
7n	*MW*	4 h	C1	0	1	0
			C2	0	2292	0
			C3	8452	2339	252

Existe determinada dificuldade no reconhecimento dos promotores a partir de via computacional (*in silico*). Esta dificuldade é determinada pela sequência dos promotores não apresentarem-se completamente conservada, ou seja, com subsequências de características únicas, que existem apenas nos promotores. Portanto, há uma alta probabilidade de encontrar sequências similares em outras

regiões genômicas que não as promotoras (HOWARD e BENSON, 2002; SIVARAMAN *et al*., 2005 *apud* DE AVILA E SILVA, 2006). De forma complementar, analisando as sequências de genes utilizadas nos dados de entrada, constatou-se que estas possuem dimensão muito maior do que as sequências de promotores e terminadores, reforçando a probabilidade de encontrar uma sequência de promotor ou terminador dentro de uma sequência de gene. Em função destas percepções, acredita-se que os genes podem ser uma fonte de ruídos, impedindo a distinção entre os três *clusters*. Além disso, analisando os agrupamentos de genes das janelas de 4n até 7n da Tabela 3, em comparação aos agrupamentos genes das janelas 2n e 3n, observa-se que os *clusters* perdem continuamente a característica de separação dos genes. Desta forma, conclui-se que janelas com vetores *X*n de dimensão elevada não são eficientes para sequências como genes e promotores.

A separação dos terminadores em um *cluster* específico deve-se à constituição das sequências dos terminadores da *E. coli*, dos quais são compostos pelo tipo intrínseco (*Rho*-independentes) e pelo tipo fator-dependente. Os intrínsecos são constituídos por uma sequência de aproximadamente 20 nucleotídeos, rica em GC. Por outro lado, os fatores-dependentes são constituídos por sequências de nucleotídeos que se estendem de 150 a 200 pares, formando uma sequência muito variada (CIAMPI, 2006; LEWIN, 2001 *apud* CORNO, 2010). Acredita-se que a menor separação dos terminadores com janela de 3n deve-se ao fato de que conjuntos menores (2n) de nucleotídeos caracterizam com maior qualidade uma sequência de dimensão menor como a de terminadores, pelo fato desta janela 2n apresentar menor perda de informação na busca de ocorrências.

A partir da Tabela 3 pode-se também constatar que não houve diferenças entre os resultados com as abordagens de *CW* e *MW* para os vetores 2n e 3n. Desta forma, optou-se por prosseguir com as simulações utilizando apenas a abordagem de *MW*, da qual assemelha-se com o processo de leitura do *DNA* pelas enzimas biológicas (ALBERTS *et al*., 2002; LEWIN, 2001).

Deste modo, percebe-se que o objetivo inicial de caracterização das sequências em três *clusters* distintos não foi alcançado. Assim, após a análise destes resultados, realizaram-se outras simulações, nas quais foi removido o conjunto de genes dos dados de entrada, em função da suspeita deste estar causando ruídos. Nesta segunda simulação aplicaram-se novamente os vetores de

2n e 3n, porém abrangendo somente *MW*, conforme resultados apresentados na Tabela 4.

Tabela 4 - *K-Means*: resultados 2ª simulação

Configuração		**Tempo de execução (aprox.)**	**Quantidade de objetos por *cluster***			
Nucleotídeos	***Window***		***Cluster***	**Promotor**	**Gene**	**Terminador**
2n	*MW*	1 min	C1	8452	-	21
			C2	0	-	231
3n	*MW*	5 min	C1	3897	-	152
			C2	4555	-	100

Analisando os resultados da segunda simulação constatou-se que a retirada dos genes qualificou a clusterização para o vetor 2n, no qual obteve-se o *cluster* C2, contendo semente terminadores, e o *cluster* C1, contendo todos os promotores e com a mesma quantidade de terminadores ruidosos apresentada na primeira simulação. No entanto, para a clusterização com o vetor 3n, permaneceu com uma divisão indefinida, reforçando a hipótese de que ocorrências de conjuntos menores de nucleotídeos caracterizam com maior qualidade uma sequência.

Desta forma, para comprovar esta segunda hipótese, realizou-se uma terceira simulação, mantendo as mesmas configurações e dados de entrada, apenas aumentando o número de nucleotídeos, conforme resultados apresentados na Tabela 5.

Tabela 5 - *K-Means*: resultados 3ª simulação

Configuração		**Tempo de execução (aprox.)**	**Quantidade de objetos por *cluster***			
Nucleotídeos	***Window***		***Cluster***	**Promotor**	**Gene**	**Terminador**
4n	*MW*	7 min	C1	4268	-	119
			C2	4184	-	133
5n	*MW*	4 min	C1	7	-	0
			C2	8445	-	252
6n	*MW*	13 min	C1	6	-	0
			C2	8446	-	252
7n	*MW*	22 min	C1	2	-	0
			C2	8450	-	252

Esta terceira simulação confirmou a hipótese de que conjuntos menores de nucleotídeos caracterizam uma sequência com mais distinção em relação aos demais segmentos. Na Tabela 5 observa-se o *cluster* C2 com um acumulo maior de objetos, tanto promotores, quanto terminadores. No entanto, nas execuções dos vetores 5n, 6n e 7n originaram-se alguns promotores ruidosos, que podem possuir significado relevante sob o aspecto biológico.

Dentre as demais simulações realizadas, constatou-se que a escolha aleatória do centroide inicial pode variar o número final de objetos no mesmo *cluster*, para uma mesma configuração, assim como o tempo final de execução, neste mesmo cenário. No entanto, por serem variações mínimas, o centroide inicial aleatório apresentou-se insignificante se considerado o conjunto total de objetos de cada segmento. Porém, se o objetivo da análise forem os ruídos, este parâmetro pode influenciar, alterando os objetos ruidosos gerados que foram obtidos no final da execução.

5.2 SIMULAÇÕES UTILIZANDO *CURE*

As simulações com o algoritmo *CURE* utilizaram como dados de entrada os atributos extraídos dos segmentos de promotores, genes e terminadores. Os parâmetros e configurações gerais utilizados estão descritos na Tabela 6.

Tabela 6 - Configurações gerais para *CURE*

	Parâmetros	**Valores**	**Descrição**
Atributos	Vetor *M*-dimensional	2n e 3n	Conjunto de nucleotídeos para os vetores *M*-dimensionais.
	Window	*CW* e *MW*	Seleção da frequência dos termos (Seção 4.3).
CURE	*Clusters*	C1 até C50	Quantidade de *clusters* no estado final do algoritmo.
	Representantes	De 4 até 8	Quantidade de representantes de cada grupo.
	Fator *alpha*	De 0,4 até 0,95	Fator *alpha* de deslocamento dos representantes de cada grupo.
	Cálculo de distâncias	*Euclidean*	*Euclidean* (4) (Seção 3.2.1).

Na primeira simulação utilizaram-se quatro representantes, aplicaram-se os vetores 2n e 3n, abrangendo *CW* e *MW*, com fator *alpha* de *0,7*. Analisando os resultados da distribuição da quantidade de objetos para cada um dos segmentos, constatou-se que *CURE* não separou os terminadores, genes e promotores, gerando apenas um *cluster* contendo a maior parte dos objetos, conforme apresentado na Tabela 7.

Tabela 7 - *CURE*: resultados 1ª simulação

Configuração		**Tempo de execução (aprox.)**	**Quantidade de objetos por *cluster***			
Nucleotídeos	***Window***		***Cluster***	**Promotor**	**Gene**	**Terminador**
2n	*MW*	30 min	C1	0	1	0
			C2	8452	4631	250
			C3	0	0	2
2n	*CW*	30 min	C1	0	0	1
			C2	8452	4632	249
			C3	0	0	2
3n	*MW*	8 h	C1	0	1	0
			C2	8452	4631	251
			C3	0	0	1
3n	*CW*	8 h	C1	0	1	0
			C2	8452	4631	251
			C3	0	0	1

Assim como nas simulações com *K-Means*, supôs-se que os genes são uma fonte de ruídos, impedindo a geração dos *clusters* distintos. Além disso, constatou-se novamente que não houve diferenças entre as abordagens de *CW* e *MW* para os vetores 2n e 3n. Desta forma, optou-se por prosseguir com as simulações utilizando a mesma configuração, porém com a abordagem de *MW* e removendo os genes dos dados de entrada, conforme apresentado na segunda simulação na Tabela 8.

Tabela 8 - *CURE*: resultados 2ª simulação

Configuração		Tempo de execução (aprox.)	Quantidade de objetos por *cluster*			
Nucleotídeos	***Window***		***Cluster***	**Promotor**	**Gene**	**Terminador**
2n	*MW*	10 min	C1	0	-	3
			C2	8452	-	249
3n	*MW*	8 h	C1	8452	-	251
			C2	0	-	1

A segunda simulação com a remoção dos genes, no entanto não trouxe resultados semelhantes aos do *K-Means*. Também não foram identificadas mudanças significativas entre vetores de atributos de 2n e 3n. Desta forma, utilizando a mesma configuração da segunda simulação, porém somente com vetor 2n, realizou-se uma terceira simulação, alterando os parâmetros do fator *alpha* e o número final de *clusters*, conforme resultados apresentados na Tabela 9.

A partir dos resultados da terceira simulação, constatou-se que a variação do fator *alpha* e o aumento do número de *clusters* não determinou que *CURE* aglomerasse os objetos em, ao menos, dois grupos distintos. Apenas verificou-se que a variação do parâmetro *alpha* ocasionou pequenas variações de quantidade de objetos não agrupados.

Com o resultado destas três simulações, constatou-se que as alterações dos parâmetros não surtiram efeito sobre os *clusters* finais gerados por *CURE*, dos quais não equivaleram aos segmentos de promotores e terminadores. A partir da revisão bibliográfica de Toronen (2004) (Seção 3.4), considerou-se que *clusters* relevantes, de promotores e terminadores, pudessem estar sendo formados em níveis intermediários da árvore binária de agrupamentos gerados por *CURE*. Portanto verificou-se que a análise de *CURE* exigia uma abordagem diferenciada através de novas execuções do algoritmo, que constituíram-se em aumentar consideravelmente o número de *clusters* finais esperados, de modo a interromper o processo de agrupamento em vários níveis anteriores as aglomerações finais. Na Tabela 10 são apresentados os resultados desta quarta simulação que tratou de encontrar o ponto correto de parada do algoritmo, equivalente ao nível de corte do dendrograma, que possui a quantidade final de *clusters* gerados. Nesta tabela são identificadas cada uma das execuções dos algoritmos com a configuração correspondente, sendo que

coluna *clusters* apresenta a quantidade final de agrupamento ao término da execução, assim como a coluna de representantes (Repres.) que mostra a quantidade de representantes utilizados. A coluna alteração representa qual alteração de configuração foi realizada entre a execução corrente e a anterior, rotulando o resultado desta alteração como efeito positivo ou negativo em relação ao comportamento esperado, apresentado pela coluna resultado.

Tabela 9 - *CURE*: resultados 3ª simulação

Configuração			**Tempo de execução (aprox.)**	**Quantidade de objetos por *cluster***			
Fator *alpha*	**Nucleotídeos**	***Window***		***Cluster***	**Promotor**	**Gene**	**Terminador**
0,7	2n	*MW*	10 min	C1	0	-	3
				C2	8452	-	249
0,7	2n	*MW*	10 min	C1	0	-	1
				C2	8452	-	249
				C3	0	-	1
				C4	0	-	1
0,7	2n	*MW*	10 min	C1	0	-	4
				C2	0	-	1
				C3	0	-	2
				C4	8452	-	243
				C5	0	-	1
				C6	0	-	1
0,4	2n	*MW*	10 min	C1	8452	-	250
				C2	0	-	2
0,4	2n	*MW*	10 min	C1	0	-	1
				C2	8452	-	249
				C3	0	-	1
				C4	0	-	1
0,4	2n	*MW*	10 min	C1	0	-	1
				C2	0	-	2
				C3	8452	-	246
				C4	0	-	1
				C5	0	-	1
				C6	0	-	1

Tabela 10 - Resultados 4ª simulação

Execução	*Clusters*	Fator *alpha*	Nucleotídeos	Repres.	Alteração	Resultado
1	2	0,7	2n	8	Representantes	Negativo
2	4	0,7	2n	8	*Clusters*	Negativo
3	6	0,7	2n	8	*Clusters*	Negativo
4	20	0,7	2n	4	*Clusters;Repres.*	Negativo
5	50	0,95	2n	4	Alpha; *Clusters*	Positivo
6	2	0,95	2n	4	*Clusters*	Negativo
7	45	0,95	2n	4	*Clusters*	Positivo
8	35	0,95	2n	4	*Clusters*	Positivo
9	20	0,95	2n	4	*Clusters*	Negativo
10	30	0,95	2n	4	*Clusters*	Positivo

Analisando as execuções 1 e 3 da Tabela 10, constatou-se que a alteração de representantes não provocou efeitos relevantes e as alterações da quantidade de *clusters* finais não atingiram o nível de corte que apresentasse grupos separados de promotores e terminadores, ou seja, resultado marcado como negativo. Na execução 4, aumentou-se o número de *clusters* para 20, retornando a quantidade de representantes para 4, porém o resultado continuo sendo negativo. Na execução 5, aumentou-se então o número de *clusters* para 50 e o fator *alpha* para 0,95, com o objetivo a aprofundar o ponto de corte no dendrograma (Seção 3.3), que confirmou-se em dois grandes grupos de objetos, sendo um deles constituído, em sua maioria, de objetos promotores e outro grupo constituído, em sua maioria, de objetos terminadores, ou seja, é um nível de corte significativo, com resultado positivo. Para comprovar que o causador deste efeito positivo não foi o fator *alpha* e sim o número de *clusters*, realizou-se a execução 6, reduzindo o número de *clusters* para 2, mantendo o *alpha* em 0,95. Desta forma, o foco das execuções 7 até 10 ficou em torno do número de *clusters*, conforme previsto inicialmente, das quais confirmou-se que o nível do corte significativo estava entre 20 e 30 *clusters* finais.

Com o objetivo de melhorar a eficácia da análise do nível de corte ideal, agregou-se na ferramenta desenvolvida uma saída parcial de dados que apresentava os agrupamentos intermediários conforme a convergência dos mesmos. Com estas informações foi possível gerar as tabelas: Tabela 11, Tabela 12 e Tabela 13 e efetuar uma análise mais consistente do comportamento de *CURE*. Estas tabelas assemelham-se a um dendrograma, onde cada linha representa um

ponto de corte em um determinado nível, que apresenta a quantidade de *clusters* formados. Neste determinado nível ressaltou-se agrupamentos relevantes, isto é, um *cluster* com muitos objetos do mesmo tipo, representados pelas colunas *clusters 1-5*. Lê-se, no nível 397*,* para um o *cluster* 1, são encontrados 3949 genes (3949G) e 4 promotores (4P), por exemplo.

Tabela 11- *Clusters* relevantes da 1ª simulação

	***Clusters* relevantes**				
Quantidade de *clusters* (nível)	***Cluster* 1**	***Cluster* 2**	***Cluster* 3**	***Cluster* 4**	***Cluster* 5**
500	2552G	1397G 3P	1969P	4066P	1542P
397	3949G 4P	4327P	3601P		
396	3949G 3605P	4327P			
250	3952G 8224P	610G			
221	4563G 8451P	20T			
169	4564G 8451P	14T			
168	4564G 8451P 14T	49T			
66	4564G 8451P 14T	163T			
57	4564G 8451P 177T	5T	3T		
19	4564G 8451P 212T	22T			
	Aglomeram-se demais objetos de *clusters* com pequenas quantidades.				
1	4632G 8452P 252T				

Legenda: G = genes; P = promotores; T = terminadores.

Analisando a Tabela 11, referente à primeira simulação, encontra-se no nível 500 algumas aglomerações de promotores, pertencentes aos *clusters* 3, 4 e 5. Na transição do nível 397 ao nível 396 observa-se que o *cluster* 1, com quantidade elevada de genes, é aglomerado a um grande grupo de promotores, perdendo assim a função de representação da separação distinta dos genes. A partir do nível 250 um novo grupo relevante de genes é gerado (610G) e a quantidade de promotores do *cluster* 1 continua elevando-se consideravelmente, descaracterizando ainda mais o grupo. No nível 221, iniciam-se as aglomerações relevantes dos terminadores (20T). Na transição do nível 169 ao nível 168 o grupo de terminadores 20T é aglomerado ao *cluster* 1, junto aos demais genes e promotores. Porém, ainda nesta transição, é

gerado um novo grupo relevante de terminadores em *cluster* 2 (49T). Na sequência, até o nível 66, observa-se que o *cluster* 1 permanece inalterado, enquanto o *cluster* 2 segue agregando elementos, gerando um grupo que distingue significativamente os terminadores (163T). A partir deste ponto de corte em diante (nível), não obteve-se agrupamentos relevantes, apenas objetos de promotores, genes e terminadores de características ruidosas e que podem ter alguma relevância biológica, pelo fato de não terem sido aglomerados em seus devidos grupos em níveis anteriores.

Ainda na análise da Tabela 11, é possível identificar no nível 500 e no nível 397, pequenos grupos ruidosos 4P e 3P que estão unificados a grandes grupos de genes. A pergunta em questão é o que estes objetos de promotores possuem de comum aos genes, e se os mesmos possuem implicação biológica. A partir desta análise, é possível observar que os grupos de promotores e terminadores são unificados diretamente ao de genes, comprovando assim o efeito poluidor deste segmento.

Tabela 12 - *Clusters* relevantes da 2ª simulação

	***Clusters* relevantes**		
Quantidade de *clusters* (nível)	***Cluster* 1**	***Cluster* 2**	***Cluster* 3**
200	8449P	33T	8T
100	8450P	137T	
79	8450P	161T	5T
78	8450P 1T	161T	5T
53	8451P 1T	197T	
52	8451P 198T	2T	
25	8452P 225T		
	Aglomeram-se demais objetos T de *clusters* com pequenas quantidades.		
1	8452P 252T		

Legenda: P = promotores; T = terminadores.

Na Tabela 12, referente à segunda simulação, verifica-se que do nível 200 ao nível 79 o algoritmo gerou um agrupamento com quase a totalidade de promotores (*cluster 1*), que expressa a qualidade no agrupamento sem a presença de genes. No mesmo nível, encontra-se também um grupo relevante de terminadores (137T) que representa 64% da totalidade dos terminadores. No nível

78 observa-se que o grupo de genes contém um terminador ruidoso, do qual pode ter alguma relevância biológica, sendo que o mesmo permanece sem alteração até o nível 52, onde um conjunto volumoso de terminadores une-se ao *cluster* 1. Assim como os resultados apresentados na Tabela 11, é possível verificar na Tabela 12 que os objetos terminadores também apresentam um certo grau de ruído. Em comparação a primeira simulação, nesta segunda constatou-se que o algoritmo aglomerou os terminadores representativos em níveis predecessores, ou seja, o algoritmo reconheceu antecipadamente que os terminadores se parecem mais entre si do que com promotores ou genes.

Tabela 13 - *Clusters* relevantes da 6ª execução da 4ª simulação

	***Clusters* relevantes**		
Quantidade de *clusters* (nível)	***Cluster* 1**	***Cluster* 2**	***Cluster* 3**
200	8207P	243P	22T
100	8450P	117T	
63	8450P	169T	
62	8450P 1T	169T	
23	8451P 1T	225T	
22	8451P 226T		
12	8452P 226T		
	Aglomeram-se demais objetos T de *clusters* com pequenas quantidades.		
1	8452P 252T		

Legenda: P = promotores; T = terminadores.

Durante a realização das execuções da quarta simulação, extraíram-se os *clusters* relevantes da execução de número 6, apresentados na Tabela 13. Optou-se por esta execução, pois a mesma possui fator *alpha* 0,95, servindo assim como base de comparação entre as extrações da Tabela 12 e Tabela 11, que utilizaram fator *alpha* 0,7. Em função desta alteração de parâmetro, observou-se que a Tabela 13 possui resultados semelhantes à Tabela 12, porém o grupo relevante de promotores (*cluster* 1) aglomera um grupo relevante de terminadores (226T) apenas no ponto de corte de nível 22, e não no 52 como apresentado na segunda simulação. A partir disto, constata-se que os terminadores foram identificados com mais objetos até este nível, tornando o grupo de terminadores mais distinto. Este fato deve-se ao

aumento na aproximação dos representantes ao centro do *cluster*, provocado pelo fator *alpha* elevado, de valor 0,95. Portanto, conforme De Amo, (2004) e Xu e Wunsch (2005) (Seção 3.3.2), quanto maior o fator *alpha*, mais próximo os representantes ficarão do centro do *cluster*, aumentando a eficácia do cálculo de distância, tornando o algoritmo reduzidamente sensível a ruídos.

De um modo geral, não foram constatadas alterações relevantes de agrupamentos em função do número de representantes utilizados nos parâmetros das simulações. Uma vez que, ao elevar o número de representantes juntamente com fator *alpha* (*alpha* de 0,4 para 0,95 e representantes de 4 para 8), constatou-se que a retração provocada torna o representantes aproximadamente equidistantes aos demais *clusters.* Desta forma, mesmo em quantidade elevada, provocam comportamentos semelhantes nas aglomerações. Portanto, um cenário interessante para ser analisado seria com um número elevado de representantes e fator *alpha* próximo à zero.

6 CONCLUSÃO

O objetivo deste trabalho consistiu em efetuar uma análise genômica do *DNA* da bactéria *E. coli*, aplicando os algoritmos de clusterização *K-Means* e *CURE*, através da construção de um programa *desktop* implementados por uma linguagem de programação. Os resultados obtidos foram analisados e comparados de modo a demonstrar a eficácia do reconhecimento de padrões por vias computacionais, contribuindo assim nos estudos de pesquisas com padrões de genes, promotores e terminadores desta bactéria.

A metodologia aplicada, que teve base nas etapas da análise de *cluster,* não produziu os resultados esperados utilizando como formatação de entrada a técnica de coocorrência implementada neste trabalho. Acredita-se que este resultado está associado à natureza complexa e ruidosa da representação dos dados biológicos. Além disso, a decisão de quais algoritmos seriam utilizados foi tomada com base na revisão bibliográfica realizada em trabalhos envolvendo esta área de pesquisa. Os algoritmos foram desenvolvidos na ferramenta *desktop*, conforme a proposta inicial e os resultados obtidos através da análise dos dados juntamente com um especialista da área.

Os *clusters* gerados apresentaram resultados significativos, considerando que ambos os algoritmos apresentaram grupos de objetos bem definidos, em pelo menos um contexto de configuração. No entanto, alguns contextos de configurações estenderam-se em tempo de execução além da viabilidade aceitável, ou não geraram grupos significativos de promotores, genes e terminadores. Com base nisto, conclui-se que, a proposta inicial de clusterizar todos os segmentos utilizando vetores de 2n até 8n (ou 10n) e obter grupos distintos, não se adequou a realidade encontrada. Desta forma, foi necessário reformular a estratégia utilizando apenas vetores de 2n e 3n, aproximando assim os resultados dos objetivos iniciais. Deste modo, percebeu-se que expoentes maiores caracterizam genes, mas se perde informação de promotores e terminadores. Além disso, a ideia inicial de obter três *clusters* distintos mostrou-se demasiado ruidosa.

As perspectivas de trabalhos futuros envolvem uma exploração mais elaborada da extração de atributos, pois esta fase é determinante para a definição das características relevantes de quaisquer objetos. Tratando-se de *DNA*, a partir das pesquisas realizadas com coocorrência, pode ser desenvolvido um trabalho

utilizando a teoria de *ATE* de Ji (2008) (Seção 3.2.2) para a contabilização de frequência de termos. Além disso, este trabalho pode ainda ser complementado através do desenvolvimento de componentes que organizam os dados em gráficos e exibem os mesmos através da ferramenta implementada. Como por exemplo, gráficos de dendrogramas para *CURE*, erro quadrático para *K-Means*, *weblogo* (CROOKS, 2004) e alinhamento de sequências de cada *cluster*. Na ferramenta desenvolvida, é possível aplicar algumas melhorias já identificadas, como formatação restritiva para o arquivo de entrada, configuração de personalização dos dados de saída e pesquisa (*online*) dos conceitos biológicos para cada uma das sequências. Agregando assim funcionalidades extras a ferramenta, tornando-a mais robusta de modo a facilitar a análise dos *clusters*.

REFERÊNCIAS

ALBERTS, Bruce; JOHNSON, Alexander; LEWIS Julian *et al*. **Molecular Biology of the Cell**. 4. ed. New York: Garland Science, 2002. 1616 p.

ALBERTS, Bruce *et al*. **Fundamentos da Biologia Celular.** 3. ed. Rio de Janeiro: Artmed, 2011. 864 p.

BARRERA, Junior; CESAR-Jr, Roberto M.; FERREIRA, João E., GUBITOSO, Marco E. An environment for knowledge discovery in biology. **Computers in Biology and Medicine**, v. 34, n.5, p. 427-447, 2004.

BERG, Jeremy M.; TYMOCZKO, John L.; STRYER, Lubert. **Biochemistry**. 5. ed. New York: W H Freeman, 2002. 1050 p.

BERGERON, Bryan P. **Bioinformatics computing.** New Jersey: Prentice Hall Professional, 2003. 439 p.

BERKHIN, Pavel. A survey of clustering data mining techniques. In:**Grouping multidimensional data**. Springer Berlin Heidelberg, 2006. p. 23-26.

BLATT, Marcelo; WISEMAN, Shai; DOMANY, Eytan. Superparamagnetic clustering of data. **Physical Review Letters,** v. 76, p.3251-3255, 1996.

CALLEBAUT, Werner. Scientific perspectivism: a philosopher of science's response to the challenge of big data biology. **Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences**, v. 43, n. 1, p. 69-80, 2012.

CHEN, Huiwen D.; FRANKEL, Gad. Enteropathogenic Escherichia coli: unravelins pathogenesis. **FEMS Microbil**. Rev., v. 29, 2005.

CHEN, Mo. **Fully vectorized kmeans algorithm**. Matlab Central, 2012. Disponível em <http://www.mathworks.com/matlabcentral/fileexchange/24616-kmeans-clustering> Acesso em: 29/05/2013.

CIAMPI, M. Sofia. Rho-dependent terminators and transcription termination. Microbiology, 2006, 152: 2515–2528.

COX, Michael M.; DOUDNA, Jennifer A.; O`DONNELL, Michael. **Biologia Molecular:** Princípios e Técnicas. Rio de Janeiro: Artmed, 2012. 944 p.

CROOKS, Gavin E. *et al*. WebLogo: a sequence logo generator. **Genome research**, v. 14, n. 6, p. 1188-1190, 2004.

DE AMO, Sandra. **Técnicas de Mineração de Dados**. Congresso da Sociedade Brasileira de Computação, 14. 2004, Salvador, Brasil. p.1 - 43.

DE AVILA E SILVA, Scheila. **Redes Neurais Artificiais aplicadas na caracterização e predição de regiões promotoras**. 2006. 144 f. Dissertação (Mestre em Computação Aplicada) - Curso de Pós-graduação em Computação

Aplicada, Departamento de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio Dos Sinos, São Leopoldo, 2006.

DE ROBERTIS, Eduardo D. P. **Bases da biologia celular e molecular**. 2. ed. Rio de Janeiro: Guanabara Koogan, 1993. 307 p.

EDWARDS, Robert G.; SOKAL, Alan D.. Generalization of the fortuin-kasteleyn-swendsen-wang representation and monte carlo algorithm. **Physical Review D,** New York, v. 38, n. 6, 07 jan. 1988.

FRALEY, Chris.; RAFTERY, Adrian E.. How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. **Technical Report**, No 329. Department of Statistics, University of Washington, USA, 1998.

FRAMEWORK, Microsoft .NET. **Windows Microsoft .NET Framework**. Disponível em: <http://www.microsoft.com/net>. Acesso em: 14/05/2013.

GIANNINI, John L. **Biological Animations**: Transcription. http://www.stolaf.edu/people/giannini/. Disponível em: <http://www.stolaf.edu/people/giannini/flashanimat/molgenetics/transcription.swf>. Acesso em: 19 ago. 2013.

GUHA, Sudipto; RASTOGI, Rajeev; SHIM, Kyuseok. CURE: An Efficient Clustering Algorithm for Large Databases. **ACM SIGMOD Record**, ACM, v. 27, n. 2, p.73-84, 1998.

HAN, Jiawei; KAMBER, Micheline. **Data Mining Concepts and Techniques.**Second Edition University Of Illinois At Urbana-champaign: Elsevier, 2001. 772 p.

HOLT, John G. **Bergey's manual of determinative bacteriology**. 9 ed Baltimode: Williams & Wilkins, 1994. 787 p.

HOWARD, Daniel; BENSON, Karl. Evolutionary computation method for pattern recognition of cis-acting sites. **BioSystems**, v. 72, n.1/2 , p. 19-27, 2002.

JAIN, Anil K.; DUBES, Richard C. **Algorithms for Clustering Data**. New Jersey: Pretice-Hall, 1998. 320 p.

JAIN, Anil K.; MURTY, M. N.; FLYNN, P.j.. **Data Clustering: A Review**. **Acm Computing Surveys**, New York - Usa, v. 31, n. 3, p.264-323, 01 set. 1999.

JI, He. Improving feature representation of natural language gene functional annotations using automatic term expansion: Computational Intelligence in Bioinformatics and Computational Biology. **CIBCB '08. IEEE Symposium On**, n. , p.173-179, 15-17 set. 2008.

JONES, Neil C.; PEVZNER, Pavel A.. **An Introduction to Bioinformatics Algorithmics.** London, England: Bradford, 2004. 435 p.

KAUFMAN, Leonard; ROUSSEEUW, Peter J. **Finding groups in data: an introduction to cluster analysis.** 1 edition New York: Wiley Interscience, 2005. 368 p.

KLEINBERG, Jon. An impossibility theorem for clustering. **Advances in neural information processing systems**, p. 463-470, 2003.

LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Introduction to algorithms.** 2. ed. Massachusetts, EUA: The MIT Press, 2001.

LEWIN, Benjamin. **Genes VII**. 1. ed. Porto Alegre: Artes Médicas, 2001. 1002 p.

LIU, Libin; HO, Yee-kin; YAU, Stephen. Clustering DNA sequences by feature vectors. **Molecular phylogenetics and evolution**, v. 41, n. 1, p. 64-69, 2006.

LI, Yanpeng *et al*. A Framework for Semi supervised Feature Generation and Its Applications in Biomedical Literature Mining: Computational Biology and Bioinformatic. **IEEE/ACM Transactions On**, v. 8, n. 2, p.294-307, mar-abr. 2011.

MACQUEEN, James. **Proceedings of the Fifth Berkley Symposium on Mathematical Statistics and Probability:** Some methods for classification and analysis of multivariate observations. Berkley: University Of California, 1967.

MAIMON, Oded; ROKACH, Lior. **Data Mining and Knowledge Discovery Handbook**. Second Edition Israel: Springer Science, 2010. 1306 p.

MICROSOFT C#. **Microsoft C# Language**. Disponível em: < http://msdn.microsoft.com/en-us/library/vstudio/ms228593(v=vs.100).aspx>. Acesso em: 15/05/2013.

MURRAY, Patrick R.; ROSENTHAL, Ken S.; PFALLER, Michael A.. **Microbiologia Médica**. 6. ed. Rio de Janeiro: Elsevier, 2010. 1072 p. Do original: Medical Microbiology, 6th edition.

PROSDOCIMI, Francisco *et al*. Bioinformática: manual do usuário. **Biotecnologia Ciência & Desenvolvimento**, v. 29, p. 12-25, 2002.

RADJIMAN, Sugiarto; LIANYI, Han; JIAN-SHENG, Wang; YU, Chen Z.. Super Paramagnetic Clustering of DNA Sequences. **Journal Of Biological Physics,** Singapore, p. 11-25. 01 jan. 2006.

SHAMIR, Ron; SHARAN, Roded. **Algorithmic Approaches to Clustering Gene Expression Data**. 2001. 41 f. Current Topics In Computational Biology (1) - Tel-aviv University, Tel-aviv 69978, 2001.

SIVARAMAN, Karthikeyan; SESHASAYEE, Aswin Sai Narain; SWAMINATHAN, Krishnakumar; MUTHUKUMARAN, Geetha; PENNATHUR, Gautam. Promoter addresses: revelations from oligonucleotide profiling applied to the Escherichia coli genome. **Theoretical Biology an Medical Modelling**, v. 2, 2005.

SZWARCFITER, Jayme L. **Grafos e algoritmos computacionais.** 2. ed. Rio de Janeiro: Campus, 1986. 216 p.

TORONEN, Petri. Selection of informative clusters from hierarchical cluster tree with gene classes. **Bmc Bioinformatics**, A. I. Virtanen Institute For Molecular Sciences, Finland, v. 5, n. 1, p.1-19, 25 mar. 2004.

TORTORA, Gerard J.; FUNKE, Berdell R; CASE, Christine L. **Microbiologia**. 8. ed. Porto Alegre: Artmed, 2005. 893p.

TRABULSI, R,L.*et al*. **Microbiologia:Escherichia**. 3°edição. São Paulo: Editora Atheneu,1999.

VISUAL STUDIO 2010, Professional. **MSDN – Microsoft Visual Studio.** Disponível em: <http://msdn.microsoft.com/en-us/library/52f3sw5c(v=vs.100).aspx>. Acesso em: 15/05/2013.

WATSON, James D.; CRICK, Francis. Molecular Structure of Nucleic Acids: A structure for deoxyribose nucleic acid. **Nature**: Scientific Journal, Worldwide, v. 171, n. 4356, p.737-738, 25 abr. 1953.

XU, Rui; WUNSCH, Donald C. Clustering algorithms in biomedical research: a review. **Biomedical Engineering, IEEE Reviews in**, v. 3, p. 120-154, 2010.

XU, Rui; WUNSCH, Donald C. Survey of Clustering Algorithms. **IEEE Transactions On Neural Networks**, Usa, v. 16, n. 3, p.645-678, 09 maio 2005.

ANEXO A – ALGORITMO *K-MEANS*

1: **Parâmetros de entrada:**

k, número de *clusters*, sendo $2 \leq k \leq n$, onde *n* é o total de objetos;

T, limite máximo de iterações;

M, matriz de dados lineares, sendo *x* um objeto correspondente a uma linha da matriz.

2: **Definição dos centroides iniciais:**

Dado um método facultativo, define-se $\{c_1^0, \ldots, c_k^0\}$ a partir de *M*.

3: **Para *t* = 1 até *T* faça**

4: **Passo de alocação**

Associa-se cada x_i, sendo $i = 1, \ldots, n$, a um *cluster k**, considerando a menor $d(x_i, c_{k*}^t)$ dada pela equação:

$$k^* = min_{z=1,..,k}\, d(x_i, c_z^t) = \sqrt[q]{\sum_{l=1}^{p} \left(\left|x_{il} - c_{zl}^t\right|\right)^q}$$

onde $q \geq 1$ e *p* = quantidade de atributos em x_i.

5: **Passo de representação**

Calcula-se os centroides c_z^t, sendo $z = 1, \ldots, k$, calculando-se o valor médio de cada atributo representado por μ_{zl}, sendo $l = 1, \ldots, p$, através da equação:

$$\mu_{zl} = \frac{1}{n_z} \sum_{i=1}^{n_z} x_{zli}$$

onde n_z = quantidade de objetos no *cluster z.*

6: **Critério de parada**

7: **Se nenhum objeto foi alocado no passo 4 então**

8: **Interrompe-se o processo**

9: **Fim se**

10: **Fim para**

Fonte: adaptação de MACQUEEN (1967)

ANEXO B – ALGORITMO *CURE*

1: **Parâmetros de entrada:**

k, número de clusters finais esperados;

c, número de representantes que serão selecionados nos *clusters*;

α, fator *alpha* utilizado na retração dos representantes;

M, matriz de dados lineares, sendo *x* e *p* um objeto correspondente a uma linha da matriz.

2: **Definição dos *clusters* iniciais:**

Define-se cada *x* de *M* como sendo um *cluster*.

3: **Calculam-se as distâncias entre todos os objetos (matriz de dissimilaridade) e constroem-se as estruturas auxiliares *k-tree* e *heap*:**

T = construir_k_tree(*M*)

Q = construir_heap(*M*), mantendo-se *Q* ordenado crescentemente pelas distâncias dos pares de objetos mais próximos.

4: **Enquanto tamanho(*Q*) > *k* faça**

5: **Extrair par de objetos que possui menor distância entre si:**

u = extrair_menor_distância(*Q*), ou seja, objeto na primeira posição de *Q*.

v = *u.mais_próximo*

6: **Unificar (*merge*) *u* e *v* formando um único *cluster w:***

w = Unificar(*u*, *v*)

7: **Calcular centroide de *w:***

$$w_c = \frac{n_u * u_c + n_v * v_c}{n_u + n_v}$$

onde w_c = o centroide de *w*, n_u = a quantidade de objetos no *cluster* u, u_c = o centroide do *cluster* u, n_v = quantidade de objetos no *cluster* v e v_c = o centroide do *cluster* v.

8: **Selecionar representantes:**

tmpSet = vazio

Para *i* de 1 até *c* faça

maxDist = 0

Para cada *p* em *w* faça

Se *i* == 1

mimDist = distância(*p*, *w.c*)

Se Não

mimDist = min { distância(*p*, *q*) : *q* Є tmpSet }

Fim Se

Se *minDist* >= *maxDist*

maxDist = *minDist*

maxPoint = *p*

Fim Se

Fim Para

tmpSet = *tmpSet* U { maxPoint }

Fim Para

Onde *tmpSet* = objetos representantes do cluster *w.*

9: **Calcular fator *alpha*:**

Para cada *p* em *tmpSet* faça

w.r = *w.r* U { *p* + α * (*w.c* - *p*) }

Fim Para

Onde *w.r* é o conjunto de representantes de *w* que são transferidos de *tmpSet* para *w* no mesmo instante em que o fator *alpha* é calculado.

10: **Reorganizar estrutura *k-tree* e *heap*:**

Remover representantes de *u* de *T*

Remover representantes de *v* de *T*

Inserir representantes de *w* em *T*

11: **Calcular objeto (*cluster*) mais próximo a *w*:**

w.mais_próximo = *x*, sendo *x* um objeto qualquer em *Q*.

Para cada *x* em *Q* faça

Se distância(*w*, *x*) < distancia(*w*, *w.mais_próximo*)

w.mais_próximo = x

Fim se

Se (*x.mais_próximo* == *u*) ou (*x.mais_próximo* == *v*)

Se distância(*x, x.mais_próximo*) < distância(*x, w*)

x.mais_próximo = cluster_mais_próximo(*z,* distância(*x, w*))

Ou seja, sendo *z* um objeto de *Q*, busca-se um *cluster* z tal que a distância(*x, z*) < distância(*x, w*). Se encontrado, então *x.mais_próximo* = *z*. Se não encontrado, então *x.mais_próximo* = *w*.

Se Não

x.mais_próximo = *w*

Fim se

Se Não, Se distância(*x*, *x.mais_próximo*) > distância(*x*, *w*)

x.mais_próximo = *w*

Fim se

Realocar *x* em *Q*, ou seja, reposicionar *x* em *Q* mantendo *Q* ordenado pelas distâncias.

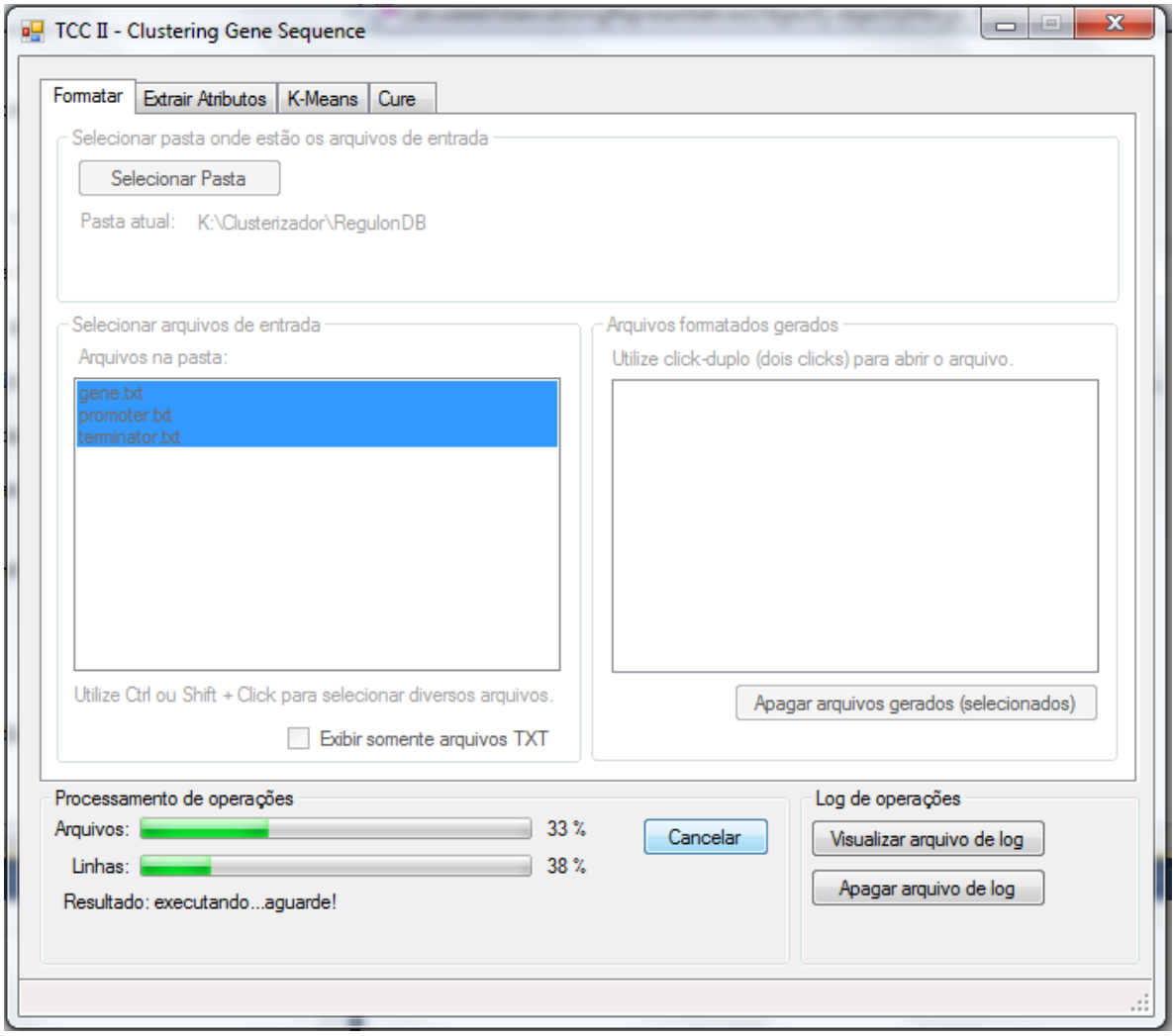
Fim Para

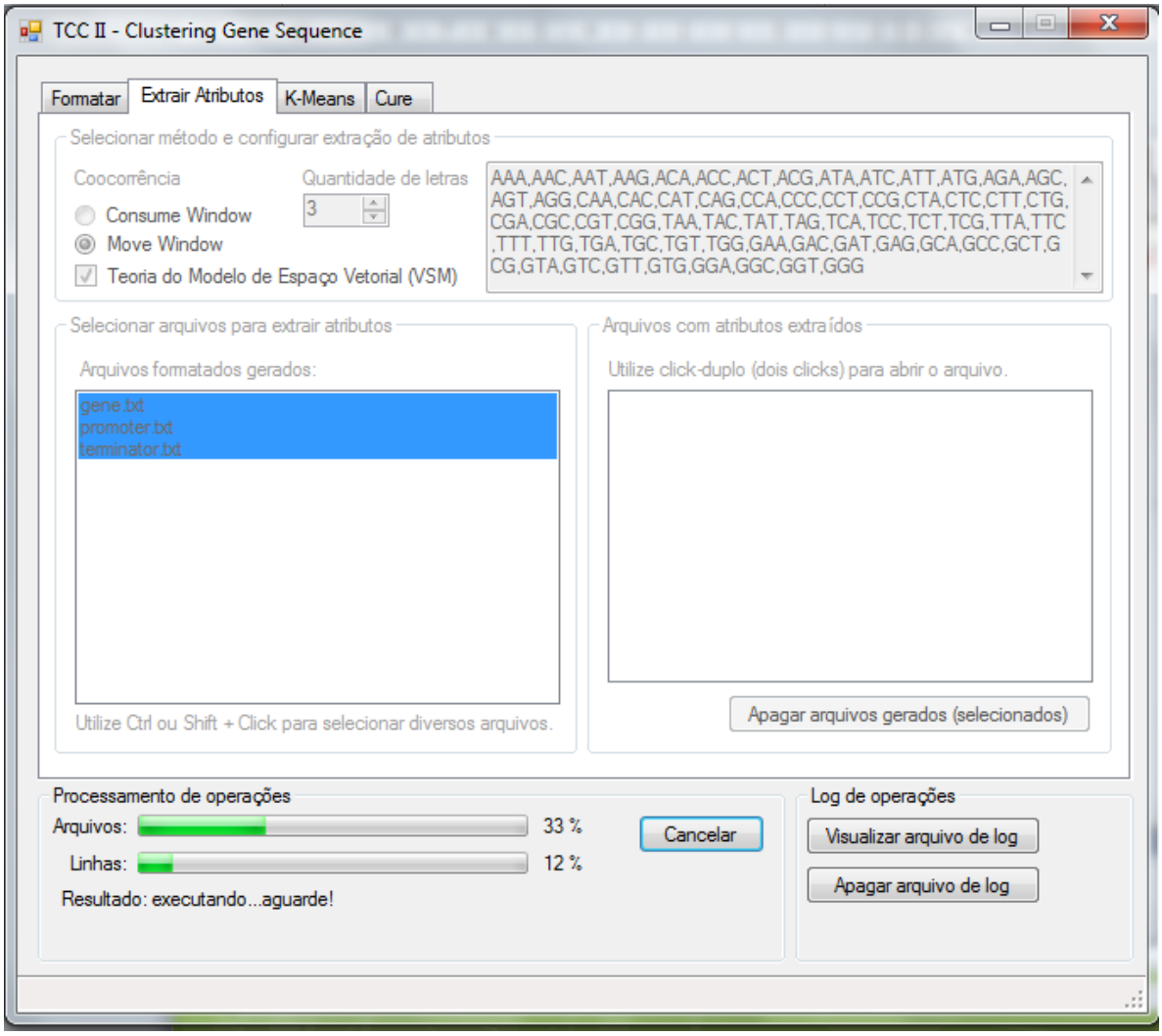
12: **Inserir *w* em *Q***

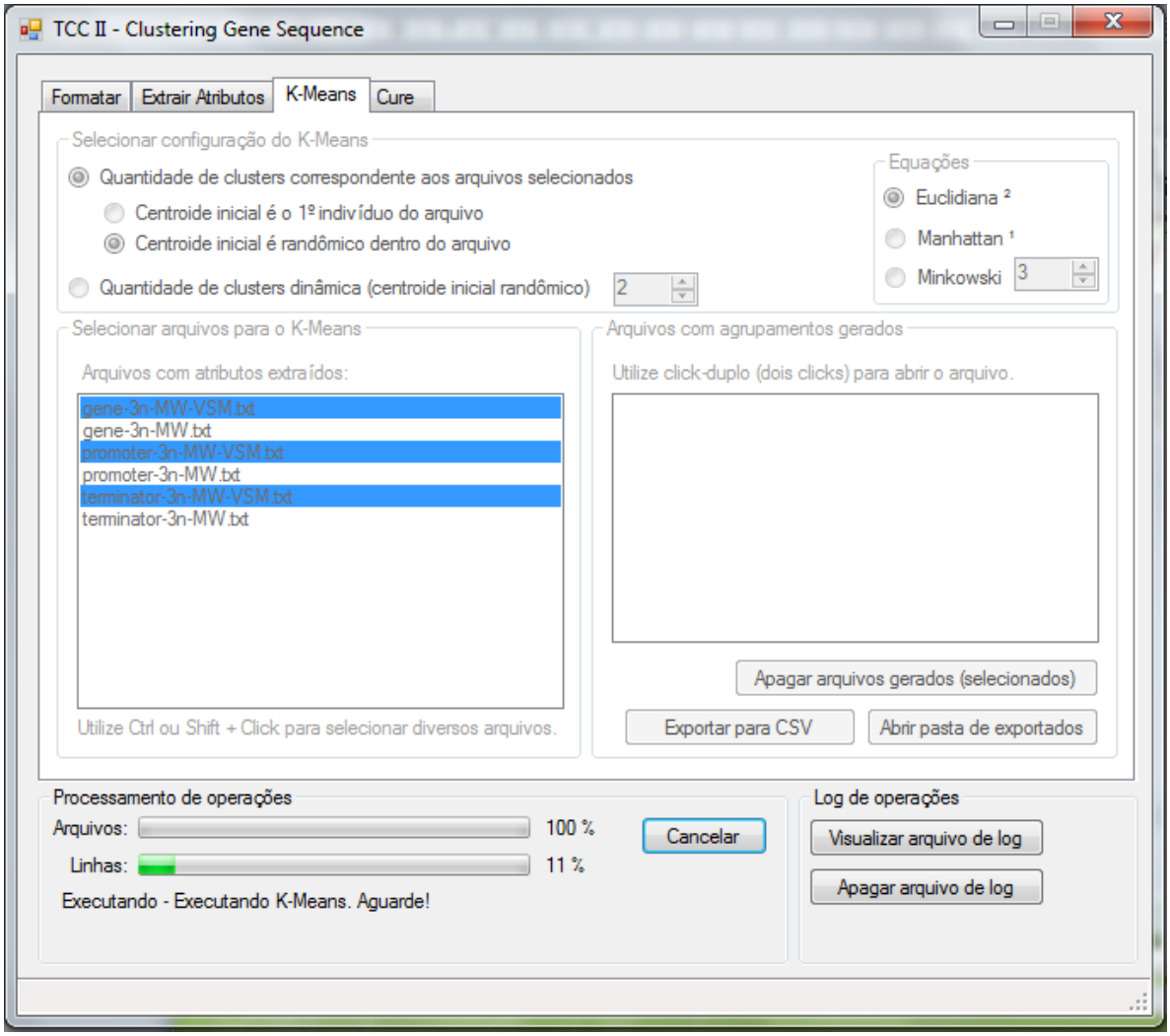
13: **Fim enquanto**

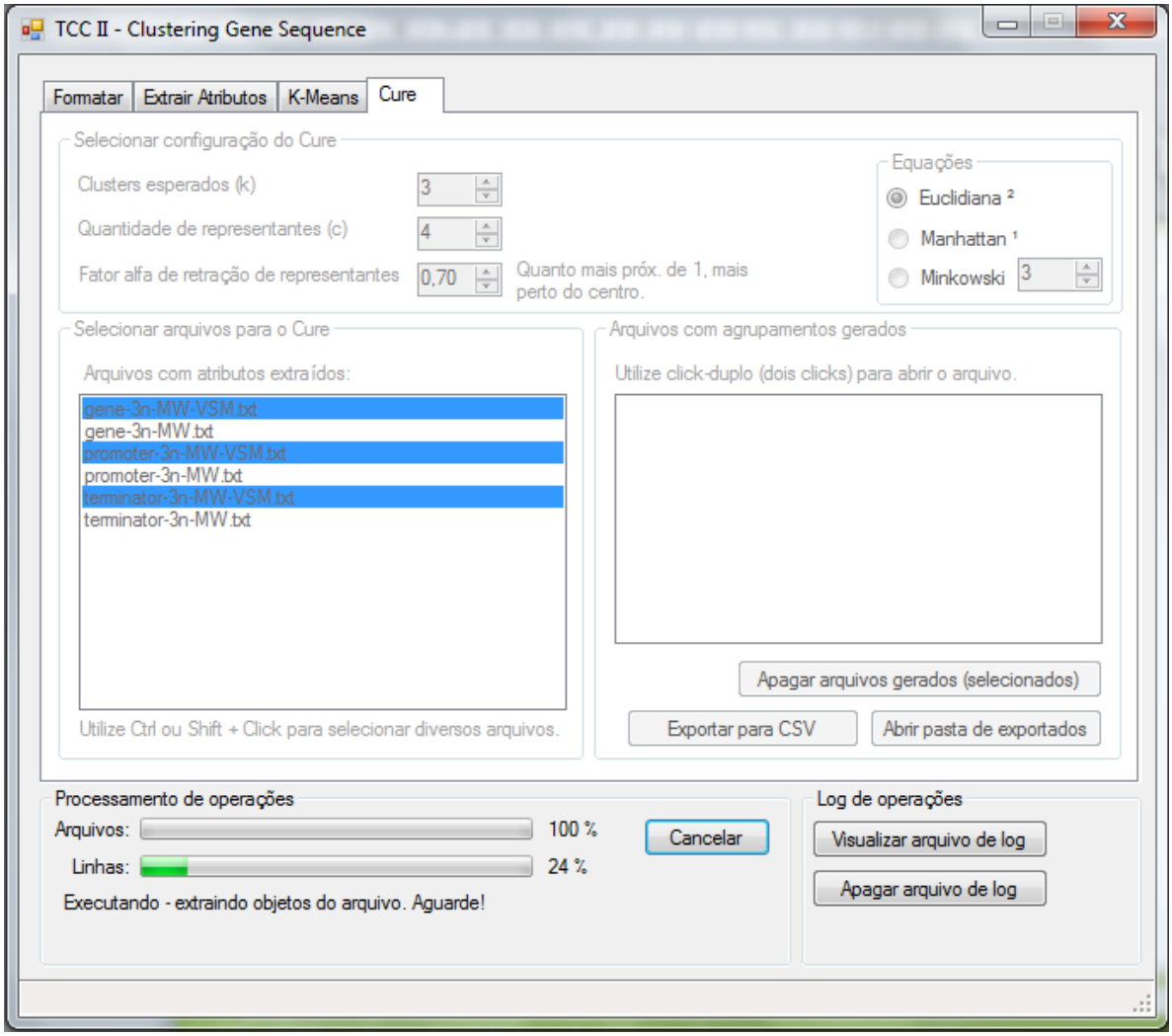
Fonte: adaptação de GUHA (1998)

ANEXO C - APLICATIVO *DESKTOP*




TCC II - Clustering Gene Sequence
Formatar
Extrair Atributos
K-Means
Cure
Selecionar método e configurar extração de atributos
Coocorrência
Quantidade de letras
3
Consume Window
Move Window
Teoria do Modelo de Espaço Vetorial (VSM)
AAA,AAC,AAT,AAG,ACA,ACC,ACT,ACG,ATA,ATC,ATT,ATG,AGA,AGC,AGT,AGG,CAA,CAC,CAT,CAG,CCA,CCC,CCT,CCG,CTA,CTC,CTT,CTG,CGA,CGC,CGT,CGG,TAA,TAC,TAT,TAG,TCA,TCC,TCT,TCG,TTA,TTC,TTT,TTG,TGA,TGC,TGT,TGG,GAA,GAC,GAT,GAG,GCA,GCC,GCT,GCG,GTA,GTC,GTT,GTG,GGA,GGC,GGT,GGG
Selecionar arquivos para extrair atributos
Arquivos formatados gerados:
gene.txt
promoter.txt
terminator.txt
Utilize Ctrl ou Shift + Click para selecionar diversos arquivos.
Arquivos com atributos extraídos
Utilize click-duplo (dois clicks) para abrir o arquivo.
Apagar arquivos gerados (selecionados)
Processamento de operações
Arquivos:
33 %
Linhas:
12 %
Cancelar
Resultado: executando...aguarde!
Log de operações
Visualizar arquivo de log
Apagar arquivo de log


TCC II - Clustering Gene Sequence
Formatar
Extrair Atributos
K-Means
Cure
Selecionar configuração do K-Means
Quantidade de clusters correspondente aos arquivos selecionados
Centroide inicial é o 1º indivíduo do arquivo
Centroide inicial é randômico dentro do arquivo
Quantidade de clusters dinâmica (centroide inicial randômico)
2
Equações
Euclidiana ²
Manhattan ¹
Minkowski
3
Selecionar arquivos para o K-Means
Arquivos com atributos extraídos:
gene-3n-MW-VSM.txt
gene-3n-MW.txt
promoter-3n-MW-VSM.txt
promoter-3n-MW.txt
terminator-3n-MW-VSM.txt
terminator-3n-MW.txt
Utilize Ctrl ou Shift + Click para selecionar diversos arquivos.
Arquivos com agrupamentos gerados
Utilize click-duplo (dois clicks) para abrir o arquivo.
Apagar arquivos gerados (selecionados)
Exportar para CSV
Abrir pasta de exportados
Processamento de operações
Arquivos:
100 %
Linhas:
11 %
Cancelar
Executando - Executando K-Means. Aguarde!
Log de operações
Visualizar arquivo de log
Apagar arquivo de log


TCC II - Clustering Gene Sequence
Formatar
Extrair Atributos
K-Means
Cure
Selecionar configuração do Cure
Clusters esperados (k)
3
Quantidade de representantes (c)
4
Fator alfa de retração de representantes
0,70
Quanto mais próx. de 1, mais perto do centro.
Equações
Euclidiana ²
Manhattan ¹
Minkowski
3
Selecionar arquivos para o Cure
Arquivos com atributos extraídos:
gene-3n-MW-VSM.txt
gene-3n-MW.txt
promoter-3n-MW-VSM.txt
promoter-3n-MW.txt
terminator-3n-MW-VSM.txt
terminator-3n-MW.txt
Utilize Ctrl ou Shift + Click para selecionar diversos arquivos.
Arquivos com agrupamentos gerados
Utilize click-duplo (dois clicks) para abrir o arquivo.
Apagar arquivos gerados (selecionados)
Exportar para CSV
Abrir pasta de exportados
Processamento de operações
Arquivos:
100 %
Linhas:
24 %
Cancelar
Executando - extraindo objetos do arquivo. Aguarde!
Log de operações
Visualizar arquivo de log
Apagar arquivo de log