UNIVERSIDADE DE CAXIAS DO SUL

Centro de Ciências Exatas e Tecnologia

Curso de Bacharelado em Ciência da Computação

JAQUELINE CELESTINI

DIAGRAMAS DE VORONOI – APLICAÇÃO NA REDE DE INTEGRAÇÃO PROTÉICA

Caxias do Sul

2009

Jaqueline Celestini

DIAGRAMAS DE VORONOI – APLICAÇÃO NA REDE DE INTEGRAÇÃO PROTÉICA

Trabalho de Conclusão do Curso
para obtenção do Grau de
Bacharel em Ciência da
Computação da Universidade de
Caxias do Sul

Daniel Luis Notari
Orientador

Caxias do Sul
2009

AGRADECIMENTOS

Primeiramente a Deus, por abençoar a mim e a minha família.

Aos meus pais, Derli e Iracilda, por todo amor e carinho dedicados ao longo de minha vida, por custearem meus estudos possibilitando chegar até aqui e pelo apoio e compreensão durante toda jornada acadêmica, aceitando minhas ausências quando os trabalhos e provas me impediam de ir visitá-los.

As minhas irmãs Daniela e Vânia, pelo companheirismo nas brincadeiras quando crianças, pelos conselhos, pela ajuda prestada nos mais diversos assuntos, mas principalmente por ter compartilhado por vários anos, as alegrias e sufocos inerentes a qualquer jornada acadêmica.

A minha grande amiga Janaina, pelo companheirismo diário, pelas risadas, choros, festas, alegrias e tristezas compartilhadas e pelo apoio e incentivo prestados durante a realização deste trabalho.

Aos meus queridos amigos e colegas com quem compartilhei grande parte de minha jornada acadêmica, pessoas que com certeza vão estar presentes em diversos momentos da minha vida.

Ao meu orientador, professor Daniel, pela dedicação, disponibilidade e credibilidade em mim depositados. Obrigado pelas idéias, pelas bibliografias fornecidas e pelos ensinamentos passados, que muito contribuíram para o desenvolvimento deste trabalho, e um agradecimento especial pela paciência e incentivo demonstrados.

A minha colega Renata de Paris, por despertar em mim o interesse pelo campo da Bioinformática.

Enfim, a todos familiares, amigos e colegas de trabalho que possibilitaram a realização da minha jornada acadêmica e deste trabalho.

EPÍGRAFE

“Nada é Impossível
Impossível é apenas uma palavra grande dita por aí
por homens pequenos, que acham mais fácil viver no
mundo que deram para eles, do que explorar
o poder que eles têm de transformá-lo.
Impossível não é um fato. É uma opinião.
Impossível não é uma declaração. É um desafio.”

Autor desconhecido

RESUMO

Atualmente inúmeras pesquisas médicas, auxiliadas pela bioinformática, buscam analisar as interações entre as proteínas, com o objetivo de decifrarem a função celular, os processos biológicos e os componentes celulares onde a proteína analisada atua, pois essas informações são de grande importância quando a proteína em questão está relacionada a fatores de impacto social, como por exemplo, as pandemias. Essa integração protéica, representada através de redes de livre escala, pode ser modelada por estruturas matemáticas conhecidas como Diagrama de Voronoi. Estudado na Geometria Computacional, o Diagrama de Voronoi tem por objetivo modelar redes de livre escala através de grupos de poliedros desenhados em torno de cada ponto da rede, no caso em questão, em torno de cada proteína. Com a aplicação dessa modelagem na integração protéica, é possível analisar informações mais aprofundadas sobre a estrutura e funcionamento molecular das proteínas relacionadas a fatores de impacto social.

Palavras-Chave: Bioinformática, Redes de Proteínas, Geometria Computacional e Diagramas de Voronoi.

ABSTRACT

There are numerous medical research these days which, aided by bioinformatics aim at analyzing interactions among proteins. The main objective of such analyses is to unfold cell functioning, biological processes and components where specific protein operates, for these excerpts of information are of great importance when the protein relates to socially impacting factors, such as in pandemic events. Such protein integration represented through scale-free networks may be modeled by mathematical structures known as Voronoi Diagram. The Diagram is studied by Computing Geometry and aims at modeling scale-free networks through polyandry groups which are drawn around each point of the network, specifically around each protein, in this case. By applying this modeling in protein integration it is possible to analyze deeper information about molecular structures and functioning of proteins related to social impact.

Keywords: Bioinformatics, Protein Networks, Computational Geometry and of Voronoi Diagrams.

LISTA DE FIGURAS

QUADROS

LISTA DE ABREVIATURAS

API	Documentação sobre bibliotecas e pacotes da linguagem Java
COOH	Grupo Carboxila
COX1	Proteína
DNA	Ácidos Desoxirribonucléicos
DDBJ	DNA Data Bank Japan
$\in$	Símbolo matemático - pertence
EBI	European Bioinformatics Institute
EMBL	European Molecular Biology Laboratorie
Embnet	European Molecular Biology Network
GDB	Genome Data Bank
JAVA	Linguagem de programação orientada a objetos
JUNG	Biblioteca gráfica em linguagem Java
KEGG	Kyoto Encyclopedia of Genes and Genome
mRNA	RNA Mensageiro
NCBI	National Center for Biotechnology Information
NH2	Grupo amino
NIGMS	Instituto Nacional de Ciências Médicas Gerais dos EUA
NSF	Fundação Nacional de Ciências dos EUA
P53	Proteína
PDB	Protein Data Bank
PIR	Protein Informacion Resource
R4	Sistema de coordenadas de quatro vetores
RNA	Ácidos Ribonucléicos
SGDB	Sistema de Gerenciamento de Bancos de Dados
SIG	Sistema de Coordenadas Geográficas
tRNA	RNA Transportador
UCS	Universidade de Caxias do Sul

SUMÁRIO

INTRODUÇÃO

Desde a década de 60 a Biologia e a Informática trabalham juntas, facilitando, por exemplo, a geração de hipóteses experimentais e a economia de tempo e recursos. Mas foi em meados de 1993, com o advento do projeto Genoma, que surgiu a BioInformática, definida como o uso de técnicas computacionais e estatísticas, matemática aplicada e criação e otimização de algoritmos na resolução de problemas práticos e formais relacionados a coleta, armazenagem e análise de dados gênomicos (Prosdocimi, 2002). O desafio atual na era pós-genoma é determinar a função e o papel biológico de cada uma das seqüências de proteínas. Se os genes são os portadores das instruções que permitem a "construção" de um determinado organismo, as proteínas são responsáveis por sua estrutura e funcionamento.

Sendo as moléculas mais abundantes e importantes nas células, as proteínas são fundamentais sob diversos aspectos da estrutura e função celulares além de expressarem a maior parte das informações genéticas (Bebek, 2007). Para cada proteína existe um segmento do DNA que guarda informações sobre ela, especificando sua seqüência de aminoácidos, sendo que a forma e outras propriedades de cada proteína são dadas pela seqüência de aminoácidos que a constitui (Raychaudhuri, 2006). A visualização desses dados é uma área especial da bioinformática e a representação gráfica é quase sempre a maneira mais efetiva de descrever, explorar e condensar um grande volume de informações numéricas.

É através da plotagem de arquivos extraídos de bancos de dados protéicos, tais como o PDB (Protein Data Bank – http://www.rcbs.org/pdb), que abriga a ficha técnica e o desenho em 3D de milhares de proteínas, que é possível visualizar a interação entre as proteínas e perceber que estas geram estruturas complexas, costumeiramente representadas sob a forma de redes, conhecidas matematicamente como redes de livre escala, por facilitar a análise e a visualização (Waterman, 2000).

Essas redes, formadas por nós e conectores, podem descrever redes sociais, de informações, tecnológicas ou biológicas. Nas redes tecnológicas, por exemplo, os nós são representados pelos computadores e os conectores são os cabos. Dentro do contexto

das redes biológicas, nas redes de proteínas, os nós são representados pelas proteínas e os conectores pelas reações químicas.

Para a manipulação dessa rede, existem alguns softwares disponíveis gratuitamente. Um desses softwares que possibilita diversas representações gráficas das redes protéicas é o Cytoscape[1]. Desenvolvido em uma parceria entre o Instituto Nacional de Ciências Médicas Gerais dos EUA (NIGMS) e a Fundação Nacional de Ciências dos EUA (NSF), esse software é uma plataforma global para a visualização e análise de complexas redes de interação moleculares e que possibilita a inclusão de novas funcionalidades, com o uso de *plugins*, disponíveis em seu *website* ou mesmo desenvolvido por usuários ou pesquisadores.

Entre as possíveis representações gráficas das redes protéicas, uma é com o uso dos Mosaicos de Voronoi, representados como um grupo de poliedros semelhantes aos encontrados em colméias, pois possuem propriedades matemáticas interessantes para a representação das redes. Uma delas é a associação entre pontos vizinhos. Através dessas associações é possível identificar e analisar as proteínas com maior número de conexões, conhecidas como *hubs*. Essas proteínas são associadas de maneira empírica a funções mais importantes dentro das células, visto que por serem identificadas como mais "velhas" possuem maior preferência nas conexões (Barabasi, 2004; Oltvai, 2004).

Os estudos dessas redes e proteínas podem elucidar o comportamento e função celular, possibilitando grandes avanços na medicina, relacionados ao tratamento e cura de doenças, principalmente as que causam impactos sociais, como as epidemias.

O objetivo deste trabalho é estudar a aplicação dos mosaicos de Voronoi na representação gráfica das redes de proteínas e desenvolver um aplicativo utilizado no software Cytoscape para a manipulação dessas redes, visando a identificação de proteínas *hubs*. Um dos possíveis algoritmos para a utilização do modelo computacional do Diagrama de Voronoi será selecionado e implementado na forma de um *plugin* com o uso da linguagem Java. Este trabalho foi divido em 6 capítulos, descritos a seguir, para melhor detalhar as informações e etapas envolvidas na criação deste aplicativo.

No capítulo 2, "Bioinformática e Biologia Molecular", é feito uma rápida explanação sobre a relação entre informática e biologia, seu conceito, como e quando surgiu e qual sua contribuição em pesquisas genéticas. Nesse capítulo são abordados

[1] Website: http://www.cytoscape.org.br

também os conceitos de biologia molecular e características de seus principais representantes: DNA, RNA e proteínas, estas de forma mais detalhada em relação a sua organização e representação estrutural – redes de livre escala, sobre a qual será aplicado o modelo computacional conhecido como Mosaico de Voronoi.

No capítulo 3, "Geometria Computacional", são abordados conceitos relacionados a utilização de modelos computacionais para a resolução de problemas matemáticos, tais como complexidade, campos de atuação e propriedades matemáticas. São comentados alguns modelos computacionais clássicos, entre eles o Mosaico de Voronoi, cuja explicação mais detalhada aborda desde sua criação em 1908, passando pela construção de uma de suas possíveis implementações até sua contribuição para a pesquisa genética, além de mostrar algumas formas naturais onde pode ser visualizado.

No capítulo 4, "Implementação", é descrito de forma detalhada como foi construída a implementação do *plugin* Voronoi, que será utilizado no software de visualização de redes protéicas Cytoscape. Esse capítulo engloba uma breve descrição do Cytoscape, o conceito de *plugin*, qual a linguagem utilizada e o porque da escolha, os diagramas de aplicação, de classes e de uso desenvolvidos e utilizados para a sua construção.

No capítulo 5, "Estudos de Caso", são mostradas algumas redes protéicas obtidas através do repositório de informações genéticas String, disponível no website http://string-db.org/, sobre as quais foram aplicadas o *plugin* desenvolvido neste trabalho.

Por fim, o capítulo 6, "Conclusão", é apresentado um resumo do trabalho desenvolvido e possíveis contribuições futuras para sua melhoria.

2 BIOINFORMÁTICA E BIOLOGIA MOLECULAR

Nesse capítulo será introduzido alguns processos e áreas de atuação da bioinformática, iniciando com um conceito geral de biologia molecular, mais detalhadamente nas proteínas e como a biologia computacional pode ajudar na representação e estudo dessas proteínas.

2.1 Bioinformática

No contexto das novas descobertas na área da Biologia, como o projeto genoma humano na década de 90 (Passarge, 2004) e suas decorrências (projetos transcriptoma - RNA e proteoma – Proteínas), os termos Biologia Computacional, Bioinformática e Biologia de Sistemas surgiram para definir e nomear as áreas que essas novas descobertas abrangem.

Biologia Computacional é o termo utilizado para identificar o uso de qualquer ferramenta matemática e/ou computacional que possibilite processar e armazenar o grande volume de dados gerados por pesquisas e experiências biológicas em qualquer âmbito (Kingsbury, 1996).

Já a Bioinformática, uma das subáreas da Biologia Computacional, é definida como o uso de técnicas computacionais e estatísticas, matemática aplicada e criação e otimização de algoritmos na resolução de problemas práticos e formais relacionados à coleta, armazenagem e análise de dados biológicos, especificamente de ácidos nucléicos e proteínas (Prosdocimi, 2002). Seu objetivo é converter as informações em linguagem biológica, seqüências de informações contidas em proteínas ou nucleotídeos, em conhecimento bioquímico e biofísico, funções estruturais, evolutivas e funcionais.

Biologia de Sistemas é o termo usado para definir o estudo das interações entre os componentes de um sistema biológico, e como essas interações fazem emergir função e comportamento no sistema (por exemplo, genes e enzimas). Sua abordagem consiste em utilizar teorias, modelagens matemática e computacional e experimentos

para descrever as interações de um sistema celular (Weston, 2004; Hood, 2004) e um dos métodos mais utilizados é a teoria dos grafos (Barabasi, 2004).

A bioinformática pode ser utilizada sob dois diferentes aspectos. Em uma primeira abordagem, a bioinformática pode ser complementar e fornecer infra-estrutura para o desenvolvimento dos trabalhos de pesquisa. Numa segunda abordagem, a bioinformática é responsável pela interpretação do conhecimento gerado, que será utilizado por outras áreas de pesquisa.

Um dos aspectos mais distintos da bioinformática é o vasto uso da web. Muitos dos programas desenvolvidos pela comunidade da bioinformática são disponibilizados através da web. Estão disponíveis na web, imensos bancos de dados com seqüências de DNA e estruturas em 3D de proteínas disponíveis para pesquisas (Cohen, 2004). Entre os bancos e programas disponíveis, alguns dos mais conhecidos:

- *NCBI (National Center for Biotechnology Information)*: Fundado em 1988, nos Estados Unidos, como um recurso nacional de informações sobre biologia molecular, onde são criados bancos de dados públicos, conduzem pesquisas em biologia computacional, desenvolvem ferramentas para análise de dados do genoma e disseminação de informações biomédicas – visando uma melhor compreensão dos processos moleculares das doenças que afetam a saúde humana. O banco de dados NCBI está disponível através do website http://www.ncbi.nlm.nih.gov/.

- *EBI (European Bioinformatics Institute)*: É uma fundação acadêmica sem fins lucrativos, que é parte integrante do Laboratório Europeu de Biologia Molecular (EMBL). O EBI funciona como um centro de investigação e serviços em bioinformática. O Instituto gerencia bases de dados biológicos que incluem informações sobre os ácidos nucléicos, seqüências protéicas e estruturas macromoleculares. O EBI está disponível através do website http://www.ebi.ac.uk/.

- *Bioinformatics Journal:* Um dos principais periódicos no campo da Bioinformática, tem seu foco principal sobre novos desenvolvimentos em genoma, bioinformática e biologia computacional. Esse periódico é uma publicação da Oxford Journals que pode ser acessado através do website http://bioinformatics.oxfordjournals.org.

- *Embnet (European Molecular Biology Network):* O Embnet é um grupo de colaboradores da biologia molecular, que presta serviços a comunidade científica da área da biologia molecular. É acessado através do website http://www.embnet.org/.

- *BioMed Central*: é uma editora de acesso público, com enfoque na área da Ciência, Tecnologia e Medicina. Além de disponibilizar diversos artigos gratuitamente, o BioMed Central possui assinatura de periódicos sobre o genoma e a biologia molecular e disponibiliza acesso a diversos serviços e produtos da área. É acessado através do website http://www.biomedcentral.com.

2.2 Biologia Molecular

A natureza química do material genético foi o propulsor de várias pesquisas para desvendá-la. Graças a isso, foi descoberto que o DNA era a molécula que armazenava a informação genética, sua estrutura química foi desvendada por Watson e Crick em 1953 e, finalmente, com a descoberta dos códigos genéticos e do fluxo da informação biológica (dos ácidos nucléicos para as proteínas), os biólogos introduziram o conceito do Dogma Central (Figura 1), estabelecendo que o DNA atua como um modelo para se replicar, é transcrito no RNA, que por sua vez é convertido em proteína (Gibas, 2001, Prosdocimi, 2002).

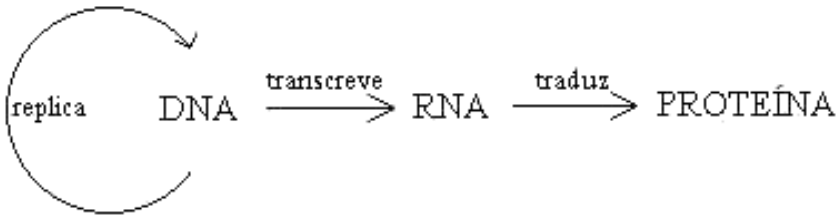


Figura 1. Dogma Central (Fonte: Prosdocimi, 2002).

O princípio do dogma central declara que a transferência de informações genéticas entre ácidos nucléicos ou entre ácido nucléico e proteína pode ser possível, mas que entre proteínas ou entre proteína e ácido nucléico ainda não é possível. Isso decorre do fato que a informação genética possui uma seqüência precisa, assim como as bases dos ácidos nucléicos e os resíduos de aminoácidos das proteínas.

Entender a herança genética sempre foi um dos problemas da biologia molecular. Desde 1865, quando Mendel determinou um modelo matemático para a herança genética, sendo o gene a unidade básica desse modelo, passando por 1944 quando o gene foi reconhecido como parte do DNA e 1953 com o descobrimento da estrutura de dupla hélice do DNA por James Watson e Francis Crick que o mundo científico tenta solucionar essa questão (Waterman, 2000).

A informação genética dos seres vivos está presente nos ácidos nucléicos, que são substâncias orgânicas complexas, que desempenham duas grandes funções nas células: coordenar a síntese das proteínas e transmitir as informações genéticas (Passarge, 2004). Os nucleotídeos que compõe os ácidos nucléicos são os mais importantes nas células, pois possuem os genes, representados por dois grandes conjuntos, explicados a seguir.

2.2.1. DNA - Ácidos Desoxirribonucléicos

O DNA é a base da hereditariedade, composto de açúcar e fosfatos, está localizado nos cromossomos, no núcleo das células e consiste numa molécula complexa, de cadeia longa, chamada ácido desoxirribonucléico. Ligado à molécula de açúcar está uma das quatro bases nitrogenadas, representadas pelas letras A (adenina), C (citosina), G (guanina) e T (timina). É a seqüência dessas bases ao longo da molécula do DNA que armazena a informação genética, que será transcrita pelo RNA.

A estrutura do DNA (Figura 2) é composta por um par de moléculas, formando uma dupla hélice, que se mantêm estabilizada por pontes de hidrogênio entre as bases presas as duas cadeias. Essa estrutura explica duas importantes funções do DNA: a replicação e a transmissão das informações genéticas. A replicação é feita por uma das cadeias da dupla hélice, que serve de molde para a replicação/formação de uma nova cadeia quando a hélice é desenrolada (Passarge, 2004). A transmissão é feita com um filamento do DNA em conjunto com o RNA e será explicado posteriormente.

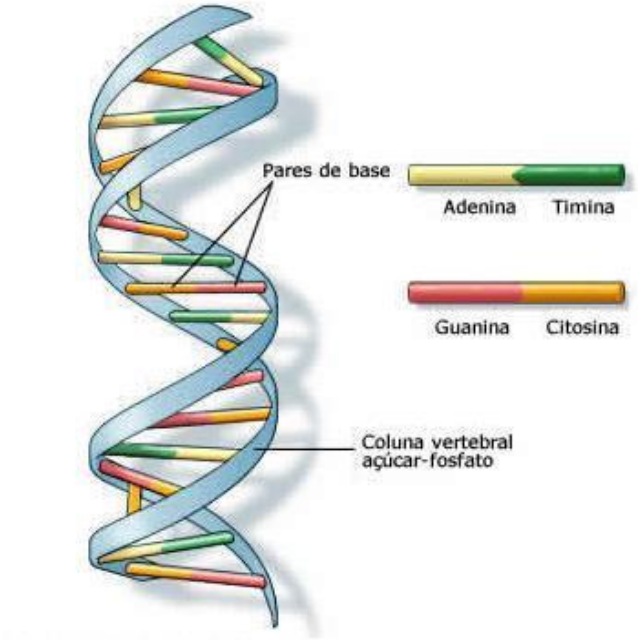


Figura 2. DNA (Fonte: Passarge, 2004)

2.2.2 RNA - Ácidos Ribonucléicos[2]

O RNA tem uma constituição semelhante a do DNA, mas tem como principais diferenças: i) suas bases hidrogenadas que são a A (adenina), C (citosina), G (guanina) e U (uracila); ii) sua estrutura, que geralmente se apresenta numa única cadeia simples e iii) sua função, de transcrição e síntese de proteínas, além de atuar como catálise em importantes reações biológicas.

Dentre as classes de RNA, as que participam da transcrição do DNA são o RNA Mensageiro (mRNA), que contém a informação genética para a seqüência de aminoácidos e o RNA Transportador (tRNA), que identifica e transporta as moléculas de aminoácido.

Na transcrição, a seqüência de nucleotídeos do DNA é transcrito no mRNA, após o desenrolamento da dupla-hélice do DNA. Em seguida, o mRNA é sintetizado por um conjunto complexo de proteínas, em seqüencias que são complementares as do filamento do DNA (Figura 3).

[2] Fonte bibliográfica utilizada em toda a seção 2.2.2: (Passarge, 2004)

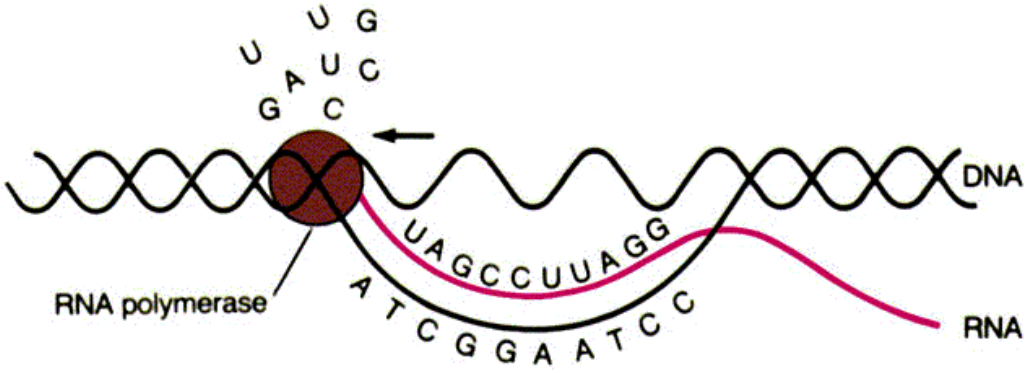


Figura 3. Transcrição do DNA (Fonte: Passarge, 2004).

O tRNA é utilizado durante a tradução da seqüência contida no mRNA. Sua função é reunir os aminoácidos, de acordo com a seqüência determinada pelas bases nucleotídicas do mRNA.

2.2.3 Proteínas

As proteínas são as moléculas mais abundantes e importantes nas células, pois são fundamentais sob diversos aspectos da estrutura e função celulares além de expressarem a maior parte das informações genéticas (Bebek, 2007). Para cada proteína existe um segmento do DNA que guarda informações sobre ela, especificando sua seqüência de aminoácidos, sendo que a forma e outras propriedades de cada proteína são dadas pela seqüência de aminoácidos que a constitui (Raychaudhuri, 2006). Como a função das proteínas está ligada à sua estrutura e interações, o conhecimento detalhado dessas características ajuda a entender o papel que cada proteína desempenha (Passarge, 2004).

Quimicamente falando, as proteínas são moléculas orgânicas formadas por uma ou mais cadeias polipeptídicas, pertencem à classe dos peptídeos, pois são formadas por aminoácidos ligados entre si por ligações peptídicas. Uma ligação peptídica é a união do grupo amino ($-NH_2$) de um aminoácido com o grupo carboxila (-COOH) de outro aminoácido, através da formação de uma amida (Figura 4).

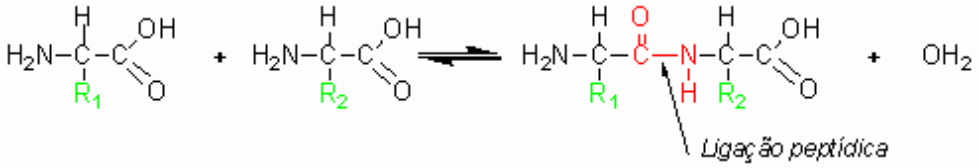


Figura 4. Ligação peptídica entre dois aminoácidos que compõe a estrutura de uma proteína. (Fonte: Passarge, 2004).

Cada proteína possui uma configuração particular (Figura 5), em sua maioria formam arranjos tridimensionais, quase globulares, e foram divididas em cinco categorias (Passarge, 2004):

- *Estrutura Primária*: são as proteínas que se apresentam em longas cadeias distendidas. Um exemplo dessa estrutura é a insulina, composta por aproximadamente 50 aminoácidos.

- *Estrutura Secundária*: são as proteínas que possuem forma helicoidal ou folha pregueada. Essa estrutura se mantém estabilizada através de pontes de hidrogênio entre o grupo NH de um aminoácido e o grupo C=O de outro aminoácido (a partir do 4º).

- *Estrutura Terciária*: são as proteínas que possuem a forma esférico-elipsóidica, ou seja, comumentemente encontrada nas proteínas globulares (hormonas, anticorpos, enzimas, etc). São mantidas graças às interações do grupo R entre diversos aminoácidos, podendo facilmente ter sua forma alterada devido à desnaturação da proteína (temperatura, pH, concentração salina, entre outros).

- *Estrutura Quaternária*: é definida pelo grau de polimerização, visto que muitas proteínas são formadas por duas ou mais subunidades.

- *Estrutura Pentanária*: duas ou mais subunidades de estruturas quaternárias unidas, forma complexos multiprotéicos (como a cadeia respiratória).

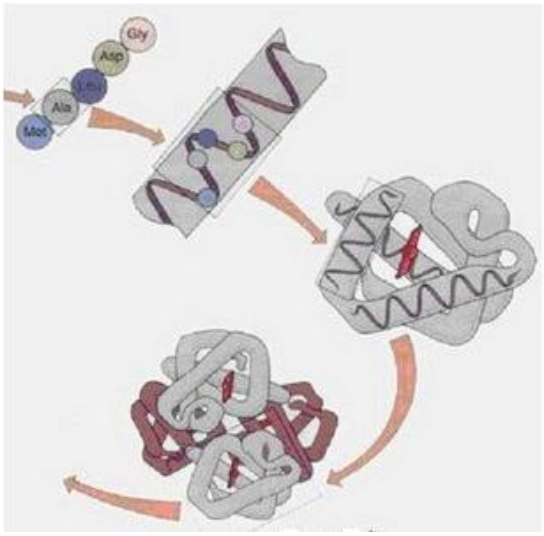


Figura 5. Estruturas de Proteínas (Fonte: Passarge, 2004).

2.2.4 Bancos de Dados Biológicos

Com o surgimento dos sequenciadores de DNA, o volume de sequências a serem armazenadas e analisadas cresceu muito, exigindo recursos computacionais cada vez mais eficientes.

Atualmente, a maior parte das informações sobre sequências de nucleotídeos e aminoácidos advêm dos projetos:

- *Genoma:* o DNA é segmentado e recursos computacionais são utilizados para montá-los e reconstituir a informação genética.

- *Transcriptoma:* neste projeto, o objetivo é reconhecer os genes expressos em diferentes tecidos e fases de desenvolvimento, identificando e caracterizando as proteínas codificadas pelo mRNA.

- *Proteoma:* a quantidade de proteínas resultantes da avaliação dos mRNAs transcritos nem sempre corresponde a quantidade de proteínas expressa na célula. Com isso não é possível relacionar essa proteína a uma função celular. Por isso, surgiu esse projeto, que trabalha com a análise das proteínas expressas, e que possibilita relacionar uma proteína a uma determinada função celular (Prosdocimi, 2002).

Essas informações estão organizadas em bancos de dados públicos, com acesso *on-line*, o que facilita a divulgação de novas descobertas. Esses bancos são classificados como bancos de estruturas primárias (armazenam sequências de nucleotídeos e proteínas) e estruturas secundárias (armazenam informações que derivam da análise e estudo dos dados armazenados nos bancos de estruturas primárias). Entre os principais bancos, pode-se citar:

- *Genbank:* foi criado nos Estados Unidos e armazena sequências de DNA (Quadro 1) e proteínas (Quadro 2). Acessado através do website http://www.ncbi.nlm.nih.gov.

Genome Mitochondrial COX 1 - Homo sapiens neanderthalensis
1 gatcacaggt ctatcaccct attaaccact cacgggagct ctccatgcat ttggtatttt
61 cgtctggggg gtgtgcacgc gatagcattg cgagacgctg gagccggagc accctatgtc
121 gcagtatctg tctttgattc ctgccccatt ccattattta tcgcacctac gttcaatatt
181 acaggcgagc atacttactg aagtgtgtta attaattaat gcttgtagga cataataata
241 acgactaaat gtctgcacag ctgctttcca cacagacatc ataacaaaaa atttccacca
301 aacccccccc cccccgcttc tggccacagc acttaaacac atctctgcca aaccccaaaa

Quadro 1. Sequência de genoma da COX1[3].

Cytochrome C Oxidase Subunit I [Homo sapiens neanderthalensis]

[3] Enzima atuante na dor de cabeça.

```
1   mfadrwlfst nhkdigtlyl lfgawagvlg talsllirae lgqpgnllgn dhiynvivta
61  hafvmiffmv mpimiggfgn wlvplmigap dmafprmnnm sfwllppsll lllasamvea
121 gagtgwtvyp plagnyshpg asvdltifsl hlagvssilg ainfittiin mkppamtqyq
181 tplfvwsvli tavllllslp vlaagitmll tdrnlnttff dpagggdpil yqhlfwffgh
241 pevyililpg fgmishivty ysgkkepfgy mgmvwammsi gflgfivwah hmftvgmdvd
301 trayftsatm iiaiptgvkv fswlatlhgs nmkwsaavlw algfiflftv ggltgivlan
```

Quadro 2. Sequência da proteína da COX1.

- *EBI (European Bioinformatics Institute):* criado na Europa, armazena sequencias de DNA. Acessado através do website http://www.wbi.ac.uk.

- *DDBJ (DNA Data Banck of Japan):* criado no Japão, armazena sequências de DNA. Acessado através do website http://www.ddbj.nig.ac.jp.

- *PDB (Protein Data Bank):* criado nos Estados Unidos, armazena estruturas tridimensionais resolvidas de proteínas. Acessado através do website http://www.rcsb.org/pdb.

- *GDB (Genome Data Bank):* é o banco de dados oficial do Projeto Genoma Humano. Acessado através do website http://www.gdb.org.

- *TIGR Databases:* armazena informações de genoma de vários organismos diferentes. Acessado através do website http://www.tigr.org/tdb.

- *PIR (Protein Information Resource):* é um banco de dados secundários de proteínas anotadas. Acessado através do website http://www-nbrf.georgetown.edu.

- *SWISS-PROT:* também é um banco de dados secundários, armazena sequências de proteínas e suas respectivas características moleculares, anotadas manualmente por uma equipe de especialistas. Acessado através do website http://www.expasy.ch/spro.

- *INTERPRO:* armazena os dados de famílias, domínios e assinaturas de proteínas. Acessado através do website http://www.ebi.ac.uk/interpro.

- *KEGG (Kyoto Encyclopedia of Genes and Genome):* é um banco de dados funcional, que disponibiliza links para mapas metabólicos de organismos com o genoma completa ou parcialmente seqüenciados, sendo um dos mais utilizados atualmente. Acessado através do website http://www.genome.ad.jp/kegg.

2.2.5 Visualização de proteínas

As primeiras imagens da estrutura das proteínas utilizadas foram em 2D, obtidas através de géis de eletroforese (Figura 6), criada em 1939 por A.Tiselius e A. E. Kabat com o objetivo de fracionar partículas de enzimas, proteínas, DNA e RNA, pois esses

compostos apresentam cargas elétricas definidas por pH específicos. Nessa metodologia não se vê a estrutura molecular das proteínas, mas sim as bandas que ficam no gel. O gel age como uma "malha" onde as proteínas ficam retidas de acordo com sua estrutura molecular (Waterman, 2000).

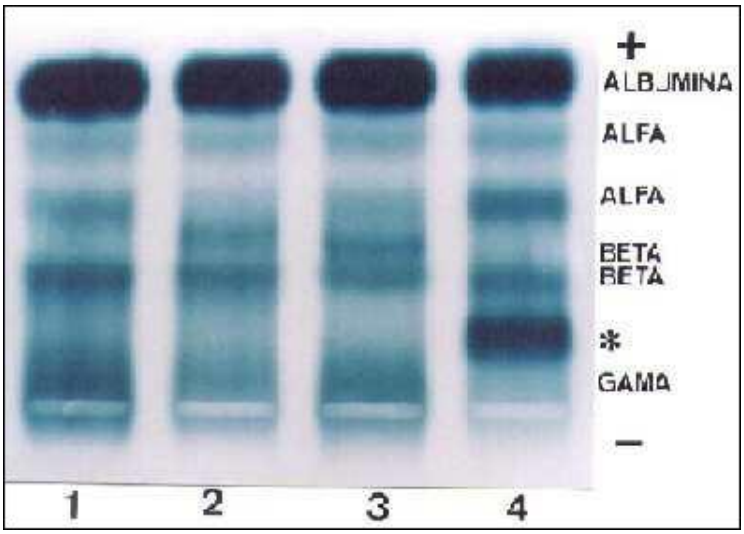


Figura 6. Eletroforese de proteínas séricas em gel de agarose, (1) hipergamaglubulinemia em processo inflamatório; (2) e (3): fracionamento normal; (4) gama monoclonal no mieloma múltiplo [4].

Outra técnica de visualização 2D das proteínas é a cristalografia (Figura 7), onde pode-se determinar a estrutura molecular da proteína. Esta é uma ferramenta necessária para a determinação da estrutura tridimensional da proteína. O uso de difração de raios-X aplicados a cristais de macromoléculas biológicas contribui significativamente, pois possibilita aos pesquisadores conhecer o arranjo estrutural das proteínas, bem como visualizar as interações entre proteínas (Oliveira, 2007).

[4] Fonte: www.ciencianews.com.br. Acessado em 28 de dez. 2008.

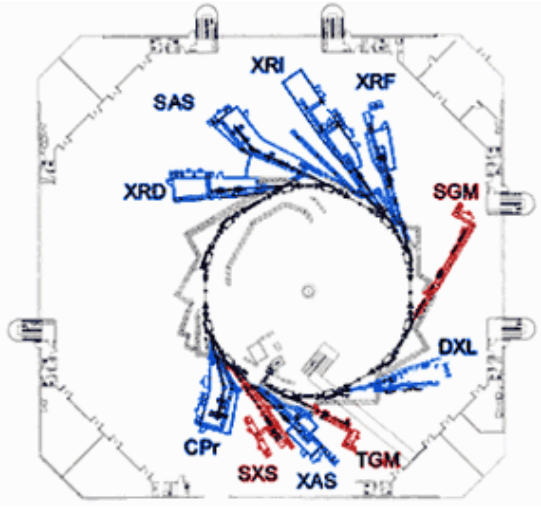


Figura 7. Cristalografia de Proteínas[5]**.**

A visualização computacional de proteínas, resultante da análise cristalográfica de raio-X tornou-se muito importante para diversas áreas da Biologia. Esta visualização consiste na análise de arquivos textos extraídos de bancos de dados biológicos que abrigam fichas técnicas e desenhos em 3D de milhares de proteínas.

Através desses arquivos é possível realizar a plotagem das ligações moleculares presentes nas proteínas, visualizar a interação entre elas e perceber que estas geram estruturas complexas, que são costumeiramente representadas sob a forma de redes por facilitar a análise e a representação (Waterman, 2000).

2.3 Biologia de Sistemas

Como dito anteriormente, Biologia de Sistemas é o termo usado para definir o estudo das interações entre os componentes de um sistema biológico, através de abordagens que envolvem teorias e modelos matemáticos e computacionais. Entre as ferramentas e técnicas utilizadas para descrever as interações de sistemas celulares estão as redes de livre escala.

Na matemática, o ramo que estuda os fundamentos das redes de integração é a Teoria dos Grafos. As redes de livre escala são as que melhor representam os processos biológicos (Barabasi, 2004; Oltvai, 2004). Paul Erdös e Alfréd Rényi (1960), inicialmente estabeleceram os conceitos matemáticos das redes de livre escala que têm sua distribuição de conectividade regida por uma lei de potência, $P(k) \sim k^{-y}$, onde o

[5] Fonte: www.lnls.br/infra/linhasluz/cpr.htm. Acessado em 05 de junho de 2009.

número de conectores *k* segue um padrão exponencial *y*. A rede livre de escala mais conhecida é a rede Barabási-Albert (BA).

As redes são formadas por nós e conectores, e podem descrever redes sociais, de informações, tecnológicas ou biológicas. Nas redes tecnológicas, por exemplo, os nós são representados pelos computadores e os conectores são os cabos. A Internet também é outro exemplo de redes tecnológicas (Figura 8), onde os computadores e roteadores são os nós e os cabos e fibras óticas são os conectores (Barabási, 2002). Assim, dentro do contexto das redes biológicas, nas redes de proteínas, os nós são representados pelas proteínas e os conectores pelas reações químicas (Figura 9).

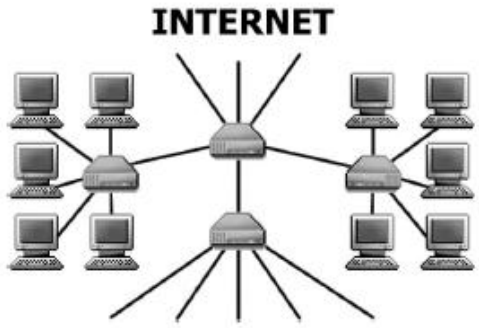


Figura 8. Representação de uma rede tecnológica. (Fonte: Albert, 2002; Barabasi, 2002).

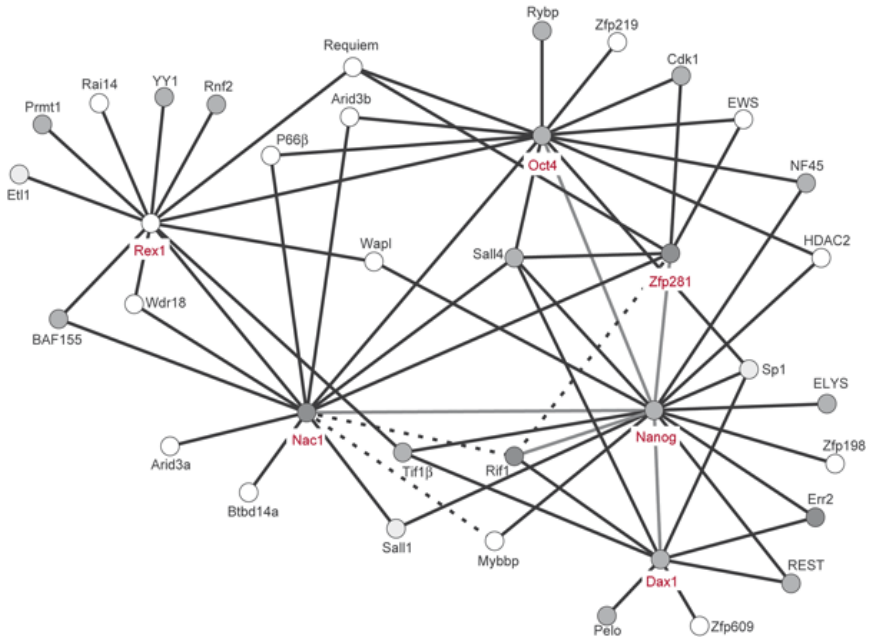


Figura 9. Representação de uma rede de interação de proteínas[6].

[6] Fonte: http://focus.hms.harvand.edu/2006/121506/research_briefs.shtml#Orkin

As interações protéicas se caracterizam pela topologia representada pelas redes de livre escala, devido a algumas características em comum: assim como a interação entre as proteínas, as redes de livre escala são altamente redundantes e pouco uniformes (Bebek, 2007), com alguns poucos nós que podem apresentar alta conectividade. Esses nós, conhecidos como *hubs* (representados na Figura 10 pelos pontos azuis), são importantes, pois representam pontos de controle da rede (Barabasi, 2004; Oltvai, 2004).

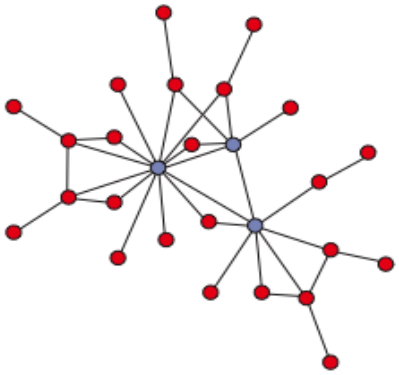


Figura 10. Proteínas com mais conexões são chamadas de proteínas hubs (Fonte: Barabasi, 2004; Oltvai, 2004).

Nas interações entre proteínas, uma comparação de genomas cruzados, de uma média evolucionária, levou a percepção de que proteínas mais "velhas" tem mais conexões em contrapartida a proteínas mais "jovens". Essa descoberta empírica demonstra uma preferência nas conexões (Barabasi, 2004; Oltvai, 2004), enfatizando que as proteínas com mais conexões podem desempenhar uma função mais importante na célula.

Um dos fatores centrais da importância da Biologia, principalmente a da genética, é sua relação com a Medicina. O estudo dessas redes de interações pode elucidar o comportamento celular de um organismo, visando auxiliar na descoberta de novas curas/tratamentos em diversas áreas da medicina (Cohen, 2004).

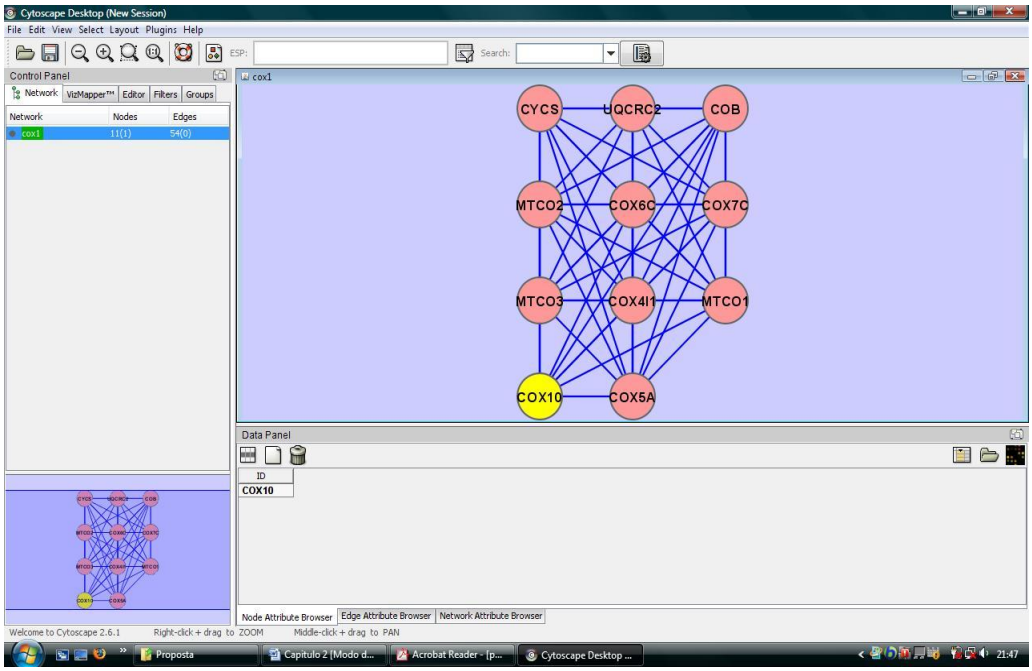


Figura 11. Ambiente de visualização do Cytoscape.

O software Cytoscape[7] (Figura 11) permite representar, através de um grafo, as informações referentes à rede de interação das proteínas, com nodos e ligações e nomenclaturas. Na figura 12 uma amostra de visualização de uma rede de interação de proteínas que pode ser obtida através do Cytoscape.

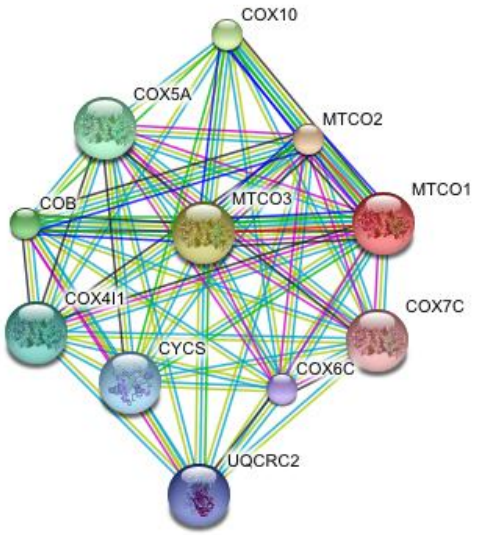


Figura 12. Rede de interação da proteína COX1 obtida no String[8].

[7] Website: http://www.cytoscape.org/

[8] Repositório de proteínas, disponível no website: http://string-db.org/

2.4 Considerações Finais

O estudo da Biologia Molecular, principalmente no campo da genética, cresce a cada ano, com novas descobertas e pesquisas de dados importantes para a manutenção da vida de organismos celulares. Nesse contexto, surgiu a necessidade de ferramentas que possam auxiliar cientistas e pesquisadores a manipular e trabalhar essas informações, de maneira rápida e preferencialmente fácil.

A informática, e as áreas que abrange e atua, está entre as ferramentas que mais auxiliam a Biologia Molecular atualmente. Usada não apenas na coleta e armazenagem de dados, a informática atua também na geração de imagens que representam estruturas microscópicas, como por exemplo, as proteínas e suas interações. A Geometria Computacional, vista no próximo capítulo, é a área da informática que trabalha a geração dessas imagens, utilizando algoritmos, teorias e modelos matemáticos.

3. GEOMETRIA COMPUTACIONAL

Geometria Computacional é definida como o estudo sistemático de algoritmos eficientes para problemas geométricos. A origem do termo é incerta, mas é atribuída a Marvin Minsky em 1969, no livro *Perceptron* e é utilizada para denotar algoritmos de modelagem de sólidos. Os dados de entrada do algoritmo são coleções de objetos e a saída geralmente são estruturas de dados geométricos. A ênfase de sua utilização é em problemas de matemática discreta, que trabalha com conjuntos de objetos e grafos (Esperança, 2002; Cavalcanti, 2002). Um problema geométrico é caracterizado pelos seguintes fatores (Figueiredo, 2005; Carvalho, 2005):

- *Dados e solução:* os dados de um problema geométrico são representados por um conjunto finito de objetos geométricos (pontos, retas, círculos, etc). A solução procurada pode ser obtida através do cálculo de números relacionados aos dados de entrada (por exemplo: a área de um polígono), ou através da construção de um novo objeto geométrico a partir dos dados iniciais (por exemplo: a interseção de dois polígonos) ou então através da verificação de propriedades do dado de entrada (por exemplo: verificar se um polígono é convexo).

- *Números reais:* objetos geométricos são definidos por números reais através de coordenadas ou equações.

- *Contínuos e discretos*[9]*:* essas duas características conferem ao objeto geométrico a possibilidade de serem representados por um computador. Por exemplo: um polígono é uma região do plano, possui um número infinito de pontos, mas seus limites podem ser representados, que por sua vez são representados pela sequência dos vértices.

- *Aspectos geométricos e topológicos:* São encontrados na solução dos problemas geométricos. Aspecto geométrico pode ser a forma ou localização de um

[9] Contínuo e Discreto: Contínuo e discreto são termos que se referem respectivamente a duas das ações básicas na elaboração da Matemática: *contar* e *medir*. A sucessão dos números naturais 1,2,3,... é a representação matemática para o discreto, enquanto que o correspondente para o contínuo, na matemática é encontrado na reta real R.(Brolezzi, 1996).

objeto. Aspecto topológico é a relação de adjacência e incidência[10] entre os objetos. A resolução do aspecto geométrico determina a solução do aspecto topológico. Solucionar a parte geométrica de forma simples, eficiente e robusta, é uma das chaves principais a resolução computacional dos problemas geométricos. O primeiro problema surgido com essa característica é o das pontes de Königsberg, (Figura 13) proposto por Euler, que consiste na existência de sete pontes, na cidade de Königsberg (Rússia), para interligar duas ilhas às margens do rio Preguel (Carvalho, 2005).

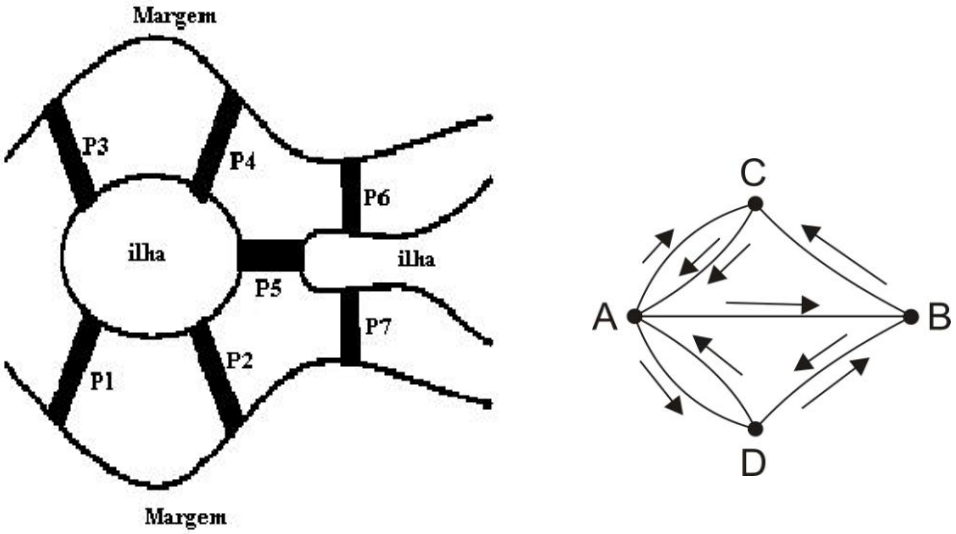


Figura 13. Problema das pontes de Königsberg e grafo representativo do problema (Fonte: Carvalho, 2005).

Quanto ao resultado desejado, os problemas geométricos podem ser classificados em quatro tipos (Figueiredo, 2005; Carvalho, 2005):

- *Seletivos:* o objetivo é selecionar um subconjunto a partir dos dados de entrada. Exemplo: triangulação (subdivisão de um plano de pontos em simplexos[11], como por exemplo, o Diagrama de Delaunay) e árvore geradora mínima.

- *Construtivos:* o objetivo é construir novos objetos geométricos a partir dos dados de entrada. Exemplo: interseção de polígonos, mosaico (também chamado de diagrama) de Voronoi (Figura 14) e geração de malhas.

[10] Adjacência e Incidência: Adjacência é o nome dado aos ângulos consecutivos cuja união constitui um semiplano. Incidência é o nome dado ao ângulo entre pontos e vértices. Ambos utilizados em Grafos – Matriz de Adjacência e Matriz de Incidência (Carvalho, 2005).

[11] Simplexos são extensões de triângulos em outras dimensões, como segmentos de reta, tetraedros, etc.

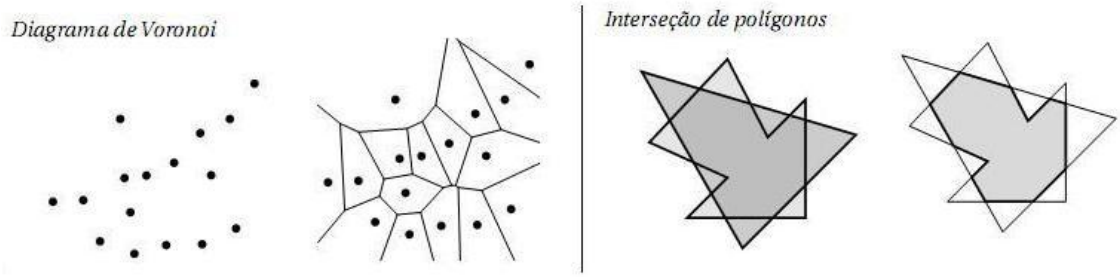


Figura 14. Exemplos de problemas geométricos (Fonte: Figueiredo, 2005; Carvalho, 2005).

- *Decisão:* o objetivo é responder sim ou não a pergunta que acompanha os dados de entrada. Exemplo: "Dado um polígono e um ponto, o ponto está fora do polígono?" ou "Dentre um conjunto de segmento de retas, há algum par que se cruza?"

- *Consulta:* o objetivo é pré-processar um conjunto fixo de objetos geométricos, de modo a responder de maneira mais eficiente as repetidas consultas sobre este conjunto. Exemplo: Dado um conjunto de pontos, a consulta é encontrar qual desses pontos está mais próximo de um determinado ponto do plano.

Dentre suas aplicações, os problemas tratados pela geometria computacional podem ser utilizados nas seguintes áreas (Figueiredo, 2005; Carvalho, 2005):

- *Computação Gráfica:* dentre os diversos problemas nessa área, um deles é detectar colisões entre objetos ao fazer uma animação, ou desenhar objetos geométricos de forma realista em 3D.

- *Robótica:* nessa área os problemas são, em sua maioria, voltados para a movimentação dos robôs, na identificação da área e qual o melhor caminho entre dois pontos e para o uso de esteiras mecânicas.

- *Sistemas de informações geométricas:* o grande volume de informações desse tipo de sistema precisa ser analisado para saber quais os objetos geométricos são próximos. Essas informações são utilizadas para verificar, por exemplo, quais cidades podem ser atingidas por um rio com risco de transbordar.

- *Circuitos integrados:* Como nesses circuitos os seus componentes eletrônicos não podem se sobrepor para evitar curto-circuitos durante seu projeto, é necessário identificar sobreposições de forma eficiente.

- *Bancos de dados:* Consultas a bancos de dados (*query* em SGDBs) que envolvam muitos campos são consideradas consultas multi-dimensional. Por exemplo, se um SGBD de uma companhia armazena, para cada funcionário, o peso, a altura, a

idade e o salário, então cada funcionário é representado por um ponto em **R**4[12] e uma consulta ao SGBD é uma busca em **R**4.

- *Sistemas de Informação Geográfica (SIG):* consiste na aplicação para sistemas que realizam o tratamento computacional de dados geográficos e recuperam informações não apenas com base em suas características alfanuméricas, mas também através de sua localização espacial. A geometria e os atributos dos dados num SIG devem estar *georeferenciados*, isto é, localizados na superfície terrestre e representados numa projeção cartográfica. Possui uma ampla gama de aplicações, que inclui temas como agricultura, florestas, cartografia, cadastro urbano e redes de concessionárias (água, energia e telefonia) (Davis, 2001; Câmara, 2001).

Nesse capítulo são abordados os conceitos de geometria computacional, complexidade de algoritmos e sobre o Mosaico de Voronoi. Em quais áreas esses conceitos podem ser aplicados, e, quanto ao Mosaico de Voronoi, quais etapas devem ser desenvolvidas para sua aplicação.

3.1 Complexidade de Algoritmos[13]

A complexidade de tempo de um algoritmo, ou simplesmente complexidade, é a quantidade de trabalho executada pelo algoritmo. Para fazer esse cálculo, é selecionada uma operação do algoritmo, chamada de operação fundamental. A complexidade é baseada na contagem do número de operações fundamentais. Podem ser escolhidas mais de uma operação fundamental com pesos diferentes. Ex: comparações ou trocas em um algoritmo de classificação, produtos, somas, etc.

Normalmente a complexidade é calculada em função do tamanho da entrada. Por exemplo, chamando E_n o conjunto de entradas possíveis de tamanho n, tomando $e \in E_n$ e chamando $c(e)$ o número de execuções da operação fundamental, tem-se, para a entrada e a fórmula $CPC(n) = max_{e \in En}\, c(e)$, que é a complexidade no pior caso. Outras medidas utilizadas são a complexidade no caso médio e a complexidade no melhor caso.

[12] Um sistema de coordenadas para $\mathbf{R}^n$ é definido por um ponto (origem) e n vetores. (Esperança, 2002; Cavalcanti, 2002).

[13] Fonte bibliográfica utilizada em toda a seção 3.1: (Rezende, 1994; Stolfi, 1994).

Para compreender o calculo da complexidade, é preciso compreender a representação assintótica e a notação *O*. Na representação assintótica, a função exata do cálculo da complexidade não interessa, pois esta depende do de outros fatores, entre eles o compilador e o hardware. O objetivo é calcular como o tempo t_n de um programa manipulando uma entrada de tamanho *n*, cresce com essa a entrada, para descobrir se é possível rodar o programa para uma entrada *n* muito grande.

Na Notação *O*, depois de definida uma função para representar a menor complexidade, essa função *f(n)* é aplicada na fórmula: $T_n = O(f(n))$ *se e somente se* $\exists$ *constantes positivas C e* n_0 *tais que* $| t_n | \leq C |f(n)| \ \forall n \geq n_0$, ou seja, a função *f(n)* é um limite superior para t_n. Com isso busca-se obter algoritmos utilizando essa fórmula, onde *f(n)* seja a "menor" função possível, para que o tempo t_n não cresça rapidamente com a entrada *n*.

Entre algumas funções calculadas na notação *O*, pode-se citar (em ordem crescente da "melhor para a "pior" função) as seguintes:

- *O(1):* complexidade constante, ou seja, o número de operações não muda, independente do tamanho da entrada;

- *O(log n):* complexidade logarítmica, por exemplo, algoritmo de busca binária;

- *O(n):* complexidade linear, por exemplo, algoritmo de pesquisa em uma lista;

- *O(nlog n):* complexidade N log N, por exemplo, algoritmo QuickSort e MergeSort;

- $O(n^2)$*:* complexidade quadrática, por exemplo, algoritmo de soma de matrizes e Bubblesort;

- $O(n^3)$*:* complexidade cúbica, por exemplo, algoritmo de soma de elementos entre vetor de índices e matriz;

- $O(2^n)$ *e* $O(3^n)$*:* complexidade exponencial, por exemplo, algoritmos de "força bruta".

As seis primeiras notações representam algoritmos polinomiais[14]. As duas últimas são algoritmos exponenciais[15]. Normalmente, consideram-se algoritmos

[14] Algoritmos Polinomiais: um algoritmo A é dito "polinomial" se existe algum polinômio (por exemplo: $f(x) = x^2 + 1$) p tal que A sempre termina depois de no máximo p(n) operações elementares, para quaisquer dados com tamanho total n.

[15] Algoritmos Exponenciais: Algoritmo com complexidade exponencial, não é executável para valores de entrada muito grandes, pois as funções exponenciais (por exemplo: $f(x) = 2^x$) crescem muito rapidamente. Por isso, problemas que somente podem ser resolvidos através de algoritmos exponenciais são ditos intratáveis.

polinomiais como tratáveis em tempo razoável e algoritmos exponenciais como intratáveis.

3.2 Algoritmos Clássicos da Geometria Computacional

Entre alguns dos algoritmos clássicos da geometria computacional, pode-se citar:

- *Fecho Convexo (Figura 15)*: Convexidade é uma propriedade geométrica bastante importante. Um conjunto de pontos é convexo se para cada par de pontos no conjunto, o segmento de reta entre eles está inteiramente contido no conjunto. O Problema do Fecho Convexo consiste em: dados *n* pontos, encontrar o invólucro convexo desses pontos. Uma aplicação prática deste problema se encontra em robótica. Se o fecho convexo de um robô não colide com obstáculos então o robô também não colide. O fecho convexo pode ser construído em tempo *O(nlog n)* por um algoritmo de divisão – e – conquista (Figueiredo, 2005; Carvalho, 2005).

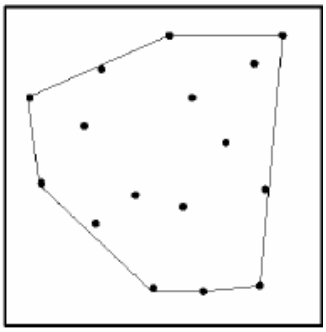


Figura 15. Problema do Fecho Convexo.

- *Par mais próximo (Figura 16):* Dados *n* pontos deseja-se encontrar dois pontos cuja distância entre eles é mínima. Uma aplicação prática deste problema está no controle de tráfego aéreo: dois aviões que estão em maior perigo de colisão são aqueles que estão mais próximos. Este problema pode ser resolvido com algoritmos num tempo $O(dn^2)$, onde *d* é a dimensão do espaço. De forma mais eficiente, este problema, pode ser resolvido por um algoritmo do tipo divisão - e - conquista em tempo *O(dn log n)* (Rezende, 1994; Stolfi, 1994).

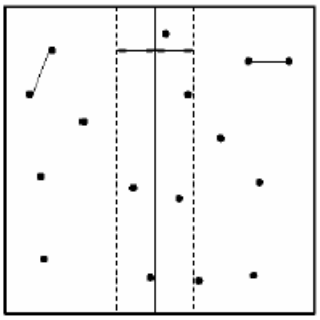


Figura 16. Problema do par mais próximo.

- *Triangulação de Polígonos*[16] *(Figura 17):* O objetivo é particionar um domínio complexo em uma coleção de objetos simples (Esperança, 2002; Cavalcanti, 2002). A região mais simples na qual pode-se decompor um objeto planar é um triângulo. Por exemplo: dado um polígono *P*, quer-se adicionar o maior número possível de diagonais (que não se cruzem) em *P*, de tal forma que o interior do polígono fique particionado em triângulos. Esse problema pode ser resolvido com um algoritmo linear *O(n)*. Esse algoritmo pode ser utilizado em problemas do tipo *Art Gallery*: imagine que as salas de galeria de arte formem um polígono. A questão é qual o menor número de guardas que são necessários para tomarem conta das salas, considerando que cada guarda fique parado em um local da galeria.

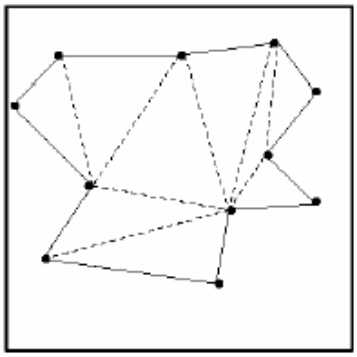


Figura 17. Problema da Triangulação de Polígonos.

- *Intersecções (Figura 18):* Um dos problemas geométricos mais básicos é o de determinar quando dois objetos se intersectam (cruzam). Esse problema é frequentemente reduzido ao problema de determinar quais pares de entidades primitivas (segmentos de retas) se cruzam e pode ser resolvido por um algoritmo de ordenação com complexidade *O(nlog n)* (Esperança, 2002; Cavalcanti, 2002).

[16] Polígonos: figura geométrica plana, limitada por retas. Exemplos: hexágono, triângulo, quadrado, entre outros. São considerados polígonos somente figuras com mais de três lados.

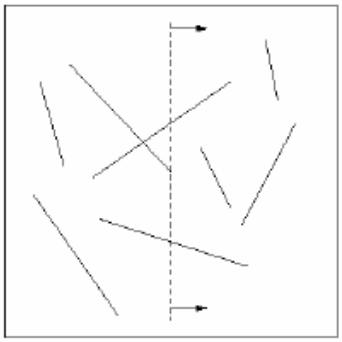


Figura 18. Problema das Intersecções.

- *Mosaicos de Voronoi:* Dado um conjunto *S* de *n* pontos no plano quer-se determinar para cada ponto *p* em *S* qual e a região *V(p)* dos pontos do plano que estão mais perto de *p* do que de qualquer outro ponto em *S*. As *n* regiões *V(p)* formam uma partição do plano chamada de Mosaico de Voronoi. Por exemplo: imagine uma vasta floresta contendo vários pontos de observação de incêndio. O conjunto das árvores que estão mais próximas de um determinado posto *p* determina a região *V(p)* das árvores que são de responsabilidade do ponto observação *p*. O Mosaico de Voronoi de um conjunto de *n* pontos pode ser construído em *O(nlog n)* por um algoritmo do tipo divisão – e – conquista. O Mosaico de Voronoi será mais detalhado na seção 3.3.

- *Triangulação de Delaunay (Figura 19):* O dual geométrico (utilizando retas) de um Mosaico de Voronoi para um conjunto *S* de pontos forma uma triangulação ao conjunto S, chamada de triangulação de Delaunay. A triangulação de Delaunay tem várias propriedades geométricas interessantes, por exemplo, contem todas as "árvores geradoras mínimas" [17] de *S* (Rezende, 1994; Stolfi, 1994).

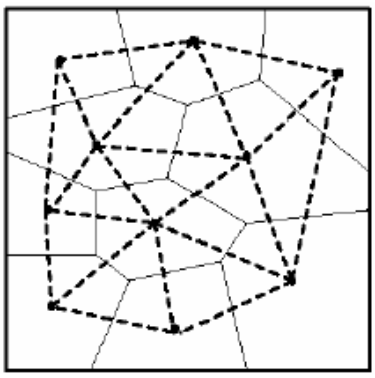


Figura 19. Problema de Triangulação de Delaunay.

[17] Árvores geradoras mínimas: Uma árvore geradora mínima é a árvore geradora de um grafo conexo *G(V, E)* que possui peso mínimo dentre todas as árvore geradoras (sub-grafos que possuem os mesmo vértices do grafo original) de *G*.

3.2.1 Etapas para trabalhar com Geometria Computacional

Para desenvolver os algoritmos utilizados na Geometria Computacional, é necessário definir o modelo computacional, fazer a prova de correção e fazer a análise de desempenho. Para executar essas etapas, é necessário tomar alguns cuidados (Rezende, 1994; Stolfi, 1994):

- *Modelo Computacional*: é a definição de quais operações serão executadas e qual o custo de cada uma (Figura 20). Assim, é possível calcular o grau de complexidade do algoritmo. A escolha do modelo computacional é importante, pois conforme o modelo selecionado, é possível definir quais algoritmos podem ser utilizados, quais os custos de suas operações básicas e inferir suas propriedades matemáticas.

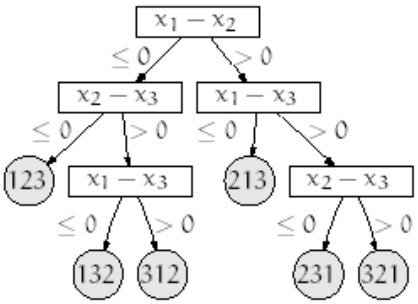


Figura 20. Modelo Computacional. Arvore de decisões algébricas (Fonte: Figueiredo, 2005; Carvalho, 2005).

- *Prova de correção:* o algoritmo deve satisfazer algumas condições para ser confiável, tais como: *i)* as operações executadas pelo algoritmo são válidas e definidas não importa os dados de entrada. *ii)* o algoritmo sempre termina após um número finito de operações. *iii)* o resultado obtido pelo algoritmo satisfaz as condições do enunciado. A prova de correção garante que o algoritmo retorne o resultado correto para cada entrada de dados fornecida.

- *Análise de desempenho:* devido à existência de mais de um algoritmo para resolver um determinado problema, é necessário escolher qual o melhor a ser utilizado. Essa escolha é baseada em algumas propriedades do algoritmo, como por exemplo, eficiência, robustez e usabilidade, entre outros. A propriedade mais utilizada, pela facilidade de quantificar, é o custo de um algoritmo. Essa medida, chamada também de

complexidade[18] ou eficiência, envolve o tempo de processamento e a memória alocada para a execução do algoritmo, e é geralmente a propriedade que define qual algoritmo é melhor.

Dentre os principais objetivos da análise de desempenho de algoritmos está a comparação entre algoritmos para um mesmo problema, possibilitando escolher o mais eficiente. Mas nem sempre isso é possível, pois a eficiência pode mudar conforme as instâncias que são aplicadas. Quando isso ocorre, a solução para escolher qual o algoritmo mais eficiente é calcular a complexidade do algoritmo. (Figueiredo, 2005; Carvalho, 2005).

Os algoritmos eficientes para a resolução de problemas geométricos são criados a partir da combinação de teoremas matemáticos relacionados à geometria do problema, técnicas algorítmicas, estruturas de dados adequadas, uma boa análise de desempenho e primitivas geométricas. Exemplo: Dados três pontos no plano, A, B e C, decidir se o ponto C está À esquerda do segmento orientado AB (Figura 21). Esse predicado pode ser resolvido considerando o sinal do produto vetorial AB x AC e, portanto, pode ser implementado através da avaliação de um determinante.

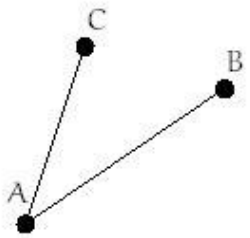


Figura 21. Exemplo de primitivas geométricas. (Fonte: Figueiredo, 2005; Carvalho, 2005).

Algumas situações são impossíveis de serem solucionadas com a ajuda da geometria computacional. Essas situações são chamadas de limitações da geometria computacional e se caracterizam por dados discretos (que incluem aproximações de fenômenos contínuos), objetos geométricos planos e dimensionalidade (problemas n-dimensionais são pouco abordados). Dentre as situações que podem ser trabalhadas estão os problemas de proximidade, que entre outros, compreende o mosaico ou algoritmo de Voronoi, que será explicado na próxima seção (Esperança, 2002; Cavalcanti, 2002).

[18] Ver seção 3.1 – Complexidade de Algoritmos

3.3 Mosaico de Voronoi

Em 1908, o matemático russo Georgi Voronoi desenvolveu um mosaico para representar a decomposição de um espaço métrico em regiões de acordo com a distância entre determinados pontos. Dado um conjunto de pontos num plano, o mosaico de Voronoi divide o plano conforme a regra do vizinho mais próximo (procura um ponto ou vértice, vizinho ao que está sendo analisado, que seja o mais próximo possível). Cada ponto é associado a uma região exclusiva (Aurenhammer, 1991).

A definição formal é: dado um conjunto *S* de *n* pontos no plano, o objetivo é determinar para cada ponto *p* de S qual a região *V(p)* dos pontos do plano que estão mais próximos de *p* do que qualquer outro ponto de *S*. Por exemplo, no problema do correio (Figura 22), o conjunto *S* seria os postos do correio, o plano seria a cidade e as regiões *V(p)* seria o conjunto de casas da cidade que seria atendidas pelo posto *p*. As regiões determinadas para cada ponto formam a partição do plano que é chamada de Mosaico de Voronoi (Figura 23) (Rezende, 1994; Stolfi, 1994).

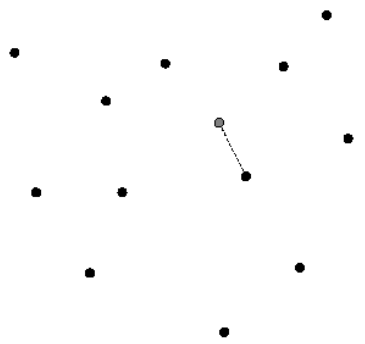


Figura 22. Clássico problema do correio (Fonte: Rezende, 1994; Stolfi, 1994).

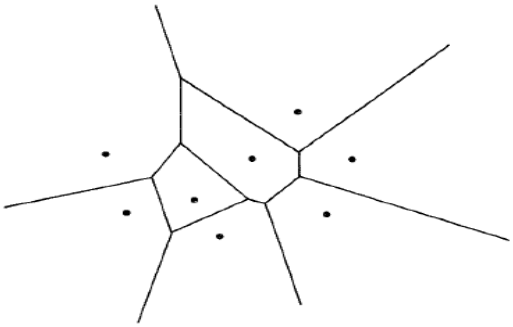


Figura 23. Mosaico de Voronoi de oito células (Fonte: Aurenhammer, 1991).

Existem três razões que transformaram o Mosaico de Voronoi numa ferramenta muito requisitada: i) A primeira é que o Mosaico de Voronoi surge na natureza em diversas situações (por exemplo: uma colméia ou as estruturas moleculares, que podem ser representadas por ele). ii) A segunda é que a estrutura do Mosaico de Voronoi auxilia no cálculo de objetos matemáticos, visto que pode ser relacionado a várias formas geométricas (por exemplo: poliedros, retas, pontos). iii) E a terceira é que o Mosaico de Voronoi provou ser uma ferramenta poderosa na resolução de problemas computacionais (análise de Cluster e detecção de colisões – robótica, entre outros) e isso tem atraído a atenção de cientistas da área computacional nas últimas décadas (Aurenhammer, 1991).

Para construir um Mosaico de Voronoi para *n* pontos num plano, a complexidade calculada para o pior caso, usando modelos computacionais algébricos, é de $\Omega(n \log n)$. Sua construção varia conforme o número de dimensões onde será aplicado. Para a construção em ambientes 2D, o mosaico geralmente é construído em três etapas (Preparata, 1988; Shamos, 1988):

- *1º etapa (Figura 24)*: Divisão do plano de pontos *S* em dois sub-planos S_1 e S_2, de tamanhos aproximados. O conjunto *S* será divido com relação a uma reta semi-vertical: o conjunto de pontos S_1 será formado pelos pontos de *S* que têm a coordenada y menor ou igual à uma mediana pré-estabelecida (geralmente o raio do plano) das coordenadas y dos pontos e o conjunto S_2 será formado pelos demais pontos de *S* (conforme algoritmo abaixo).

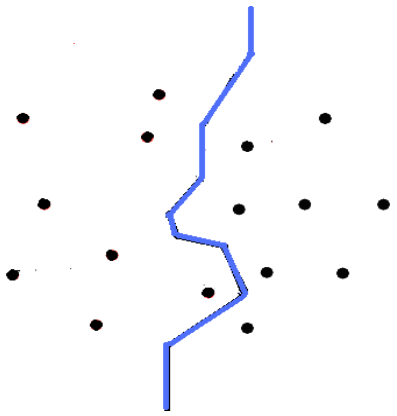


Figura 24. Divisão do plano em dois sub-planos (Fonte: Preparata, 1988; Shamos, 1988).

Um possível algoritmo seria:

```
Procedure Etapa1
    float mediano;                                          S1[j].y := S[i].y;
   //tamanho do conjunto S                                  j := j + 1;
    int i, tamS;                                        }
    int j, l;  //indices arrays S1 e S2                 Else
    array S, S1, S2;                                        S2[l].x := S[i].x;
    mediano := METADE(S);                                   S2[lj].y := S[i].y;
    tamS := length(S);                                      l := l + 1;
    i := 0;  j := 0;  l := 0;                           }
    while (i < (tamS-1))                                    i := i + 1;
        If (S[i].y <= mediano) then                 End  //while
                S1[j].x := S[i].x;                  End
```

- *2º etapa (Figura 25)*: Construção das regiões para os sub-planos S_1 (Figura 23a) e S_2 (Figura 23b) de forma recursiva (ver algoritmo de divisão e conquista). Esta etapa corresponde às chamadas recursivas onde cada sub-plano será resolvido independentemente. A base da recursão é atingida quando o conjunto de pontos em questão tem no máximo k_0 pontos, onde k_0 é uma constante.

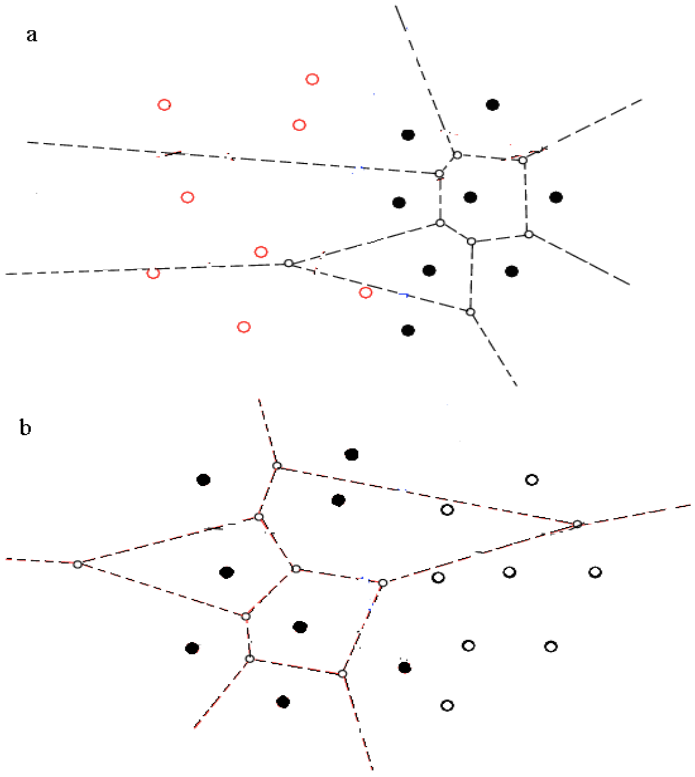


Figura 25. Construção das regiões para o sub-plano S_1 (a) e sub-plano S_2(b) (Fonte: Preparata, 1988; Shamos, 1988).

Um algoritmo do tipo de divisão e conquista pode ser representado por:

```
Procedure Etapa2(S) {                       if (| S | = 2) then {
```

```
        // suponha S={a,b}
        if (a>b) return (a,b)
        else return (b,a)
      }
    else  {
            divide  S  em  S1  e  S2;
//etapa 1
                                        (max1, min1) = Etapa2(S1)
                                        (max2, min2) = Etapa2(S2)
                                         return          (MAX(max1,
                                max2), MIN(min1, min2))
                                        }
                                      End
```

- *3º etapa (Figura 26)*: Eliminar as arestas do sub-plano S_2 que estão a esquerda da divisão do plano *S* e as arestas do sub-plano S_1 que estão a direita da divisão do plano *S*. Ao mesclar as regiões é gerado a região do plano *S* (ver algoritmo do merge). Nesta etapa é preciso escolher um bom algoritmo para que essa etapa possa ser executada em tempo *O(nlog n)* para todo o algoritmo. O trabalho desta etapa é encontrar duas retas tangentes a S_1 e S_2: uma tangenciando os fechos convexos abaixo (inferior) da reta e a outra tangenciando acima (superior). A partir destas tangentes é fácil construir o fecho convexo $(S_1\ S_2)$ em tempo linear. Denotando por *ui* e *vj* os vértices de S_1 e S_2, respectivamente, a tangente $T=ui\ vj$ é o segmento de reta que determina a tangente inferior. A tangente superior pode ser encontrada de maneira análoga.

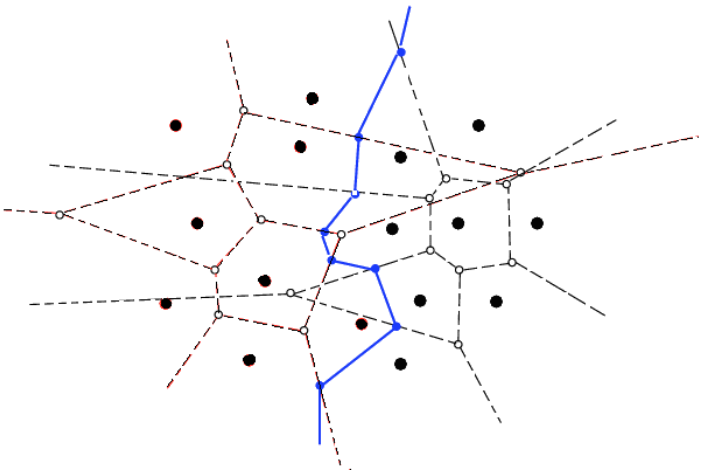


Figura 26. Sub-planos S_1e S_2 sobrepostos (Preparata, 1988; Shamos, 1988).

Um possível algoritmo para mesclar dois sub-conjuntos seria:

```
Procedure Etapa3 (S1 [], int ini, int fim,
int meio, S2 []) {

        int i = ini;
        int k = ini;
        int j = meio+1;
        while ((k <= meio) && (j <=
fim)) {

            S2[i++] = S1[k] <= S1[j]
? S1[k++] : S1[j++];
        }
        if (k > meio) {
```

```
            while (j <= fim) S2[i++] = S1[j++];
        }
        else {
            while (k <= meio) S2[i++] = S1[k++];
        }
        For (i=ini; i <= fim; i++) S1[i] = S2[i];
    }
```

3.4 Aplicações do Mosaico de Voronoi

No contexto das estruturas moleculares, o Mosaico de Voronoi começou a ser utilizado em 1974 por F. Richards, para avaliar o volume dos átomos na proteína globular. Nessa abordagem foram descobertas algumas falhas do Voronoi, quando utilizado para representar volumes atômicos. Mas para modelar a estrutura de proteínas não é necessário conhecer suas coordenadas atômicas, o conjunto de pontos utilizado está relacionado à forma da proteína e as suas interações (Poupon, 2004).

Além da aplicação na matemática e na representação de estruturas moleculares, o Mosaico de Voronoi é encontrado em diversas outras áreas[19], entre elas:

- *na natureza* (Figura 27a), no formato de colméias, nas manchas das girafas, em alguns corais e na lama seca.

- *em representações gráficas* (Figura 27b), na transformação de imagens, utilizando computação gráfica. Nessa aplicação, quanto maior o número de células do Mosaico de Voronoi aplicadas na imagem real, mais parecido será o resultado obtido na imagem computadorizada. Por exemplo, na Figura 27b, na representação da constelação estelar e da flor de lótus, se o número de células do Mosaico de Voronoi aplicadas nas imagens fosse, digamos, 900(novecentas) células, a imagem obtida seria muito mais próxima à real do que as das imagens da Figura 27b, que foram geradas por aproximadamente 100(cem) a 200(duzentas) células do Mosaico de Voronoi.

- *no corpo humano* (Figura 27c), na forma dos tecidos musculares esquelético (que vão revestir e formar ossos e músculos), cardíaco (que revestem e formam o coração) e liso (que revestem e formam diversos órgãos do corpo humano, principalmente a pele). Nos tecidos musculares, os Mosaicos de Voronoi são perceptíveis através dos cortes transversais dos tecidos, e uma curiosidade interessante, é que quanto mais células do Mosaico de Voronoi presentes no tecido, menos rígido é o

[19] Fonte: http:// www.cefala.org/~leoca/voronoi/72.html

tecido e mais lenta sua contração.

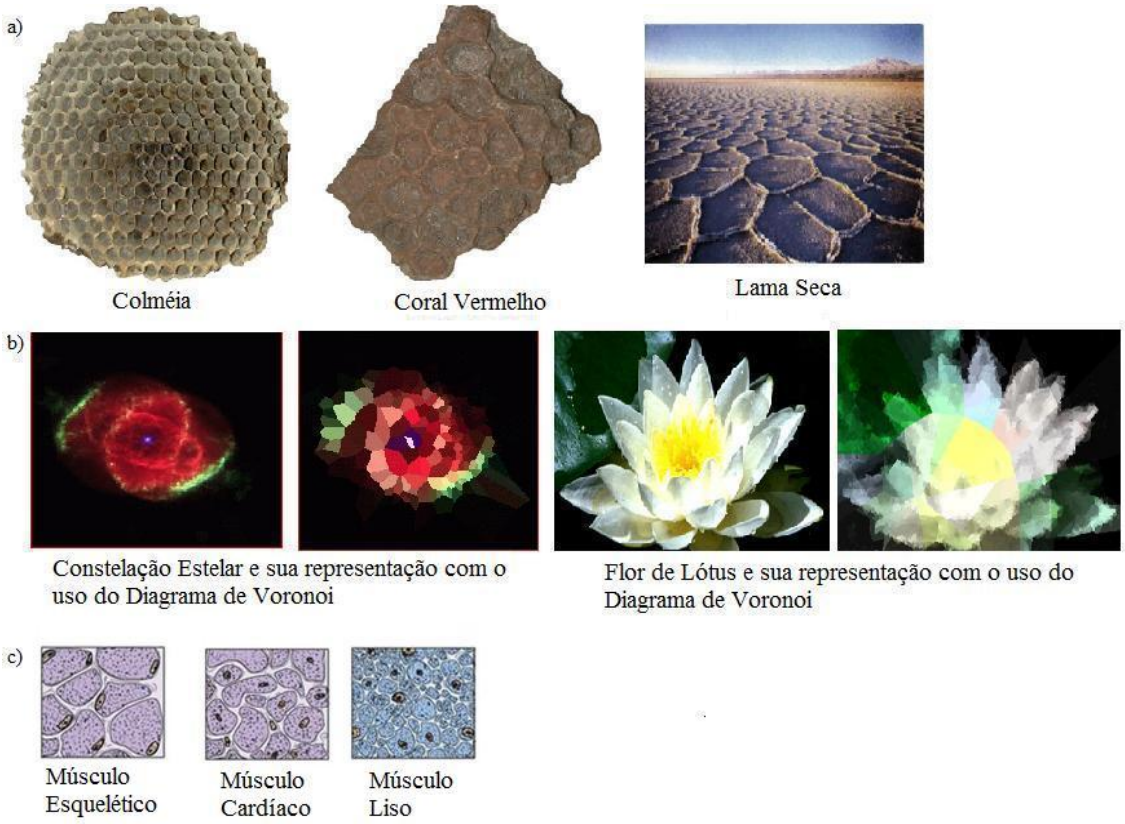


Figura 27. Exemplos de Voronoi em áreas adversas da matemática.

3.5 Considerações Finais

A Geometria Computacional utiliza algoritmos e modelos matemáticos para resolver e representar problemas em diversas áreas, como robótica, sistemas de informação geográficos e computação gráfica. Entre os modelos matemáticos utilizados está o Mosaico de Voronoi, usado na resolução e representação de problemas relacionados a redes, como a interação entre as proteínas.

O Mosaico de Voronoi possui diversas implementações, algumas inviáveis devido à uma complexidade muito grande. Uma dessas implementações será descrita no próximo capítulo.

4 IMPLEMENTAÇÃO

Neste capítulo será descrito a implementação de um *plugin*, utilizado no programa Cytoscape, para desenhar o Mosaico de Voronoi de redes de proteínas. Para tanto, nas seções seguintes serão apresentados os artefatos de software que compõem a modelagem do sistema e os softwares a serem utilizados.

4.1 Cytoscape[20]

O software Cytoscape, disponível no website http://www.cytoscape.org/, é um software de código fonte aberto, desenvolvido em 2002 em uma parceria entre o Instituto Nacional de Ciências Médicas Gerais dos EUA (NIGMS) e a Fundação Nacional de Ciências dos EUA (NSF). Com o objetivo original de atuar na busca de dados biológicos, hoje é uma plataforma global para a visualização e análise de complexas redes de interação moleculares.

Composto por funcionalidade básicas, dentre elas, mostrar a nomenclatura de cada proteína da rede, seus perfis de expressão gênicas e suas interações (Figura 11, Capítulo 2), o Cytoscape possui funcionalidades adicionais obtidas através da utilização de *plugins*, desenvolvidos com base na linguagem Java, utilizando a API do próprio Cytoscape. Entre os recursos disponíveis no software, pode-se citar:

- *Integração de dados:* o Cytoscape tem suporte a vários formatos de arquivos (SIF, XGMML, BioPAX, PSI-MI, MS Excel, entre outros), interoperabilidade, atua como *Web Service Clients* (atualmente suporta Pathway Commons, Intacta, BioMart, NCBI Entrez Gene e PICR) e *Session File* (todo o trabalho realizado com o software salvo em um único arquivo).

- *Visualização:* o Cytoscape permite personalizar a visualização de uma rede com o WizMapper, utilizando cores, espessuras e outros atributos configuráveis pelo

[20] Fonte bibliográfica utilizada em toda a seção 4.1: Cytoscape User Manual <http://www.cytoscape.org/manual/Cytoscape2_6Manual.pdf>.

usuário, aplicar diversos layouts a rede (por exemplo, disposto de forma cíclica ou em árvore), navegar pela rede com o uso do zoom e exportar imagens nos formatos PDF, EPS, SVG, PNG, JPEG e BMP.

- *Análise:* o Cytoscape permite a aplicação de filtros para a seleção de subconjuntos de nós e/ou interações, encontrar módulos e clusters para reconhecimento de expressões gênicas e pesquisar nodos e arestas com o uso de *queries*.

Com o arquivo dos dados de uma determinada proteína, obtidos nos bancos de dados proteômicos, o Cytoscape extrai os vértices que compõem a rede de interação. Esses vértices serão repassados para o *plugin* Voronoi, que irá processar esses dados e desenhar a rede com a forma do Mosaico de Voronoi.

4.2 Requisitos da Aplicação

Este trabalho tem por objetivo construir um *plugin* para ser utilizado no software Cytoscape. A função deste *plugin* é facilitar a visualização de uma rede de proteínas, mais especificamente das proteínas hubs[21], utilizando Mosaicos de Voronoi. O processo de visualização da rede de proteína através desse software está representado por um diagrama de caso de uso (Figura 28), explicado a seguir.

[21] Proteínas hubs: ver Capítulo 2, seção 2.3.

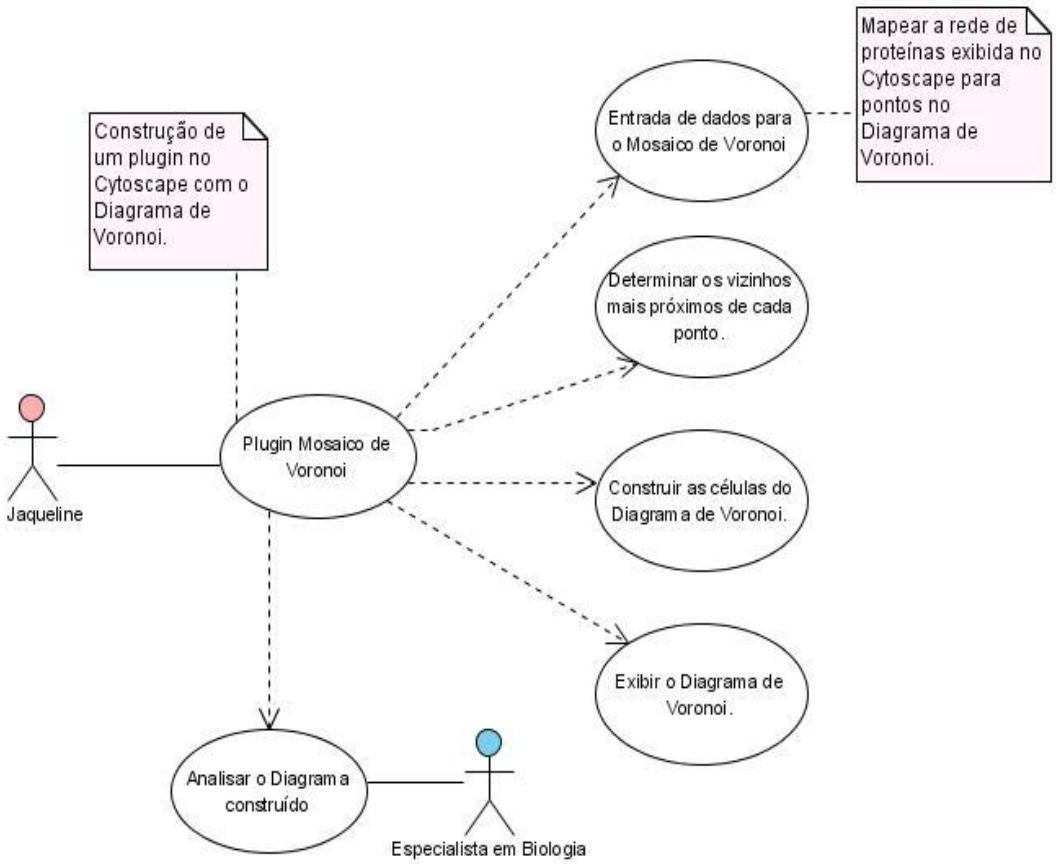


Figura 28. Requisitos da aplicação.

Para a construção do *plugin*, é necessário mapear a rede de proteínas. Para isso o software Cytoscape fornece as coordenadas *x* e *y* de cada proteína, correspondente a uma rede de interação, e cada coordenada corresponderá a um ponto no plano do Mosaico de Voronoi. Após determinar o conjunto desses pontos, que corresponde a 1º Etapa da construção de um Diagrama de Voronoi (Capítulo 3, seção 3.3), é necessário definir quais os pontos (vizinhos) mais próximos a um determinado ponto, ou seja, quais as proteínas que interagem entre si. Essa verificação corresponde a um dos processos da 2º Etapa da construção do Mosaico de Voronoi (Capítulo 3, seção 3.3).

Após definir o conjunto de pontos e seus respectivos vizinhos, é necessário calcular o bissetor entre um ponto e cada um de seus vizinhos. Os pontos bissetores encontrados serão utilizados para calcular os vértices das arestas que constroem as células do Voronoi. Esses cálculos também estão compreendidos na 2º Etapa da construção do Mosaico de Voronoi (Capítulo 3, seção 3.3). Quando todos os pontos do conjunto tiverem suas respectivas células desenhadas, o Mosaico de Voronoi está pronto para ser exibido na tela, correspondendo a 3º Etapa da construção do Mosaico de

Voronoi (Capítulo 3, seção 3.3). Na visualização é possível utilizar bibliotecas gráficas para manusear e filtrar a rede.

4.3 Modelo Conceitual

Na figura 29 tem-se a representação do diagrama UML de classes da aplicação descrito a seguir.

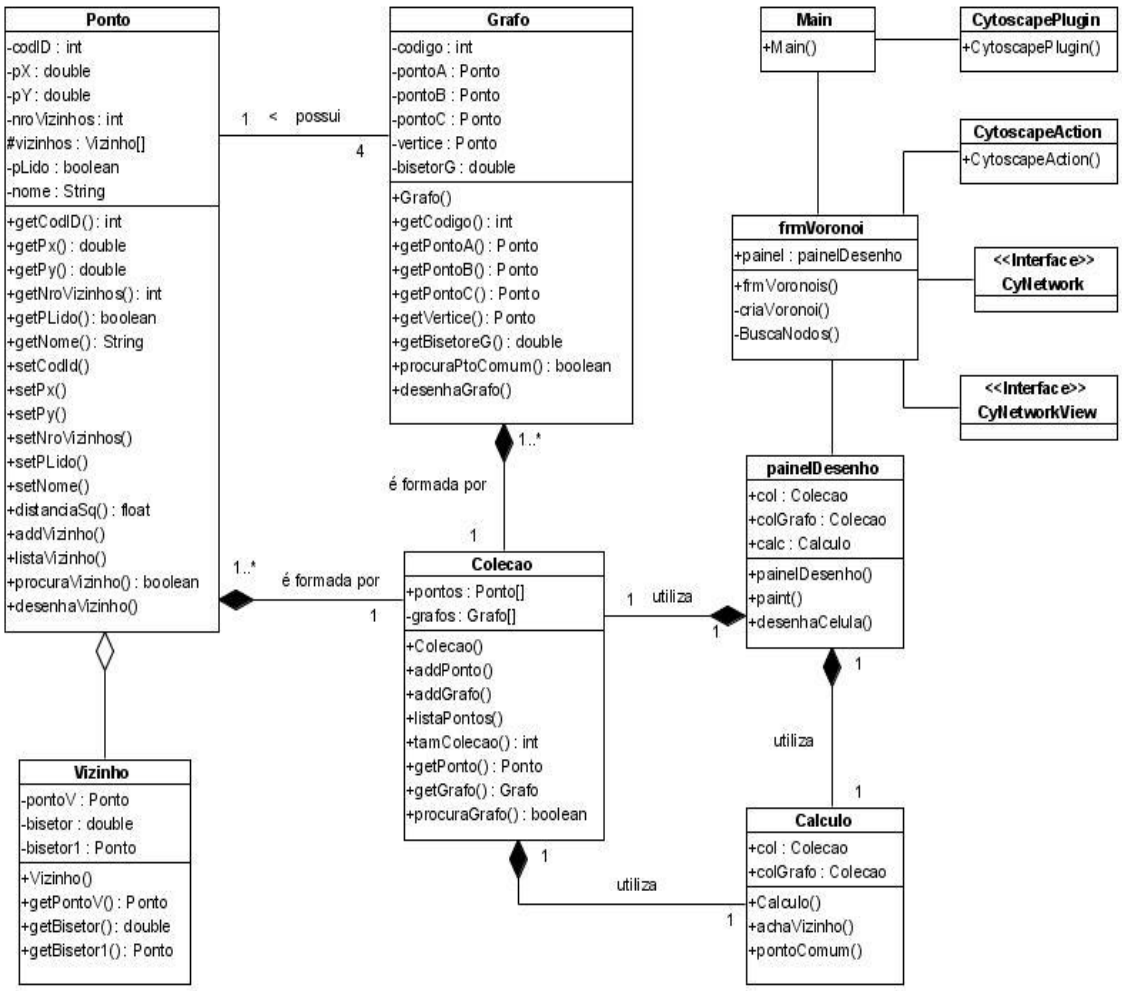


Figura 29. Diagrama de Classes da aplicação.

As principais classes da aplicação são:

- a classe *Colecao:* é a classe que abriga os vértices extraídos do Cytoscape em uma lista. Constituída basicamente pelo método de adicionar pontos, é a classe através da qual serão manipulados os pontos do Diagrama de Voronoi. Esta classe também abriga uma lista dos pontos vértices que formam o Mosaico de Voronoi.

- a classe *Ponto:* é a classe que armazena as informações de cada vértice da rede, tais como sua localização no plano, a lista e o número de pontos vizinhos. Seus principais métodos são:

- *distanciaSq()*: responsável por calcular a distancia entre dois pontos do plano, é utilizado na classe Calculo().

- *desenhaVizinho*: responsável por desenhar o bisetor entre o ponto e seus vizinhos. O bisetor será o vértice utilizado para desenhar o poliendro da célula de Voronoi.

Os métodos utilizados nesta classe correspondem a 2º Etapa da construção de um Diagrama de Voronoi (Capítulo 3, seção 3.3).

- a classe *Vizinho:* é a classe que irá abrigar para cada ponto quais outros pontos são sues "vizinhas" na rede. Composta basicamente por métodos get() e set().

- a classe *Grafo*: é a classe que armazena para cada três (entre o ponto da lista e dois vizinhos) um ponto comum. Esse ponto será o vértice para o desenho do Mosaico de Voronoi. Seus principais métodos são *procuraPtoComum(),* que busca se já existe uma combinação para os três pontos, e *desenhaGrafo().*

As classes auxiliares da aplicação são:

- a classe *painelDesenho:* é a classe responsável por desenhar o Mosaico de Voronoi. Tem os métodos *paint(),* responsável por desenhar os pontos e arestas do Mosaico de Voronoi e o método *desenhaCelula(),* responsável por desenhar as arestas do Mosaico de Voronoi. Esse método é utilizado na 3º etapa da construção de um Diagrama de Voronoi (Capítulo 3, seção 3.3).

- a classe *Calculo:* é a classe responsável pelos cálculos que envolvem a construção do Mosaico de Voronoi. Tem como métodos a função *achaVizinho(),* que calcula o valor da bissetriz entre dois pontos para verificar se são vizinhos. Esse método é utilizado na 2º etapa da construção de um Diagrama de Voronoi (Capítulo 3, seção 3.3), e a função *pontoComum(),* que calcula para cada três pontos, entre um ponto da lista e dois vizinhos, um ponto em comum para ser o vértice do desenho do Mosaico de Voronoi.

Para fazer a ligação deste aplicativo ao software Cytoscape é necessário estender algumas de suas classes, que são:

- a classe *CytoscapePlugin:* estendida pela classe *Main(),* é utilizada para definir um padrão na construção e no carregamento dos *plugins* utilizados no Cytoscape. O

construtor padrão, utilizado nos plugins do Cytoscape pode ser exemplificado pelo trecho de código mostrado na Figura 30.

```
package br.ucs.tcc;

import cytoscape.plugin.CytoscapePlugin;
import cytoscape.*;

public class Main extends CytoscapePlugin{

   public Main() throws Exception {
      frmVoronois frameV = new frmVoronois();

      frameV.setPreferredMenu("Plugins");

       Cytoscape.getDesktop().getCyMenus().addAction(frameV);
   }

   /**
    * Gives a description of this plugin.
    */
   @Override
   public String describe() {
      StringBuffer sb = new StringBuffer();
      sb.append("This is a test to create a plugin into Cytoscape. This application
creates a graph for fun");
      return sb.toString();
   }
}
```

Figura 30. Trecho de código do uso da classe CytoscapePlugin.

- a classe *CytoscapeAction*: estendida pela classe frmVoronoi(), é utilizada para verificar as ações entre o Cytoscape e o plugin, evitando por exemplo, que o plugin seja chamado quando não foi aberto nenhuma rede de integração.

- as interfaces *CyNetwork* e *CyNetworkView:*: utilizadas na classe frmVoronoi, são instanciadas para que as informações da rede de proteínas que está sendo visualizada no Cytoscape possam ser manipuladas pelo plugin.

4.4 Implementação

Como dito anteriormente, a linguagem de programação escolhida para a implementação do *plugin* do Diagrama de Voronoi foi o Java SE 6 (http://www.java.com/pt_BR/download/), por facilitar a integração com o software Cytoscape, cuja API (Aplication Programming Interface) é baseada nela. A opção de desenvolver em forma de *plugin* foi por facilitar o desenvolvimento e utilização, e pelo

fato de que a própria comunidade do Cytoscape estimula o desenvolvimento de *plugins*, visto que inúmeras de suas características adicionais, tais como redes e perfis de análises moleculares e novos layouts estão disponíveis nesse formato (http://chianti.ucsd.edu/cyto_web/plugins/index.php), muitos de forma gratuita.

Os *plugins* são pedaços de programa desenvolvidos para oferecer funcionalidades específicas de forma a acrescentar isso a uma aplicação. Esses mini-programas rodam embutidos, utilizando a interface de um programa principal e, normalmente, possuem limitações definidas de suas funcionalidades (Carvalho, 2005; Vera, 2005).

Para a implementação de algumas funcionalidades gráficas do *plugin*, tais como zoom, é possível utilizar a biblioteca JUNG[22] – Java Universal Network Graph (http://jung.sourceforge.net/), um framework que permite a visualização de grafos em um applet, que entre suas características, as que podem ser utilizadas nessa aplicação são: *i)* a função de zoom; *ii)* a possibilidade de mover toda a rede, apenas um vértice ou uma região selecionada, *iii)* a definição de espessura das arestas e *iv)* a aplicação de filtros.

Por fim a arquitetura do *plugin* pode ser representada pela Figura 31, onde através do aplicativo Cytoscape serão extraídos os vértices (coordenadas x e y) das proteínas. Esses vértices serão armazenados no *plugin*, sob a forma de uma lista de pontos, que será percorrida para aplicar os conceitos do Diagrama de Voronoi. Após verificar os pontos vizinhos e desenhar as células do Voronoi, as bibliotecas de visualização são utilizadas para manipular o desenho, com aplicações de zoom e filtros na busca das proteínas hubs (Capítulo 2, seção 2.3).

Uma implementação mais completa seria com o uso de algoritmos recursivos, pois possibilita trabalhar com grandes redes protéicas. A recursividade não foi utilizada neste trabalho por não dominar a elaboração de algoritmos recursivos e por não haver tempo hábil para a mudança.

[22] Fonte Bibliográfica das bibliotecas: (Junior, 2008).

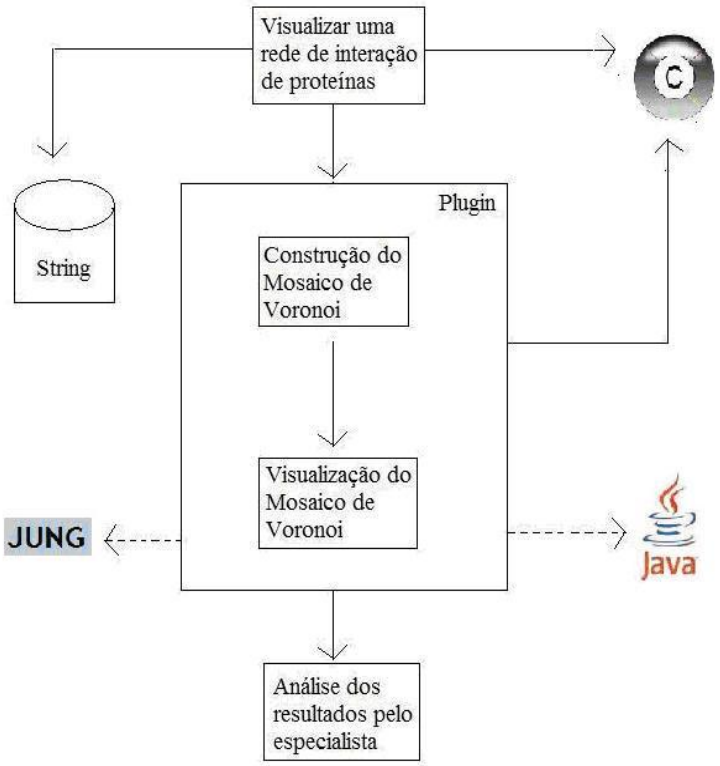


Figura 31. Arquitetura de Software

4.5 Considerações Finais

O software Cytoscape é uma ferramenta de auxilio na análise de redes de proteínas, cujas diversas funcionalidades estão disponíveis através de *plugins* que, por ser de código fonte aberto, podem ser desenvolvidos pela comunidade do Cytoscape ou por usuários e pesquisadores do programa.

Os requisitos da aplicação são úteis para auxiliar durante o desenvolvimento, quais etapas devem ser implementadas para garantir uma aplicação mais robusta e completa. O diagrama de classes, criado a partir da análise dos requisitos necessários, é outra ferramenta de auxílio para o desenvolvimento da aplicação, onde são detalhadas algumas informações importantes para o desenvolvedor, tais como os tipos de variáveis, os métodos necessários e principalmente, as relações entre as classes que compõem a aplicação. Por fim, o diagrama que representa a arquitetura da aplicação mostra quais os processos relacionados ao uso do *plugin* desenvolvido, e quais ferramentas são utilizadas nesses processos, seja elas no desenvolvimento, na obtenção de dados ou na manipulação dos resultados.

Essas ferramentas são também utilizadas durante a fase de testes, mostrada no próximo capítulo, para validar a implementação, verificando se todos os requisitos foram atendidos, se as relações entre as classes foram atendidas e se os processos da aplicação como um todo seguem o fluxo correto de execução.

5 ESTUDOS DE CASO[23]

Neste capítulo são apresentados alguns testes efetuados com o *plugin* do Mosaico de Voronoi no ambiente do software Cytoscape. Foi utilizado nos testes três redes proteícas: *i)* a COX1, que possui 11 nodos e está presente na cadeia respiratória, atuando na catálise de redução de oxigênio da água; *ii)* a P53, que possui 356 nodos e que atua como um supressor tumoral em muitos tipos de tumor, além de induzir o crescimento ou retardo, dependendo de circunstâncias como o tipo celular e fisiológico; *iii)* e a physical, que possui mais de quatro mil nodos. Sobre essas redes foram aplicados testes com o uso de escalas diferentes para as coordenadas x e y. Os resultados obtidos foram:

Proteína COX1 no repositório de proteínas String:

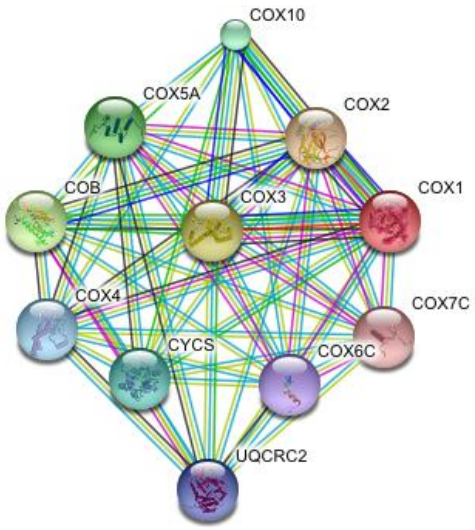


Figura 31. COX1 no repositório de proteínas String.

- Proteína COX1 sem uso de escala:

[23] As informações que constam neste capítulo foram todas extraídas do repositório de proteínas String, disponível no website < http://string-db.org/>

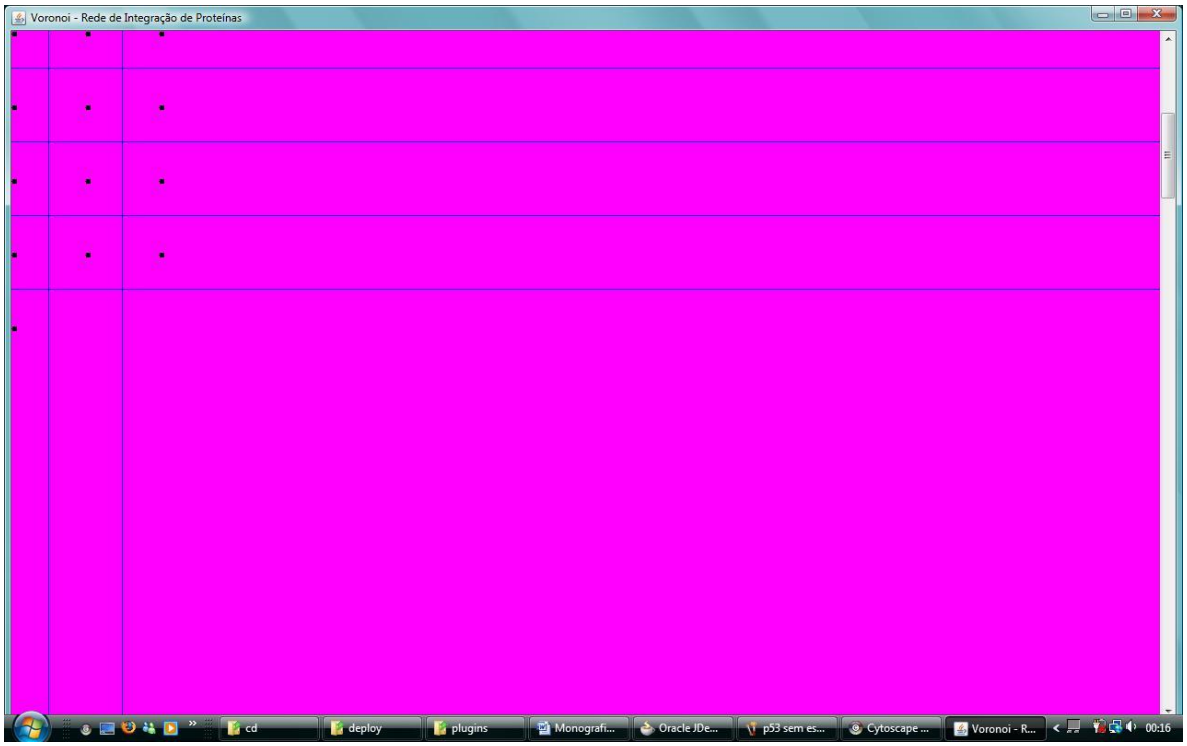


Figura 32. COX1 com o Voronoi sem escala.

- Proteína COX1 com escala: coordenada x = x/2 e coordenada y = y/2.

Figura 33. COX1 com o Voronoi com escala.

Proteína P53 no repositório de proteínas String:

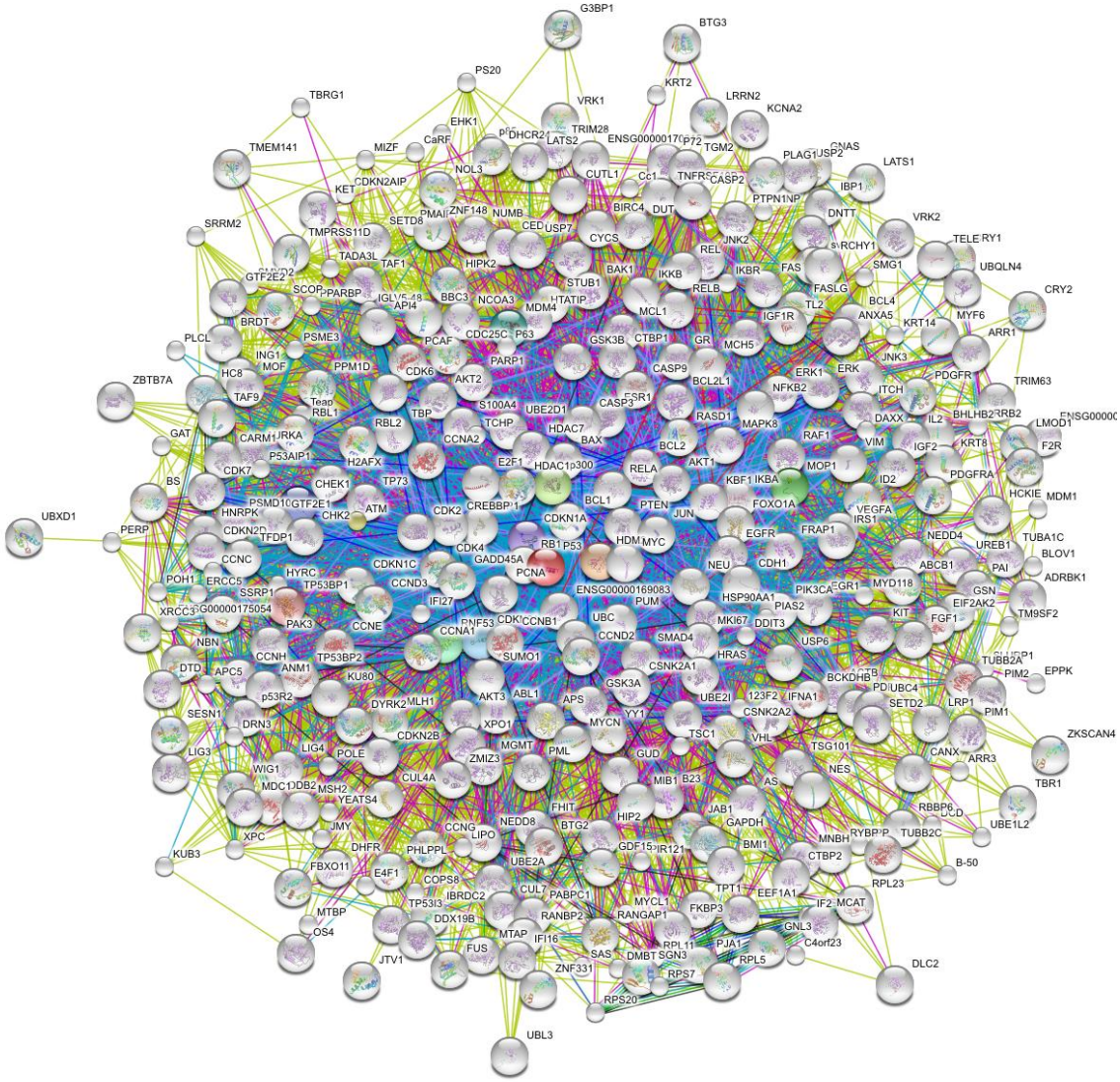


Figura 34. P53 no ambiente repositório de proteínas String.

- Proteína P53 sem uso de escala:

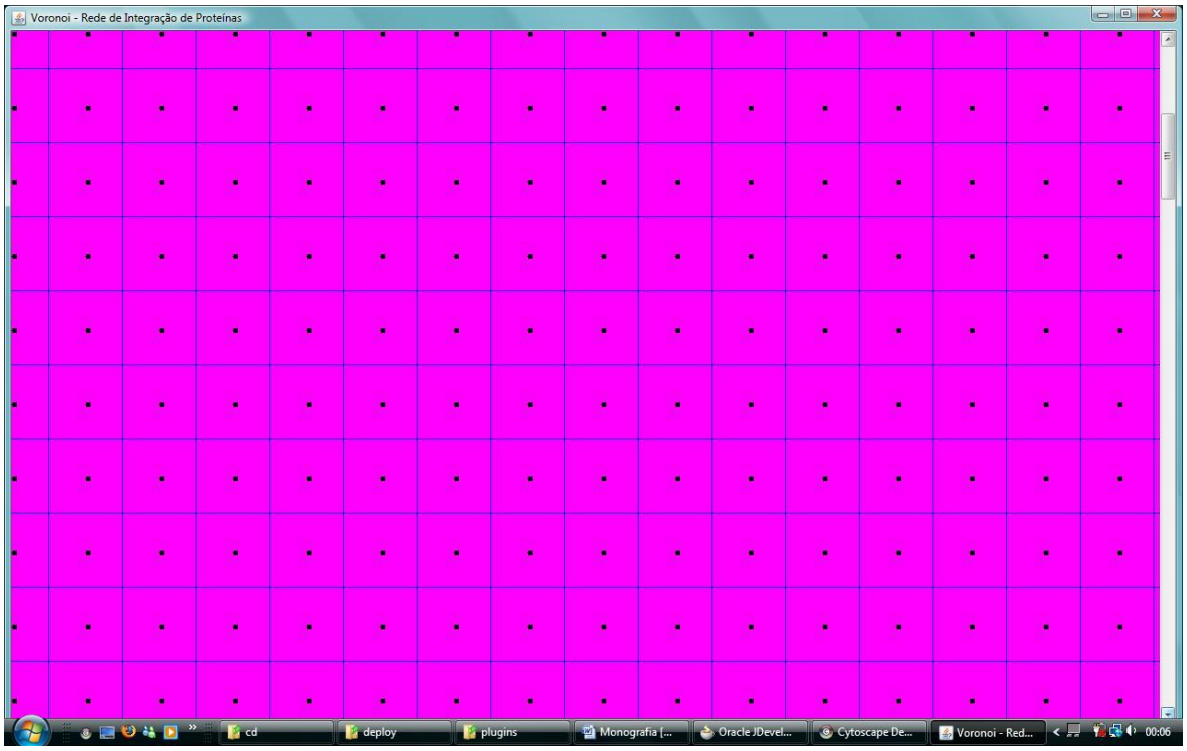


Figura 35. P53 com o Voronoi sem escala.

- Proteína P53 com escala: coordenada x = x/2 e coordenada y = y/2.

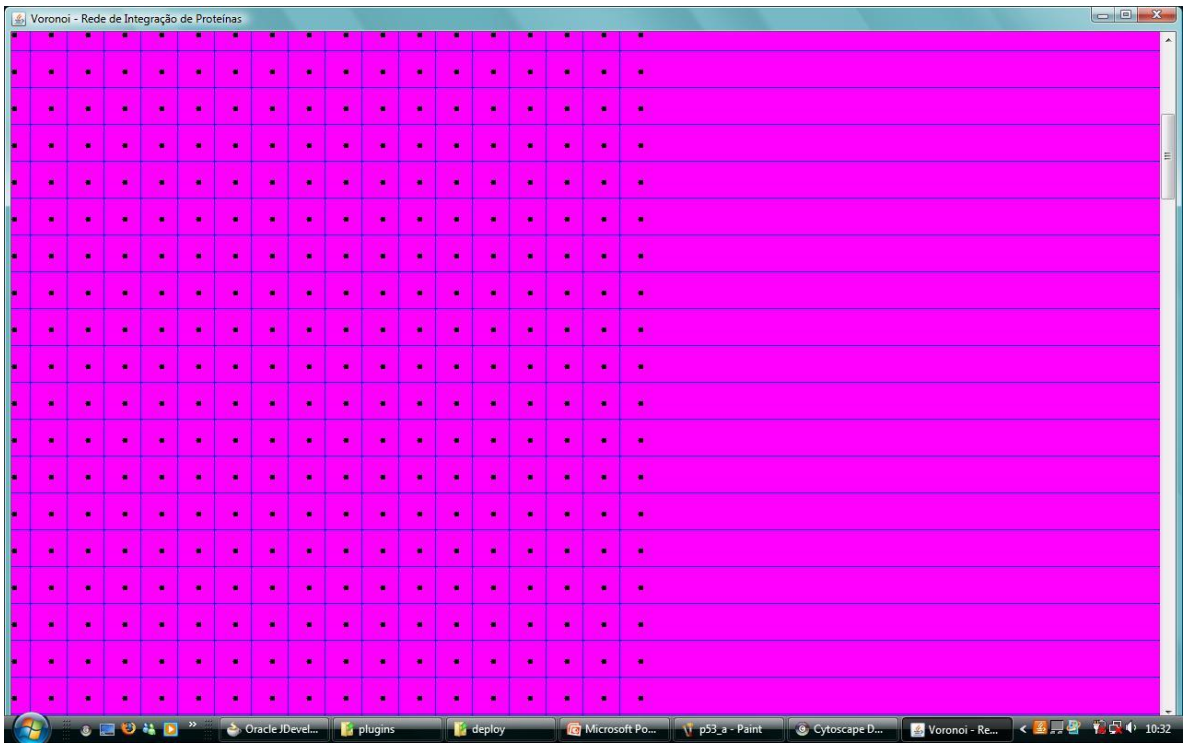


Figura 36. P53 com o Voronoi com escala.

Proteína PhysicalInt_21979 no ambiente do Cytoscape:

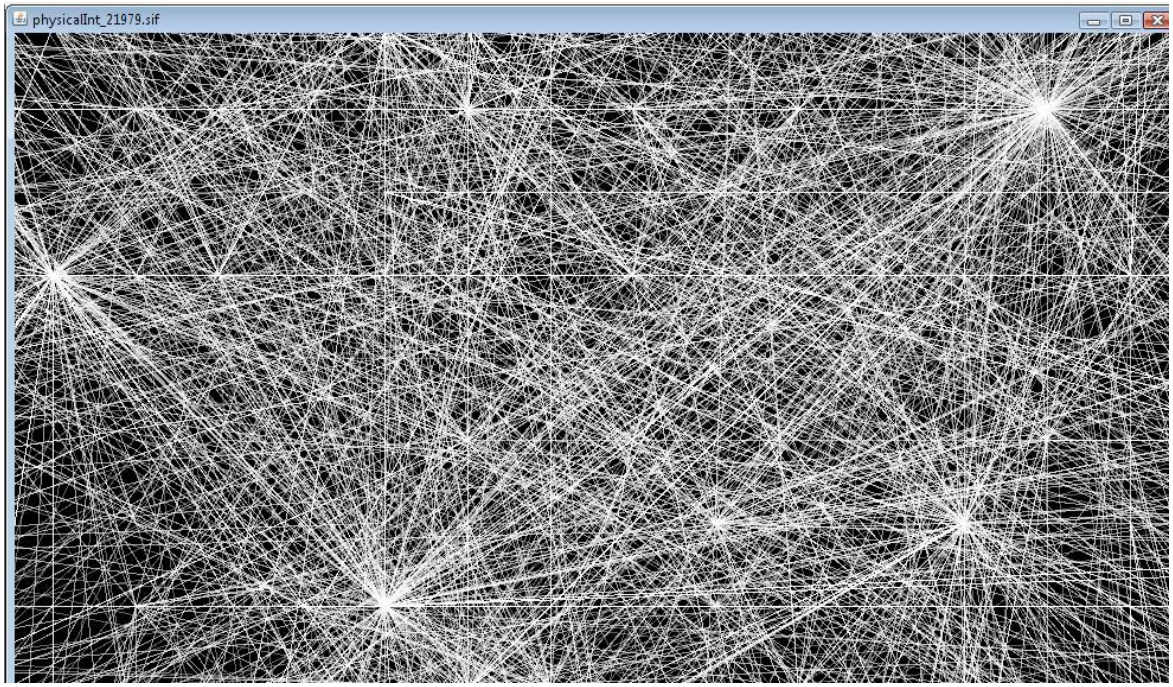
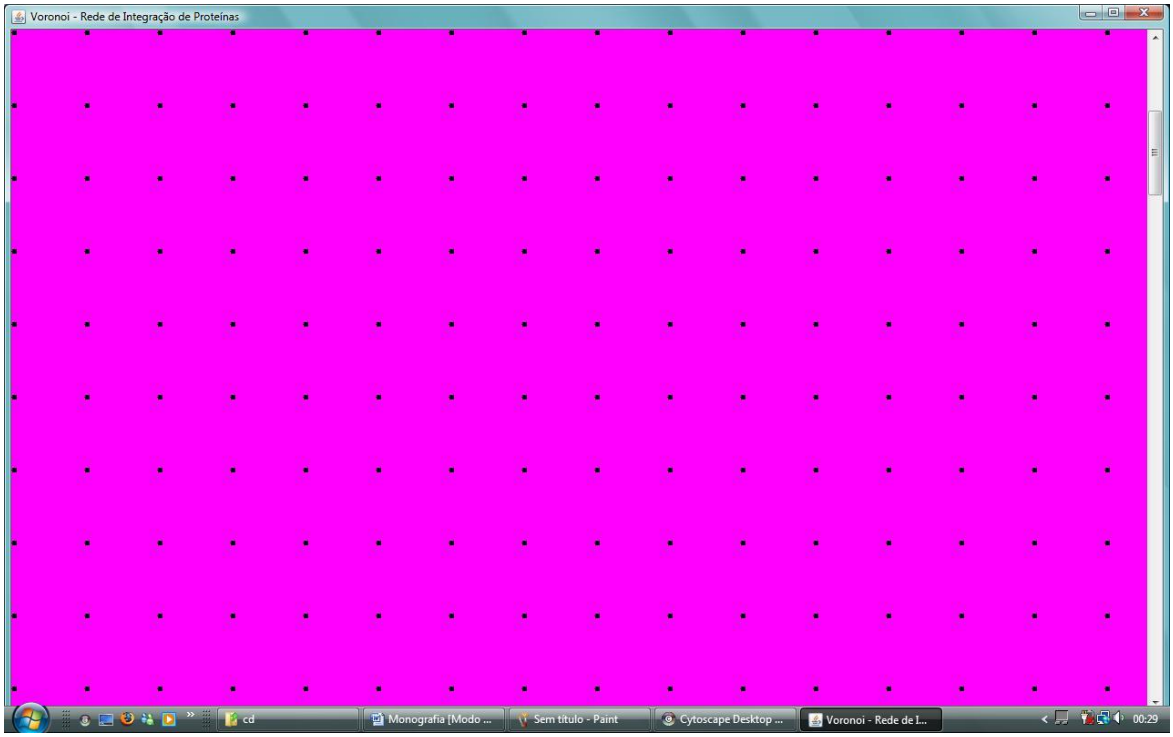
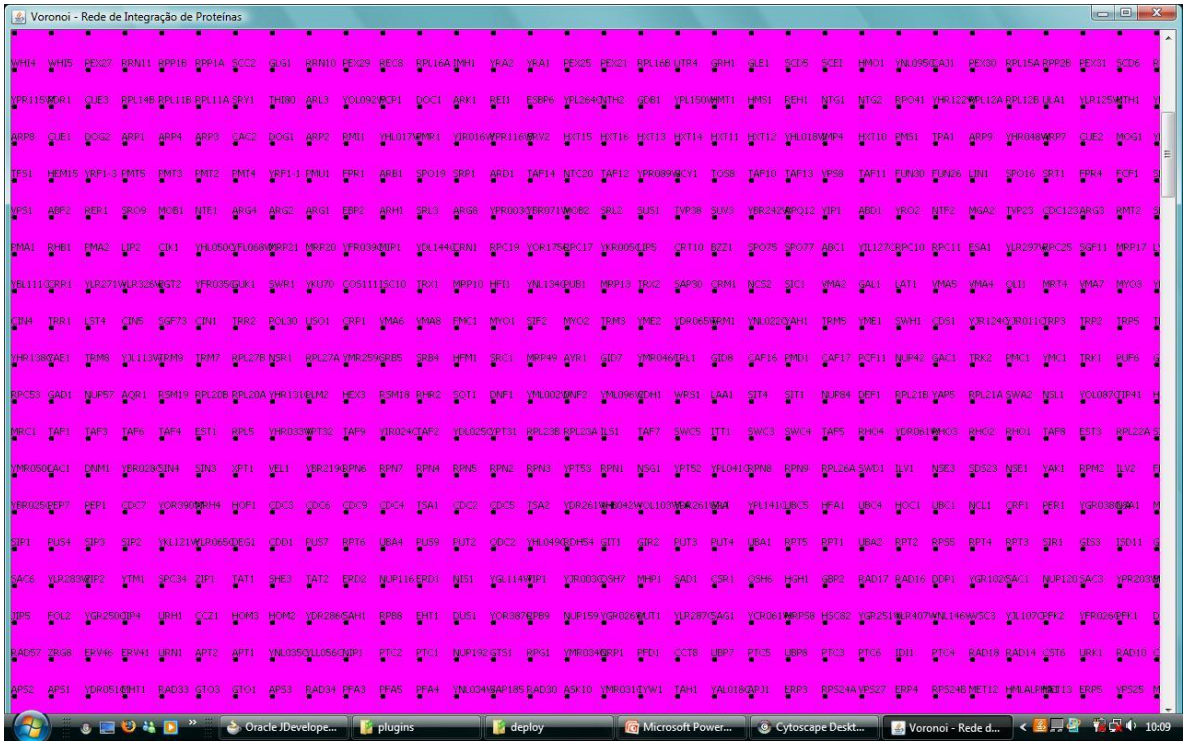


Figura 37. PhysicalInt_21979 no ambiente Cytoscape.

- Proteína PhysicalInt_21979 sem uso de escala: não foi possível gerar a visualização do Mosaico devido ao tempo de processamento. Somente os pontos.



- Proteína PhysicalInt_21979 com escala: coordenada x = x/2 e coordenada y = y/2. Não foi possível gerar a visualização do Mosaico devido ao tempo de processamento. Somente os pontos com o nome da proteína.



5.1 Considerações Finais

A utilização de estudos de caso é muito importante, pois possibilita a validação de aplicações desenvolvidas quanto a execução das tarefas solicitadas nos requisitos da aplicação, ao tempo de processamento, a facilidade de uso e a ocorrência de falhas.

Através desses estudos foi possível verificar, no contexto deste trabalho, se o *plugin* calcula e desenha corretamente o Mosaico de Voronoi para redes protéicas de tamanhos diferenciados, o comportamento do tempo de processamento em função da quantidade de dados processados, a verificação de sua utilização, se é viável ou não, visto que a maioria das redes protéicas são formadas por um grande número de proteínas, e se possibilita a identificação das proteínas hubs.

Foi possível constatar também que as coordenadas x e y das proteínas no Cytoscape possuem inúmeros alinhamentos, o que não possibilita gerar uma boa representação do Mosaico de Voronoi, principalmente na delimitação das células com o infinito.

6 CONSIDERAÇÕES FINAIS

O crescente número de pesquisas médicas relacionadas ao comportamento de organismos celulares, e principalmente a importância das proteínas e suas interações nesses organismos, necessitam de sistemas e ferramentas que possibilitem trabalhar com maior rapidez e facilidade, utilizando as informações disponíveis nos diversos bancos de dados proteômicos. Um dos sistemas disponíveis atualmente é o software Cytoscape, um sistema de código fonte aberto, que possibilita mapear e analisar as redes protéicas com o uso de ferramentas *plugins*, que podem ser desenvolvidas pelos próprios pesquisadores, conforme suas necessidades. O uso dos Diagramas de Voronoi para a representação de redes protéicas é a base para a implementação do *plugin* Voronoi descrito neste trabalho.

6.1 Conclusões

O *plugin* Voronoi implementado neste trabalho para ser utilizado no software Cytoscape, tem por objetivo utilizar o modelo computacional conhecido como Diagrama de Voronoi para representar graficamente as proteínas e suas interações na rede, visando identificar quais são proteínas *hubs*, consideradas mais importantes em relação a suas funções dentro da célula.

A validação do *plugin* foi com o estudo de caso de três proteínas de tamanhos diferenciados. Para os testes foram usados dois critérios: com ou sem escala de representação das coordenadas dos nodos das proteínas e com ou sem a impressão dos nomes das proteínas. Essas duas opções foram utilizadas para verificar qual opção resultaria numa melhor representação da rede. Foi possível verificar que a melhor representação foi com o uso de escalas sem a impressão dos nomes, facilitando a visualização do mosaico e a conseqüente localização das proteínas hubs.

Nos testes, foi possível avaliar que, se a rede for manipulada previamente no Cytoscape e posteriormente aplicada no *plugin*, a representação pode ser prejudicada na localização de algumas coordenadas que podem ficar negativas.

Foi constatado também que sem um algoritmo recursivo no cálculo do vizinho mais próximo e sem o uso de escalas a representação de redes acima de 600 nodos se torna inviável.

6.2 Trabalhos Futuros

O uso dos componentes *JPanel* e *JFrame*, inviabilizou a navegação pela visualização do mosaico com as barras laterais. Como trabalho futuro fica a opção de trocar o *JPanel* por um *applet* ou incluir outros componentes que possibilitem a navegação. Fica também como sugestão, a utilização de bibliotecas gráficas, como a JUNG, citada no capítulo 4, seção 4.4, possibilitando a aplicação de zoom, o que facilitaria a visualização e manuseio das proteínas *hubs*.

Outra sugestão seria a utilização do próprio código fonte do Cytoscape para mapear a rede e verificar as proteínas que interagem entre si.

Outra sugestão de trabalho futuro seria decorrente dos resultados mostrados pelo *plugin*. Através da identificação das proteínas hubs, pode-se fazer a análise dessas proteínas num nível mais aprofundado e detalhado de sua estrutura.

7 REFERÊNCIAS

Albert R, Barabási A-L. Statistical physics of complex networks. Reviews of Modern Physics 74, 47 (2002).

Aurenhammer, F. Voronoi Diagrams – a Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys, Vol 23, Nº 3, 345-397 (1991).

Bajuelos, Antonio L. Problemas de proximidade: Diagramas de Voronoi. Universidade de Aveiro. Disponível em: <http://www2.mat.ua.pt/pessoais/Leslie/geocom/slides /gc_0708_7_diagramas_voronoi.pdf >. Acessado em Maio 2009.

Barabasi AL, Oltvai ZN. Network biology: understanding the cell's functional organization. Nature Reviews. Genetics 5, 101-113 (2004).

Bebek, Gurkan. Analyzing and Modeling Large Biological Networks: Inferring Signal Transduction Pathways. Case Western Reserve University, 10-41 (2007).

Brolezzi, Antonio Carlos. A Tensão entre o Discreto e o Contínuo na História da Matemática e no ensino da Matemática. Tese (Doutor em Educação) – USP, São Paulo (1996). Disponível em <http://www.ime.usp.br/~brolezzi/publicacoes/teses /brolezzidr.pdf>. Acessado em Maio de 2009.

Carvalho, Marcelo T. M. de; Vera, Mário de S. Desenvolvimento de Aplicativos com a TerraLib. Bancos de Dados Geográficos. Editora MundoGEO, Curitiba – PR 461-490 (2005).

Carvalho, Marco Antônio de. Teoria dos Grafos – Uma Introdução. UNICAMP, São Paulo (2005). Disponível em <http://www.ceset.unicamp.br/~magic/ST069/ Apografos_ceset.pdf>. Acessado em Abril de 2009.

Cohen, Jacques. Bioinformatics—An Introduction for Computer Scientists. Brandeis University, 122–158 (2004).

Cytoscape User Manual. Disponível em < http://www.cytoscape.org/manual/ Cytoscape2_6Manual.pdf>. Acessado em Dezembro de 2008.

Davis, Clodoveu; Câmara, Gilberto - Introdução à ciência da geoinformação. UFMG, Capítulo 3, 1-35 (2001). Disponível em <http://www.csr.ufmg.br/geoprocessamento/ centrorecursos/relacionadas/cap3-arquitetura.pdf>. Acessado em Abril de 2009.

Esperança C., Cavalcanti, P.R. Geometria Computacional. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1-27 (2000).

Figueiredo, L. H., Carvalho, P. C. P. Notas de Geometria Computacional. IMPA, Rio de Janeiro, 1-15 (2005).

Gibas, C. Desenvolvendo Bioinformática: Ferramentas de Software para Aplicações em Biologia. Rio de Janeiro, Campus (2001).

Junior, Valdir A. L. Estudo de Bibliotecas Java para visualização de redes/grafos. UNIVALI, 1-27 (2008).

Kingsbury, David T. Computacional Biology. ACM Computing Surveys, Vol. 28, No. 1, 101-103, (1996).

Oliveira, Marcos B. et al. A base molecular da vida: uma introdução à biologia molecular. Universidade de São Paulo e Ed. Poligono, 29-65 (2007)

Passarge, Eberhard. Genética: texto e atlas – 2.ed. Artmed Editora, 20-45 (2004).

Preparata, Franco P; Shamos, Michael Ian. Computacional Geometry – An Introduction. 2.ed. Springer-Verlag, x-y (1988).

Poupon A. Voronoi and Voronoi-related tessellations in studies of protein structure and interaction. Current Opinion in Structural Biology 14, 233-241 (2004).

Prosdocimi, F. et al. Bioinformática: manual do usuário. Biotecnologia, Ciência & Desenvolvimento. Ano 5. v 29. nov/dez 2002. p. 12-25. Disponível em: <http://www.biotecnologia.com.br/revista/bio29/bioinfo.asp>. Acesso em Dezembro de 2008.

Raychaudhuri, Soumya. Computacional Text Analysis for Functional Genomics and Bioinformatics. Oxford University Press, 42-52 (2006).

Rezende, P. J., Stolfi, J. Fundamentos de Geometria Computacional. Universidade Estadual de Campinas, São Paulo, 179-200 (1994).

Waterman, Michael S. Introduction to Computational Biology: Maps, sequences and genomes. Chapman & Hall/CRC, 29-40 (2000).

Weston, Andrea D; Hood, Leroy. Systems Biology, Proteomics, and the Future of Health Care: Toward Predictive, Preventative, and Personalized Medicine. Journal of Proteome Research, 3, 179- 196 (2004).