Carla Alexandra Marques Gregório

Análise de Dados de Desnaturação Proteica Obtida por Simulações de Dinâmica Molecular

Carla Alexandra Marques Gregório

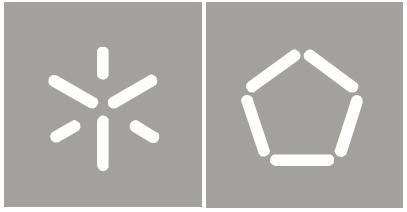
UMinho | 2012

Universidade do Minho
Escola de Engenharia

Carla Alexandra Marques Gregório

Análise de Dados de Desnaturação Proteica Obtida por Simulações de Dinâmica Molecular

Outubro de 2012



Universidade do Minho
Escola de Engenharia

Carla Alexandra Marques Gregório

Análise de Dados de Desnaturação Proteica Obtida por Simulações de Dinâmica Molecular

Tese de Mestrado
Mestrado em Engenhara Informática - Bioinformática

Trabalho efectuado sob a orientação do
Professor Doutor Paulo Jorge de Sousa Azevedo

Outubro de 2012

DECLARAÇÃO

Nome: Carla Alexandra Marques Gregório

Endereço electrónico: PG17350@alunos.uminho.pt Telefone: 967234602

Número do Bilhete de Identidade: 13607098

Título dissertação/tese: Análise de Dados de Desnaturação Proteica Obtida por Simulações de Dinâmica Molecular

Orientador: Professor Doutor Paulo Jorge de Sousa Azevedo

Ano de conclusão: 2012

Designação do Mestrado: Mestrado em Engenharia Informática

Universidade do Minho, Outubro de 2012

Assinatura: Carla Alexandra Marques Gregório

Agradecimentos

Acredito que nada do que produzimos é apenas fruto de nós mesmos. Podemos ter a maior fatia do esforço e de dedicação mas é a interacção com outros, a troca de experiências e o apoio e motivação de amigos e familiares que completam qualquer vivência ou qualquer trabalho. Assim, resta-me agradecer a quem contribuiu para que este objetivo, não só académico e profissional, mas também pessoal, fosse atingido.

Ao Professor Paulo Azevedo o meu sincero agradecimento por todo interesse e disponibilidade demonstrada ao longo de todo projeto, pelos conhecimentos transmitidos, pelo rigor exigido e por todas as críticas e conselhos.

Ao Professores Rui Brito e à doutora Cândida Silva pelo suporte, tanto no esclarecimento de dúvidas para total compreensão, definição e discussão do problema e conceitos mais relevantes, como na elaboração e análise do caso de estudo e sugestões para o desenvolvimento da ferramenta.

A todos os que me forneceram material de estudo e que me ajudaram na resolução de problemas técnicos.

Aos meus pais e irmãos pelo apoio e motivação, bem como aos restantes familiares e amigos.

Ao Rogério por ser o meu porto de abrigo.

Por último, quero agradecer o facto deste trabalho fazer parte do Projeto Estratégico do CCTC – Refª PEst-OE/EEI/UI0752/2011.

Resumo

A Polineuropatia Amiloidótica Familiar, mais conhecida em Portugal por Doença dos Pezinhos ou por Paramiloidose, é uma doença incurável, apresentando uma rápida evolução e podendo conduzir à morte do paciente. Esta patologia é desencadeada por um processo de desnaturação da Transtirretina (TTR) – uma proteína produzida pelo organismo humano – que provoca a acumulação de elevadas quantidades de substâncias fibrilares da proteína mutada em diversos tecidos. Esta acumulação condiciona o normal funcionamento do organismo.

A dinâmica molecular é uma técnica de simulação computacional que permite derivar os diferentes estados de desnaturação da TTR. Foram efetuadas diferentes simulações de desnaturação proteica representando diferentes condições iniciais sobre a TTR e uma sua variante mais amiloidogénica (Leu55Pro). Destas simulações foram consideradas dez para aplicação de algoritmos de *Graph Mining* da biblioteca ParMol, para extração de subgrafos (fragmentos). Os fragmentos foram posteriormente organizados em grupos conforme a zona da proteína em que se encontram. Estes fragmentos são a base dos caminhos de desnaturação.

O objetivo desta tese é o desenvolvimento de uma ferramenta de visualização que permita o estudo do fluxo de evolução da TTR ao longo das simulações de desnaturação. Nomeadamente pretende-se identificar as interações entre resíduos que compõem a TTR e comandam o processo de desnaturação. A ferramenta (Subgraph Paths) permite a visualização e análise não só dos fragmentos extraídos mas também dos caminhos de *folding/unfolding* associados a estes. Um caminho é uma trajetória de evolução de um fragmento ao longo do tempo numa simulação. Esta evolução pode ser composta por momentos de expansão e de retração do fragmento, em termos de ganhos ou perdas de ligações. Outro conceito importante abordado na dissertação é o de procura de caminhos noutras simulações. A procura permite expor semelhanças e diferenças no comportamento dos resíduos entre simulações de diferentes variantes da proteína.

Abstract

Familial Amyloid Polyneuropathy, best known in Portugal as "Doença dos Pezinhos", has no cure and has a very fast evolution, being quickly fatal. This pathology is triggered by a Transthyretin denaturation process – Transthyretin, a protein produced in human organism - causing the accumulation of large amounts of mutated protein fibrillar substances in various tissues, affecting normal body functioning.

Molecular dynamics is a computer simulation technique that allows deriving different TTR denaturation states. Different denaturation simulations were performed considering different initial conditions about TTR and about its more amyloidogenic variant (Leu55Pro). Ten of these simulations were considered in order to apply graph mining algorithms from ParMol library, aiming to extract frequents sub graphs (fragments). The fragments were subsequently organized into groups according to the part of the protein where they belong. These fragments are the paths basis.

The purpose of this thesis is the development of a visualization tool that allows the study of TTR evolution flow along the unfolding simulations. In particularly, it is intended to identify the interactions between residues that compose TRR which command the unfolding process. The tool (Subgrapg Paths) permits to visualize and to analyze not only the extracted fragments but also the folding/unfolding paths related to them. A path is a fragment evolution trajectory along a run. An evolution may be composed of moments of expansion or retraction of the fragment, in terms of gains or losses of connections. Another important discussed concept is the search of paths in other simulations. The search exposes similarities and differences in residues behavior between different simulations that may give more clues about Transthyretin unfolding process.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Fórmulas

Lista de Algoritmos

Lista de Siglas e Acrónimos

ASS	Amiloidose Senil Sistémica
CAP	Cardiomiopatia Amiloidótica Familiar
GPL	*GNU General Public License*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
PAF	Polineuropatia Amiloidótica Familiar
PIM	*Platform-Independent Model*
TTR	Transtirretina
UML	*Unified Modeling Language*

Capítulo 1

Introdução

O estudo de qualquer doença envolve uma panóplia de áreas de conhecimento de forma a acelerar e completar o estudo em diversas vertentes. Uma destas áreas é sem dúvida a informática.

A informática tem vindo a ter um papel crucial no estudo de várias doenças devido ao elevado volume de dados envolvidos em experiências de investigação quer de cura, quer de causas. Para análise desses dados tipicamente utilizam-se técnicas de *Data Mining* de modo a extrair padrões e informação sobre o comportamento de compostos e/ou substâncias. Além disto, as ferramentas e plataformas informáticas, tanto para alojamento, organização e filtragem, como para análise de dados provenientes de simulações e experiências, são sempre uma mais-valia e encurtam o tempo de qualquer investigação.

Esta dissertação apresenta o trabalho realizado no âmbito do desenvolvimento de uma ferramenta informática em que serão usadas técnicas de *Data Mining*. Esta ferramenta vem em prol da assistência à investigação das causas de uma doença específica - a Paramiloidose. Porém, poderá também ser extensível a outras patologias cujo processo de desencadeamento seja semelhante.

A Polineuropatia Amiloidótica Familiar, mais conhecida em Portugal por Doença dos Pezinhos ou Paramiloidose, é uma doença ainda sem cura e que, em muitos casos, poderá ser mortal. Esta doença é causada pela mutação de uma proteína produzida pelo organismo humano – a Transtirretina – que, quando mutada, provoca a acumulação de

fibras de amiloide em diversos tecidos, condicionando assim o funcionamento de certos órgãos. Contudo, ainda não é possível explicar totalmente como são formadas as fibras de amiloide a partir da proteína mutada. O trabalho apresentado compreende a utilização de técnicas de *Data Mining*, nomeadamente de *Graph Mining,* de modo a extrair fragmentos (sub-moléculas) da proteína mutada e nativa associada à doença. Estes fragmentos são obtidos a partir de simulações de dinâmica molecular da Transtirretina. Usando estes fragmentos, pode-se traçar trajetórias de evolução com o objetivo de expor o seu comportamento.

A ferramenta permitirá não só analisar os fragmentos relevantes obtidos, visualizando-os graficamente, mas também traçar trajetórias de evolução, denominadas de caminhos. Por sua vez, estes caminhos poderão ser visualizados e o utilizador poderá navegar nestes, estudando o comportamento de um fragmento em pormenor. A ferramenta contará com algumas funcionalidades adicionais como a possibilidade de refazer e expandir um caminho, e de procurar um caminho em diferentes simulações visando encontrar semelhanças entre os fenómenos ocorridos em diferentes variantes da proteína associada à Paramiloidose.

1.1. Motivação e Objetivos

A Paramiloidose, por ser uma doença ainda sem cura e mortal, é continuamente alvo de estudos. Além disto, por pertencer ao grupo das amiloidoses, em que se incluem por exemplo as doenças de Alzheimer, de Parkinson e de Huntington, torna-se um caso de estudo interessantíssimo, pois os resultados obtidos são passíveis de se estender a outras patologias do grupo.

Apesar dos muitos estudos já realizados, ainda não existe cura para nenhuma amiloidose, nem as suas causas e aparecimento estão completamente clarificados. Mais concretamente, no caso da Paramiloidose, ainda é necessário estudar aprofundadamente o comportamento da proteína que mutada leva ao aparecimento da doença. Ou seja, pretende-se saber que interações entre os resíduos que compõem a Transtirretina comandam o processo de desnaturação. Assim, esta tese surge no sentido de colmatar a necessidade da existência de um novo estudo que, com base em trabalhos e simulações de dinâmica molecular já realizados, permita uma nova abordagem a este problema de modo a lançar mais pistas sobre o comportamento da Transtirretina.

Usando técnicas de *Data Mining*, pretende-se extrair fragmentos das simulações de desnaturação proteica associadas à doença e, posteriormente, derivar informações relevantes sobre estes. Para isto, será usado o conceito de caminho e de procura de similaridade, tendo como foco a transformação e manuseamento destes fragmentos que por só si – dado o seu volume e natureza – não conseguem fornecer diretamente informações relevantes. Assim, o objetivo principal é desenvolver uma ferramenta que permita assistir investigadores, tanto no estudo da Transtirretina, como de outras proteínas associadas às amiloidoses. A elaboração desta ferramenta terá em conta os resultados de outros trabalhos já publicados em que foram usadas técnicas *Data Mining* e trabalhos em que foi estudada a Transtirretina, não esquecendo os requisitos impostos pelos utilizadores.

1.2. Considerações Linguísticas

Por imposição da área em que esta tese está a ser desenvolvida – a informática – torna-se impossível a não recorrência a termos estrangeiros, sobretudo anglófonos. Não se trata, contudo, de qualquer desrespeito pela língua portuguesa, mas antes da quase inexistência de termos que lhe correspondam em português. Nalguns casos, apesar de haver correspondência linguística, os termos são ainda pouco conhecidos por estarem fracamente disseminados. Este facto poderia levar a interpretações erradas ou a confusão por parte de quem lê. Deste modo, a opção por manter os termos originais prende-se maioritariamente com o desejo de haver clareza na exposição e explicação. Acrescente-se que esta tese foi escrita de acordo com as regras do recente acordo ortográfico.

1.3. Estrutura do Documento

Esta dissertação está dividida em seis partes:

- Uma primeira parte em que o problema é contextualizado e em que são explicados os fundamentos teóricos necessários para a sua compreensão. Esta parte compreende os capítulos 2 e 3. No capítulo 2 é apresentada a doença e a sua relação no grupo das amiloidoses. No capítulo 3 são introduzidas algumas definições da biologia molecular;

- Na segunda parte é apresentado o estado da arte, que corresponde ao capítulo 4;
- Na terceira parte é desenvolvida a parte teórica o trabalho. Esta parte corresponde aos capítulos 5, 6 e 7. No capítulo 5 é apresentado todo o trabalho que envolve a extração de subgrafos frequentes; no capítulo 6 a teoria, definições e regras associadas ao traçar de caminhos; e no capítulo 7 é apresentado o conceito de procura de caminho numa corrida e todas as definições, teoria e regras envolvidas;
- Na quarta parte é apresentado o resultado final do trabalho desenvolvido e a sua implementação. Esta parte corresponde ao capítulo 8;
- Na quinta parte é abordado um caso de estudo. Esta abordagem encontra-se no capítulo 9;
- Na sexta e última parte são ditadas as conclusões e o trabalho futuro. Esta parte corresponde ao capítulo 10.

Em anexo encontra-se toda a informação complementar do estudo.

Capítulo 2

Polineuropatia Amiloidótica Familiar

A Polineuropatia Amiloidótica Familiar (PAF), mais conhecida em Portugal por Doença dos Pezinhos ou Paramiloidose, foi identificada em 1939 pelo neurologista Corino de Andrade. Trata-se de uma doença neurológica de transmissão genética, crónica e progressiva e que apresenta uma evolução muito rápida, podendo conduzir em pouco tempo à morte do paciente.

Segundo o Grupo de Apoio à Paramiloidose[1], distinguem-se essencialmente quatro períodos ao longo da investigação desta doença:

- Um primeiro período de incubação, que decorre entre 1939 (data das primeiras observações de doentes na Póvoa do Varzim e em Vila do Conde) e 1952 (data da descrição dos princípios que caracterizam a Paramiloidose);
- Um segundo período de estado que durou trinta anos - até 1980 - em que se multiplicaram estudos multidisciplinares, muitos deles ligados ao Centro de Estudos de Paramiloidose criado por Corino de Andrade em 1960;
- Um terceiro período de explosão do conhecimento científico da doença em bases bioquímicas e de biologia molecular;
- Um quarto período que se inicia com a terapêutica pelo transplante hepático em 1991 e que vai até aos nossos dias, avançando no século XXI com um leque de perspetivas sempre mais alargado.

[1] Informações em http://paramiloidose.no.sapo.pt/paramiloidose.html.

Apesar de ser uma doença mortal, esta pode ser retardada através de um transplante hepático que diminui os altos níveis de proteína anormal no sangue. Contudo, este traz problemas ao nível da rejeição do órgão e da falta de órgãos disponíveis para o transplante. Em casos mais graves, já não há benefícios com o transplante devido às lesões já avançadas existentes no paciente [Adams 2000].

Em termos de causas, sabe-se que esta deve-se a uma mutação numa proteína chamada Transtirretina, sendo que existem diversos tipos de mutações associados a esta proteína. A mutação mais frequente em Portugal é conhecida como Val30Met e consiste na substituição de um único aminoácido, a Valina, por uma Metionina na posição 30 da cadeia de aminoácidos da proteína. Porém, esta e outras mutações ocorrem também noutros pontos do globo nomeadamente no Japão, na Suécia, na ilha de Maiorca em Espanha, no Brasil e em Itália [Coutinho 1989].

A Figura 1 apresenta a distribuição mundial da PAF em que a magnitude dos círculos representados está relacionada com o número de pacientes em cada localização [Ando 2005].

Figura 1 - Distribuição mundial da PAF.

Esta doença caracteriza-se essencialmente por:

- Perda da destreza motora, da sensibilidade táctil e da dor [Coutinho 1976];
- Problemas cardíacos [Ando 2005];

- Disfunção erétil e urinária [Freitas 1976];
- Disfunções gastrointestinais [Ando 2005];
- Disfunções renais [Lobato 2003];
- Perturbações oculares [Ando 2005].

2.1. Outras Doenças Relacionadas

A PAF, a doença de Alzheimer, de Parkinson e de Huntington fazem parte do grupo das amiloidoses e partilham várias características: todas estas são muito restritivas, não têm cura e são desencadeadas por um mecanismo comum de acumulação de substâncias fibrilares de proteínas em tecidos. As amiloidoses são doenças que têm origem na acumulação, sob a forma de placas, de quantidades elevadas de proteína incorretamente enrolada em tecidos [Quintas 2001]. De acordo com [Brito 2004], apesar das proteínas envolvidas diferirem em sequência, estrutura e função, a parte amiloidótica destas doenças partilha os mesmos mecanismos moleculares.

A doença de Alzheimer, descrita pela primeira vez em 1906 pelo psiquiatra e neuropatologista Aloïs Alzheimer, afeta a memória e a saúde mental, podendo levar à demência e à dependência total do paciente, acompanhados por distúrbios comportamentais [Raskind 1995]. Existem, contudo, inúmeros tratamentos que visam minimizar os sintomas, proteger o sistema nervoso e retardar a evolução da doença [Sereniki 2008].

A doença de Parkinson foi descrita pela primeira vez em 1817 pelo médico James Parkinson e é principalmente caracterizada pela rigidez muscular, tremores e lentidão de movimentos [Jankovic 2008]. 90% dos doentes sofrem também de algum tipo de desordem psicológica nalgum ponto das suas vidas [Menza 2005]. Atualmente, quer os diversos medicamentos existentes, quer a cirurgia, não mais fazem do que controlar sintomas, continuando a ocorrer a degradação das células responsáveis pelo agravamento progressivo da doença (fonte: Médicos de Portugal[2]).

[2] Informações em http://medicosdeportugal.saude.sapo.pt/utentes/doencas_neurologicas/parkinson/ .

Por último, a doença de Huntington, descrita pela primeira vez por George Huntington em 1882 é maioritariamente caracterizada por movimentos involuntários do rosto, braços e pernas, demência e depressão [Walker 2007].

Hoje em dia existem cerca de 10 000 pessoas em todo mundo com PAF (fonte: Associação Portuguesa de Paramiloidose[3]), 25 milhões com Alzheimer (fonte: Alzheimers Association[4]) e mais de 6 milhões de pessoas com Parkinson (fonte: World Parkinson Congress[5]). Para a doença de Huntington, a prevalência mundial é de 5 a 10 casos por cada 100 mil habitantes [Sharon 2010]. Apesar de, em comparação com as outras doenças, a PAF apresentar menor prevalência, esta continua a ser um caso de estudo interessante pois ainda não é possível explicar totalmente o processo de desnaturação proteica da Transtirretina que leva à formação da doença.

[3] Informações em http://www.paramiloidose.com/portal.html.
[4] Informações em http://www.alz.org/
[5] Informações em http://www.worldpdcongress.org/ .

Capítulo 3

Proteínas

A mutação genética na Transtirretina que desencadeia a PAF provoca a acumulação de fibras de amiloide da proteína em diversos órgãos, impedindo o seu normal funcionamento. Quantos mais depósitos, mais estes condicionam o funcionamento do organismo. Além disto, é sabido que estas fibras de amiloide são muito resistentes à dissolução e à degradação. Neste capítulo, será feita uma abordagem geral às proteínas e aos vários conceitos biológicos importantes para a compreensão do problema.

Este capítulo está dividido em 5 secções: na primeira são abordados os conceitos base relativos às proteínas; na segunda é abordado o fenómeno de naturação proteica; na terceira é apresentada a Transtirretina, proteína associada à PAF; na quarta apresenta-se o conceito de desnaturação proteica que está na origem da doença; e, por fim, são abordadas as simulações de dinâmica molecular de desnaturação da Transtirretina.

3.1. Conceitos Base

O organismo humano, tal como o organismo de todos os seres vivos, é composto por várias substâncias distintas. Das substâncias mais importantes destacam-se as proteínas por desempenharem funções vitais a vários níveis. Dentro destas funções encontram-se as estruturais, enzimáticas, hormonais, nutritivas, de defesa e de coagulação sanguínea, e de transporte (ver mais informações no anexo I - A). O próprio termo Proteína deriva do grego *proteíos*, que significa "o mais importante". Segundo [Morrison 1992], esta designação foi atribuída justamente pela elevada importância das proteínas.

O ciclo de vida de uma proteína envolve várias etapas. No início de vida, uma proteína não é mais que uma cadeia de aminoácidos (ou cadeia polipeptídica), sendo esta a sua estrutura primária. À semelhança do que é mostrado na Figura 2, um aminoácido é constituído por um carbono alfa que está ligado a um grupo Carboxilico - ácido orgânico COOH, a um grupo Amina - H2N e a um átomo de hidrogénio – H. O carbono alfa também se liga a uma cadeia lateral - designada por R na imagem - que varia de aminoácido para aminoácido. Sendo assim, as características de cada aminoácido variam consoante a composição química da cadeia lateral. Estas características poderão ser, por exemplo, propriedades físicas e químicas (fonte: Gallant's Biology Stuff[6]).

Uma cadeia polipeptídica é única e específica para cada proteína e uma alteração num único aminoácido é suficiente para trazer graves consequências. Associadas à PAF, existem vários tipos de mutações que variam de acordo com o aminoácido substituído na cadeia. Atualmente são conhecidas mais de 80 variantes patogénicas da Transtirretina [Luis 2006]. De todas as mutações, segundo [Brito 2003], a mais amiloidogénica é a Leu55Pro, em que um aminoácido Prolina substitui um aminoácido Leucina na posição 55 da cadeia. A mutação mais prevalente é a Val30Met, em que uma Metionina substitui uma Valina na posição 30 da cadeia.

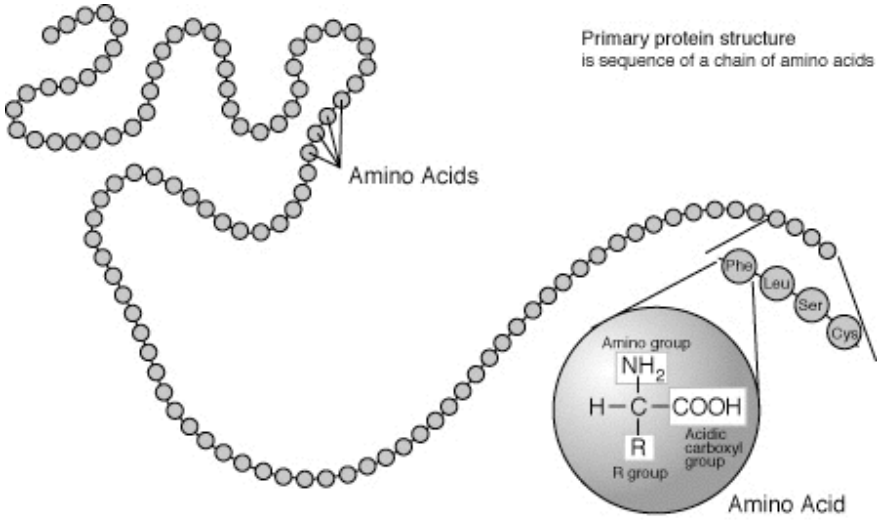


Figura 2 - Representação da estrutura primária de uma proteína.

[6] Informações em http://kvhs.nbed.nb.ca/gallant/biology/biology.html.

De modo a que uma proteína possa adquirir a sua função biológica, esta necessita de adquirir a sua forma tridimensional, sendo esta conseguida através de um processo de enrolamento, denominado de naturação proteica.

3.2. Naturação Proteica

Eventualmente, os aminoácidos de uma cadeia polipeptídica começam a arranjar-se espacialmente, criando novas ligações com os aminoácidos que se encontram mais próximos. Os 2 tipos de formas de ligações mais comuns são as α-hélices e as folhas β.

- α-hélice: é a forma mais comum de estrutura secundária e caracteriza-se por uma hélice em espiral estabilizada pelas pontas de hidrogénio.
- folha β: caracteriza-se pela forma de uma folha de papel dobrada em pregas. Esta estrutura é também estabilizada pelas pontas de hidrogénio.

A Figura 3 (fonte: Gallant's Biology Stuff[7]) mostra uma representação de uma α-hélice e de uma folha β.

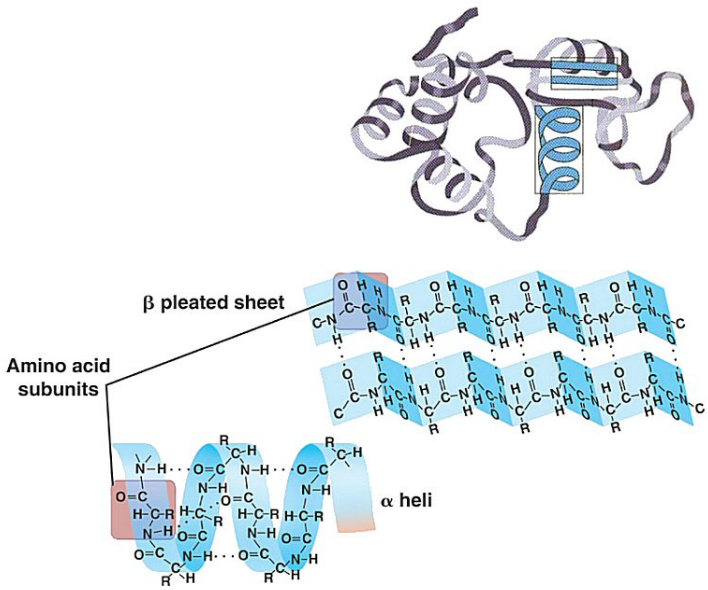


Figura 3 - Representação de uma α-hélice e de uma folha β.

Poderá haver também o arranjo espacial global entre aminoácidos que poderão estar distantes. Este arranjo da proteína dá origem a dobras da cadeia sobre si mesma, dando origem à estrutura terciária. Por último, algumas proteínas podem ter mais do que uma

[7] Informações em http://kvhs.nbed.nb.ca/gallant/biology/biology.html.

cadeia polipeptídica. Nestes casos, as ligações entre estas cadeias têm a mesma natureza das que ocorrem na estrutura terciária, originando a estrutura quaternária. Como exemplos de tais proteínas, veja-se os casos da Transtirretina e da Hemoglobina, duas proteínas com uma estrutura quaternária por serem constituídas por quatro cadeias polipeptídicas.

A Figura 4 (fonte: National Human Genome Research Institute[8]) apresenta um esquema que representa os 4 níveis possíveis de uma estrutura de uma proteína.

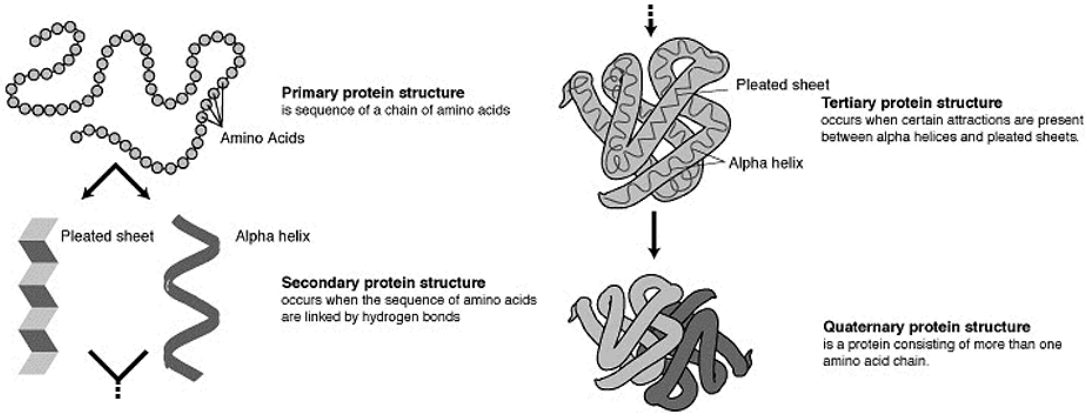


Figura 4 - Representação dos vários níveis de estrutura de uma proteína.

3.3. Transtirretina

A Transtirretina (TRR) é uma proteína libertada quase exclusivamente pelo fígado [Felding 1982], mas produzida também em pequenas quantidades pelos plexos coroideus do cérebro [Dickson 1985]. Tem como função ligar-se a algumas hormonas de modo a transportá-las no sangue pelas artérias e veias e distribui-las pelo organismo inteiro [Hamilton 1993]. O próprio nome da proteína é uma sigla que deriva da sua função de transportar as hormonas Tiroxina e o Retinól (Vitamina A) [Richardson 2007].

Para além da PAF, a Transtirretina está envolvida noutras duas patologias, a Amiloidose Senil Sistémica (ASS) e a Cardiomiopatia Amiloidótica Familiar (CAP). No caso da ASS, os depósitos de amilóide correspondem à Transtirretina no seu estado normal e afecta cerca de 25% da população acima dos 80 anos, tratando-se de uma doença com poucos sintomas [Westermark 1990]. Relativamente à CAF, esta está maioritariamente

[8] Informações em http://www.genome.gov/.

relacionada com uma mutação em que uma Isoleucina substitui, na posição 122, uma Valina na cadeia polipeptídica da Transtirretina. Nesta patologia, os depósitos de amilóides são encontrados principalmente no coração [Jiang 2001].

A Figura 5 mostra a constituição da Transtirretina (fonte: Centro de Estudos em Paramiloidose Antônio Rodrigues de Mello[9]) que tem uma estrutura quaternária - composta por 4 monómeros (representados pelas letras A,B,C e D), cada um composto por 127 aminoácidos [Luís 2006].

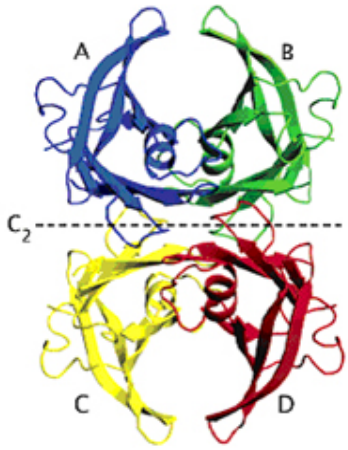


Figura 5 - Estrutura da Transtirretina.

A Figura 6 [Rodrigues 2010] representa a estrutura de um monómero da Transtirretina.

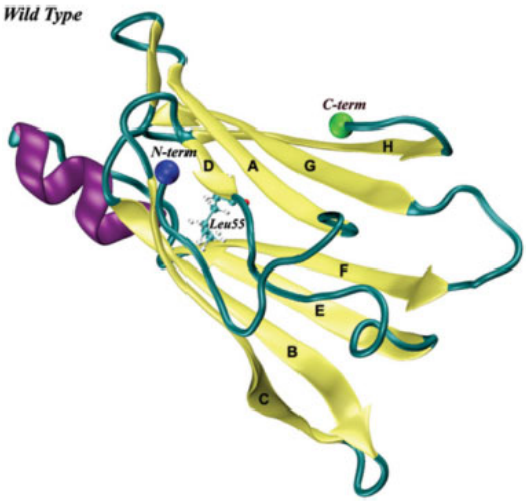


Figura 6 - Representação da estrutura de um monómero da Transtirretina.

[9] Informações em http://www.ceparm.com/.

A TRR é uma proteína composta por subunidades idênticas. Cada monómero é constituído por uma α-hélice e por 2 folhas β. Cada folha é construída por 4 cadeias β e por sete voltas que interligam as cadeias β [Brito 2003]. As folhas são as DAGH e CBEF – na Figura 6, as letras de A-H representam cadeias β. Assim, a maior parte dos resíduos estão envolvidos em cadeias β e poucos na única α-hélice.

3.3.1. Organização dos Aminoácidos

Num monómero da variante nativa (não mutada) da Transtirretina, denominada de Wild-Type, é já conhecida a localização dos aminácidos – designados de resíduos – considerando a sua estrutura secundária. Em cada monómero, os resíduos estão localizados nas cadeias β e na α-hélice, conforme mostra a Tabela 1 [Silva 2007]. Os resíduos que não aparecem na tabela não estão dispostos em nenhuma destas estruturas.

Saber a localização dos resíduos é importante no âmbito da ferramenta proposta neste estudo, pois, tal como será possível concluir mais à frente, as localizações permitem partir para estudos sobre o comportamento da proteína.

A Tabela 1 descreve a relação entre os diversos resíduos constituintes da proteína e a sua localização face à estrutura secundária da proteína. Os resíduos estão identificados pelo seu número de ordem da cadeia polipeptídica da estrutura primária. Complementarmente, nos Anexos I-B e I-C encontra-se a designação de todos os aminoácidos que constituem a Transtirretina face ao seu número de ordem na cadeia polipeptídica da estrutura primária.

Localização	Resíduos (representados pela sua ordem na cadeia polipeptídica)
Cadeia β A	12, 13, 14, 15, 16, 17, 18
Cadeia β B	28, 29, 30, 31, 32, 33, 34, 35, 36
Cadeia β C	40, 41, 42, 43, 44, 45, 46, 47, 48, 49
Cadeia β D	54, 55
Cadeia β E	67, 68, 69, 70, 71, 72, 73
Cadeia β F	91, 92, 93, 94,95, 96, 97
Cadeia β G	104, 105, 106, 108, 109, 110, 111, 112
Cadeia β H	115, 116, 117, 118, 119, 120, 121, 122, 123
α-hélice	75, 76, 77, 78, 79, 80, 81, 82
Volta BC	37, 38, 39

Tabela 1 - Localização dos resíduos num monómero da Transtirretina.

3.4. Desnaturação Proteica

A desnaturação proteica é a alteração da estrutura tridimensional de uma proteína que leva à perda da sua função biológica. A desnaturação pode, ou não, ser reversível [Shortle 1996] e pode ser desencadeada de várias formas [Stefani 2004]: variações extremas de temperatura; radiações ultravioletas e ionizantes; variações de pH; detergentes; pressão ou agitação; entre outros.

A desnaturação proteica não altera a estrutura primária da proteína em questão, ou seja, as ligações que existem na estrutura primária continuam sempre a existir depois da desnaturação. Contudo, parte das restantes ligações são perdidas. De forma a clarificar o significado da desnaturação proteica, seguem-se alguns exemplos da ocorrência quotidiana deste fenómeno:

- A gelatina é um caso flagrante de desnaturação proteica, visto que a gelatina é colagénio desnaturado proveniente de ossos, peles e outros tecidos de animais;
- Um ovo, quando cozinhado, provoca o branqueamento e a solidificação da clara devido à desnaturação das proteínas contidas na clara (em particular da Albumina);
- No fabrico de queijos, a desnaturação é provocada no leite de modo a este coalhar.

Assim, a desnaturação nem sempre é um processo indesejado. A desnaturação proteica pode contribuir para a digestibilidade de algumas proteínas. Para os organismos animais, quanto mais próxima da estrutura primária a proteína estiver, mais fácil será a sua digestão [Sgarbieri 1996]. No caso da PAF, o processo de desnaturação que ocorre na Transtirretina não se trata, naturalmente, de um processo desejável, pois origina a doença pela acumulação de fibras de amiloide.

A Figura 7 [Rodrigues 2010] é uma representação do processo de desnaturação de um monómero da Transtirretina ao longo de 10 nano segundos de simulação. Nesta, é possível observar a sequência de estados de desnaturação com as respectivas perdas de ligações.

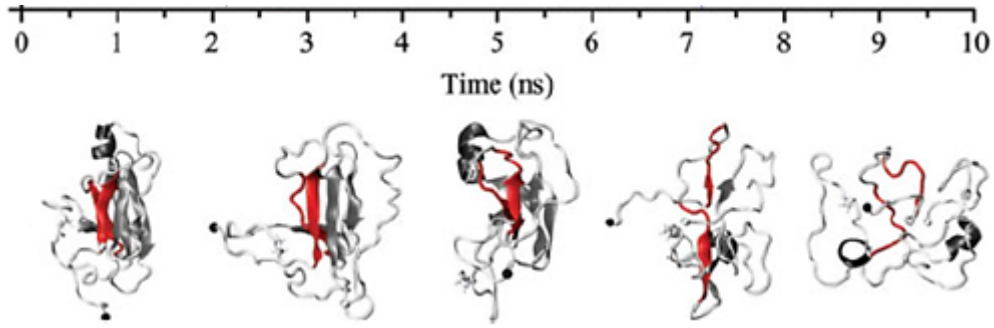


Figura 7 - Representação do processo de desnaturação de um monómero.

3.5. Simulações de Dinâmica Molecular

Nas simulações de dinâmica molecular, o comportamento do sistema molecular é obtido pela integração das equações de movimento de Newton e da função da energia potencial [Hoing 1999]. O resultado da simulação é uma série temporal de conformações, ou seja, revela a trajetória seguida por cada átomo de acordo com as leis de mecânica clássica de Newton. Desta forma, obtém-se uma trajetória que descreve as posições, velocidades e acelerações dos átomos e a sua variação ao longo do tempo. As simulações de dinâmica molecular para obtenção dos dados em estudo neste trabalho foram realizadas através do programa NAMD [Kalé 1999], usando a versão 27 do campo de forças CHARMM [Brooks 1983].

Neste trabalho, cada simulação representa 10 nano segundos do processo de desnaturação das variantes da TTR, sendo registados 10000 momentos de observação distintos por simulação. Em cada momento, mediram-se as distâncias relativas mínimas entre os 127 aminoácidos que compõem a proteína em estudo relativamente a todos os outros num dado monómero [Wei 2004]. Estas distâncias ficam representadas numa matriz.

Para cada simulação haverá 10000 matrizes, cada uma destas representa um momento de observação com 127 linhas e 127 colunas. Cada linha e cada coluna representam um aminoácido e o valor de cada célula indica a distância mínima que um aminoácido tem relativamente a outro no momento de observação correspondente. Os dados das simulações serão referidos como corridas. As corridas WT2, WT3, WT4, WT5 e WT6 correspondem às simulações da variante nativa e as corridas L55P1, L55P2, L55P3, L55P4 e L55P5 às simulações da variante mais amiloidogénica. Cada simulação representa uma série temporal de sequências de conformação de cada variante.

Todos os aspetos importantes relacionados com as simulações de dinâmica molecular estão relatados em [Brito 2004]. O objetivo deste trabalho é, não só expor todos os detalhes destas simulações, mas também enquadrá-las como um desafio de *Data Mining*. Este relata também que não foi explorada a exaustiva coleção de todas as possíveis propriedades relacionadas com as simulações e que ainda não é claro quais as interações físico-químicas que têm um papel determinante em cada fase do processo de desdobramento. Outro aspeto interessante deste trabalho é a ideia que este transmite em termos de recursos necessários para fazer as simulações. Para um *cluster* de 36 máquinas Pentium 4, o tempo necessário para obtenção de uma simulação é 60 dias. A trajetória produzida ocupa 5 GB.

A Figura 8 [Vilaça 2008] apresenta a caracterização das séries de dados das simulações de desnaturação da Transtirretina ao longo dos 10 nano segundos de simulação. Nesta é possível observar que cada *frame* – representada numa matriz - corresponde a um estado de desnaturação da proteína ao longo do tempo.

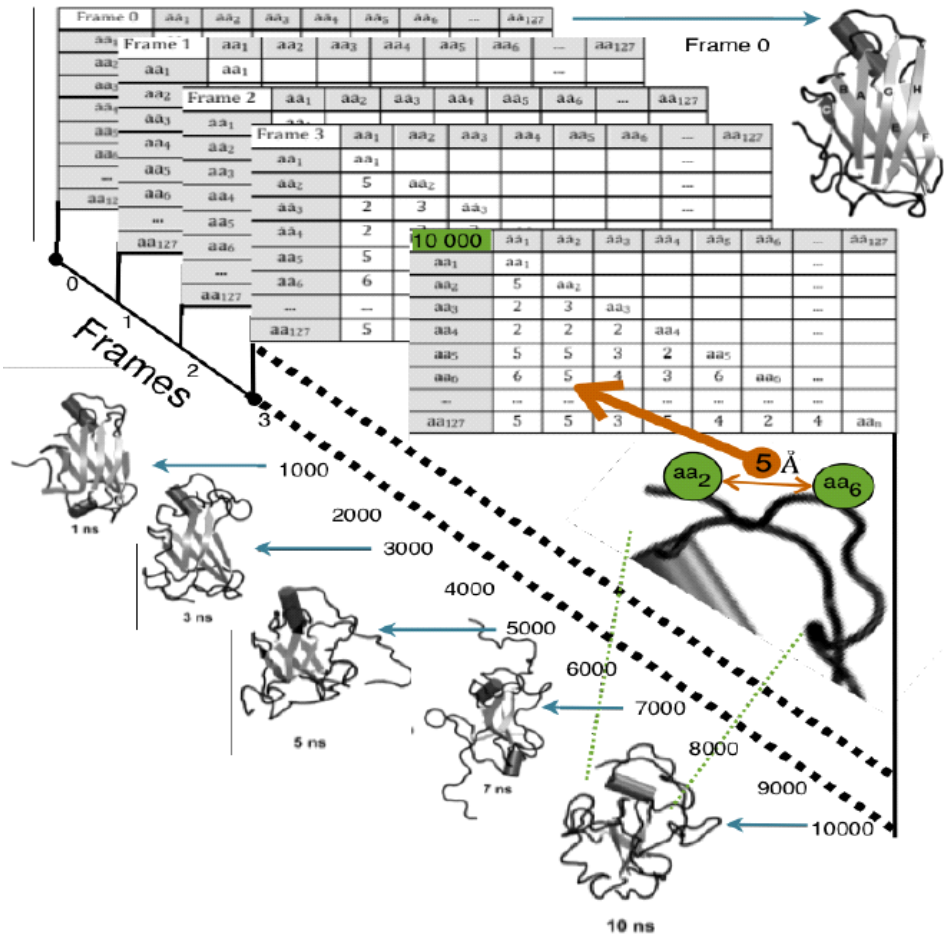


Figura 8 - Caracterização das séries de dados.

Ainda relativamente às simulações de dinâmica molecular, foi desenvolvido um sistema de *Data Warehousing* para correr e armazenar as simulações [Berrar 2005]. Foram integrados também serviços pré-definidos de *Data Mining*, nomeadamente detecção de padrões, *clustering* e classificação que trabalham sobre as simulações guardadas no *Data Warehouse*. Este projecto vem da necessidade de providenciar um repositório público que possibilite aos investigadores correr, reunir e partilhar os dados das simulações das proteínas, dado o vasto número de investigadores que pretendem estudar as proteínas, o esforço computacional para gerar os dados, os elevados volumes de dados e os diferentes tipos de análises que necessitam de realizar.

Capítulo 4

Estado da Arte

O objetivo deste capítulo é organizar todas as informações recolhidas em termos do estado da arte. Este compreenderá todos os trabalhos relevantes feitos até à data em que tenham sido consideradas as simulações de dinâmica molecular da Transtirretina, nomeadamente pela utilização de técnicas de *Data Mining*.

A Figura 9 apresenta um ciclo de estudos do processo de desnaturação da Transtirretina.

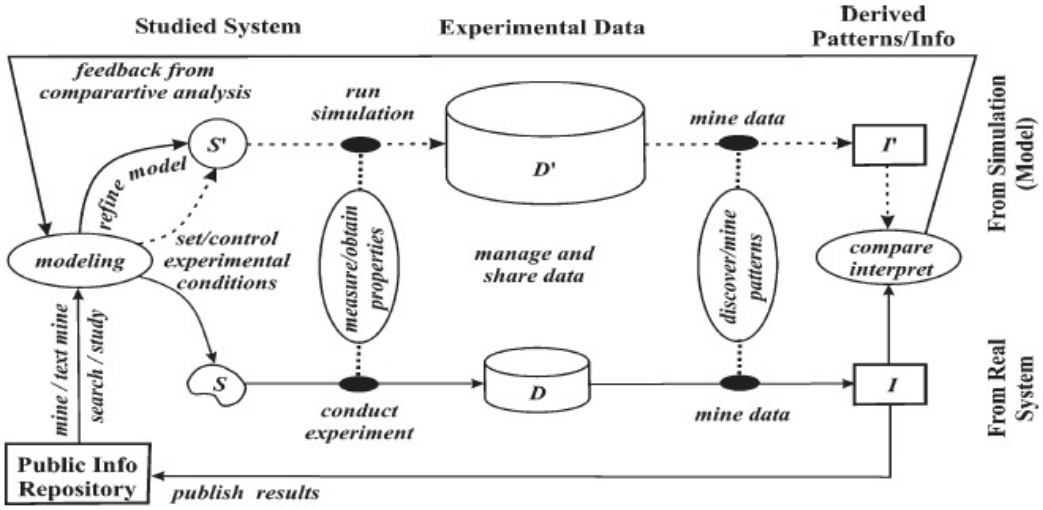


Figura 9 - Ciclo de estudos do processo de desnaturação da Transtirretina.

Em [Brito 2004], é claro o importante papel do *Data Mining* na análise de dados provenientes de simulações de dinâmica molecular. Tal como é possível observar na Figura 9, as técnicas de *Data Mining* fazem parte de um ciclo em que o seu papel é

derivar padrões e extrair informação de dados provenientes de experiências e de simulações de dinâmica molecular. Estes resultados são depois usados para refinar novos modelos que conduzirão a novas simulações e a novas experiências. O objetivo é que este ciclo termine quando tiverem sido expostas todas as informações sobre o processo de desnaturação da Transtirretina.

[Rodrigues 2010] relata uma análise feita às simulações de desnaturação proteica da Transtirretina cujo objectivo é comparar o comportamento da variante nativa e da Leu55Pro. Esta análise permitiu chegar a diversos resultados, entre estes:

- Os monómeros da variante Leu55Pro desnaturam mais cedo e em maior extensão que na variante nativa;
- A única α-hélice presente nos monómeros da TTR desnatura completamente na maioria das simulações Leu55Pro e mantém-se intacta nas simulações nativa;
- A variante Leu55Pro forma conformações propensas à agregação pelo deslocamento das cadeias C e D;
- A variante Leu55Pro apresenta uma severa perda das ligações da cadeia H, o que provoca a destabilização da folha β CBEF;
- A variante nativa forma conformações compatíveis com a agregação apenas tardiamente nas simulações e apenas quando o monómero se encontra extensivamente desdobrado.

Em [Rodrigues 2010] conclui-se que, em comparação com a variante nativa, a variante Leu55Pro apresenta uma maior probabilidade de formar conformações transientes compatíveis com a agregação e formação de amilóides.

Em [Azevedo 2005] é relatado que, usando de regras de associação, é possível identificar um grupo bem definido de resíduos que tendem a formar um *cluster* hidrofóbico aquando do processo de desdobramento da Transtirretina. Como conclusão do estudo foram encontrados 28 grupos resíduos. Estes conjuntos de resíduos têm duas características importantes: i) nunca se exporem ao solvente; ii) mostrarem uma relação espacial constante durante a maior parte do processo de desdobramento.

Outra utilização de regras de associação foi relatada em [Brito 2006]. Neste estudo, as regras de associação extraídas foram divididas em 4 tipos de acordo com as potenciais interações físico-químicas mais importantes e de acordo com padrões de similaridade em termos de exposição ao solvente. Como resultado, foram identificadas na variante nativa 98 regras, envolvendo 21 resíduos hidrofóbicos; e, na variante mais amiloidogénica, 14 regras, envolvendo 12 resíduos hidrofóbicos.

[Silva 2007] apresenta um estudo em que, usando *clustering* hierárquico, tendo como medida de similaridade a medida de correlação de Pearson, designada por TPCM, foram expostos certos padrões importantes e correlações entre resíduos. A utilização de *clustering* hierárquico como uma ferramenta útil foi também relatada em [Ferreira 2007], desta vez usando como medida de similaridade a variação da superfície molecular acessível ao solvente, designada por SASA, para cada aminoácido, de modo a construir um dendrograma (estrutura usada para avaliar um conjunto de dados). Os autores provaram que a sua abordagem foi útil na caracterização de um sub-cluster e na descoberta de novas relações entre resíduos distantes.

Noutro estudo [Ferreira 2006] propôs-se um algoritmo de extração de padrões em séries temporais. Este algoritmo recebe uma amostra representativa da base de dados de sequências temporais comparáveis e encontra padrões temporais – conhecidos por *motifs*. O algoritmo foi desenvolvido para ser usado em dados obtidos de simulações de desnaturação proteica. Contudo, este poderá ser usado noutros contextos. O algoritmo foi testado usando dados provenientes de simulações de dinâmica molecular da Transtirretina, tendo como objetivo a caracterização de eventos simultâneos deste processo de desnaturação. É razoável concluir que este algoritmo é bastante interessante e que deve ser tido em conta em problemas de descoberta de padrões em séries temporais.

Outro algoritmo de extração de padrões em séries temporais – MrMotif – é relatado em [Castro 2010]. O propósito deste algoritmo é poder ser usado em diferentes tipos de estudos em áreas distintas, desde as telecomunicações até à medicina. O algoritmo foi utilizado de modo a extrair os 10 padrões mais frequentes das simulações de dinâmica molecular da Transtirretina, com o tamanho de 64 pico segundos.

[Fernandes 2008] reporta o uso de duas técnicas da *Data Mining* para processar os dados provenientes das simulações do desdobramento da Transtirretina, nomeadamente

clustering hierárquico e regras de associação. Esta abordagem passou por dividir o processo em duas fases:

- Inicialmente os dados foram organizados em 2 partições, uma relacionada com as simulações da variante *Wild-Type* da Transtirretina e outra relacionada com a variante mais amiloidogénica (Leu55Pro).

- Em seguida, foi construída uma segunda partição usando 126 aminoácidos, excluindo o aminoácido da posição 55 e usando as 10 simulações de modo a organizar todos os aminoácidos (partição de consenso); depois, o *pivot* de cada classe foi usado para procurar eventos relevantes; os 127 aminoácidos foram usados para procurar por outro evento. No último passo, foram encontradas regras de associação para cada variante da Transtirretina (Leu55Pro e *Wild-Type*) usando os eventos relatados.

Em [Fernandes 2009] é apresentado um método de descoberta de conhecimento das simulações. Este método analisa a variação da distância dos átomos carbono-alfa, dos 127 aminoácidos que compõe a TTR, em relação ao centro de massa da proteína. Neste trabalho foram usadas as 10 simulações descritas anteriormente, *clustering* e regras de associação, de forma a caracterizar cada variante da TTR. Os resultados obtidos são uma contribuição para a discriminação entre as variantes amiloidogénicas e as não amiloidogénicas.

Em [Ferreira 2009] é apresentado um método de *clustering* chamado *Trajectoty Spatial Clustering*, cujo objectivo é identificar resíduos quem têm uma trajectória sincronizada durante o processo de desnaturação da TTR. Este método vem no sentido de analisar o comportamento e as relações entre os resíduos ao longo de uma simulação e tem como objectivo revelar pistas sobre quais resíduos têm um papel crucial no processo de desdobramento da proteína. O nome do método vem do facto do *clustering* ser determinado pela localização espacial do carbono-alfa de cada resíduo e é um método inspirado na técnica *Frequent Itemset Mining*. Este método provou ser útil na identificação de *clusters* de resíduos síncronos.

A GraphBrowser [Vilaça 2008] é uma ferramenta de exploração de fragmentos frequentes que foi desenvolvida especificamente para o caso de estudo das simulações de dinâmica molecular da desnaturação proteica da Transtirretina. Esta exploração é

feita pela visualização e análise dos fragmentos frequentes, nomeadamente ao nível da sua persistência e da relação com outros fragmentos.

A GraphBrowser encontra-se bem documentada, permite uma visualização clara dos fragmentos e permite a aplicação filtros, nomeadamente filtrar fragmentos que contêm certos resíduos ou com uma dada persistência. Contudo, esta apresenta algumas limitações:

- Como se trata de uma ferramenta *offline,* não se encontra ligada a nenhum repositório de simulações e de fragmentos e portanto não permite a partilha de informação;

- A ferramenta não produz nenhum relatório, o que seria interessante visto que da exploração de um certo fragmento poderiam ser organizados dados sobre este;

- A ferramenta é um pouco lenta mesmo correndo numa máquina com capacidade de processamento e de memória elevados, nomeadamente aquando do carregamento dos ficheiros de fragmentos. Além disto, a ferramenta contém alguns *bugs* que dificultam a sua utilização e para além de explorar os fragmentos através de filtros não os trabalha de forma a realizar conhecimento.

Estudar a GraphBrowser levanta questões acerca das limitações deste tipo de ferramentas e revela aspetos a serem pensados aquando do desenvolvimento de outra ferramenta com os mesmos objetivos e/ou que utilizem os dados provenientes das simulações de dinâmica molecular da desnaturação proteica da Transtirretina.

Capítulo 5

Extração de Fragmentos Frequentes

Sendo o objetivo principal da tese o desenvolvimento de uma ferramenta para a produção e visualização gráfica de sequências de sub-moléculas, usou-se *Graph Mining* para extrair fragmentos frequentes dos dados provenientes das simulações de desnaturação proteica da Transtirretina. Esta técnica é atualmente muito usada em bioquímica e em bioinformática envolvendo as mais variadas moléculas, uma vez que os fragmentos frequentes extraídos possuem informação implícita acerca do comportamento da substância envolvida.

Tal como é relatado em [Borgelt 2009], o desenvolvimento de um novo fármaco poderá levar muitos anos – normalmente entre 10 a 12 anos -, devido às normas de segurança envolvidas e aos custos envolvidos na sua investigação. Uma forma de diminuir o tempo de investigação é tentar melhorar a descoberta e a otimização da escolha de substâncias candidatas. Para isto, um grande número de potenciais moléculas são sujeitas a um determinado teste, como por exemplo, testar se são capazes de proteger uma célula humana contra um determinado vírus. O resultado é uma base de dados de informações sobre a atividade de compostos químicos. As técnicas de *Graph Mining* tornam-se essenciais quando o objetivo principal é identificar subestruturas frequentes em moléculas ativas e infrequentes em moléculas inativas.

No âmbito deste estudo, um fragmento é uma parte de um monómero da Transtirretina que é constituído por aminoácidos – referidos como resíduos – e pelas suas ligações – referidos como contactos. Procurar fragmentos frequentes em moléculas – as proteínas

são macromoléculas - pode ser obtido pela procura de subgrafos numa base de dados de grafos. Os subgrafos ocorrem nessa base de dados com uma certa percentagem mínima (frequência) ou com um número mínimo de ocorrências (suporte).

Este capítulo está organizado em cinco secções: na primeira são apresentadas as razões para a modelação do problema em grafos e como esta modelação é feita; na segunda secção é apresentada a seleção do método de extração; na terceira secção é apresentada a transformação das matrizes em grafos, preparação e filtragem dos dados; na quarta é apresentado o processo de extração e os fragmentos resultantes; e, por fim, a quinta secção conclui o capítulo.

5.1. Modelação do Problema em Grafos

Um grafo é uma estrutura matemática usada para guardar e representar informações sobre objectos e os seus relacionamentos. Os grafos estão presentes em vários domínios e conseguem representar diversos conceitos, problemas e sistemas do mundo real. De facto, a Teoria de Grafos é aplicada em larga escala nas áreas da Informática, da Engenharia de Sistemas, da Economia, da Sociologia (por exemplo nas redes sociais), da Bioquímica, entre outras.

Matematicamente, um grafo G é representado numa estrutura $G\ (V, A, f)$ em que V é o conjunto de vértices, A o conjunto de arestas que une vértices e que conta com uma função f de mapeamento de arestas em vértices $f: A -> V \times V$. O mapeamento é obtido da seguinte forma: uma aresta a de A tem um relacionamento representado como $f(a) = (v_i, v_j)$ em que v_i e v_j são vértices de V. Neste caso, v_i e v_j como partilham uma aresta são chamados de adjacentes.

A Figura 10 apresenta o exemplo de um grafo G com 6 vértices e 6 arestas, que tem um conjunto de vértices $V = \{1,2,3,4,5,6\}$, um conjunto de arestas $A = \{a, b, c, d, e, f\}$, e uma função f, tal que $f(a) = (1,2)$, $f(b) = (1,5)$, $f(c) = (2,3), f(d) = (2,5), f(e) = (3,4), f(f) = (4,5), f(g) = (4,6)$.

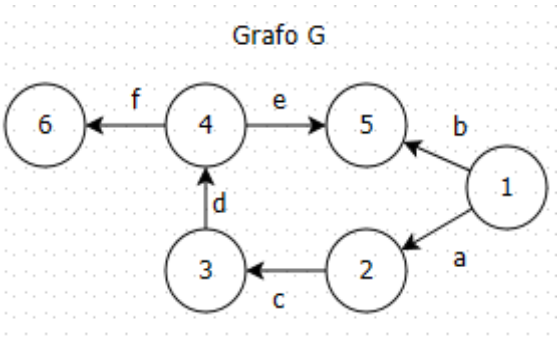


Figura 10 - Exemplo de um grafo.

Um grafo poderá ser também representado na forma matricial, nomeadamente através de uma matriz de adjacências. Dado um grafo G com n vértices, este poderá ser representado numa matriz $n \times n$. O conteúdo de cada célula da matriz depende da definição do próprio grafo: num grafo sem pesos nas arestas, uma célula contém 1, se os nós correspondestes à célula são adjacentes, e 0 no caso contrário; num grafo com peso nas arestas, o valor das células é o valor do peso das arestas. Caso o grafo seja não orientado, a sua matriz de adjacência é simétrica. Nos grafos orientados esta característica não se verifica. Continuando com o exemplo da Figura 10, a matriz de adjacência correspondente ao grafo G seria a matriz apresentada na Figura 11.

Grafo G	1	2	3	4	5	6
1		a			b	
2			c			
3				d		
4					e	f
5						
6						

Figura 11 - Matriz de adjacência.

As simulações de dinâmica molecular produzem como resultado matrizes de adjacência em que, para cada instante simulado, cada célula da matriz contém a distância mínima entre os aminoácidos que compõem a Transtirretina.

Para este estudo, os dados provenientes das simulações de dinâmica molecular da desnaturação da Transtirretina serão modelados numa sequência de grafos não orientados e sem pesos nas arestas. Basta apenas registar para cada aminoácido da proteína – ou seja, para cada vértice - a quais é que este se encontra ligado. São também

grafos simples, na medida que entre dois vértices só existe no máximo uma aresta, e não têm laços – não tem arestas a ligar um vértice a si próprio. Dado que as simulações produzem como resultado matrizes de adjacência, a transformação destas matrizes em grafos é feita de forma bastante simples e direta.

Nos estudos de compostos químicos em que se recorre a *Graph Mining* é muito comum modelar-se os grafos também na sua forma canónica [Borgelt 2009]. Esta modelação descreve um grafo como uma palavra que consegue restaurar totalmente a sua estrutura inicial.

5.2. Escolha do Método de Extração de Subgrafos Frequentes

A escolha do método de extração de subgrafos frequentes tem de ser ponderada cuidadosamente devido, principalmente, à dimensão dos dados provenientes das simulações e a sua consequente exigência em termos de computação. Dada como medida de extração o suporte mínimo – medida mais utilizada em *Graph Mining* –, são extraídos da base de dados subgrafos que respeitem essa medida. Para uma base de dados D, o suporte Sup de um subgrafo S_g é dado por:

- $Sup\ (S_g) = número\ de\ grafos\ de\ D\ que\ contém\ o\ subgrafo\ /$ $número\ total\ de\ grafos\ de\ D.$

Esta medida tem uma propriedade bastante importante: propriedade antimonótona [Washio 2003], segundo a qual o suporte de um subgrafo não pode ser menor do que o suporte de nenhum dos seus supergrafos.

- $G(S_y)\ subgrafo\ G(S_x) \rightarrow \quad Sup\ (G(S_x)) \leq Sup\ (G(S_y))$.

Considerar esta propriedade aquando da extração de subgrafos frequentes permite podar certos caminhos de procura, reduzindo assim o esforço de computação associado.

Outra característica importante de um subgrafo extraído é se este é ou não fechado. Um grafo é fechado se não existir um supergrafo deste com o mesmo suporte. Considerar este filtro permite reduzir o número de subgrafos extraídos e, além disso, faz com que sejam descartados subgrafos com informação redundante face a outros.

O método de extração de fragmentos terá que cumprir os seguintes requisitos:

- Suportar as elevadas dimensões dos dados das simulações – é necessário garantir que todos os 10000 momentos de observação poderão ser usados para extrair fragmentos;
- Executar em tempo útil, permitindo trabalhar com níveis de suportes interessantes – não é desejado que apenas seja possível extrair fragmentos a altas frequências pois poderão haver fragmentos interessantes presentes a frequências mais baixas;
- Permitir a alteração de parâmetros interessantes como o tamanho mínimo dos fragmentos extraídos, de forma a reduzir o número de fragmentos extraídos. Torna-se interessante eliminar os fragmentos cujo tamanho, medido em número de contactos, é baixo;
- De preferência permitir filtrar fragmentos fechados e podar considerando a propriedade antimonótona pelas razões anteriormente apresentadas;
- Ter documentação associada ou o código disponível de modo a facilitar a sua utilização.

A melhor forma de escolher um algoritmo de extração de subgrafos frequentes é testar vários algoritmos para a base dados em estudo. De facto, os autores deste tipo de algoritmos, para provar a eficiência de um algoritmo, apresentam *benchmarks*. Os *benchmarks* demostram que, para certas bases de dados, os resultados obtidos pelo algoritmo proposto são os melhores.

5.2.1. ParMol

O ParMol [Meinl 2006] é uma biblioteca onde estão implementados quatro dos mais populares algoritmos de extração de subgrafos frequentes: Gaston (GrAph/Sequence/Tree extractiON) [Nijssen 2004a] [Nijssen 2004b], GSpan (Graph–Based Substructure Pattern Mining) [Yan 2002], MoFa (Molecule Fragment Miner) [Borgelt 2002] e FFSM (Fast Frequent Subgraph Mining) [Huan 2003].

O projeto ParMol foi iniciado em 2004 e teve como objetivo inicial a implementação e a comparação dos 4 *graph miners* mais conhecidos ao nível do tempo de processamento e de memória consumida. O ParMol usa a mesma estrutura computacional de forma a garantir uma comparação justa entre *miners*.

Os algoritmos de *Graph Mining* recorrem normalmente à construção de um reticulado em que este é percorrido em profundidade, ou em largura. As ocorrências de um subgrafo num grafo são guardadas numa lista de ocorrências. Outros algoritmos como, por exemplo, o FFSM trabalham com matrizes de adjacência e fazem testes de isomorfismo de grafos, guardando os resultados em lista de *embeedings*. Estes testes de isomorfismo são normalmente feitos considerando uma ordem lexicográfica entre os nós de um grafo de modo a evitar duplicados.

O ParMol é uma biblioteca bem documentada, o código está disponível sob a *GNU General Public License* (GPL) e tem a possibilidade de alteração de vários parâmetros inerentes à execução dos algoritmos. Estes parâmetros são o tamanho máximo e mínimo dos fragmentos extraídos, a frequência mínima e máxima dos fragmentos extraídos e a possibilidade de podar fragmentos considerando a propriedade antimonótona. Para alguns algoritmos (gSpan e MoFa) permite ainda extrair apenas os fragmentos fechados. Outra característica muitíssimo importante do ParMol é o facto de permitir *multithreading*.

A Figura 12 [Meinl 2006] apresenta dois gráficos que, para cada algoritmo que compõe o ParMol, apresenta a variação do suporte mínimo face ao tempo de processamento e a variação do suporte mínimo face à memória exigida. Os dados foram verificados pelos autores da biblioteca, utilizando uma plataforma computacional composta por processador Dual Itanium 2 PC a 1.3 GHz, 10GB de RAM e espaço de memória reservado de 8GB. A série estudada é composta por 42689 grafos com 27 arestas cada (em média).

Pela análise da Figura 12 pode-se verificar que, para a série estudada:

- O algoritmo mais rápido é o Gaston, sendo também o que mais consome memória;

- Quando o suporte mínimo baixa, para o MoFa, o tempo de computação aumenta quase exponencialmente, sendo que os restantes mantêm-se com um tempo praticamente constante;

- Em termos de memória, o gSpan apresenta melhores resultados, seguido do MoFa.

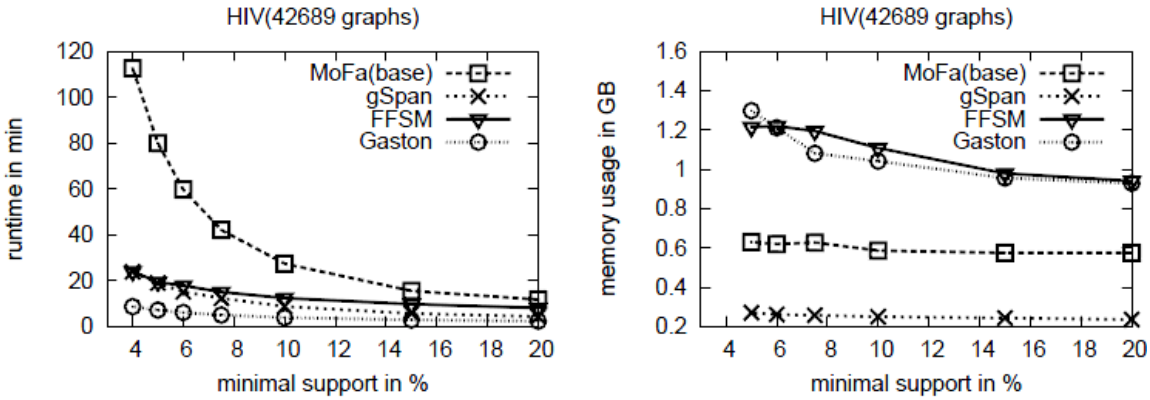


Figura 12 - Variação do suporte mínimo face ao tempo de processamento e memória dos vários algoritmos do ParMol.

Os valores apresentados foram calculados tendo em conta sempre a mesma base de dados. Contudo, o desempenho dos algoritmos poderá ser afetada por outros fatores para além do suporte mínimo. Um destes factores é o número de nós e de arestas da série. Outro estudo [Wörlein 2005], em que participaram parte dos autores do ParMol, resulta nas seguintes conclusões:

- Relativamente ao número de grafos da base de dados, estes tendem a aumentar, de forma linear, o tempo de computação. No caso do MoFa, o tempo aumenta num fator mais elevado do que os restantes;

- Em termos do número de arestas, para todos os algoritmos, com exceção do MoFa, o tempo de computação aumenta ligeiramente face ao aumento de arestas. Para o MoFa, o tempo de computação aumenta de forma proporcional com um fator elevado.

Estes dados encontram-se nas Figura 13 e Figura 14. A Figura 13 apresenta um gráfico que mostra, para cada algoritmo do ParMol, a variação do tempo de execução face ao número de grafos e a Figura 14 um gráfico com a variação do tempo de execução face à densidade de arestas.

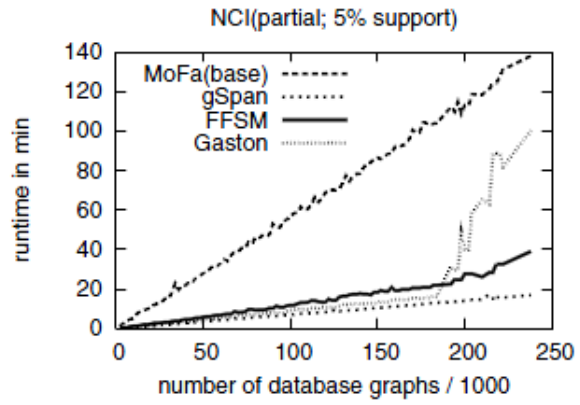


Figura 13 - Variação do tempo de execução face ao número de grafos.

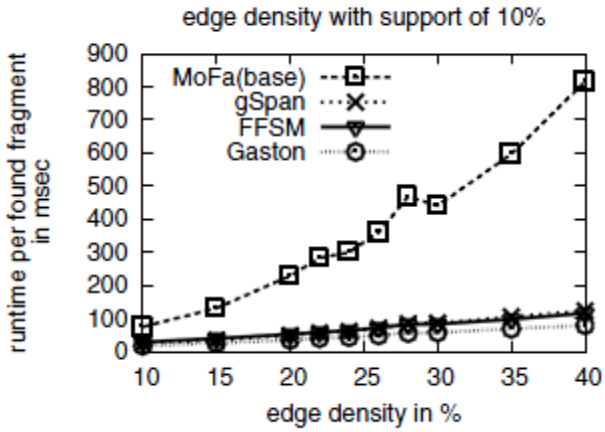


Figura 14 - Variação do tempo de processamento por fragmento face à densidade de arestas.

Usando a biblioteca ParMol, foi feito um estudo [Vilaça 2008] comparando os 4 algoritmos, usando os dados extraídos das simulações de dinâmica molecular da Transtirretina anteriormente descritas. Como resultado do estudo, observa-se que o melhor desempenho é registado pelo Gaston, apesar de necessitar de mais memória, logo seguido do MoFa e, por último, o GSpan (embora este seja o menos exigente em requisitos de memória). É possível concluir que os requisitos das séries de dados da Transtirretina, principalmente no que toca ao número de contactos, são mais exigentes do que as séries em [Wörlein 2005]. Isto deve-se a apesar de esta ser mais extensa em número de grafos (42 689 grafos), em contrapartida, a média do número de arestas é claramente menor (27) do que o número de arestas das simulações da Transtirretina (entre 163 e 197 arestas).

5.2.2. GraphSig

GraphSig [Ranu 2005] é uma abordagem que propõe reduzir o tempo de computação inerente ao processo de extração de subgrafos frequentes em grandes volumes de dados. Para obter este resultado, a estratégia passa por analisar os dados não apenas pelo seu suporte mas também pela sua significância estatística.

A Figura 15 apresenta uma esquematização da abordagem GraphSig.

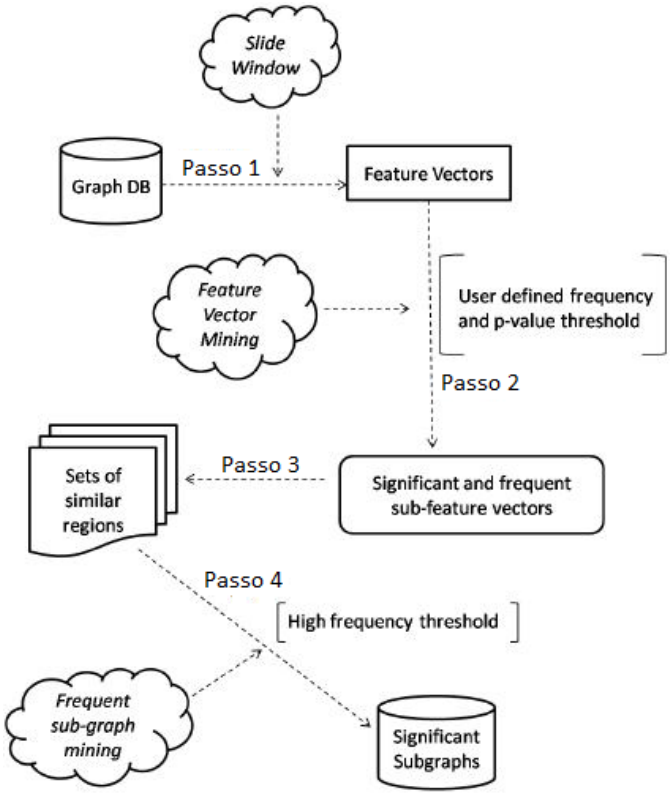


Figura 15 - Abordagem GraphSig.

A abordagem GraphSig segue os seguintes passos:

1) Inicialmente são derivados vetores de *features* da base de dados de grafos: dada uma base de dados de grafos G, para cada nó de cada grafo $g \in G$ haverá um vetor de *features* que o representa;

2) As *features* são agrupadas em conjuntos conforme o nó em que derivaram. A Figura 16 apresenta os vetores obtidos para quatro grafos $G1, G2, G3\ e\ G4$ para o nó a.

Vector	a-b	a-d	a-e	a-f	b-c	b-d	c-e	c-f	d-f
G_1	2	0	3	0	1	1	0	0	0
G_2	4	0	0	0	2	1	0	0	1
G_3	3	0	0	0	1	2	1	1	0
G_4	0	3	0	3	0	0	0	0	2

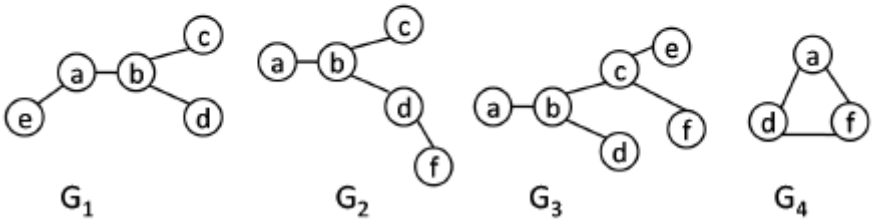


Figura 16 - Exemplo dos vetores obtidos para o nó do tipo *a* dos grafos G1-G4.

Em seguida, dentro de cada conjunto de *features* são identificados os vetores de *sub-features* fechados. Um vetor de *sub-features* é identificado pela presença de um conjunto de *features* sem valores nulos. Estes vetores são fechados pois não há nenhum vetor *super-feature* com um suporte igual ao deste. A Figura 17 apresenta um exemplo de um vetor de *sub-features* fechado.

Vector	a-b	a-d	a-e	a-f	b-c	b-d	c-e	c-f	d-f
G_1	2	0	3	0	1	1	0	0	0
G_2	4	0	0	0	2	1	0	0	1
G_3	3	0	0	0	1	2	1	1	0
G_4	0	3	0	3	0	0	0	0	2

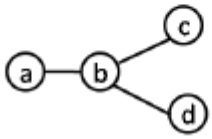


Figura 17 - Exemplo de um vetor de sub-*features* fechado.

Dentro de cada conjunto, são eliminados os vetores de *sub-features* que não respeitem os valores de *p-value* e de frequência dentro de cada conjunto;

3) Para cada um dos vetores de *sub-features* fechados que foram encontrados no passo anterior, é formado um grupo com os grafos da base de dados original que o contém;

4) Por fim, dentro de cada grupo de vetores, são então extraídos os subgrafos frequentes com um *threshold* alto.

A implementação desta abordagem encontra-se disponível para uso académico. Porém a documentação associada é insuficiente e o código também não está disponível. Não

existem assim garantias quanta à correção desta implementação. Além disto, o formato de *input* desta implementação requer que, para este caso de estudo em particular, o *input* tenha 7GB de tamanho por corrida – enquanto no ParMol o tamanho máximo encontrado num ficheiro de *input* são 12MB. Este tamanho deve-se ao facto de ser necessário associar a todos os grafos uma lista de identificação dos seus vértices. Esta identificação acaba por ser redundante, pois, para este caso de estudo, esta identificação é sempre a mesma, dado que se trata sempre da mesma molécula apenas com novas ligações criadas e/ou desfeitas. Adicionalmente, o *output* produzido não contém a lista de persistência dos fragmentos obtidos e não há estudos sobre o seu desempenho em termos de tempo de processamento e de memória.

5.2.3. Outros métodos

Existem outros métodos de extração de subgrafos frequentes. No entanto, os métodos mais utilizados e, portanto, mais explorados e documentados já se encontram implementados no ParMol. O GraphSig é o único método alternativo que considera a significância dos subgrafos frequentes extraídos e cuja implementação está disponível. Assim não se torna necessário considerar outras alternativas.

5.2.4. Conclusão

Dadas as limitações existentes em termos de documentação, pelo *output* não produzir uma lista de persistência, e por não haver estudos sobre o desempenho ou correção da implementação, o GraphSig não será considerado como opção para a extração de subgrafos frequentes. No entanto, é de realçar a sua abordagem, uma vez que poda os fragmentos obtidos pela sua significância estatística, e, por esta razão, os fragmentos obtidos são mais interessantes.

A escolha recai então sobre o ParMol. Trata-se de uma biblioteca bem documentada, o código está disponível sob a GPL e tem a possibilidade de alteração de vários parâmetros inerentes à execução dos algoritmos. Uma característica fulcral do ParMol é o facto de esta permitir *multithreading* e, portanto, tirar partido da máquina disponível, uma vez que a extração de fragmentos frequentes em grandes volumes de dados poderá ser um processo custoso em termos de processamento e memória. Como em [Vilaça 2008], o Gaston foi o algoritmo que apresentou melhor desempenho, este é o algoritmo escolhido para a extração de subgrafos frequentes.

Para o presente estudo, dentro dos algoritmos que compõe o ParMol, a estratégia utilizada pelo Gaston para extrair fragmentos frequentes face aos outros algoritmos não é por si só relevante. O que realmente é pertinente é o seu desempenho em termos de tempo de processamento e de memória exigida. Contudo, a próxima secção apresenta uma abordagem geral a este algoritmo.

5.3. Gaston

A procura de fragmentos frequentes compreende a construção e exploração de um reticulado do espaço de procura. Todos os algoritmos disponíveis na biblioteca ParMol fazem a travessia do espaço *Depth First*. Isto é, primeiro em profundidade e só depois em largura. Este tipo de procura tem como vantagem consumir menos memória pois o número de lista de aparências que precisa guardar é proporcional à profundidade do reticulado (isto é, ao tamanho do maior grafo) [Wörlein 2005]. A Figura 18 [Meinl 2006] apresenta um exemplo de um reticulado de procura para uma base de dados com dois grafos. Em cada nível, é acrescentada uma nova ligação a um subgrafo, partindo assim de um subgrafo vazio à base de dados.

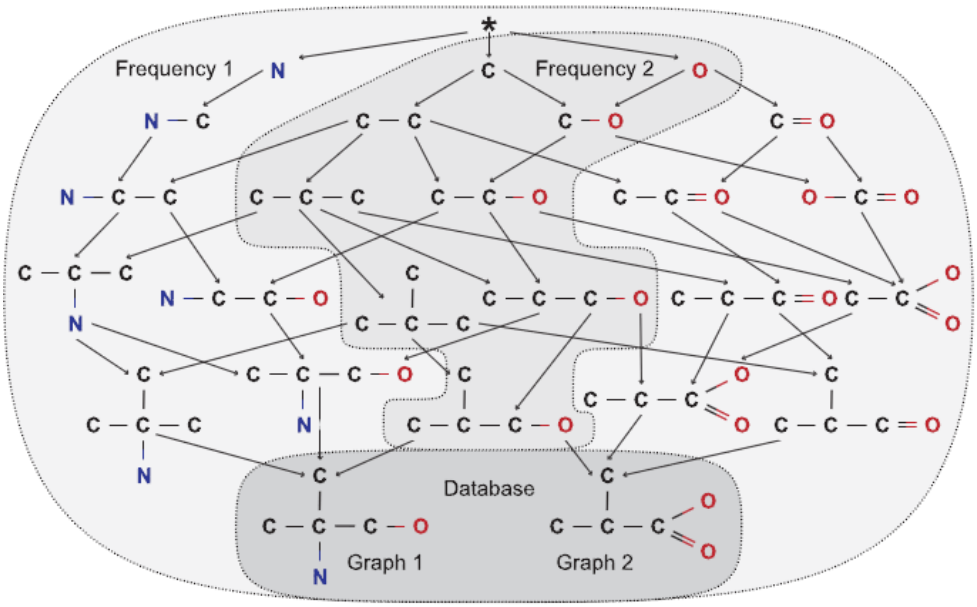


Figura 18 - Exemplo de um reticulado.

O Gaston tira partido da propriedade antimonótona. Ou seja, o reticulado pode ser podado pela frequência, pois se um fragmento é infrequente, um superfragmento deste

também o é. Em termos de fragmentos fechados, no ParMol, a extração considerando apenas estes fragmentos só é possível para os algoritmos gSpan e MoFa.

O algoritmo Gaston divide o seu processamento em 3 etapas em que primeiro considera os caminhos simples, depois as árvores sem ciclos e, por fim, os subgrafos cíclicos [Nijssen 2004b]. Esta estratégia promete extrair subgrafos de forma mais eficiente pois normalmente muitos dos subgrafos cíclicos são podados por serem supergrafos infrequentes [Meinl 2006]. A Figura 19 apresenta esta divisão em 3 etapas: por caminhos (*Paths*), por árvores sem ciclos (*Free Trees*) e por subgrafos cíclicos (C*yclic graphs*).

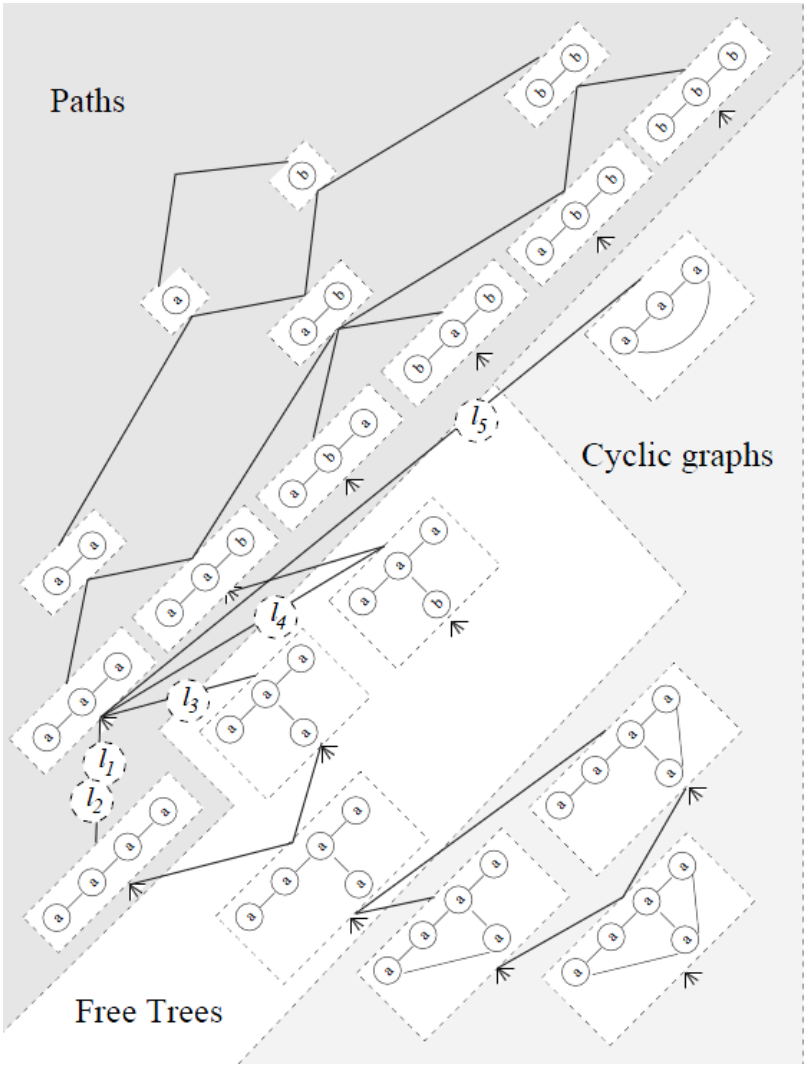


Figura 19 - Etapas do algoritmo Gaston.

Este algoritmo conta também com lista de *embeddings* para cada fragmento. Um *embedding* é um mapeamento entre os nós e as arestas de um fragmento e os nós e

arestas de um grafo em que o fragmento ocorra [Meinl 2006]. A utilização destas listas é bastante importante, dado que permite calcular a frequência de um fragmento contando apenas as ocorrências do isomorfismo na lista de *embeddings*. Da mesma forma, também é possível encontrar todas as extensões de fragmentos, procurando novos isomorfismos apenas na vizinhança dos isomorfismos do grafo-semente. Esta característica melhora consideravelmente o desempenho em bases de dados de grafos extensas, mas apresenta a desvantagem de maiores requisitos de memória [Meinl 2006].

5.4. Transformação das Matrizes em Grafos, e Preparação e Filtragem dos Dados

As matrizes de adjacência associadas às simulações foram tratadas de forma a poderem ser lidas como *input* do ParMol. Para executar esta transformação foi desenvolvido um *parser* que conta com os seguintes filtros:

- Considerar ou não os contactos nativos. Os contactos nativos são aqueles que estão presentes no início do processo de desnaturação, i.e., são aqueles contactos que aparecem na primeira *frame* das simulações. Os contactos não nativos são todos os restantes.

  Na transformação, para cada simulação serão considerados os dois tipos de *input*: incluindo contactos nativos e apenas com contactos não nativos;

- Filtrar os contactos entre pares de resíduos cuja distância entre este é inferior ou igual a um certo valor. Este valor é um parâmetro do *parser*.

  Na transformação são apenas considerados os contactos entre resíduos cuja distância entre estes é inferior ou igual a 4.2 Å (Angström), pois é esta a distância mínima necessária para se poder considerar que dois resíduos estão em contacto [Hubbard 1993];

- Filtrar os contactos entre pares de resíduos cujo número de ordem na cadeia polipeptídica dista pelo menos um certo valor. Este valor é um parâmetro do *parser*.

  Na transformação, serão desconsiderados os contactos entre pares de resíduos (i,j) com $j \leq i+3$ pois interagem como resultado da ligação da cadeia

polipeptídica [Shmygelska 2005]. Assim, apenas serão considerados os contactos cujo número de ordem na cadeia polipeptídica dista de, pelo menos, 4 posições;

- Filtrar os contactos a partir de um ficheiro.

  Na transformação não foi utilizada esta opção;

- Considerar apenas um determinado intervalo de *frames*.

  Na transformação não foi utilizada esta opção.

O *parser* demora cerca de 18 segundos a correr numa plataforma computacional Quad-Core i5-460M, com velocidade de processamento de 2.53 GHZ com Turbo Boost até 2,80GHZ e 4 GB de memória RAM. O *parser* gera um ficheiro de integração no ParMol de uma dimensão bastante reduzida – em média, 5 MB não considerando os contactos nativos e 11.2 MB considerando os contactos nativos.

Para outros estudos, o *parser* pode ser utilizado com parâmetros diferentes.

5.5. Execução do ParMol e Organização dos Fragmentos Obtidos

De forma semelhante ao que aconteceu em [Vilaça 08], extraiu-se fragmentos das corridas usando o algoritmo Gaston. Esta execução ocorreu numa plataforma computacional Quad-Core i5-460M, com velocidade de processamento de 2.53 GHZ com Turbo Boost até 2,80GHZ e 4 GB de memória RAM usando os seguintes parâmetros:

- Tamanho máximo dos fragmentos: 50 contactos;
- Tamanho mínimo dos fragmentos: 3 contactos;
- Frequência mínima: 70%;
- Frequência máxima: 100%;
- Método de serialização dos fragmentos: DotGraphParser;

Os dados relativos a estas extrações estão organizados na Tabela 2 que relaciona cada corrida face ao tempo de computação, ao número de fragmentos extraídos e ao tamanho máximo dos fragmentos extraídos.

Corrida	Tempo de computação (em segundos)	Número de fragmentos extraídos	Tamanho máximo dos fragmentos extraídos (em número de arestas)
L55P1	6	4	4
L55P2	8	55	6
L55P3	900	56554	17
L55P4	105	3913	14
L55P5	41	1990	13
WT2	73	4717	13
WT3	4089	61047	19
WT4	236	17958	14
WT5	248	15820	16
WT6	68	4308	11

Tabela 2 - Tabela comparativa de tempos de computação de cada corrida, considerando contactos nativos, face ao número de fragmentos extraídos e o tamanho máximo destes, usando o algoritmo Gaston, com frequência mínima igual a 70%.

Analisando os dados da Tabela 2, é significativa a diferença entre o número de fragmentos extraídos em cada corrida e o seu tamanho. Por exemplo, na corrida L55P1 foram extraídos apenas 4 fragmentos, enquanto na WT3 foram extraídos 61047 fragmentos. Em termos de tamanho, na corrida L55P1, o tamanho máximo dos fragmentos encontrados é 4 arestas, enquanto na corrida WT3 é 19 arestas.

Após uma análise mais detalhadas aos fragmentos extraídos, tiraram-se as seguintes conclusões:

1. Os fragmentos poderiam ser organizados em grupos de acordo com a zona da proteína em que se encontram;

2. Não é necessário extrair fragmentos com um número de contactos baixo já que estes são sempre subfragmentos de outros extraídos. O número de fragmentos

com estas características é muito elevado, representando às vezes mais do que 90% dos fragmentos extraídos.

As conclusões obtidas permitem partir para uma metodologia de organização de fragmentos com o intuito de elevar a qualidade da informação extraída. Esta organização passa por agrupar os fragmentos conforme a zona da proteína em que se encontram (1) e retirar os fragmentos de menor dimensão que são subgrafos de outros já extraídos (2). Em cada grupo, é sabido a que zonas da proteína pertencem e que o tamanho destes varia no máximo em 4 contactos. Esta divisão dos fragmentos extraídos em grupos levou ao desenvolvimento de uma outra pequena aplicação de forma a filtrar e dividir os fragmentos extraídos.

No anexo II encontra-se uma tabela que relaciona os grupos de fragmentos face ao número de fragmentos presentes, ao tamanho máximo e mínimo encontrado (calculado em número de arestas) e à frequência média. Os dados presentes na tabela mostram que o número de fragmentos disponíveis diminui razoavelmente. As Figura 20 e 21 mostram a localização dos grupos de fragmentos em cada monómero da Transtirretina em que é considerada a localização face à estrutura secundária da proteína.

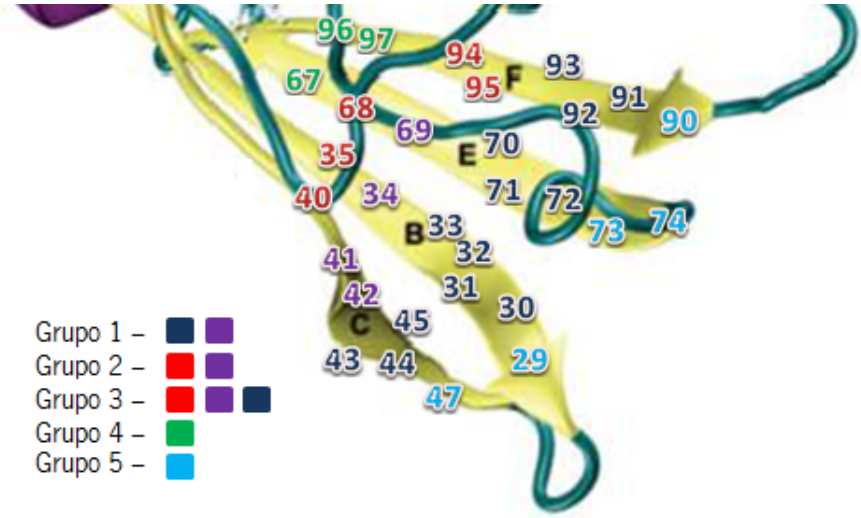


Figura 20 - Organização dos grupos de fragmentos extraídos das cadeias C,B,E e F.

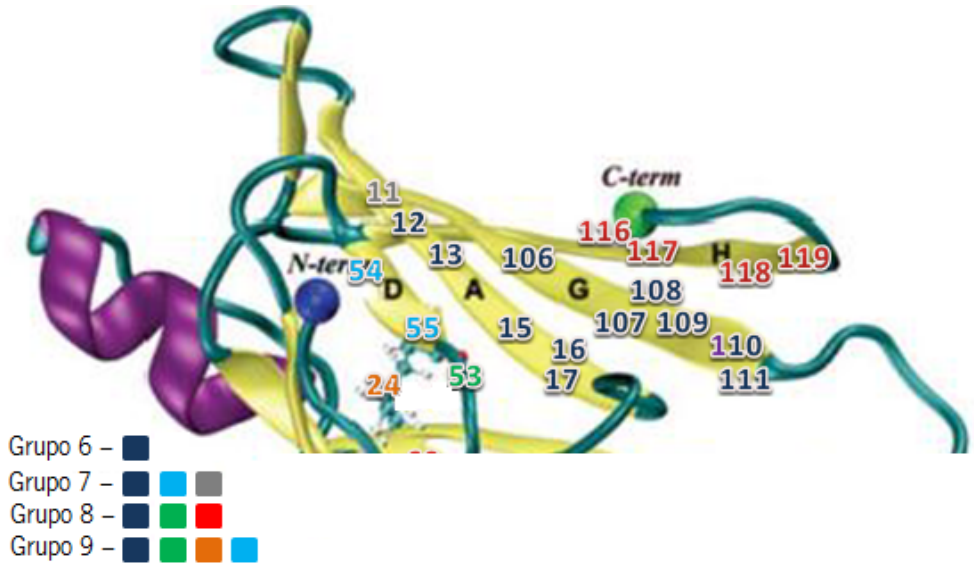


Figura 21 - Organização dos grupos de fragmentos extraídos das cadeias D, A, G e H.

De forma a cobrir fragmentos que só ocorrem a baixa frequência, foram extraídos mais fragmentos, desta vez usando como frequência mínima 25% e máxima 50%. Foram usadas as corridas que não consideram os contactos nativos pois são as únicas que permitem esta computação. Esta extração considerou os seguintes parâmetros:

- Tamanho máximo dos fragmentos: 20 contactos;
- Tamanho mínimo dos fragmentos: 2 contactos;
- Frequência mínima: 25%;
- Frequência máxima: 50%;
- Método de serialização dos fragmentos: DotGraphParser.

Os dados relativos a estas extrações estão organizados na Tabela 3 que relaciona cada corrida face ao tempo de computação, ao número de fragmentos extraídos e ao tamanho máximo dos fragmentos extraídos.

Tal como é possível observar na Tabela 3, a extração destes fragmentos não nativos é uma computação bastante rápida, cujo número de fragmentos obtidos é significativamente menor do que resultantes a altas frequências. Isto deve-se ao facto das corridas sem os contactos nativos apresentarem uma dimensão mais reduzida. Ao contrário dos fragmentos extraídos a alta frequência, estes não serão divididos em

grupos, pois em cada corrida o número de fragmentos extraídos é baixo, o que por si só já facilita a escolha e posterior utilização dos fragmentos.

Corrida	Tempo de computação (em segundos)	Número de fragmentos extraídos	Tamanho máximo dos fragmentos extraídos (em número de arestas)
L55P1	8	14	3
L55P2	5	13	3
L55P3	6	142	5
L55P4	5	59	7
L55P5	6	118	6
WT2	4	4	2
WT3	6	27	4
WT4	5	59	3
WT5	4	4	2
WT6	4	21	3

Tabela 3 - Tabela comparativa de tempos de computação de cada corrida, não considerando contactos nativos, face ao número de fragmentos extraídos e o tamanho máximo destes, usando o algoritmo Gaston, com frequência entre 25% e 50%.

5.6. Conclusão

A transformação dos dados provenientes das simulações da Transtirretina em grafos é um passo crucial para o desenvolvimento de uma ferramenta que contempla a produção e visualização gráfica de caminhos traçados a partir de fragmentos frequentes do processo de desdobramento da Transtirretina. A escolha do ParMol, e mais especificamente do algoritmo Gaston, para extração dos fragmentos, pelos motivos anteriormente referidos, revela-se a acertada pois a extração tornou-se num processo simples e relativamente rápido.

Dentro dos fragmentos extraídos a altas frequências, analisando a lista de persistência de qualquer um dos fragmentos, é possível observar que estes têm uma presença bastante oscilante. Estes desaparecem e aparecem nas *frames* muitas vezes ao longo da simulação em que, basicamente, o fragmento nunca se desfaz completamente. Este vai perdendo e ganhando um pequeno número de ligações dentro do seu núcleo de resíduos, permanecendo presente até ao fim da corrida. No caso dos fragmentos extraídos a

baixas frequências, estes tendem a não oscilar muito em termos de persistência. Ou seja, quando aparecem na corrida são persistentes até desaparecem totalmente.

Uma vez que se encontra documentada não só a localização de cada grupo de fragmentos na estrutura da proteína, mas também as suas dimensões e frequências médias esperadas dentro de cada grupo para cada corrida, torna-se uma interessante vantagem a organização de fragmentos por grupos.

Neste ponto, conhecidos já os fragmentos extraídos das simulações, é possível partir para a definição de caminhos.

Capítulo 6

Caminhos

É interessante saber o que acontece a um fragmento em todos os momentos de observação, ou seja, como este evolui ao longo do tempo. Esta evolução poderá acontecer por acrescento de contactos (e também de resíduos) e/ou por perda de contactos (e também de resíduos). A trajetória de evolução de um fragmento ao longo do tempo será referida como caminho.

Os caminhos poderão ser traçados tendo em conta várias restrições. Estas restrições são relativas à forma como o fragmento evolui, mais concretamente, em termos dos resíduos que poderão ser usados para expandir o fragmento. As restrições poderão ser feitas pelo fornecimento de uma lista de resíduos ou de um fragmento final. Outro conceito importante relacionado com os caminhos é o de caminho de fragmentação. Este tem como objetivo mostrar como um dado fragmento não nativo se fragmentou. Ou seja, como este aparece na corrida desde a sua primeira ocorrência – primeira *frame* em que este surge – até à primeira *frame* da corrida. É, portanto, construído de forma temporalmente invertida.

Este capítulo introduz o conceito de caminho e as suas possíveis variantes e características. Todos os pressupostos sobre o que é, ou não, interessante na perspectiva bioquímica no traçar de caminhos e os conceitos assentes neste capítulo foram discutidos com a equipa que forneceu as simulações – Rui M. M. Brito[10] e Cândida G.

[10] Página: http://www.uc.pt/fctuc/dquimica/rbritolab/members/rmb/ .

Silva[11] –, estando, consequentemente, em concordância com a visão e com os objetivos dos utilizadores. O capítulo divide-se em cinco secções: na primeira secção é feita a definição do caminho e dos conceitos relacionados; na segunda secção é abordada a estratégia de traçar de caminhos, que passa por recursivamente expandir um fragmento respeitando certas regras; na 3 é abordada uma variante dos caminhos: o caminho de fragmentação, de modo a esclarecer como um dado fragmento aparece na corrida; a secção 4 apresenta os algoritmos de produção de caminhos; e, por fim, a secção 5 conclui o capítulo.

6.1. Definição de Caminho

Um caminho é especificado por uma quíntupla (F, E, s, t, o), onde:

- F é um fragmento;
- E é o conjunto finito e ordenado de estados do caminho. Cada estado é especificado por um conjunto finito de contactos $C = \{c_1, \ldots, c_n\}$. Cada contacto $c \in C$ refere-se à ligação entre dois resíduos;

  Cada caminho de um fragmento é formado por estados que refletem a conjuntura do fragmento num certo instante. Cada estado poderia até mesmo corresponder a um momento de observação, todavia isso levaria a uma granularidade muito fina e o caminho tornar-se-ia numa estrutura de difícil análise dado o elevado número de estados em cada caminho e dado também o número de caminhos que geraria. O objetivo é tornar um caminho numa fonte de informação direta para o utilizador. De facto, como de uma *frame* para a outra, o número de contactos ganhos e perdidos é muito elevado, visto que um contacto é desfeito e refeito várias vezes numa corrida, é interessante tirar partido de estados espaçados em números de *frames*;

- s é o tamanho de salto entre estados.

  Um estado poderá não corresponder a um momento de observação, havendo ao invés saltos de evolução entre estados. O valor dos saltos depende de vários fatores: por exemplo, se forem considerados todos os 10000 momentos de

[11] Página: http://www.uc.pt/fctuc/dquimica/rbritolab/members/csilva/ .

observação, o salto entre estados tem de ser elevado, por forma a ser possível uma análise global dos caminhos. Contudo, após uma análise global de vários caminhos, é expectável que um deles seja escolhido de modo a observá-lo de forma mais detalhada numa determinada parte da corrida. Ao reduzir-se o tamanho da observação pode ser desejável diminuir também o tamanho dos saltos;

- t é o valor de *threshold* entre estados.

  Para que o fragmento que representa um estado seja o mais correto, no sentido de ser o que representa melhor a evolução do fragmento no intervalo entre estados, haverá o conceito de *threshold* associado a um caminho. O *threshold* tem um efeito de *smoothing* sobre os fragmentos iniciais para dar origem aos fragmentos do caminho. Este parâmetro terá a percentagem de vezes que um contacto terá de existir entre dois estados, da seguinte forma:

  - Dados dois estados e_i e e_{i+1}, em que e_i é o estado imediatamente anterior a e_{i+1}, e um valor de *threshold* t, tem-se que t é a percentagem de vezes que um dado contacto terá que aparecer entre os estados e_i e e_{i+1}, de modo a ser considerado um contacto de e_{i+1}.

  Por indicação, este valor idealmente rondará os 70-80%. Porém, tal como o valor do salto, este será fornecido aquando da produção dos caminhos pelo utilizador;

- o é o tamanho de observação do caminho.

  Um caminho é sempre traçado para uma determinada janela de *frames*. No limite, poderá ser traçado para a totalidade da corrida.

6.2. Estratégia para Traçar Caminhos

Uma estratégia para traçar os vários caminhos de um fragmento poderia passar por formar um caminho por cada nova ligação para o fragmento. Cada ligação nova geraria assim um novo caminho, e o processo repetir-se-ia recursivamente. Contudo, esta estratégia traz problemas ao nível do número de caminhos gerados dado ao elevado número de novos contactos ganhos de um estado para outro. Este elevado número de caminhos é indesejado, uma vez que, na prática, é desmotivante analisar um volume tão

expressivo de caminhos. O objetivo é que estes sejam analisáveis apenas pela sua observação sem recurso a outra ferramenta.

De modo a resolver o problema inerente ao elevado número de caminhos, uma nova abordagem teria a seguinte variante: em vez de ser considerado um novo caminho por cada nova ligação, será explorado um novo caminho para cada conjunto de novas ligações formadas para cada resíduo. Eventualmente será considerado o conjunto de resíduos associados a cada conjunto de ligações. A redução de número de caminhos pelo agrupamento das expansões por resíduo faz com que a granularidade de cada caminho seja menos fina. No entanto, não há perda de informação.

O traçar de caminhos respeita as seguintes regras:

1) Dado um fragmento, o primeiro estado corresponderá ao próprio fragmento. Os caminhos podem serem traçados até ao final da corrida mesmo que o fragmento já se tenha extinguido – considera-se que um fragmento se extingue imediatamente após a *frame* em que aparece pela última vez;

2) Dado um caminho, para cada novo estado haverá um conjunto de resíduos que expandem $R = [r_1, \dots, r_n]$. O caminho será ramificado para cada resíduo $r \in R$;

3) Em continuação da regra 2, cada uma das ramificações de um caminho não poderá mais expandir pelos restantes resíduos de R;

4) Um caminho c tem sempre uma lista de resíduos $R' = [r'_1, \dots, r'_n]$ pelos quais não pode expandir. Os caminhos que surgirem da ramificação de c herdam R';

5) Um fragmento poderá perder ligações com resíduos que não pertencem ao fragmento inicial. Porém, caso se trate de um resíduo do fragmento inicial, é ramificado um novo caminho contemplando esse resíduo que se desligou;

6) As expansões poderão ser restringidas por um conjunto de resíduos ou por um fragmento final. Estas restrições limitam o espectro de resíduos a serem usados para expandir um fragmento;

A Figura 22 apresenta 3 momentos de observação de um fragmento em que cada resíduo está representado por uma letra. Este exemplo é apenas teórico e tem como propósito servir de base para o estudo dos caminhos.

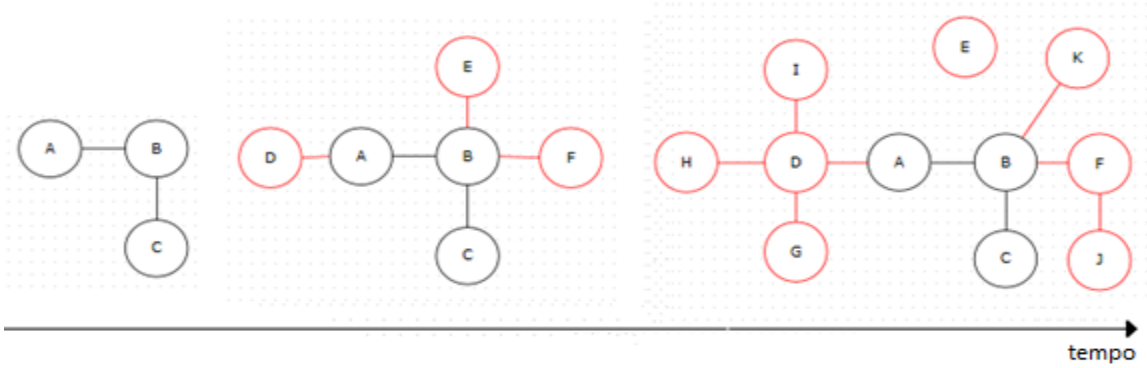


Figura 22 - 3 momentos de observação de um fragmento.

A Figura 23 mostra 3 caminhos gerados para os 3 momentos de observação ilustrados na Figura 22 com um tamanho de salto igual a uma *frame* (neste caso justifica-se um tamanho tão baixo dado o reduzido tamanho da amostra). Os 3 caminhos foram produzidos usando as regras atrás definidas. Assim:

- No primeiro estado, para qualquer caminho, o fragmento correspondente é o fragmento inicial (pela regra 1);

- Para o segundo estado, como no segundo momento de observação os resíduos A e B ganharam novas ligações, então serão ramificados dois caminhos: um para cada conjunto de ligações ganhos resíduos (pela regra 2);

- No segundo estado do primeiro caminho não foi considerada a nova ligação do resíduo B com o F, bem como no segundo caminho não foi considerada a ligação do resíduo A com o resíduo D. Isto deve-se a que, no primeiro estado, o caminho 1 foi expandido pelo resíduo A e o segundo caminho pelo resíduo B (pela regra 3);

- No terceiro estado, recursivamente, para o primeiro caminho, apenas o resíduo D ganha novas ligações, e portanto o caminho é expandido usando as novas ligações ganhas (pela regra 2);

  Para o segundo caminho, como o fragmento na transição do segundo estado para o terceiro ganha uma nova ligação no resíduo B e uma nova ligação no resíduo F, então este caminho é ramificado em dois considerando as novas ligações destes dois resíduos (pela regra 2);

- No terceiro estado, o caminho 3 não expandiu pelo resíduo A, à semelhança do que acontece com o caminho 2 e o caminho 1 não expandiu pelo resíduo B (pela regra 4);
- No terceiro estado, o caminho 3 não expandiu pelo resíduo K (pela regra 3);
- No terceiro estado do caminho 2, o resíduo E deixou de ser considerado pois trata-se de um resíduo que não pertence ao fragmento inicial e que perdeu ligação com o fragmento (pela regra 5);
- Caso houvesse outro momento de observação, teria que haver outro estado e, dependendo de que ligações se formariam e/ou perderiam, novos caminhos poderiam ser formados.

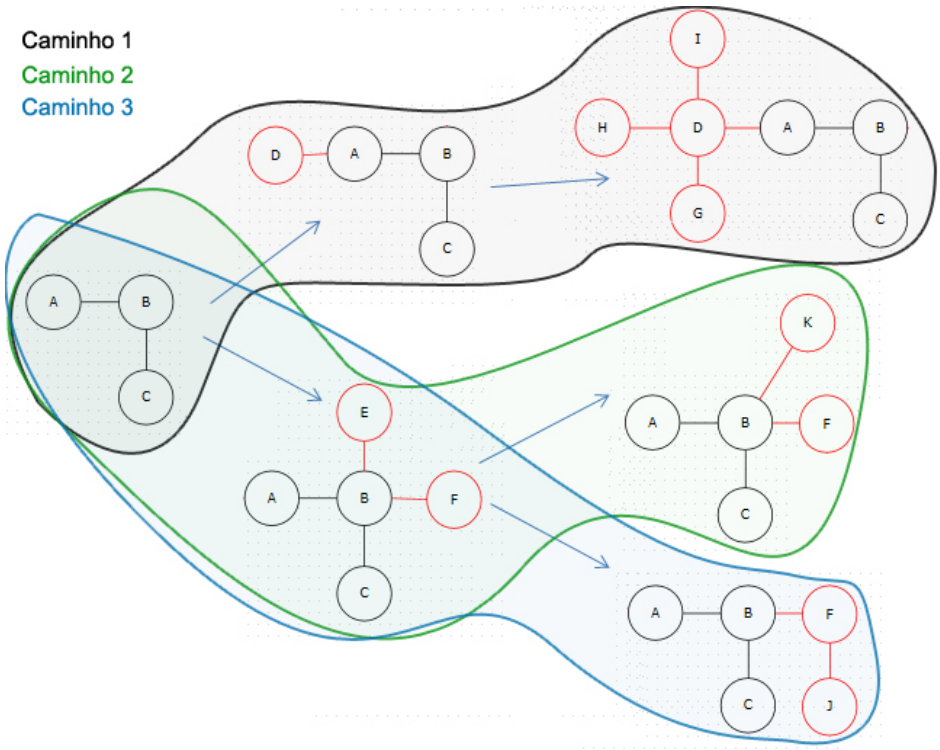


Figura 23 - Caminhos traçados para o exemplo da Figura 22.

Relativamente aos contactos presentes nos caminhos, quer o fragmento em causa seja nativo ou não nativo, no traçar dos caminhos serão considerados todos os contactos nativos e não nativos pois todos interferem na evolução de um fragmento. Em termos dos fragmentos não nativos, muitas vezes, um ou mais resíduos destes fragmentos evoluem apenas ligados por contactos próximos, e portanto este tipo de contactos terão que ser considerados apenas para fragmentos não nativos. Os contactos próximos são os

que envolvem pares de resíduos (i,j) com $j \leq i+3$. Nestes contactos, pelo menos um dos resíduos envolvidos faz parte do fragmento inicial.

6.2.1. Escolha de Resíduos

Aquando da produção de caminhos para um fragmento, é pertinente a possibilidade de escolha das direções de evolução mais importantes. Pode-se definir à partida o objectivo de observar a interação do fragmento com certos resíduos e que a interação com outros não é tão importante. Isto porque, dependendo da localização do fragmento na estrutura da proteína, haverá direções de evolução cuja descoberta de informações seja mais relevante

Assim sendo, pode-se definir quais os resíduos que devem ser tidos em conta aquando da evolução de um fragmento. Além disto, torna-se essencial libertar a aplicação da escolha do método mais indicado para expandir um fragmento. Não é possível saber à partida quais as direções que levarão a caminhos interessantes do ponto de vista bioquímico, já que estas dependem do tipo de estudo que o investigador tem em mente e do tipo de informação que este quer extrair. A escolha de resíduos traz imensos benefícios em termos de redução do tempo de computação. O número de caminhos produzidos é proporcional ao conjunto de resíduos parametrizados.

6.2.2. Fragmento Final

Um conceito parecido com o da escolha de resíduos é o de fragmento final, na medida em que estes limitam o espectro de resíduos para que um caminho poderá expandir. Contudo, são conceitos diferentes. No caso da escolha dos resíduos, o objetivo é perceber que ligações são ganhas e perdidas dentro de um núcleo de resíduos. Porém, com um fragmento final, o objetivo é chegar a um certo estado final que é representado pelo fragmento final dado.

Os caminhos traçados com um fragmento final terminam quando o fragmento final é encontrado ou quando a observação termina.

6.3. Caminho de Fragmentação

É igualmente interessante saber não só como evolui um fragmento, mas também como este é originado. Por outras palavras, que evolução na molécula leva à formação de um

determinado fragmento, nomeadamente no caso de fragmentos que envolvem contactos não nativos (no caso dos contactos nativos estes estão presentes logo na primeira *frame*). De forma a cobrir esta necessidade, existe o conceito caminho de fragmentação, i.e., um caminho que pretende mostrar como um fragmento que não está presente na estrutura nativa da proteína se forma.

A Figura 24 apresenta a relação existente entre os caminhos até então definidos e um caminho de fragmentação. Nesta, é possível observar que o caminho de fragmentação é traçado de forma temporalmente inversa aos outros caminhos, uma vez que começa na primeira ocorrência do fragmento e é traçado de forma a terminar no início da corrida.

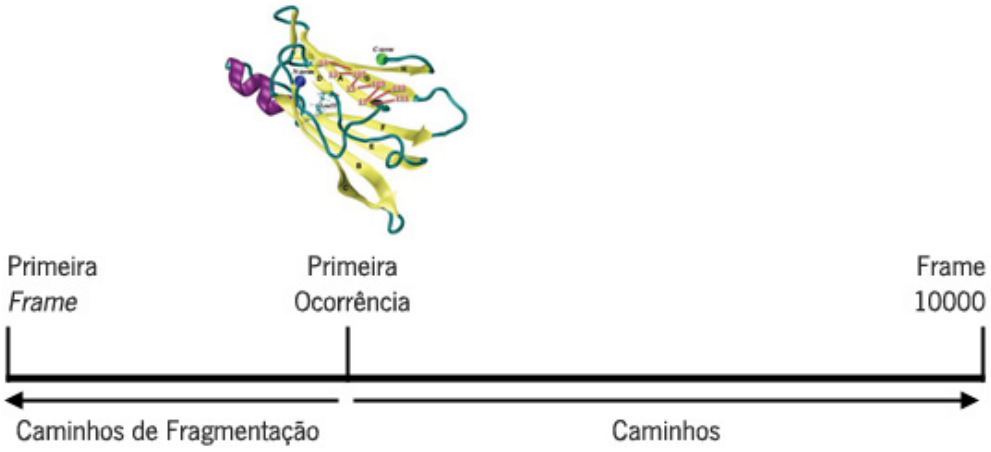


Figura 24 - Caminhos e o caminho de fragmentação.

Dado o propósito de um caminho de fragmentação ser diferente dos restantes e sendo este traçado de forma temporal inversa, é necessário definir uma nova estratégia para traçar caminhos de fragmentação. Esta estratégia considera as seguintes regras:

1. Não é possível traçar caminhos de fragmentação para fragmentos nativos – assim chamados por terem sido extraídos usando contactos nativos –, pois estes aparecem imediatamente na primeira *frame* da corrida de onde foram extraídos;

2. Para qualquer fragmento, o primeiro estado de um caminho de fragmentação corresponderá ao próprio fragmento;

3. Em cada novo estado são apenas consideradas as ligações em que entrem os resíduos do fragmento inicial e um segundo nível de ligações. Este segundo nível vem no sentido de exibir melhor o comportamento da molécula que

envolve o fragmento sem, no entanto, fazer crescer demasiado o fragmento em cada estado;

4. Entre estados são sempre considerados os valores de salto e de *threshold*;

5. Em cada novo estado, as novas ligações são procuradas nas *frames* anteriores, parando quando não houver mais *frames* para analisar. Se no fim do caminho houver uma quantidade de *frames* inferiores ao salto, estas serão consideradas de modo a que o estado final corresponda obrigatoriamente à primeira *frame* da corrida;

6. Serão sempre considerados os contactos entre pares de resíduos (i,j) com $j \leq i+3$, sendo, pelo menos um destes, um resíduo do fragmento inicial. Isto deve-se ao facto destes caminhos só serem produzidos para fragmentos não nativos.

6.4. Algoritmos de Produção de Caminhos

Foi desenvolvido um algoritmo (Algoritmo 1) de produção de caminhos que permite não só corresponder aos requisitos de memória, bem como a uma execução em tempo útil. Este algoritmo corre de forma recursiva, *multithreading* e por etapas. Por ser *multithreading* permite tomar partido da plataforma computacional e por ser por etapas não compromete a memória disponibilizada para a execução da ferramenta.

Relativamente ao Algoritmo 1:

- O processamento por etapas é feito pelo ciclo da linha 7 em que são processados um certo número de *frames* de cada vez;

- O processamento *multithreading* está nas linhas 13 e 14 em que são criadas *threads* que expandem os caminhos incompletos;

- A função *calculaObservacao* calcula o número de *frames* a processar face ao tamanho da observação e ao número de caminhos incompletos a expandir. Este cálculo é necessário para garantir que a memória não é comprometida visto ser um processo de expansão de caminhos recursivo em que podem resultar centenas ou até milhares de caminhos.

Algoritmo ProduzirCaminhos(F,s,t,f,o,exp, listaExp, frag)

Pré-condição: F é o fragmento inicial
Pré-condição: s é o tamanho do salto
Pré-condição: t é o valor do threshold
Pré-condição: f é a frame de início
Pré-condição: o é o tamanho da observação
Pré-condição: exp é o tipo de expansão {1 – lista de resíduos; 2- sem restrições; 3-fragmento final}
Pré-condição: listaExp é a lista de expansão
Pré-condição: frag é o fragmento final
Pós-condição: L contém os caminhos {L é uma variável global}

1: $E_1 \leftarrow F$; {o primeiro estado é o fragmento inicial}
2: $P' \leftarrow adicionarEstado(E_1, \emptyset)$; {cria o caminho inicial}
3: $n \leftarrow calculaObservavao(o)$; {calcula o nº inicial de estados a processar}
4: $expandeCaminho(P', s, t, f, n, exp, listaExp, frag)$; {produz os caminhos iniciais; guarda em L}
5: $f' \leftarrow f$;
6: $nEst \leftarrow n$; {actualiza o número de frames observadas}
7: **while** $(nEst < o)$ **do** {enquanto houver estados para produzir}
8: $f' \leftarrow f' + n$;
9: $n \leftarrow calculaObservavao()$; {calcula o valor de n}
10: $nEst \leftarrow nEst + n$; {actualiza o número de frames já observadas}
11: $L' \leftarrow L$;
12: $L \leftarrow \emptyset$;
13: $T \leftarrow criarThreads(nº\ de\ processadores)$; {cria threads}
14: **for** *each thread* $thr \in T$ **do**
15: $LP \leftarrow getCaminhosIncompletos(L')$; {copia parte dos caminhos incompletos}
16: **for** *each path* $p \in LP$ **do** {expande cada caminho; guarda em L}
17: $expandeCaminho(p, s, t, f', n, exp, listaExp, frag)$;

Algoritmo 1 - Algoritmo ProduzirCaminhos.

A função *expandeCaminho* está descrita no Algoritmo 2. Acerca deste algoritmo:

- A função *expandeCaminho* ramifica um caminho pelo acrescento de um novo estado;
- Na linha 3, são extraídos os contactos entre as *frames* correspondentes ao novo estado e ao estado anterior que respeitem o valor de *threshold* e que sejam uma continuação do estado anterior;
- Na linha 4, da lista de contactos da linha 3, são extraídos os resíduos que expandem considerando a lista de expansões e o tipo de expansão escolhido pelo utilizador;

- A partir da linha 5, o caminho é ramificado para cada resíduo que expande;
- Nas linhas 10 e 11, caso não haja mais estados para processar ou caso seja encontrado o fragmento final, então o novo caminho P' é adicionado à variável global de caminhos L;
- A partir da linha 12, caso haja mais estados para processar ou caso não seja encontrado o fragmento final, o caminho é expandido recursivamente. Neste caso, a *frame* de início, o tamanho da observação e a lista de expansões são actualizados.

Algoritmo expandeCaminho (P,s,t,f,n,exp, listaExp,frag)

Pré-condição: P é um caminho incompleto
Pré-condição: s é o tamanho do salto
Pré-condição: t é o valor do threshold
Pré-condição: f é a frame de início
Pré-condição: n é o tamanho da observação
Pré-condição: exp é o tipo de expansão {1 – lista de resíduos; 2- sem restrições; 3-fragmento final}
Pré-condição: listaExp é a lista de resíduos que podem expandir
Pré-condição: frag é o fragmento final
Pós-condição: P é expandido

```
1: E_ultimo ← getUltimoEstado(P);                                   {extrai o último estado}
2: e ← f + s;                                                       {calcula a frame de limite}
3: listaContactos ← extraiContactos(E_ultimo, s, t, e);             {extrai contactos}
4: listaResidues ← extraiResiduos(listaContactos, listaExp, exp);   {extrai resíduos que expandem}
5: for each resíduo r ∈ listaResidues do
6:       F' ← expande(E_ultimo, listaContactos, r);                 {expande o último estado}
7:       E_novo ← F';                                               {cria um novo estado}
8:       P' ← P;                                                    {cria um novo caminho igual}
9:       P' ← adicionaEstado(E_novo);                               {adiciona o novo estado}
10:      if (n − s < 0 || encontrouFrag(frag, E_novo)) then         {se chegou ao fim do caminho}
11:            adicionaCaminho(P') ;                                {adiciona o caminho P'}
12:      else
13:            listaExp' ← actualizaLista(r, exp, listaExp);        {actualiza a lista de expansões}
14:            n ← n − s;                                           {actualiza a observação}
15:            f' ← f + s;                                          {actualiza o valor da frame}
16:            expandeCaminho (P',s,t,f,n,exp, listaExp', frag)     {recursivamente expande P'}
```

Algoritmo 2 - Função *expandeCaminho*.

No caso dos caminhos de fragmentação, o algoritmo corre também de forma recursiva. Contudo, visto ser apenas produzido um caminho, não existe a necessidade de haver processamento por etapas nem *multithreading*. Trata-se de uma computação leve. O algoritmo é o seguinte:

Algoritmo ProduzirCaminhosFrag(F,s,t)

Pré-condição: F é o fragmento inicial
Pré-condição: s é o tamanho do salto
Pré-condição: t é o valor do threshold
Pós-condição: FP é o caminho de fragmentação {FP é uma variável global}

1: $E_1 \leftarrow F$; {o primeiro estado é o fragmento inicial}
2: $P' \leftarrow adicionarEstado(E_1, \emptyset)$; {cria caminho inicial}
3: $f \leftarrow getPrimeiraOcorrência(F)$; {calcula a primeira *frame*}
4: $n \leftarrow f$; {calcula o tamanho da observação}
5: $expandeCaminhoFrag(P', s, t, f, n)$; {expande P'; guarda caminho em FP}

Algoritmo 3 - Algoritmo ProduzirCaminhosFrag.

A função *expandeCaminhoFrag* está descrita no Algoritmo 4. Acerca do Algoritmo 4:

- Na linha 3 são extraídos todos os contactos que expandem o caminho;
- Na linha 4 é criado um novo fragmento composto pelos contractos extraídos na linha 3;
- Nas linhas 8 e 9, caso não haja mais *frames* para processar, o caminho de fragmentação está terminado;
- Das linhas 10 a 16, caso haja um número de *frames* inferiores ao tamanho do salto, estas são consideradas para criar um novo estado. O novo estado é assim adicionado ao caminho e o caminho de fragmentação está terminado;
- A partir da linha 17 tem-se o caso de haver mais *frames* para processar. O caminho é expandido recursivamente.

Algoritmo expandeCaminhoFrag (P,s,t,f,n)

Pré-condição: P é um caminho incompleto
Pré-condição: s é o tamanho do salto
Pré-condição: t é o valor do threshold
Pré-condição: f é a frame de início
Pré-condição: n é o tamanho de observação
Pós -condição: P é expandido

```
1: E_ult ← getUltimoEstado(P);                          {extrai o último estado}
2: e ← f − s;                                           {calcula a frame de limite}
3: listaContactos ← extraiContactosFrag(E_ult, s, t, e); {extrai contactos}
4: F' ← expandeFrag(listaContactos);                    {expande o último estado}
5: E_novo ← F';                                         {cria um novo estado}
6: P ← adicionaEstado(E_novo);                          {adiciona o novo estado ao caminho}
7: n ← n − s;                                           {actualiza o tamanho da observação}
8: if (n == 0 ) then                                    {se chegou ao fim da observação}
9:     FP ← P;                                          {FP é o caminho de fragmentação}
10: else if (n < s) then                                {se a observação é menor que o salto}
11:    E_ultimo ← getUltimoEstado(P);
12:    listaContactos ← extraiContactosFrag(s, t, n);   {extrai contactos}
13:    F' ← expandeFrag(E_ultimo, listaContactos);      {expande o último estado}
14:    E_novo ← F';                                     {cria um novo estado}
15:    P ← adicionaEstado(E_novo);                      {adiciona o novo estado ao caminho}
16:    FP ← P;                                          {FP é o caminho de fragmentação}
17: else
18:    f' ← f − s;                                      {actualiza o valor da frame}
19:    expandeCaminhoFrag (P,s,t,f',n);                 {recursivamente expande o caminho P}
```

Algoritmo 4 - Função *expandeCaminhoFrag*.

Em termos de complexidade, no pior caso, a produção de caminhos seria feita sem considerar nenhuma restrição e para um elevado tamanho de observação. Além disto, teoricamente, em cada estado, cada resíduo do fragmento expandiria para uma só ligação. Para este cenário, o Algoritmo 3 tem complexidade exponencial $O(2^n)$. Contudo, este cenário muito dificilmente se verificaria pois em cada novo estado um caminho expande, normalmente, para vários resíduos. Além disto, as regras 1, 2, 3 e 4 citadas na secção 6.2. permitem que o número de caminhos produzidos não aumente sempre exponencialmente.

O Algoritmo 4 tem complexidade linear $O(n)$.

6.5. Conclusão

A definição de caminho e o estabelecimento claro de todas as suas regras e variantes inerentes permitem avançar para a implementação de uma ferramenta de produção e visualização de caminhos. É de salientar que esta definição teve em conta as informações disponibilizadas pela equipa que forneceu as simulações e também teve em consideração as informações recolhidas pela observação dos fragmentos extraídos. Por exemplo, saber que um fragmento está continuamente ligado a muitos resíduos e que, de *frame* para *frame,* o número de contactos ganhos e perdidos é elevado, levou a que fosse repensada a forma de expandir um fragmento. Assim, as expansões são agrupadas por resíduos de modo a não gerar um número elevado de caminhos, impossíveis de analisar face ao seu volume.

Uma das mais-valias dos caminhos é permitirem explorar as corridas variando diversos parâmetros. Ao serem considerados o tamanho de salto, o valor de *threshold*, o tamanho de observação e as restrições quanto às expansões, os caminhos obtidos são totalmente ajustados ao estudo que o investigador tem em mente.

Traçar um caminho de fragmentação para fragmentos não nativos permite entender como estes se formam na corrida e não apenas as suas evoluções posteriores. Estes caminhos são assim uma peça chave para entender o processo de desnaturação da Transtirretina.

Capítulo 7

Procura de Caminhos em Corridas

A compreensão total do mecanismo de desnaturação de uma proteína exige a observação e comparação de várias simulações de modo a extrair informações novas e pertinentes. Quer pela similaridade entre partes de simulações, quer pela falta desta, pode-se obter novos conhecimentos. Dos vários caminhos extraídos numa determinada corrida, alguns poderão ser considerados importantes não só pelas informações que estes transmitem dentro da simulação de onde derivaram mas também pela análise do caminho noutra corrida. Procurar semelhanças e diferenças no comportamento dos resíduos ao longo do tempo de simulação, poderá dar mais pistas sobre o processo de desnaturação da Transtirretina.

Numa primeira análise, verificaram-se vários cenários possíveis na procura de um caminho numa corrida:

- O fragmento inicial de onde derivou o caminho não existe na corrida da procura;
- O caminho encontra-se integralmente na corrida exatamente na mesma localização temporal ou então afastado desta;
- O caminho encontra-se apenas parcialmente presente na corrida, havendo a necessidade de utilizar um teste para medir o quão similar é este caminho incompleto com o original da procura.

Este capítulo divide-se em quatro secções. Primeiro serão apresentados os objetivos da procura de caminhos e o método a ser usado nesta tendo em conta a observação dos caminhos e das corridas disponíveis. Na segunda secção será abordado o problema do cálculo da similaridade que engloba a transformação dos dados em vetores de *features* e o uso de um teste de hipóteses. A terceira secção apresenta o algoritmo de procura. A última secção conclui o capítulo.

7.1. Objetivos e Método de Procura

A observação de diferentes caminhos e a sua comparação face às várias corridas permite chegar às seguintes conclusões:

a) Visto que um caminho é produzido a partir de um fragmento e que o objetivo de um caminho é justamente explorar o comportamento de tal fragmento, então não tem sentido procurar um caminho numa corrida em que tal fragmento nem sequer existe;

b) Pela observação simples de vários caminhos nas diversas corridas existentes, constata-se que por vezes, e principalmente para fragmentos de maior dimensão, o mesmo caminho encontra-se em várias corridas. Este encontrar é feito pela procura do caminho usando exatamente o mesmo tamanho de salto e valor de *threshold* e considerando exatamente as mesmas *frames*. Neste caso, diz-se que o caminho está sincronizado na corrida;

c) Para além do caso em que o fragmento nem sequer existe na corrida de procura ou do caso em que o caminho está sincronizado, existe a possibilidade de o caminho estar traçado na corrida noutro espaço temporal. Este caminho poderá existir nas *frames* anteriores ou posteriores às consideradas no caminho original. Esta afirmação vai de encontro ao que é sabido sobre o facto de, por vezes, para corridas diferentes, o mesmo fenómeno poder ocorrer mais cedo ou mais tarde;

d) Observou-se também que um caminho, ou se encontra temporalmente próximo do caminho original, ou então não está presente na corrida, nem mesmo de forma incompleta;

e) Existem ainda casos de caminhos que não são abrangidos por qualquer um dos cenários anteriormente relatados. Nestes, o fragmento inicial do caminho

encontra-se na corrida mas o caminho não se encontra em qualquer parte da corrida. São avistadas apenas partes deste em que faltam determinados contactos em determinados estados do caminho.

Assim, o método de procura terá que ter em conta:

- Se um fragmento que deu origem a um caminho não se encontra na corrida, então qualquer caminho originado desse fragmento também não se encontra (a);
- Um caminho poderá ser encontrado sincronizado numa corrida se todos os contactos de todos os estados estiverem na corrida exatamente no mesmo espaço temporal, considerando o mesmo tamanho de salto e de *threshold* (b); ou então presente, mas dessincronizado (c);
- Na procura serão apenas consideradas as *frames* mais próximas (d);
- Haverá casos em que apenas parte de um caminho poderá ser encontrado (e).

O objetivo é então não só procurar um caminho, fazendo um *matching* básico no sentido em que só um caminho só é considerado presente quando ocorre exatamente nas mesmas *frames* numa corrida, mas também contemplar situações em que o caminho está dessincronizado, ou que ocorre apenas parcialmente na corrida. Neste último caso, usando um teste de hipóteses, pode determinar-se quanto um caminho parcial é similar do caminho original. O método de procura da similaridade está descrito na próxima secção.

7.2. Procura da Similaridade

A procura da similaridade entre um caminho e outro incompleto segue duas etapas. A primeira é a transformação de cada um dos caminhos em vetores de *features* cujo objetivo é facilitar a análise do caminho pela simplificação da sua estrutura. Em seguida, usando o teste de hipóteses Goodness-of-fit de Pearson [Pearson 1900] é determinado se o caminho incompleto é suficientemente similar ao caminho original. As próximas subsecções relatam em pormenor estas duas etapas.

7.2.1. Transformação dos Caminhos em Vetores de *Features*

Quando um caminho apenas se encontra de forma parcial numa corrida, a decisão sobre se este caminho parcial é ou não similar o suficiente com o original torna-se mais fácil se tanto o caminho original, quanto o caminho incompleto, sofrerem uma transformação vetorial. Esta transformação simplifica o cálculo da similaridade, bem como dilui os problemas de isomorfismo de grafos, como será possível observar mais à frente.

Um caminho original é então transformado num vetor de *features*. Para isto, primeiro é determinado um conjunto de *features* F = $\{f_i, \dots, f_n\}$, em que cada *feature* corresponde a um tipo de contacto presente no caminho original. Em seguida, para cada estado e de um caminho, haverá um vetor de *features* $x = \{x_i, \dots, x_n\}$que o representa, em que x_i = 1 se $f_i \subseteq e$; caso contrário x_i = 0;

Partindo do caminho 2 da Figura 23, tem-se o conjunto de *features* F = {(a,b), (b,c), (b,e), (b,f), (b,k)}. Para cada um dos estados do caminho 2, os vetores de *features* são x_1= [1,1,0,0,0], x_2= [1,1,1,1,0] e x_3= [1,1,0,1,1]. A Figura 25 apresenta os valores destes vetores de *features* em que cada linha representa os dados de um vetor para um dado estado.

	(a,b)	(b,c)	(b,e)	(b,f)	(b,k)
Estado 1	1	1	0	0	0
Estado 2	1	1	1	1	0
Estado 3	1	1	0	1	1

Figura 25 - Representação em *features* do caminho 2 da Figura 23

A transformação de um caminhos em vetores de *features* contorna questões de isomorfismo, uma vez que as *features* têm em conta a ordem dos resíduos na cadeia polipeptída da proteina, da seguinte forma:

- $feature\ (x, y) \rightarrow$
  $x\ tem\ número\ de\ ordem\ na\ cadeia\ polipetidica\ inferior\ ao\ de\ y.$

Assim sendo, se existir a *feature* (x, y) não existirá a *feature* (y, x). Estas considerações são possíveis, uma vez que não existe uma orientação na ligação dos resíduos (os grafos são não orientados).

A representação vetorial de uma trajetória permite recuperar a estrutura desta na sua forma original de grafo, uma vez que, tal como é dito no capítulo 3, estes grafos são não orientados e sem pesos nas arestas. Um caminho incompleto será representado da mesma forma que o caminho original. Porém, usa o conjunto de *features* derivado do caminho original.

Tendo os caminhos em formato de vetores de *features*, é possível determinar se estes se assemelham o suficiente de modo a que o caminho parcial possa ser considerado um resultado interessante da procura do caminho. O teste utilizado para averiguar esta similaridade é o teste de hipóteses não paramétrico para valores categóricos Goodness-of-fit de Pearson [Pearson 1900].

7.2.2. Teste Goodness-of-fit de Pearson

Um teste de hipóteses é um procedimento estatístico utilizado para decidir se uma dada hipótese se verifica. Os testes Goodness-of-fit são tradicionalmente usados para verificar a adequação de uma distribuição teórica em relação a uma amostra. Porém, poderão ser usados também para verificar se duas amostras foram retiradas da mesma população. A conclusão que se tira do teste de hipóteses deve ser sempre formulada no contexto do problema de pesquisa que está a ser estudado. No contexto do problema da procura da similaridade, afirmar que um caminho e o seu parcial provêm da mesma população significa dizer que o comportamento das corridas associadas aos caminhos – original e parcial –, nas partes das simulações confinadas a estes, é semelhante.

Foi escolhido o teste Goodness-of-fit de Pearson por se tratar de um teste não paramétrico para amostras de valores categóricos. Como hipóteses tem-se a hipótese nula (H_0), que dita que os caminhos provêm da mesma população, ou seja, têm um comportamento idêntico; e a hipótese alternativa (H_1), que indica que os caminhos provêm de populações diferentes.

Neste teste, considerando duas amostras A e B, as observações destas - respetivamente N_a e N_B - são agrupadas em K ($K > 2$) categorias. Neste caso de estudo, a amostra A será a representação vetorial do caminho original e B a representação vetorial do caminho parcial encontrado. As categorias são o conjunto de *features* determinado.

Para cada amostra é contada a frequência absoluta (N_k) e calculada a frequência esperada (e_k) para cada k_i categoria, com $k_i \in K$, usando as seguintes fórmulas:

$$N_k = N_{kA} + N_{kB}$$

Fórmula 1 - Frequência absoluta de cada categoria.

$$N = N_A + N_B$$

Fórmula 2 - Frequência absoluta total.

$$e_{kA} = N_A \times \frac{N_k}{N}$$

Fórmula 3 - Valor esperado de cada categoria para a primeira amostra.

$$e_{kB} = N_k - e_{kA}$$

Fórmula 4 - Valor esperado de cada categoria para a segunda amostra.

$$\sum e_{kA} = N_A$$

Fórmula 5 - Somatório dos valores esperados para a primeira amostra.

$$\sum e_{kB} = N_B$$

Fórmula 6 - Somatório dos valores esperados para a segunda amostra.

Após a obtenção dos valores destas variáveis, é possível calcular a estatística do teste. Para o teste de Pearson esta é calculada segundo a Fórmula 7:

$$TS = \sum_{k=1}^{K} \frac{(N_{kA} - e_{kA})^2}{e_{kA}} + \sum_{k=1}^{K} \frac{(N_{kB} - e_{kB})^2}{e_{kB}}$$

Fórmula 7 - Estatística do teste.

Neste teste, a hipótese nula é rejeitada, ou seja, as amostras não pertencem a populações idênticas, se para um dado grau de significância α, $TS > X^2_{K-1}(\alpha)$. O valor de α varia entre [0,1]. Tipicamente, o valor de referência é de 5%. No entanto, depende do contexto em que é aplicado, sendo um valor a definir pelo investigador. Assim, por omissão, será utilizado o valor tradicional de 0,05 – 5%.

Como resultado, tem-se então que se o valor da estatística do teste (TS) for inferior ou igual ao valor tabelado de $X^2_{K-1}(\alpha)$, o caminho parcial em teste será considerado como um resultado interessante da procura.

7.3. Algoritmo de Procura

Face ao exposto sobre a procura de um caminho numa corrida, o algoritmo de procura é o seguinte:

Algoritmo ProcurarCaminho (P,c)

Pré-condição: P é um caminho
Pré-condição: c é uma corrida
Pós -condição: R é a resposta da procura
Pós -condição: listaInc é a lista de caminhos incompletos obtidos na função procuraDesincr
Pós-condição: listaR é a lista de caminhos incompletos que passaram no teste

```
1: f ← getFragmentoInicial(P);
2: passo1 ← procuraFragmento(f,c);                    {procura fragmento na corrida}
3: if (passo1 ≠ −1) then                              {se o caminho foi encontrado}
4:       passo2 ← procuraSincr(passo1, P, c);         {procura sincronizado}
5:       if (passo2 ≠ −1) then                        {se foi encontrado}
6:             R ← ENCONTRADO_E_SINCRONIZADO;
7:       else
8:             listaInc ← ∅;
9:             listaR ← ∅
10:            passo3 ← procuraDesincr(passo1,P, c);  {procura dessincronizado}
11:            if (passo3 ≠ −1) then                  {se foi encontrado}
12:                  R ← ENCONTRADO_MAS_DESSINCRONIZADO;
13:            else
14:                  for each caminho cam ∈ listaInc do
15:                        res ← testeGOF(cam);       {testa a similaridade}
16:                        if (res ≠ −1) then         {se o caminho passou no teste}
17:                              listaR ← listaR + cam; {adiciona o caminho}
18:                  if (listaR ≠ ∅) then
19:                        R ← ENCONTRADOS_CAMINHOS_INCOMPLETOS;
20:                  else
21:                        R ←CAMINHO_NAO_ENCONTRADO;
22: else
23:      R ← CAMINHO_NAO_ENCONTRADO
```

Algoritmo 5 - Algoritmo PorcurarCaminho.

Relativamente ao Algoritmo 5:

- Na linha 2, a função *procuraFragmento* retorna o número da *frame* em que foi encontrado o fragmento. Esta procura é feita procurando, em primeiro lugar, as *frames* mais próximas da localização do fragmento na corrida original;

- Na linha 4, a função *procuraSincr* procura o caminho na mesma localização temporal. A função retorna -1 se o caminho não estiver sincronizado;

- Na linha 10, a função *procuraDesincr* procura o caminho fora da mesma localização temporal. Nesta função de procura são guardados na variável global *listaInc* os caminhos incompletos encontrados. A função retorna o valor da *frame* onde começa o caminho dessincronizado ou -1 caso o caminho não seja encontrado;

- Da linha 14 à 17 são testados os caminhos incompletos encontrados. O teste considera a transformação dos caminhos em vetores de *features* e a aplicação do teste Goodness-of-fit de Pearson.

A função *procuraDesincr* está descrita no Algoritmo 6. Neste algotimo, a procura que considera os caminhos incompletos (linhas 8 e 16 do algoritmo) tem, por omissão, as seguintes restrições:

- O número de contactos em falta em cada estado terá que ser inferior a 1 contacto para estados com menos de 10 contactos e inferior a 2 para os restantes;

- O número total de contactos em falta num caminho incompleto terá que ser inferior a 10% do total de contactos do caminho original;

Os caminhos incompletos são guardados na variável global *listaInc.* Ainda sobre o Algoritmo 6:

- Por omissão, o valor de MAX é 5;

- O valor de MAXOBS é determinado para cada caminho. Dado um caminho de procura, com um tamanho de observação o:

    - MAXOBS $= tamanho\ total\ da\ corrida - o$.

Algoritmo procuraDesincr(pos,P,c)

Pré-condição: pos é a posição inicial da procura
Pré-condição: P é o caminho
Pré -condição: c é a corrida
Pós -condição: res é o resultado da procura

```
1: nIncompletos ← 0;
2: it ← 0;
3: while (true) do
4:     ant ← −1;
5:     prox ← −1;
6:     if(pos − it ≥ 0 && it ≤ 1500) then          {procura apenas nas frames mais próximas}
7:         if(nIncompletos < MAX) then
8:             ant ← procura(pos − it, 0);          {procura por caminhos incompletos;
                                                     actualiza nIncompletos}
9:         else
10:            ant ← procura(pos − it, 1);          {não procura por caminhos incompletos}
11:    if(ant ≠ 1) then                             {encontrado um caminho dessincronizado}
12:        res ← ant;                               {ant é a frame onde começa o caminho}
13;        break;
14:    if(pos + it < MAXOBS && it ≤ 1500) then      {procura apenas nas frames mais próximas}
15:        if(nIncompletos < MAX) then
16:            prox ← procura(pos + it, 0);
17:        else
18:            prox ← procura(pos + it, 1);
19:    if(prox ≠ 1) then                            {encontrado um caminho dessincronizado}
20:        res ← prox;                              {ant é a frame onde começa o caminho}
21;        break;
22:    if(ant == −1 && pos == −1) then              {a procura termina}
23;        res ← −1;                                {nenhum caminho encontrado}
24:        break;
25:    incrementa(it);
```

Algoritmo 6 - Função procuraDesincr.

O algoritmo da procura (Algoritmo 5) tem complexidade quadrática $O(n^2)$.

7.4. Conclusão

A procura de um caminho numa corrida resulta num processo feito por etapas, podendo esta acabar rapidamente pelo facto do fragmento não estar presente na corrida, ou, pelo contrário, exigir a procura de caminhos incompletos e o cálculo da similaridade. Assim, poderá haver resultados de procura bastantes diferentes consoante o caminho e a corrida usados na procura.

A passagem de grafos para vetores veio facilitar a forma como os caminhos são manipulados para o cálculo da similaridade, diminuindo assim a complexidade em termos de isomorfismo de grafos sem perder informação sobre a estrutura inicial destes. Esta passagem não afeta a qualidade do teste, já que, em termos da informação usada para o teste, esta seria exatamente a mesma se fossem usadas as amostras em forma de grafos. Assim, não há perda de informação. Há apenas uma reestruturação mais adequada ao problema.

O teste de similaridade trabalha com as contagens de cada tipo de contacto. Estas contagens são calculadas com base nos contactos do caminho de procura. Assim, na procura dos caminhos incompletos, em cada estado, apenas são considerados os contactos referentes ao caminho de procura. Isto faz com que um caminho incompleto seja sempre um "subcaminho" do seu caminho procura. Mais concretamente, em cada estado, o conjunto de contactos de um caminho incompleto é subconjunto do conjunto de contactos do caminho original de procura.

Assim, não é possível que haja um caminho incompleto usado no teste de similaridade que não seja "subcaminho" do caminho de procura. Além disto, o algoritmo de procura de caminhos incompletos dita que, em cada estado, existe controlo sobre o número máximo de contactos em falta. Por exemplo, o número total de contactos em falta, por omissão, deve ser inferior a 10% do total de contactos do caminho de procura. Consequentemente, cada caminho incompleto encontrado tem 90% da estrutura do caminho de procura.

Face ao exposto, é possível concluir que não existe a possibilidade de ser considerado um caminho para teste que não seja estruturalmente muito semelhante ao caminho de procura. Assim, tem cabimento o uso da contagem de contactos no teste estatístico para obter similaridade de caminhos em diferentes simulações.

Capítulo 8

Implementação e Apresentação da Ferramenta

Um conjunto de fragmentos frequentes por si só não é suficiente para formar conhecimento sobre o comportamento da Transtirretina. O elevado número de fragmentos extraídos em cada variante da proteína e a própria natureza destes, torna praticamente impossível analisar um volume tão elevado de fragmentos e trabalhá-los de forma a extrair conhecimento sem recurso a uma ferramenta. Neste sentido, foi desenvolvida a Subgraph Paths, uma ferramenta em Java que permite não só visualizar fragmentos, mas também produzir caminhos e observá-los graficamente. Além disto, permite verificar se uma trajetória está presente noutras corridas.

Neste capítulo é apresentado, em primeiro lugar, a implementação e as considerações iniciais sobre a ferramenta. Em seguida, é apresentada a ferramenta, que engloba a apresentação das funcionalidades oferecidas. A terceira secção apresenta dois exemplos da utilização da ferramenta. No final, é apresentada a conclusão do capítulo.

8.1. Implementação e Considerações Iniciais

A Subgraph Paths foi desenvolvida em Java 7.0, usando o *Integrated Development Environment* (IDE) NetBeans. Foram utilizadas as bibliotecas Prefuse [Chi 1999] para a visualização dos fragmentos e dos caminhos e a JavaHelp System [Lew 2000] para a construção da documentação da ferramenta. Para o desenvolvimento gráfico da

aplicação foi usado Swing. O projeto da ferramenta foi modelado usando a *Unified Modeling Language* (UML), sendo que o modelo de domínio, os casos de uso e o *Platform-Independent Model* (PIM) encontram-se no anexo V. Além disto, a ferramenta possui o Javadoc atualizado e o manual de utilização também está disponível.

8.1.1. Prefuse

O Prefuse é uma biblioteca desenvolvida para a visualização de dados baseada no padrão *information visualization reference model*, proposto em 1989 por Ed Huai-Hsin Chi, em que, internamente, as estruturas de dados estão otimizadas para a representação de tabelas de grafos e esquemas em árvores.

Desenvolvida em tecnologia Java (Java 2D), é orientado para integrar componentes Swing e *applets web*, estando também disponível para as tecnologias ActionScript e Adobe Flash Player. Graficamente, permite a criação de animações fluidas, com algoritmos de disposição de componentes bem conseguidos, disponibilizando bons mecanismos para a navegação nos modelos, como, por exemplo, filtros, *zoom* geométrico e semântico, *querys* dinâmicas, e manipulação direta de cada um dos elementos. A biblioteca permite ainda a personalização por reescrita/extensão da maioria dos mecanismos, nomeadamente ao nível da cor, da forma, e dos tamanhos dos elementos de um diagrama, dos algoritmos para a representação gráfica, das opções para manipulação direta dos elementos, bem como do comportamento e estrutura de um modelo.

8.1.2. JavaHelp System

A JavaHelp System utiliza documentos *HyperText Markup Language* (HTML) como base para guardar e representar a informação. Uma vez que este componente utiliza um *standard* tão flexível como o HTML, pode-se facilmente elaborar documentos de ajuda muito completos e com boa aparência, que poderão ser reaproveitados para vários fins como, por exemplo, para uma página de ajuda online.

8.2. Apresentação da Aplicação

A Sugraph Paths é uma ferramenta *desktop* compatível com qualquer plataforma que suporte Java. Na continuação deste capítulo serão apresentadas apenas as

funcionalidades que se encontram implementadas e testadas a 100%, prontas para a utilização imediata por parte de investigadores e estudiosos das proteínas. Ainda é de referir que a aplicação está toda em inglês por requisito dos próprios utilizadores.

Ao executar a ferramenta, é apesentado um menu inicial que conta com um botão de ajuda, um botão para guardar um relatório das acções e resultados obtidos na ferramenta e três separadores.

A Figura 26 apresenta o menu inicial, nomeadamente o separador *Fragments*.

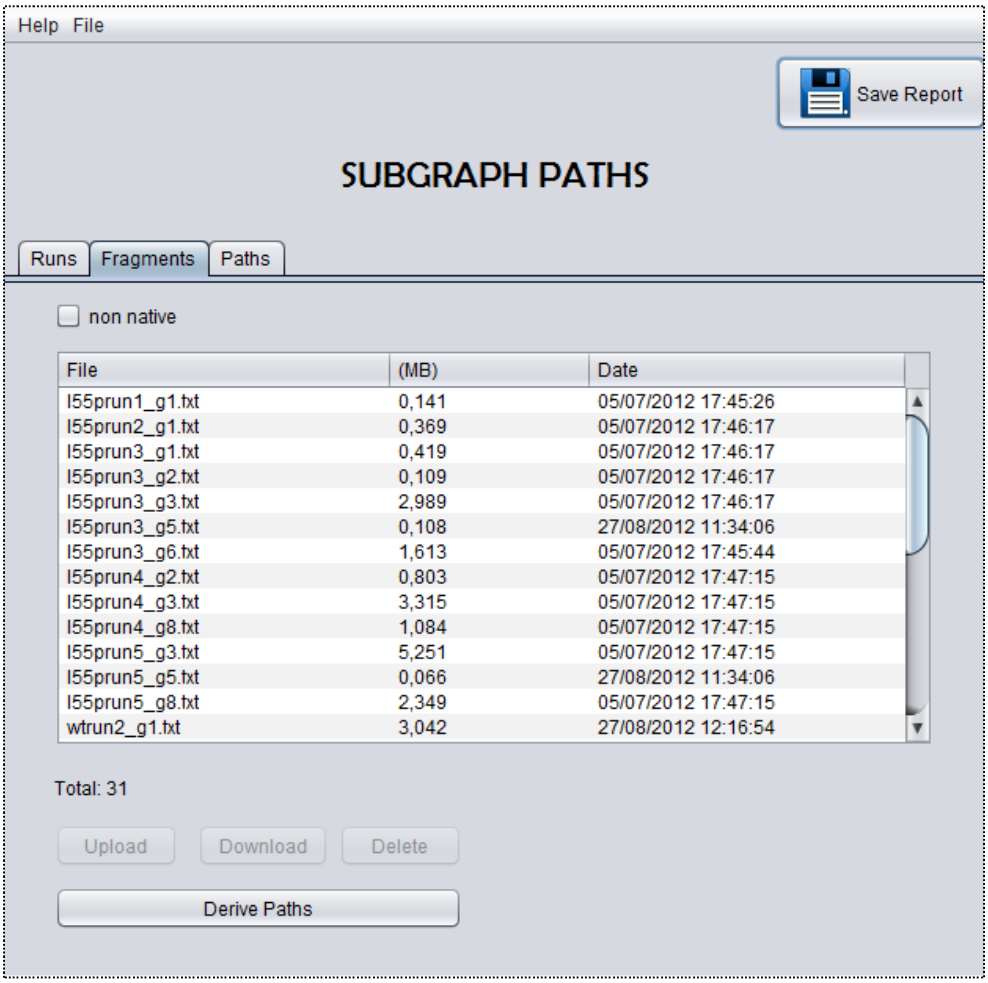


Figura 26 - Apresentação do menu inicial.

Os separadores organizam as informações da seguinte forma:

- O separador *Runs* contém uma lista de corridas disponíveis. Contudo, de momento, não existem funcionalidades disponíveis associadas às corridas;

- O separador *Fragments* contém uma lista de ficheiros de fragmentos disponíveis. Selecionando um ficheiro, é possível visualizar os fragmentos lá contidos e produzir caminhos usando o botão *Derive Paths*;

- O separador *Paths* contém uma lista de ficheiros de caminhos produzidos. Ao selecionar um destes e imprimindo o botão *Visualize Paths* é possível analisar os caminhos e aceder às funcionalidades de expandir, refazer e procurar um caminho.

8.2.1. Visualizar Fragmentos e Produzir Caminhos

Selecionando um ficheiro de fragmentos no menu inicial é possível:

- Observar cada um dos fragmentos que o compõe;

- Ter acesso às informações relativas à sua persistência, aos números de resíduos e de contactos, à primeira e à última ocorrência;

- Visualizar os resíduos envolventes ao fragmento;

- Visualizar, através de cores, a localização de cada resíduo face à estrutura secundária da proteína.

Na Figura 27 encontra-se o exemplo da visualização de um fragmento. Nesta é possível observar:

- Uma lista de fragmentos com algumas informações sobre estes e um painel em que o fragmento selecionado é mostrado;

- Na zona inferior, uma barra de persistência – neste caso, pode-se verificar que este fragmento persiste durante toda a corrida;

- Do lado direito, surge uma legenda de cores com duas funcionalidades cumulativas: visualização através de cores da localização do fragmento e visualização das ligações. A Figura 28 apresenta, para o fragmento usado como exemplo na Figura 27, a seleção de cada uma destas funcionalidades. O botão *Save JPG* permite guardar a imagem do fragmento que aparece no painel.

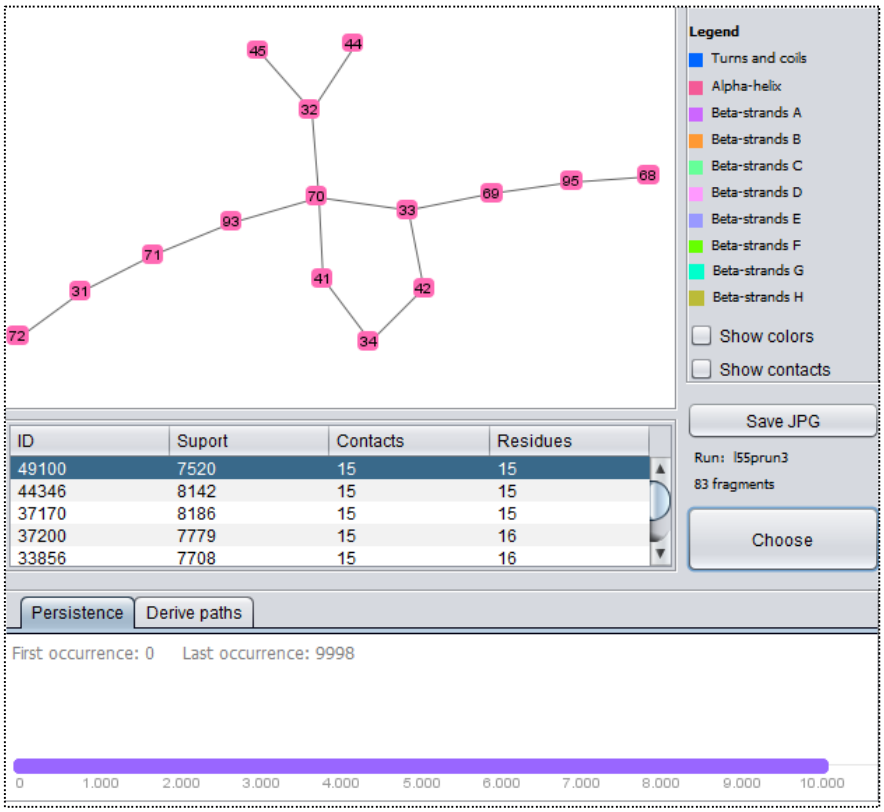


Figura 27 - Apresentação do menu de visualização e análise de fragmentos.

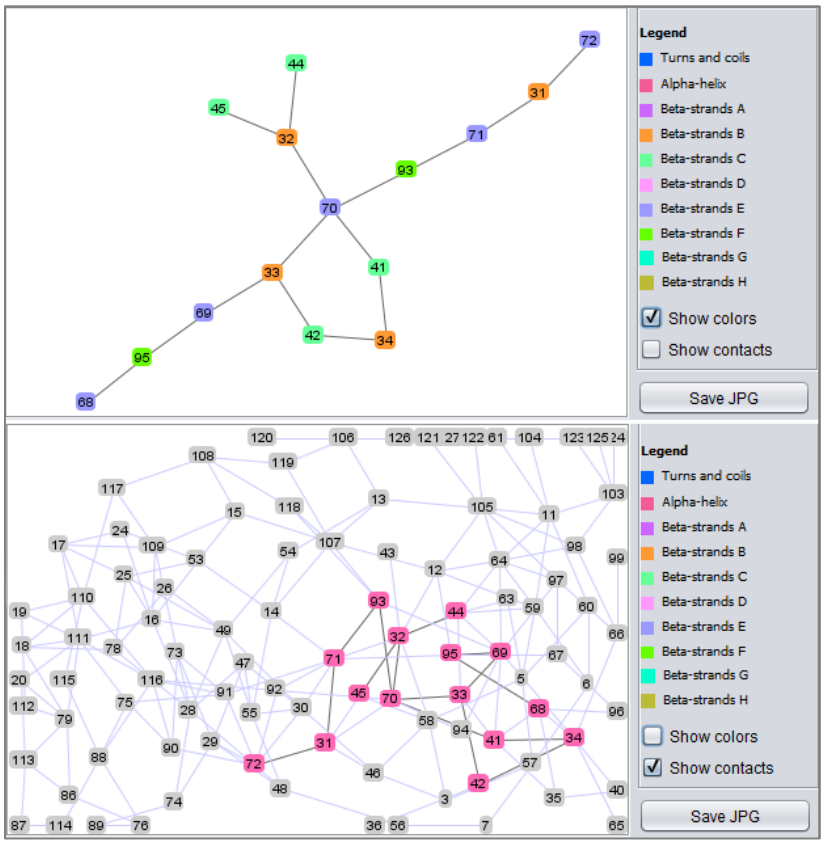


Figura 28 - Apresentação das funcionalidades de cores e de contactos.

Para produzir caminhos é necessário aceder ao separador *Derive paths*. Neste existe um formulário de informações que é necessário preencher. Estas informações são:

- Tamanho do salto e valor do *threshold*;
- Primeiro estado e tamanho de observação;
- Escolha de um fragmento final ou de resíduos para expansão;
- Decisão de produzir também um caminho de fragmentação.

Os dados inseridos poderão estar incorretos por incoerência entre si e/ou por incoerência relativamente às corridas, sendo nestes casos mostrada a respectiva mensagem de erro (ver anexo IV). Contudo, caso o preenchimento dos dados esteja correto, os caminhos são produzidos e é mostrada uma mensagem ao utilizador indicando o número de caminhos produzidos.

A Figura 29 mostra o exemplo de produção de caminhos na Subgraph Paths. Neste exemplo:

- O utilizador escolheu um tamanho de salto igual a 5 *frames* e um *threshold* igual a 70%;
- O caminho começará na *frame* 250 e foi selecionado um tamanho de observação igual a 500 *frames*. Assim, os caminhos resultantes terminarão na *frame* 750;
- Foi escolhida uma expansão sem direção;
- Foi escolhido o ficheiro *paths49100* para guardar os caminhos.

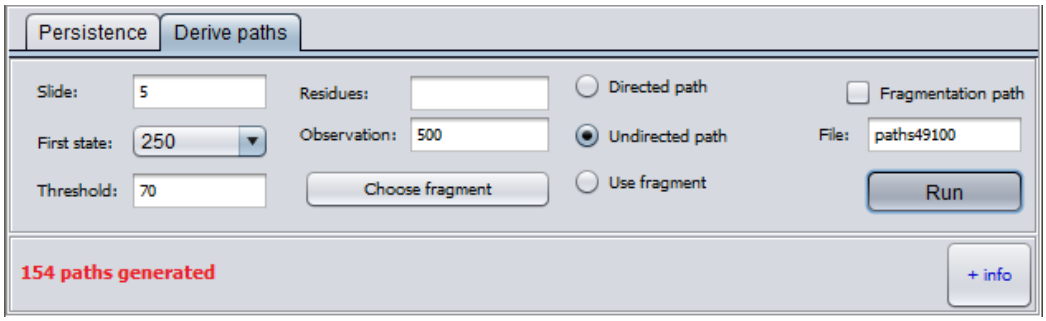


Figura 29 - Exemplo do preenchimento do formulário para a produção de caminhos.

Como resultado, temos que, no ficheiro *paths49100* foram guardados os 154 caminhos produzidos. Tal como a figura Figura 29 sugere, existe um botão + *info* que fornece mais informações acerca da funcionalidade de produção de caminhos.

Os caminhos produzidos poderão agora ser visualizados e analisados tal como será apresentado na próxima secção.

8.2.2. Visualização e Análise de Caminhos

Um ficheiro de caminhos poderá conter centenas ou até milhares de caminhos. Selecionando um ficheiro de caminhos e o botão *Visualize Paths* do menu inicial:

- É possível visualizar graficamente em pormenor cada um dos caminhos que o compõe e navegar pelos estados do caminho tanto no sentido de evolução, quanto de regressão temporal. Esta navegação é feita usando as setas do teclado (← e →);
- São mostradas informações acerca de cada caminho, nomeadamente o tamanho da observação, o número de estados, o tamanho de salto e de *threshold* usado, o suporte do fragmento inicial e os identificadores do fragmento e da corrida;
- São mostradas informações correspondentes a cada estado, nomeadamente o número do estado e da *frame* correspondente na corrida, o suporte do fragmento do estado e o número de contactos e de resíduos.

Os caminhos produzidos no exemplo da secção anterior poderão ser visualizados selecionado o ficheiro *paths49100* do separador *Paths* do menu inicial e o botão *Visualize Paths*. A Figura 30 mostra a janela que é aberta aquando da seleção do ficheiro *paths49100*.

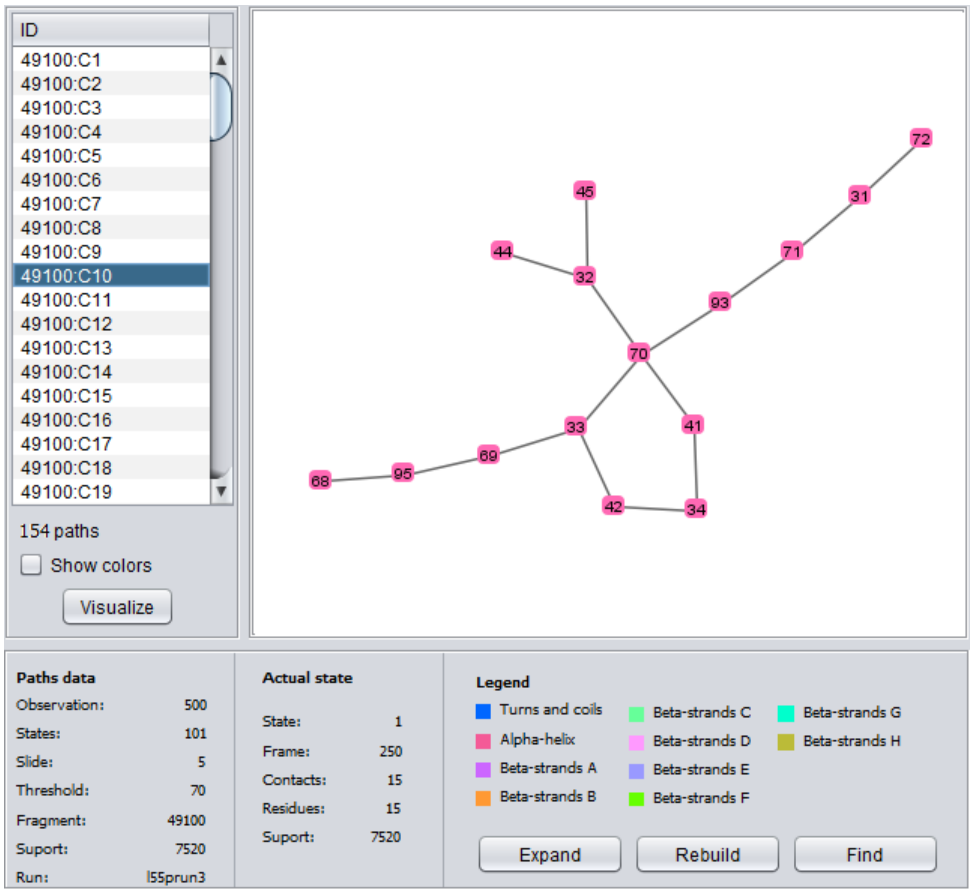


Figura 30 - Menu de visualização e análise dos caminhos.

Na Figura 30 é visível:

- Do lado esquerdo, a lista de todos os caminhos produzidos, e do lado direito um painel de visualização dos caminhos;
- Em baixo, um painel informativo sobre os caminhos e sobre o estado atual (notar que a imagem mostra que o primeiro estado corresponde à *frame* 250 e tem 15 contactos e 15 resíduos, sendo o seu suporte igual a 7520 *frames*).

Ainda relativo ao exemplo anterior, a Figura 31 mostra a visualização de dois estados: o estado 51, que corresponde à *frame* 500 da corrida e o estado 101, que é o último do caminho e que corresponde à *frame* 750 (valores que novamente correspondem totalmente aos esperados).

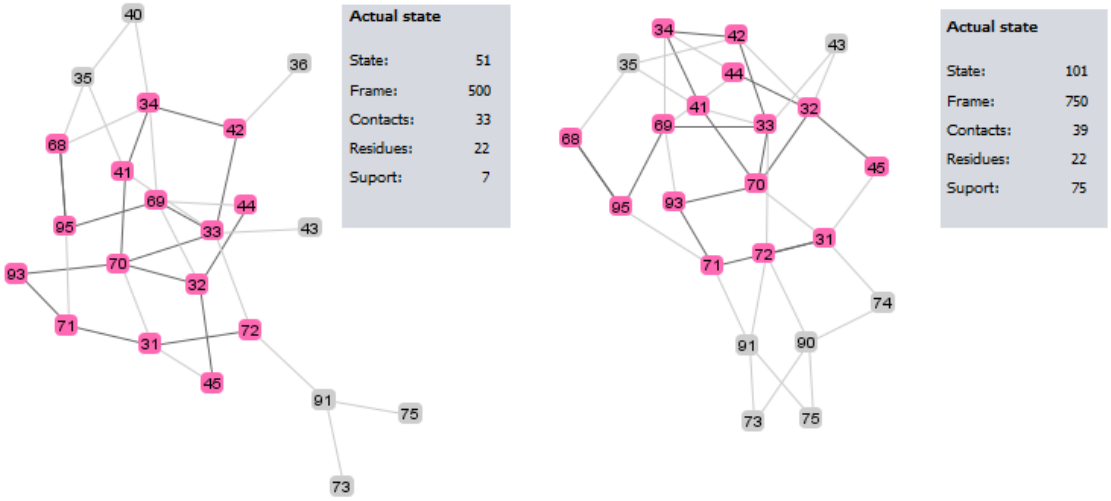


Figura 31 - Exemplo da visualização de um caminho em dois estados distintos, 51 e 101.

8.2.3. Funcionalidades Relativas a Caminhos

No menu de visualização de caminhos existem 3 funcionalidades disponíveis: expandir, refazer e procurar um caminho.

Expandir

Expandir um caminho consiste em prolongá-lo pelo seu fim. Para isto, o utilizador necessita de fornecer:

- Tamanho de observação;
- Resíduos usados para expandir o caminho (podem ser fornecidos códigos referentes às estruturas da proteína);
- Ficheiro em que serão guardados os caminhos.

Esta opção está disponível através botão *Expand* no qual é apresentado um menu tal como mostrado na Figura 32.

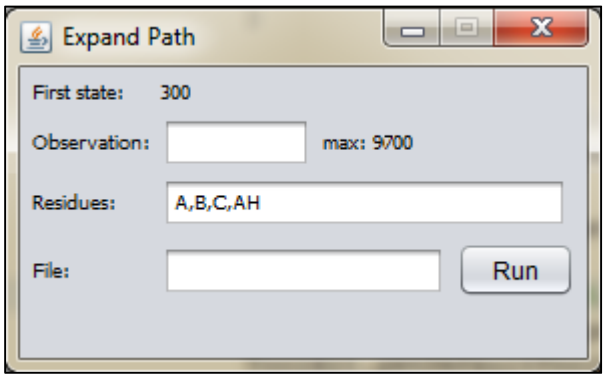


Figura 32 - Menu de expandir um caminho.

Refazer

Refazer um caminho consiste em, usando o mesmo grupo de resíduos de expansões e pela mesma ordem de expansão, considerar parte do espaço de observação do caminho usando diferentes valores de salto e/ou de *threshold*.

O objetivo é analisar de forma mais pormenorizada um intervalo particular da simulação até ao máximo da totalidade de um caminho. Para isto, o utilizador necessita de fornecer:

- Tamanho do salto e valor do *threshold*;
- Valor do primeiro estado e do tamanho de observação;
- Ficheiro em que serão guardados os caminhos.

Esta opção está disponível no botão *Rebuild* em que é apresentado um menu tal como mostrado na Figura 33.

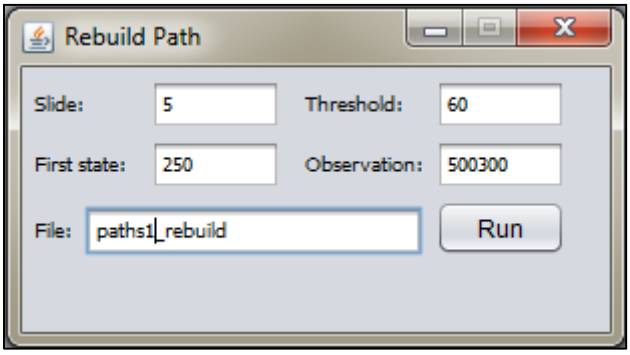


Figura 33 - Menu de refazer um caminho.

Procurar

Na procura de um caminho numa corrida, o utilizador apenas necessita de escolher, dentro de uma lista, a corrida pretendida. Após a procura é enviada uma mensagem a indicar o resultado. Esta funcionalidade de procura está disponível pelo botão *Find* em que é apresentado um menu tal como mostrado na Figura 34. O menu de procura conta também com um botão + *info,* de modo a assistir a interpretação dos resultados.

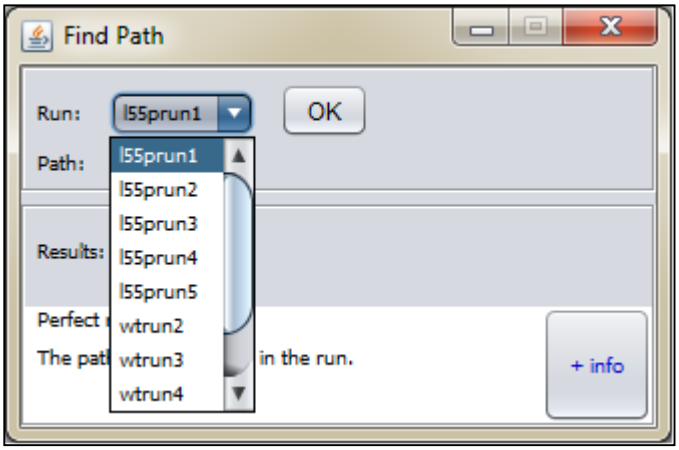


Figura 34 - Menu de procura de um caminho.

Caso a procura resulte em caminhos incompletos encontrados, são disponibilizados ao utilizador apenas os caminhos que passaram no teste. Para cada um destes caminhos é apresentando o valor da estatística do teste, o *p-value* e o número de contactos em falta. Adicionalmente, existe a possibilidade de selecionar cada um destes caminhos incompletos e visualizá-los individualmente (através do botão *Partial Path*). Este cenário encontra-se exemplificado na Figura 35.

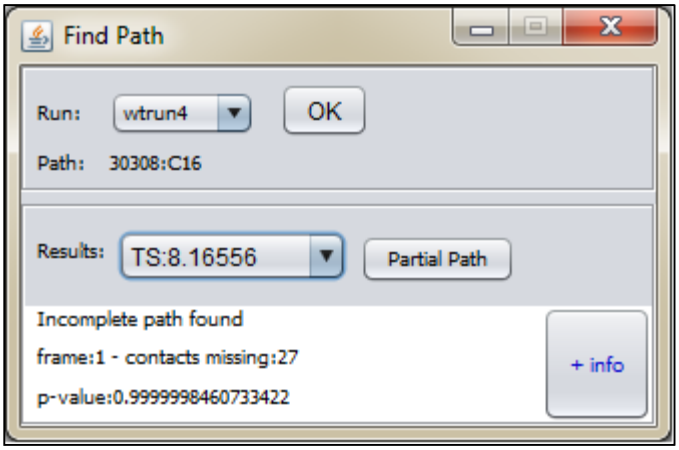


Figura 35 - Menu de procura de um caminho - caminhos incompletos.

8.2.4. Outras Funcionalidades

Para além do manual de utilizador, a Subgraph Paths conta ainda com um menu de ajuda. Este menu não somente apresenta a ferramenta, mas tem também como objetivo esclarecer as dúvidas do utilizador face ao projeto por trás da ferramenta, ao autor, às suas funcionalidades e às regras e conceitos envolvidos na elaboração de caminhos e da procura do caminho.

A Figura 36 apresenta a página de ajuda, nomeadamente o *Quick Start*.

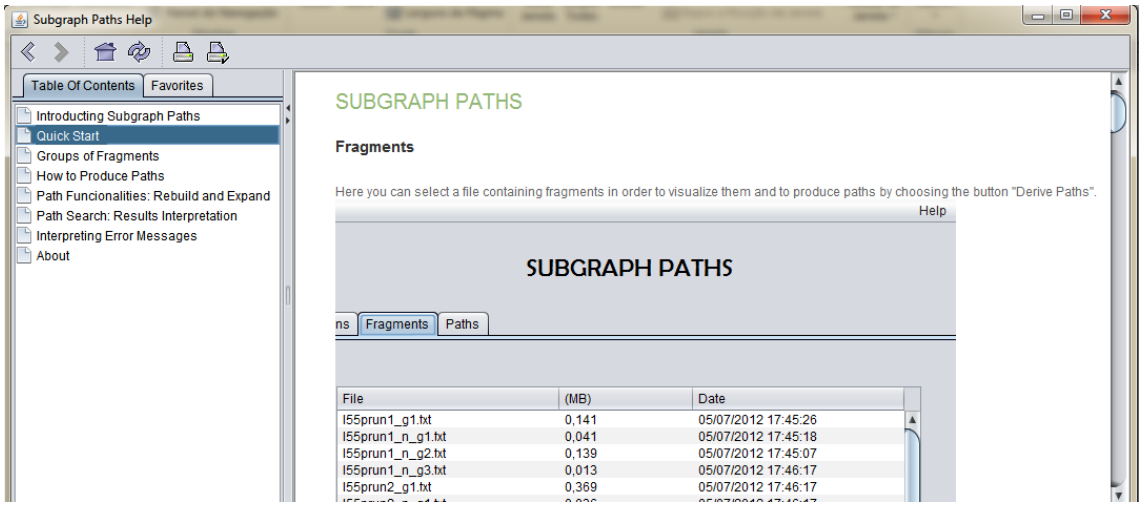


Figura 36 - Apresentação da Ajuda.

8.3. Exemplos da Utilização da Subgraph Paths

Nesta secção serão apresentados dois exemplos da utilização da ferramenta. O primeiro exemplo prevê a produção de caminhos para um fragmento extraído a alta frequência, e a escolha de um destes caminhos para procura nas 10 corridas disponíveis. O segundo exemplo considera a produção de um caminho de fragmentação para um fragmento não nativo, extraído a baixa frequência, e a procura deste nas restantes 10 simulações.

Traçar Caminhos para um Fragmento Nativo

Foi selecionado um fragmento frequente da corrida L55P3 referente à variante mais amiloidogénica. Este fragmento encontra-se representado na Figura 37. O fragmento é constituído por 8 resíduos e 8 contactos, tem frequência igual a 76,46%, encontra-se na primeira *frame* da corrida, e extingue-se na *frame* 9961.

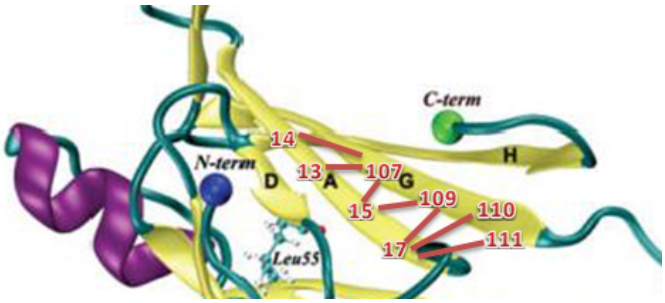


Figura 37- Fragmento extraído a alta frequência da corrida L55P3.

As figuras 38 e 39 foram retiradas da Subgraph Paths. A Figura 38 apresenta a visualização do fragmento e a Figura 39 apresenta o gráfico de persistência. Na Figura 38 os resíduos a azul correspondem aos que se encontram na cadeia G e os resíduos a roxo os que se encontram na cadeia A. As cores estão segundo a estrutura secundária da proteína.

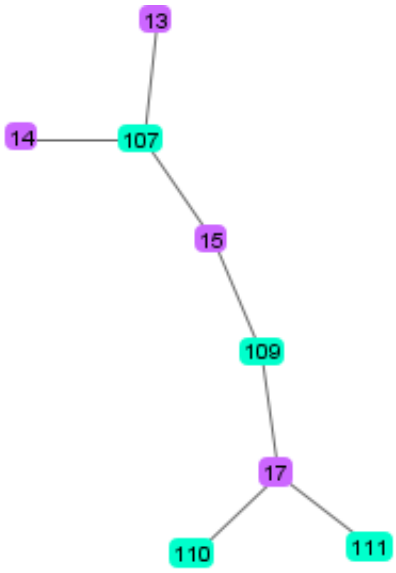


Figura 38 - Fragmento - Primeiro exemplo.

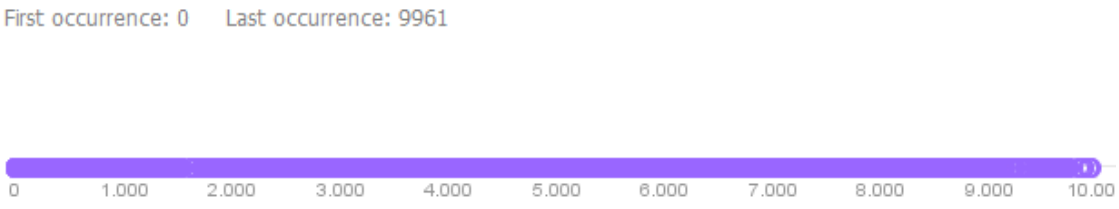


Figura 39 - Gráfico de persistência - Primeiro exemplo.

Foram produzidos caminhos para este fragmento, considerando os seguintes parâmetros:

- Começam na primeira *frame* da corrida;

- O tamanho de observação é igual a 300 *frames;*
- O valor do salto igual a 10 *frames*, resultando assim em 31 estados;
- O valor de *threshold* é de 80%;
- Para este caso de estudo, o objetivo é não apenas saber como as ligações dentro do fragmento evoluem, mas também como este fragmento interage com os resíduos das cadeias β A, G e H. Assim, serão considerados os resíduos destas cadeias como forma de expansão.

Como resultado foram produzidos 65 caminhos. Este é um número razoável de caminhos visto haver o agrupamento das expansões por resíduo. Porém este número aumentaria se fosse diminuído o tamanho do salto e/ou aumentado o tamanho de observação como mostra a Tabela 4. A Tabela 4 mostra a relação entre o número de caminhos produzidos e a variação do tamanho de salto e do tamanho da observação.

Número de caminhos produzidos	Tamanho do salto (em número de *frames*)	Tamanho da observação (em número de *frames*)
65	10	300
99	10	500
332	10	1000
95	5	300
129	5	500
510	5	1000

Tabela 4 - Relação entre o número de caminhos produzidos e o tamanho do salto e da observação.

Pela análise dos caminhos produzidos pode-se verificar:

- Durante toda a observação, o fragmento manteve-se sempre ligado a todos os resíduos das cadeias β onde pertence (cadeias A e G);
- Durante toda a observação, o fragmento manteve-se também sempre ligado a todos os resíduos da cadeia β H.

- Em alguns caminhos percebe-se o comportamento mais instável do resíduo Leucina, da posição 12 da cadeia polipeptídica. Este resíduo é o que aparece e desaparece mais vezes nos caminhos.

Foi selecionado um dos 65 caminhos produzidos, escolhido por mostrar bem a ligação existente entre o fragmento utilizado neste caso de estudo e os restantes resíduos das cadeias A, G e H. A procura tem como objectivo procurar este comportamento nas 10 corridas disponíveis. A Tabela 5 apresenta os resultados desta procura para cada corrida.

Tal como a Tabela 5 indica:

- Apenas na corrida L55P3 (corrida onde ocorreu a produção dos caminhos) é que o caminho foi encontrado de forma completa;
- Nas corridas L55P1 e L55P2 o caminho não foi encontrado nem incompleto.
- Nas restantes corridas, o caminho foi encontrado incompleto.
- Em termos dos caminhos incompletos, dado que o caminho selecionado para a procura um total de 617 contactos distribuídos por 31 etapas, o número de contactos em falta nos caminhos incompletos encontrados não é significativo.

Corrida	Resultado
L55P1	Não encontrado
L55P2	Não encontrado
L55P3	Encontrado e sincronizado
L55P4	Encontrado mas incompleto 14 contactos em falta
L55P5	Encontrado mas incompleto 5 contactos em falta
WT2	Encontrado mas incompleto 15 contactos em falta
WT3	Encontrado mas incompleto 7 contactos em falta
WT4	Encontrado mas incompleto 4 contactos em falta
WT5	Encontrado mas incompleto 10 contactos em falta
WT6	Encontrado mas incompleto 5 contactos em falta

Tabela 5 - Resultados da procura do caminho.

No anexo III - A encontra-se apresentado o caminho utilizado neste exemplo de procura.

Traçar Caminho de Fragmentação para um Fragmento Não Nativo

Foi selecionado um fragmento não nativo, extraído a baixa frequência da corrida L55P3. Este fragmento tem uma dimensão bastante reduzida - à semelhança do que acontece com a maior parte dos fragmentos não nativos. O fragmento é constituído por 3 resíduos e 2 ligações e está apresentado na Figura 40. O fragmento ocorre na *frame* 4711 e tem um suporte de 3538 *frames*.

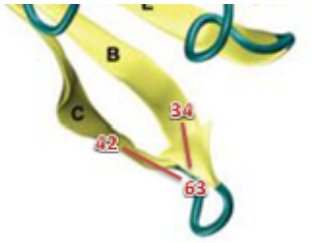


Figura 40 - Fragmento extraído a baixa frequência da corrida L55P3.

A Figura 41 apresenta o fragmento tal como este é visualizado na Subgraph Paths. O resíduo a laranja pertence à cadeia B, o resíduo a verde à cadeia C e o resíduo a azul não pertence a nenhuma cadeia e nem à α-hélice.

Figura 41 - Fragmento - Segundo exemplo.

Traçar um caminho de fragmentação vai permitir perceber como este fragmento se formou. Ou seja, que evolução teve a proteína que levou à formação na *frame* 4711 deste fragmento não nativo. Este caminho foi produzido com tamanho de salto igual a 50 *frames*, resultando portanto em 96 estados. O valor de *threshold* escolhido é 80%.

O caminho produzido encontra-se parcialmente apresentado no anexo III – B. Pela análise do caminho pode-se verificar que o fragmento é formado pela proximidade

existente entre as cadeias β B,C e E: os resíduos 34 e 42, por pertencem respetivamente às cadeias B e C, têm naturalmente, pela proximidade, ligações com os resíduos da cadeia E. Por sua vez, os resíduos da cadeia E têm ligações com o resíduo 63 pela sua ordem na cadeia polipeptídica. Esta proximidade faz com que o resíduo 63 venha a criar ligações com os resíduos das cadeias B e C.

Foi efetuada a procura deste caminho de fragmentação nas restantes corridas. O resultado desta procura encontra-se na Tabela 6.

Corrida	Resultado
L55P1	Não encontrado – *frame* 5427
L55P2	Não encontrado – *frame* 9343
L55P3	Encontrado e sincronizado.
L55P4	Não encontrado – *frame* 6334.
L55P5	Não encontrado – fragmento não encontrado
WT2	Não encontrado – fragmento não encontrado
WT3	Não encontrado – *frame* 4507.
WT4	Não encontrado – fragmento não encontrado
WT5	Não encontrado – *frame* 5245.
WT6	Não encontrado – fragmento não encontrado

Tabela 6- Resultados da procura do caminho de fragmentação.

A Tabela 6- Resultados da procura do caminho de fragmentação. mostra que o caminho de fragmentação apenas aparece na corrida em que foi extraído, sendo que nas corridas L55P5, WT2, WT4 e WT6 o fragmento nem sequer está presente. Nas restantes corridas – L55P1, L55P2, L55P4, WT3 e WT6 – o fragmento encontra-se presente, mas em *frames* longínquas face à primeira ocorrência original, o que faz com que o caminho seja considerado como não encontrado.

8.4. Conclusão

A Subgraph Paths em termos de tecnologias incorporadas faz um uso *standard* das tecnologias actuais e os algoritmos são relativamente fáceis de implementar após definidos todos os conceitos associados. Contudo, pelas funcionalidades que oferece e

pelos conceitos inerentes a estas, torna-se uma ferramenta bastante interessante para a análise do comportamento da Transtirretina e de qualquer outra proteína.

De forma geral, os menus são todos simples e intuitivos, usando termos simples e uma linguagem visual igualmente simples, tal como era desejado pelos utilizadores da ferramenta. Além disto, a visualização gráfica dos fragmentos e dos caminhos, bem como a interação em tempo real ficou muito bem conseguida graças às funcionalidades disponibilizadas pelo Prefuse. A Subgraph Paths alcança assim os objetivos previstos de expor o comportamento da Transtirretina ao longo do processo de desnaturação. Adicionalmente, conta com outras funcionalidades como são a possibilidade de refazer e de expandir um caminho considerado interessante, de modo a poder observá-lo usando diferentes parâmetros.

Em relação à procura de um caminho numa corrida, esta é feita num menu muito simples. O utilizador apenas escolhe a corrida em que ocorrerá a procura e é mostrada uma mensagem a indicar o resultado desta procura. Assim, o utilizador não fica a par do algoritmo usado, parecendo portanto um processo de procura rápido e simples. O algoritmo de procura segue todos os prossupostos e procedimentos descritos no capítulo 7.

Em termos do menu de ajuda, este está bastante bem organizado, com conteúdo pertinente para os utilizadores, visualmente atrativo e que utiliza tecnologias muito recentes como HTLM 5 e CSS revelando-se, portanto, a JavaHelp System uma escolha acertada.

Capítulo 9

Caso de Estudo

Com o objetivo de demonstrar as potencialidades da ferramenta Subgraph Paths em extrair conhecimento acerca do processo de desenrolamento de proteínas, nomeadamente a partir da produção e visualização de caminhos de evolução de conjuntos de contactos nativos de proteínas e também pela procura de caminhos em múltiplas simulações de dinâmica molecular, foi elaborado um caso de estudo.

Este estudo foi realizado em colaboração com os investigadores Rui M. M. Brito e Cândida G. Silva. Foi utilizada uma plataforma computacional Quad-Core i5-460M, com velocidade de processamento de 2.53 GHZ com Turbo Boost até 2,80GHZ e 4 GB de memória RAM.

9.1. Apresentação do Caso de Estudo

Pretende-se analisar a estabilidade da proteína Transtirretina nativa e da sua variante mutada Leu55Pro através da evolução dos contactos nativos ao longo de várias simulações. Para isto foram analisados os fragmentos frequentes disponíveis e foram traçados caminhos para alguns fragmentos.

9.2. Resultados Obtidos

Pela utilização da Subgraph Paths foi possível a obtenção de resultados que discriminam o comportamento da proteína nativa e mutada.

Dos resultados obtidos, verifica-se que maior parte dos fragmentos descrevem contactos nativos entre resíduos nas cadeias β CBEF. Além disso, observa-se também que os fragmentos obtidos para as simulações da WT são constituídos por mais resíduos do que a L55P (ver Figura 20 e anexo II). É também possível constatar que muitos dos fragmentos obtidos para a L55P são "subfragmentos" da WT. Estas observações parecem indicar que a folha β CBEF é mais estável na WT do que na L55P, o que está de acordo com resultados computacionais e experimentais descritos por diversos autores [Liu 2000, Liu 2002, Rodrigues 2010].

O afastamento as cadeias β C e D e consequente exposição da interface formada pelas cadeias β A e B é um passo importante para a agregação monómero-monómero na TTR. Seguindo uma análise distinta, Rodrigues e colaboradores em [Rodrigues 2010] concluíram que este evento é muito precoce na L55P e mais tardio na WT. De facto, contactos nativos persistentes ao longo da simulação entre as cadeias β D e A apenas são observados em duas simulações da WT (grupos 7 e 9 de fragmentos) e em nenhuma das simulações da L55P. Por outro lado, sabe-se que a cadeia β C (resíduos 40 a 49) se vai afastando da cadeia β B a partir do extremo C-terminal (resíduo 49) até aproximadamente aos resíduos na posição 43 e 44 [Liu 2000]. Analisando os fragmentos obtidos é possível verificar que os contactos nativos persistentes entre resíduos nestas duas cadeias β se observam maioritariamente entre os resíduos 40 a 44 da cadeia β C e resíduos da cadeia β B (grupos 1 a 3), e que estes contactos nativos são persistentes em todas as simulações da WT mas apenas em três das simulações da L55P.

Foram traçados caminhos para 2 fragmentos. O primeiro fragmento foi escolhido da corrida WT4 e é constituído por resíduos das cadeias β A,G e D. O segundo fragmento foi escolhido da corrida L55P2 e é um subfragmento do primeiro.

O primeiro fragmento encontra-se representado na Figura 42: do lado esquerdo encontra-se o fragmento enquadrado na estrutura da proteína e do lado direito apresenta-se o fragmento tal como este é visualizado na Subgraph Paths. O fragmento é constituído por 10 resíduos e 9 contactos, tem frequência igual a 73,96%, encontra-se na primeira *frame* da corrida, e extingue-se na *frame* 9999. Os resíduos a azul correspondem aos que se encontram na cadeia β G, os resíduos a roxo aos que se encontram na cadeia β A e os resíduos a cor-de-rosa aos que se encontram na cadeia β D. As cores estão segundo a estrutura secundária da proteína.

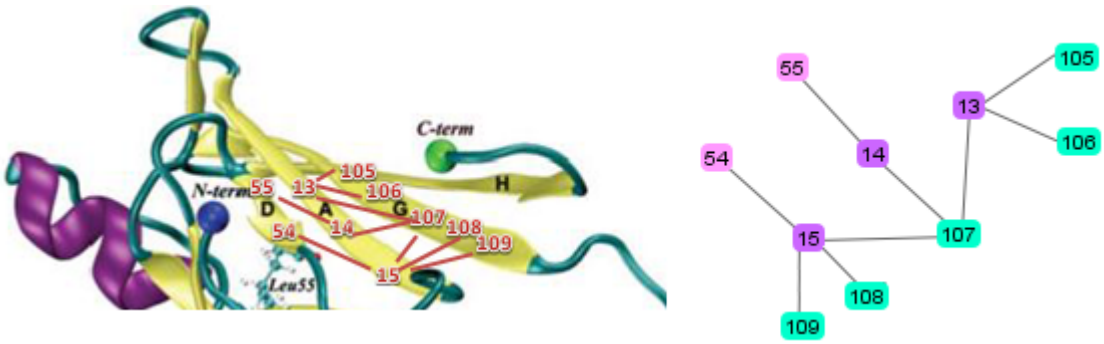


Figura 42 - Fragmento 32344 - WT4

O fragmento seleccionado para a L55P é "subfragmento" do fragmento escolhido da corrida *Wild-Type*. Este fragmento encontra-se representado na Figura 43. O fragmento é constituído por 5 resíduos e 4 contactos, tem frequência igual a 73,56%, encontra-se na primeira *frame* da corrida, e extingue-se na *frame* 8279.

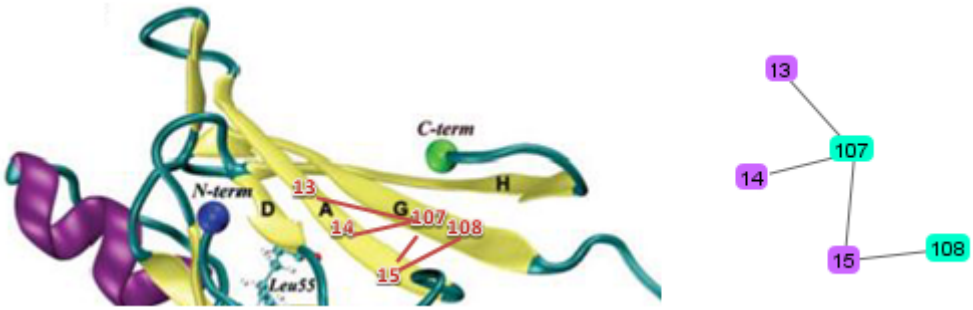


Figura 43- Fragmento 30144 – L55P2

Para ambos os fragmentos, foram produzidos caminhos considerando os mesmos parâmetros. O objetivo é perceber diferenças no comportamento das duas variantes da TTR. Os parâmetros são:

- Começam na primeira *frame* da corrida e o tamanho de observação é igual a 3000 *frames;*
- O valor do salto igual a 10 *frames*, resultando assim em 301 estados;
- O valor de *threshold* é de 70%;
- Para este caso de estudo, o objetivo é não apenas saber como as ligações dentro do fragmento evoluem, mas também como este fragmento interage com os resíduos das cadeias β D, A e G. Assim, serão considerados os resíduos destas cadeias como forma de expansão.

Como resultado foram produzidos 61 caminhos para o primeiro fragmento (pertencente à corrida WT4) e 64 caminhos para o segundo fragmento (pertencente à corrida L55P2). Para o fragmento da WT, cada caminho tem, em média, um total de 5304 contactos e 4426 resíduos. Enquanto os caminhos do fragmento da L55P têm, em média, um total 4227 contactos e 4086 resíduos. Assim, analisando os caminhos gerados para estes dois fragmentos, facilmente se constata que os caminhos na WT são mais ricos - no sentido em que envolvem mais resíduos- e tendem a manter-se por períodos de tempo mais longos e com maior suporte (threshold), o que indica uma maior estabilidade da proteína.

A Figura 44 apresenta o último estado de dois caminhos. À esquerda o último estado de um caminho produzido para o fragmento da L55P2 e à direita o último estado de um caminho produzido para o fragmento da WT4. O exemplo da Figura 44 é ilustrativo de como na corrida WT4 envolvem mais resíduos e mais contactos entre os resíduos.

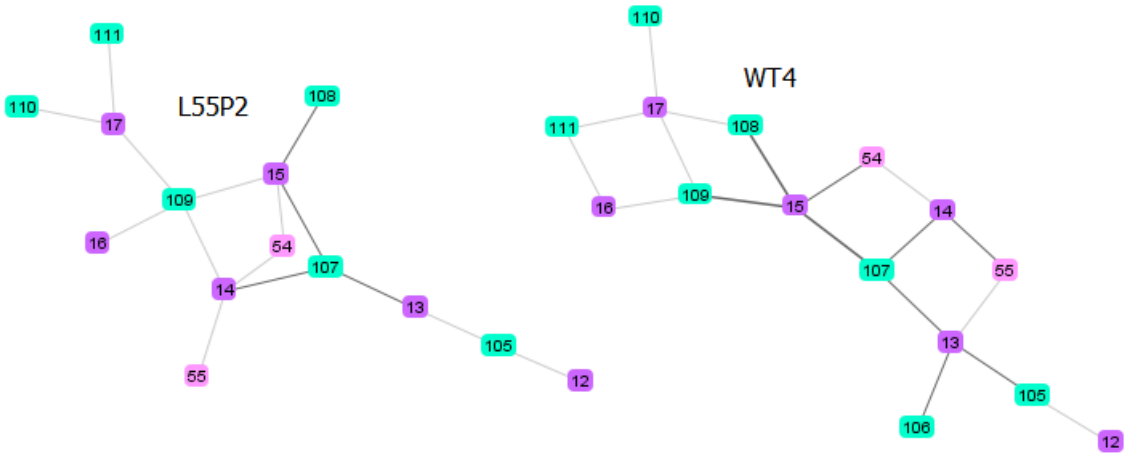


Figura 44 - Último estado de dois caminhos: à esquerda o último estado de um caminho da L55P2 e à direita da WT4.

Adicionalmente, os caminhos do fragmento da WT4 foram procurados nas corridas da variante Leu55Pro e os caminhos do fragmento da L55P2 foram procurados nas corridas WT.

Os resultados da procura indicam que nenhum caminho traçado para o fragmento da WT4 foi encontrado nas corridas L55P. Por outro lado, os caminhos da L55P2 foram encontrados incompletos em algumas corridas WT. A Figura 45 apresenta um estado de um caminho incompleto encontrado na WT4. Desenhados a vermelho encontram-se os

contactos/resíduos em falta no caminho – neste caso, falta a ligação do resíduo 107 com o 12.

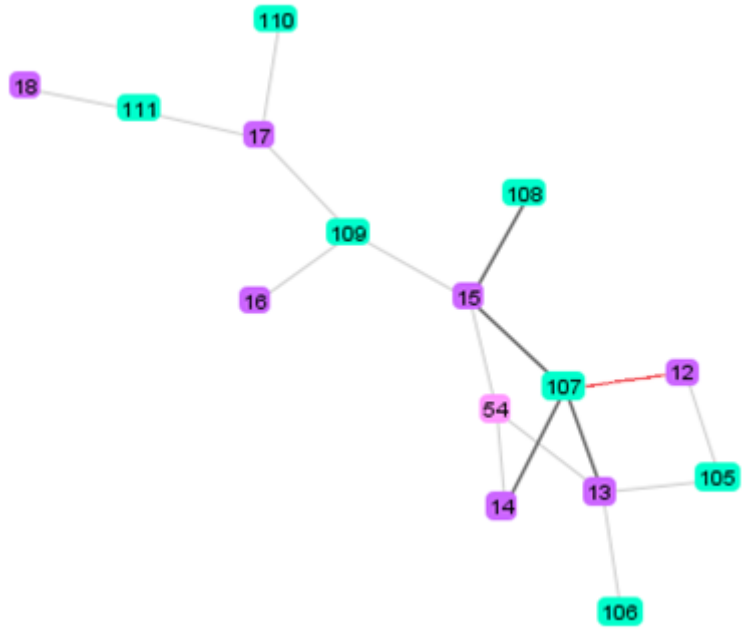


Figura 45 - Estado de um caminho incompleto de procura.

Estes resultados permitem-nos concluir que as corridas WT têm caminhos/fragmentos mais específicos que não se conseguem reproduzir nas corridas L55P.

9.3. Conclusão

Tanto pela observação, como pela produção de caminhos é possível extrair resultados interessantes acerca do processo de desenrolamento da Transtirretina. Como os caminhos produzidos são gerados com base nas preferências do utilizador relativamente ao modo de expansão, ao tamanho de observação, de salto e de *threshold*, estes exploram exatamente o tipo de informação que o utilizador pretende.

Os resultados do caso de estudo traduzem-se em conhecimento retirado das simulações de dinâmica molecular da Transtirretina, nomeadamente através da análise de fragmentos e de caminhos traçados a partir de fragmentos frequentes e pela procura destes caminhos noutras simulações. Dado o elevado número de fragmentos extraídos das corridas, todo o processo desenvolvido para estes casos de estudo poderá ser aplicados a muitos destes fragmentos. Estes estudos resultam assim num elevado volume de conhecimento retirado das simulações.

Capítulo 10

Conclusão e Trabalho Futuro

Já várias conclusões foram ditadas ao longo da dissertação no final dos capítulos. Contudo, serão apresentadas em seguida conclusões face ao trabalho realizado ao longo da tese como um todo. Estas apresentam os pontos fortes e mais fracos do trabalho, as principais dificuldades encontradas e reflexões sobre os resultados finais obtidos. Adicionalmente, a secção 10.1 apresentará todas as ideias e projetos existentes para trabalho futuro.

O desenvolvimento de uma ferramenta que permita assistir investigadores em estudos sobre o comportamento de uma proteína envolve competências ao nível do desenvolvimento de *software* por parte do *developer*. Contudo, exige também que este estude todos os termos, conceitos e dinâmicas associadas às proteínas para a compreensão do problema e envolvimento total na sua resolução. Assim requer um considerável investimento de tempo.

A extração de subgrafos frequentes, designados de fragmentos, usando a biblioteca ParMol permitiu, variando diversos parâmetros, personalizar a extração, adequando os fragmentos aos requisitos pretendidos. Contudo, não permite ter em conta a significância estatística destes, o que seria bastante interessante. A questão da similaridade nesta fase não poderia ser resolvida pela complexidade inerente à integração de um sistema que tenha em conta a significância estatística. Isto deve-se a não haver nenhuma ferramenta ou biblioteca disponível compatível com os requisitos exigidos para a extração de subgrafos frequentes. Esta situação levaria ao

desenvolvimento de raiz de tal plataforma. Contudo, a escolha do ParMol mostrou-se acertada na medida em que permitiu uma extração rápida, eficiente e personalizada. Além disto, o ParMol, por ter tanta documentação associada e o código estar disponível sob a GPL, torna-se fácil de utilizar.

A Subgraph Paths permite, então, não só visualizar graficamente os fragmentos frequentes, mas também traçar e visualizar caminhos. A análise dos fragmentos é completa, pois mostra os dados mais relevantes sobre estes, para além de permitir visualizar não só a sua localização relativamente à estrutura secundária da proteína mas também as ligações envolventes ao fragmento. Além disto, a produção de caminhos é bastante personalizável e segue todos os requisitos propostos, sendo, deste modo, um ponto bem concretizado.

Em termos de análise dos caminhos, é possível navegar temporalmente nestes e ter acesso em tempo real aos dados de cada estado. É também possível refazer e expandir um caminho, de modo a perceber mais detalhadamente o comportamento daquela parte da simulação da proteína. Adicionalmente é possível procurar cada um dos caminhos produzidos em todas as corridas disponíveis. A procura de um caminho numa corrida é feita de forma bastante simples e os dados obtidos permitem comparar o comportamento de diferentes variantes da proteína.

A Subgraph Paths mostra-se então interessante na exposição do comportamento da proteína, quer pelo estudo de cada simulação isoladamente, quer pelo cruzamento dos dados das simulações na procura de caminhos.

10.1.Trabalho Futuro

Como trabalho futuro existem diversas variantes de análise, nomeadamente em termos de melhoramento da qualidade dos fragmentos extraídos que servem de base para a produção de caminhos, de melhoramento das funcionalidades da ferramenta e de alargamento desta a outras patologias.

Os fragmentos obtidos pelo ParMol têm como característica de extração a frequência, não considerando assim a significância dos fragmentos. Contudo, sabe-se que, em alguns casos, os fragmentos significativos podem não ser os que ocorrem a frequências mais elevadas. Neste trabalho, a extração de fragmentos usando duas gamas de

frequências, uma mais baixa e outra mais alta, veio de certo modo colmatar esta falha, de modo a captar este tipo de fragmentos. Contudo, seria de interessante, futuramente, a elaboração de um estudo sobre a significância destes fragmentos. Este estudo poderia resultar, por exemplo, num filtro sobre os fragmentos já extraídos e em que estes seriam escolhidos pela significância.

Relativamente à ferramenta, existe uma série de melhoramentos que serão futuramente feitos, nomeadamente a ligação da ferramenta a um servidor onde não só estarão alojados os fragmentos, as corridas e os caminhos, mas também onde correrão os algoritmos de produção de caminhos. Posteriormente, serão também introduzidas novas funcionalidades ainda por debater. Uma destas, em princípio, será usando o Chimera [Pettersen 2004], possibilitar a visualização dos resíduos de um fragmento ou de um caminho num monómero da proteína de forma tridimensional.

À semelhança do que foi ditado na introdução da dissertação, espera-se que a ferramenta possa ser utilizada para estudar outras patologias para além da Paramiloidose, nomeadamente outras doenças do grupo das amiloidoses. Assim sendo, um dos projetos futuros será a extração de fragmentos frequentes das simulações das proteínas associadas a essas patologias e utilização desses fragmentos na Subgraph Paths. O mesmo se aplica para outras variantes da TTR (por exemplo, para a Val30Met).

Anexos

Anexo I – Proteínas

A. Funções das Proteínas

Função	Descrição	Exemplo
Enzimática	Catalisadores biológicos - são moléculas que aumentam a velocidade de uma reacção, diminuindo a energia de activação.	Enzimas
Transporte	Ligam-se e transportam especificamente moléculas de um órgão para o outro.	Hemoglobina
Nutritiva	Proteínas com alto teor nutritivo.	Albumina
Estrutural	Servem como filamentos, cabos ou lâminas para dar firmeza ou protecção a estruturas biológicas	Colagénio
Defesa	Defendem o organismo de invasões.	Imunoglobulina
Hormonal	Ajudam a regular a atividade celular ou fisiológica.	Insulina

Tabela 7 - Principais funções das proteínas (fonte: About.com Biology[12]).

[12] Informações em http://biology.about.com/od/molecularbiology/Molecular_Biology.htm.

B. Tabela de aminoácidos

Família do Aminoácido	Designação	Símbolo com 3 letras
Básico	Lisina	LYS
	Arginina	ARG
	Histidina	HIS
Ácido	Ácido Aspártico	ASP
	Ácido Glutâmico	GLU
Polar (hidrofólios)	Asparagina	ASN
	Glutamina	GLN
	Serina	SER
	Treonina	THR
	Tirosina	TYR
Não Polar (hidrofóbicos)	Triptofano	TRP
	Glicina	GLY
	Valina	VAL
	Leucina	LEU
	Isoleucina	ILE
	Prolina	PRO
	Fenilalanina	PHE
	Metionina	MET
	Alanina	ALA
	Cisteína	CYS

Tabela 8 - Tabela de aminoácidos [Bray 1994].

C. Identificação dos Aminoácidos presente na Transtirretina

Número de ordem	Aminoácido	Número de ordem	Aminoácido	Número de ordem	Aminoácido
1	GLY	44	PHE	87	PHE
2	PRO	45	ALA	88	HSD
3	THR	46	SER	89	GLU
4	GLY	47	SLY	90	HSD
5	THR	48	LYS	91	ALA
6	GLY	49	THR	92	GLU
7	GLU	50	SER	93	VAL
8	SER	51	GLU	94	VAL
9	LYS	52	SER	95	PHE
10	CYS	53	GLY	96	THR
11	PRO	54	GLU	97	ALA
12	LEU	55	LEU	98	ASN
13	MET	56	HSD	99	ASP
14	VAL	57	GLY	100	SER
15	LYS	58	LEU	101	GLY
16	VAL	59	THR	102	PRO
17	LEU	60	THR	103	ARG
18	ASP	61	GLU	104	ARG
19	ALA	62	GLU	105	TYR
20	VAL	63	GLU	106	THR
21	ARG	64	PHE	107	ILE
22	GLY	65	VAL	108	ALA
23	SER	66	GLU	109	ALA
24	PRO	67	GLY	110	LEU
25	ALA	68	ILE	111	LEU
26	ILE	69	TYR	112	SER
27	ASN	70	LYS	113	PRO
28	VAL	71	VAL	114	TYR
29	ALA	72	GLU	115	SER

30	VAL	73	ILE	116	TYR
31	HSD	74	ASP	117	SER
32	VAL	75	THR	118	THR
33	PHE	76	LYS	119	THR
34	ARG	77	SER	120	ALA
35	LYS	78	TYS	121	VAL
36	ALA	79	TRP	122	VAL
37	ALA	80	LYS	123	THR
38	ASP	81	ALA	124	ASN
39	ASP	82	LEU	125	PRO
40	THR	83	GLY	126	LYS
41	TRP	84	ILE	127	GLU
42	GLU	85	SER		
43	PRO	86	PRO		

Tabela 9 - Identificação dos aminoácidos presentes na TRR em relação ao seu número de ordem na cadeia polipeptídica inicial.

D. Variantes da Transtirretina

Variante	Alteração	Fenótipo	Focos geográficos (origem étnica)
Gly6Ser	CGT→AGT	Não amiloidogénica	Caucasiana
Cys10Arg	TGT→CGT	NA, O, C, PN	EUA (Húngara)
Leu12Pro	CTG→CCG	NA, SNC, L, LM	Reino Unido
Met13Ile	ATG→ATC	Não amiloidogénica	Alemanha
Asp18Asn	GAT→AAT	C	EUA
Asp18Glu	GAT→GAA/G	NA, PN	América do Sul
Aps18Gly	GAT→GGT	SNC, LM	Hungria
Val20Ile	GTC→ATC	STC, C	Alemanha, EUA
Ser23Asn	AGT→AAT	O, C, PN	EUA (Portuguesa)
Pro24Ser	CCT→TCT	STC, C, PN	EUA
Ala25Ser	GCC→TCC	C, PN	EUA
Ala25Thr	GCC→ACC	SNS, PN	Japão
Val28Met	GTG→ATG	NA, PN	Portugal
Val30Ala	GTG→GCG	NA,C	EUA (Alemã)
Val30Gly	GTG→GGG	SNS,O,LM	EUA
Val30Leu	GTG→CTG	NA, C, R, PN	Japão, EUA
Val30Met	GTG→ATG	NA, O, LM, PN	Argentina, Brasil, China, Finlândia, França, Alemanha, Grécia, Itália, Japão, Portugal, Suécia, Turquia, EUA
Phe33Cys	TTC→TGC	STC, O, R, C	EUA
Phe33Ile	TTC→ATC	O, PN	Israel (Polaca)
Phe33Leu	TTC→CTC	NA, PN	EUA (Polaca e Lituana)
Phe33Val	TTC→GTC	NA, PN	China, Reino Unido
Arg34Thr	AGA→ACA	C, PN	Itália
Lys25Asn	AAG→AAC/T	NA, C, PN	França
Ala36Pro	GCT→CCT	SNC, STC, O, PN	Grécia, Itália, EUA
Asp38Ala	GAT→GCT	NA, C, PN	Japão
Trp41Leu	TGG→TTG	O	EUA (Russa)
Glu42Asp	GAG→GAC/T	C	França

Glu42Gly	GAG→GGG	NA, C, PN	Japão, Rússia, EUA
Phe44Ser	TTT→TCT	NA, O, C, PN	EUA, Japão
Ala45Asp	GCC→GAC	C, PN	Irlanda, Itália, EUA
Ala45Ser	GCC→TCC	C	Suécia
Ala45Thr	GCC→ACC	C	Irlanda, Itália, EUA
Gly47Ala	GGG→GCG	NA, C, PN	Alemanha, Itália, França
Gly47Arg	GGG→C/AGG	NA, PN	Japão
Gly47Glu	GGG→GAG	C, R, PN	Alemanha, Itália
Gly47Val	GGG→GTG	NA, STC, C, PN	Sri Lanka
Thr49Ala	ACC→GCC	STC, C, PN	França, Itália
Thr49Ile	ACC→ATC	C, PN	Japão
Thr49Pro	ACC→AGG	C	EUA
Ser50Arg	AGT→AGG	NA, C, PN	França, Itália, Japão
Ser50Ile	AGT→ATT	NA, C, PN	Japão
Glu51Gly	GAG→GGG	C	EUA
Ser52Pro	TCT→CCT	NA, C, R, PN	Inglaterra
Gly53Glu	GGA→GAA	SNC, LM, N	França
Glu54Gly	GAG→GGG	NA, O, PN	Inglaterra
Glu54Lys	GAG→AAG	NA, B, PN	Japão
Leu55Arg	CTG→CGG	LM, PN	Alemanha
Leu55Gln	CTG→GAG	NA, O, PN	EUA (Espanhola)
Leu55Pro	CTG→CCG	NA, O, C, PN	Taiwan, EUA (Holandesa, Alemã)
His56Arg	CAT→AAT	C	EUA
Leu58Arg	CTC→CGC	NA, STC, O, C	Japão
Leu58His	CTC→CAC	STC, C	Alemanha, EUA
Thr59Lys	ACA→AAA	NA, C, PN	Itália, EUA (Chinesa)
Thr60Ala	ACT→GCT	STC, C, PN	Austrália, Alemanha, Irlanda, Reino Unido, EUA, Japão
Glu61Lys	GAG→AAG	PN	Japão
Phe64Leu	TTT→CTT	STC, C, PN	Itália, EUA
Phe64Ser	TTT→TCT	SNC, E, LM, PN	Canada (Italiana), Inglaterra
Ile68Leu	ATA→T/CTA	C	Alemanha, EUA

Tyr69His	TAC→CAC	E	Escócia, EUA
Tyr69Ile	TAC→ATC	STC, C	Japão
Lys70Asn	AAA→AAC/T	STC, O, PN	Alemanha, EUA
Val71Ala	GTG→GCG	STC, O, PN	França, Espanha
Ile73Val	ATA→GTA	NA, PN	Bangladeche
Asp74His	GAC→CAC	Não amiloidogénica	Alemanha
Ser77Phe	TCT→TTT	NA, PN	França
Ser77TyR	TCT→TAT	C, R, PN	França, Alemanha, EUA
Tyr78Phe	TAC→TTC	PN, STC, P	França (Italiana)
Ala81Thr	GCA→ACA	C	EUA
Ile84Asn	ATC→AAC	STC, E, C	EUA (Italiana)
Ile84Ser	ATC→AGC	STC, E, C, LM	Hungria, Suíça, EUA
Ile84Thr	ATC→CAG	NA, C, PN	Alemanha, Reino Unido
Glu89Gln	GAG→CAG	CTC, C, PN	Itália
Glu89Lys	GAG→AAG	NA, C, PN	EUA
His90Asn	CAT→AAT	Não amiloidogénica	Alemanha, Portugal
Ala91Ser	GCA→TCA	NA, STC, C, PN	França
Gln92Lys	GAG→AAG	C	Japão
Ala97Gly	GCC→GGC	C, PN	Japão
Ala97Ser	GCC→TCC	C, PN	China, França, Taiwan
Gly101Ser	GGC→AGC	Não amiloidogénica	Japão
Pro102Arg	CCC→CGC	Não amiloidogénica	Alemanha
Arg103Ser	CGC→AGC	C	EUA
Arg104Cys	CGC→TGC	Não amiloidogénica	EUA
Arg104His	CGC→CAC	Não amiloidogénica	Japão, EUA (Chinesa)
Ile107Met	ATT→ATG	C, PN	Alemanha
Ile107Val	ATT→GTT	STC, C, PN	Alemanha, EUA, Japão
Ala108Ala	GCC→GCT	Não amiloidogénica	Portugal
Ala109Ser	GCC→TCC	PN	Japão
Ala109Thr	GCC→ACC	Não amiloidogénica	Portugal
Ala109Val	GCC→GTC	Não amiloidogénica	EUA
Leu111Met	CTG→ATG	STC, C	Dinamarca
Ser112Ile	AGC→ATC	C, PN	Itália

Tyr114Cys	TAC→TGC	NA, SNC, O, C, LM, PN	Japão
Tyr114His	TAC→CAC	CNS	Japão
Tyr116Ser	TAT→TCT	NA, STC, PN	França
Tyr119Met	ACG→ATG	Não amiloidogénica	Portugal, EUA
Ala120Ser	GCT→TCT	NA, H C PN	Caraíbas
del Val122	del GTC	SNC, STC, C, PN	EUA (Espanhola)
Val122Ala	GTC→GCC	O, C, PN	Reino Unido, EUA
Val122Ile	GTC→ATC	C	África, Portugal, EUA
Pro125Ser	CCC→TCC	Não amiloidogénica	Itália

Abreviaturas: C = Coração; F = Fígado; LM = Leptomeninges; N = Neuropatia; NA = Neuropatia Autonómica; O = Olhos; P = Pele; PN = Polineuropatia; R = Rins; SNC: Sistema Nervoso Central; STC = Síndrome do Túnel Carpal;

Tabela 10 - Lista das variantes amiloidogénicas e não-amiloidogénicas da TTR, adaptada de [Connors 2003].

Anexo II – Grupos de fragmentos

Divisão em grupos dos fragmentos extraídos das corridas considerando apenas os contactos nativos.

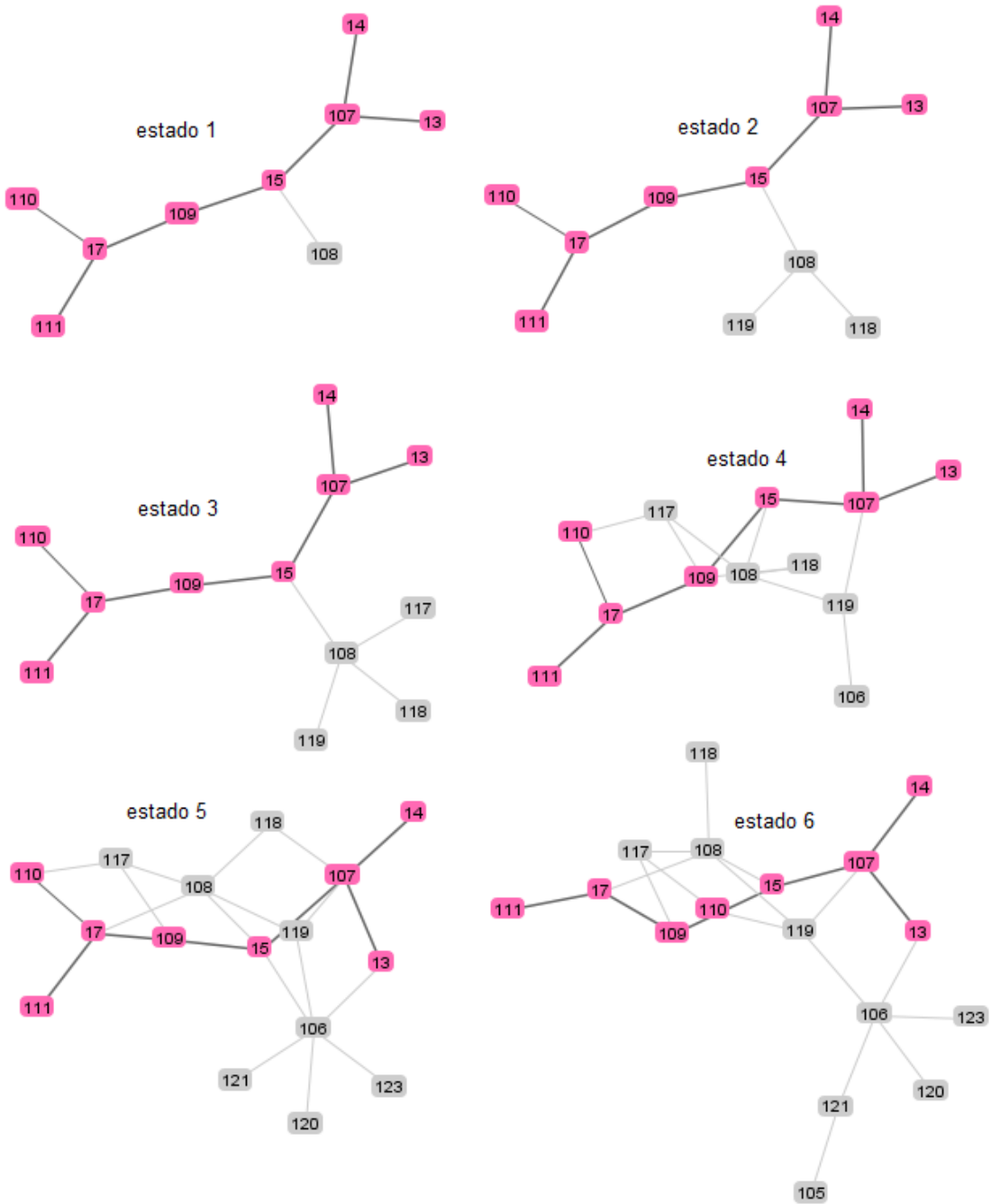
Corrida	Grupo	Número de fragmentos	Tamanho mínimo encontrado (# de arestas)	Tamanho máximo encontrado (# de arestas)	Frequência média (em percentagem)
L55P1	1	4	3	4	76,03
L55P2	1	11	4	6	72,42
	6	25	4	6	73,28
L55P3	1	11	4	6	81,40
	2	3	4	4	77,34
	3	83	15	16	77,13
	5	3	3	3	76,55
	6	44	7	9	78,68
L55P4	2	23	4	7	74,95
	3	98	12	13	72,96
	8	32	6	8	72,39
L55P5	3	154	10	12	73,07
	8	70	8	9	72,01
WT2	1	89	7	10	73,30
	2	21	3	5	76,88
	3	154	11	13	72,17
	6	57	4	7	73,42
WT3	1	1907	3	7	82,51
	2	183	3	7	80,91
	3	872	16	19	76,63
	5	9	3	3	84,14
	6	137	9	12	80,46
	7	195	10	13	76,07
	8	584	9	13	76,15
	9	144	12	14	73,57

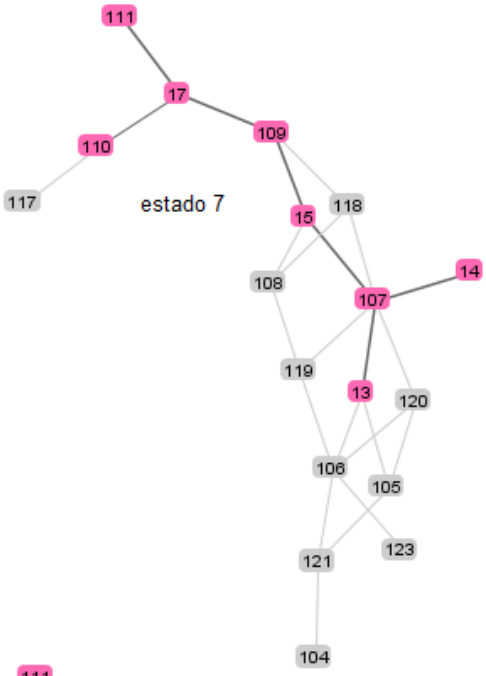
WT4	1	137	8	10	73,65
	2	71	3	7	76,07
	3	777	12	14	72,69
	7	195	9	11	73,28
WT5	1	211	9	12	73,42
	2	71	3	7	76,06
	3	505	13	16	72,29
WT6	1	171	5	8	74,64
	2	89	3	6	75,83
	3	583	9	11	73,50
	9	207	6	9	73,84

Tabela 11 - Relação de cada corrida face aos grupos existentes e ao tamanho mínimo e máximo e a frequência média dos fragmentos para cada grupo.

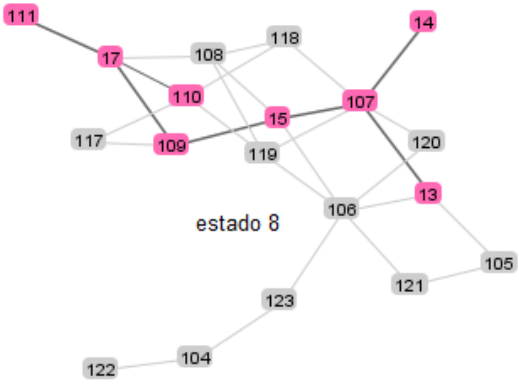
Anexo III – Caminhos

A. Primeiro Exemplo

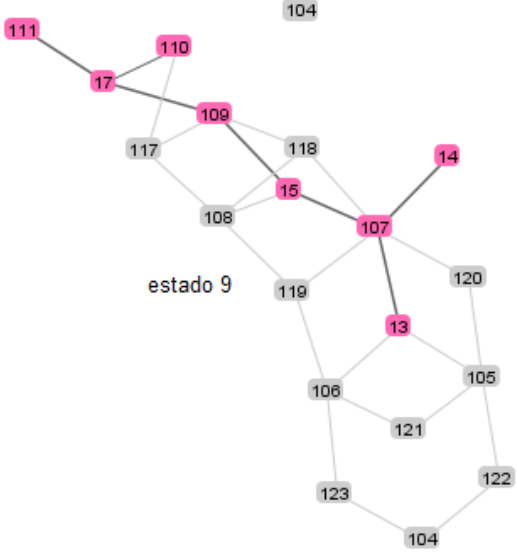




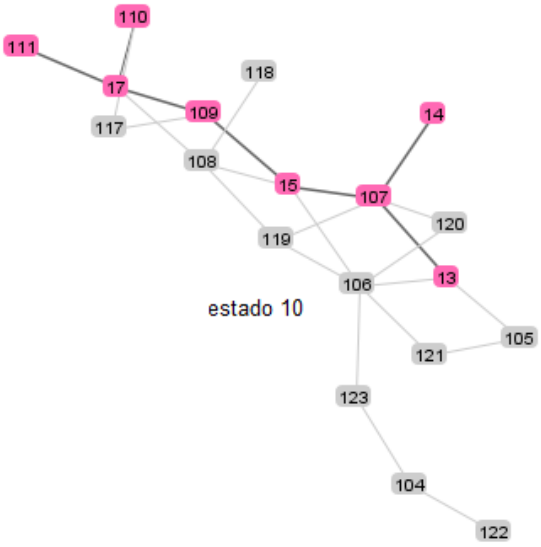
estado 7



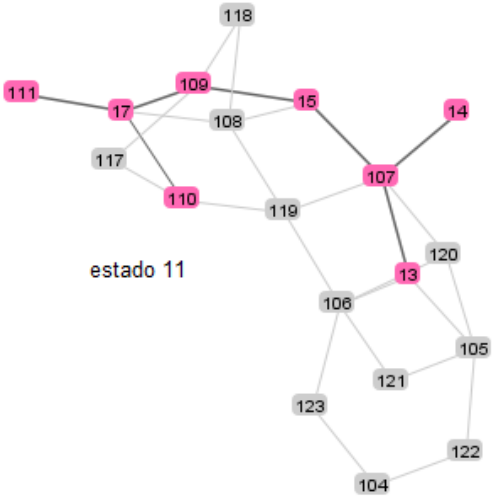
estado 8



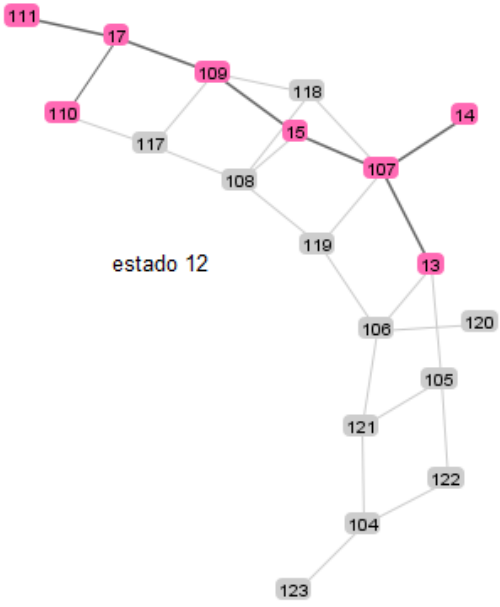
estado 9



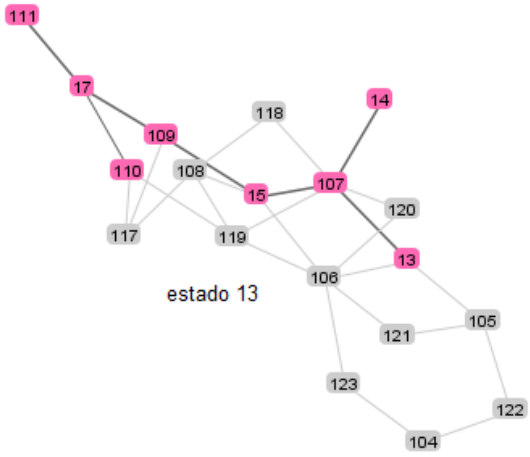
estado 10



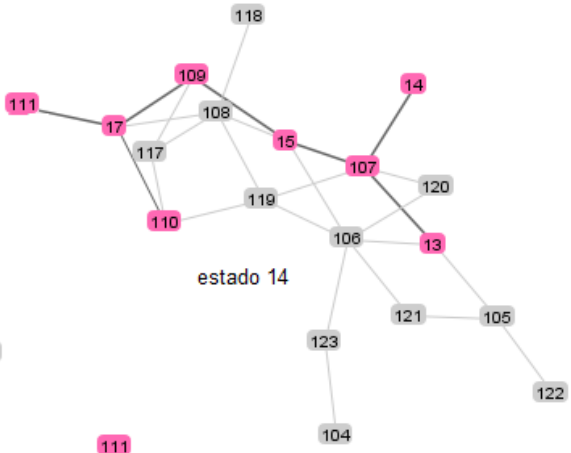
estado 11



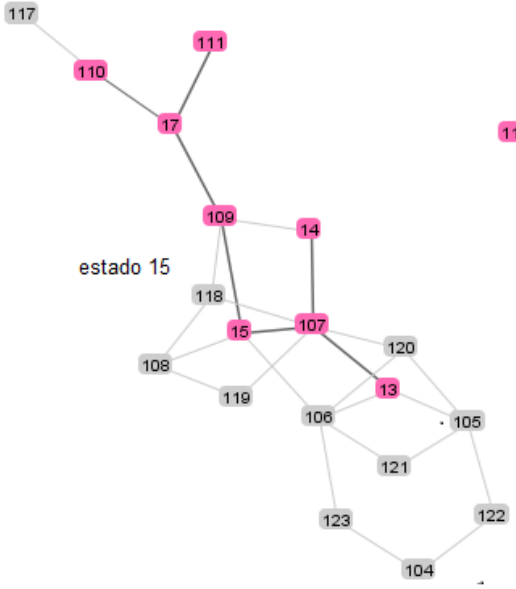
estado 12



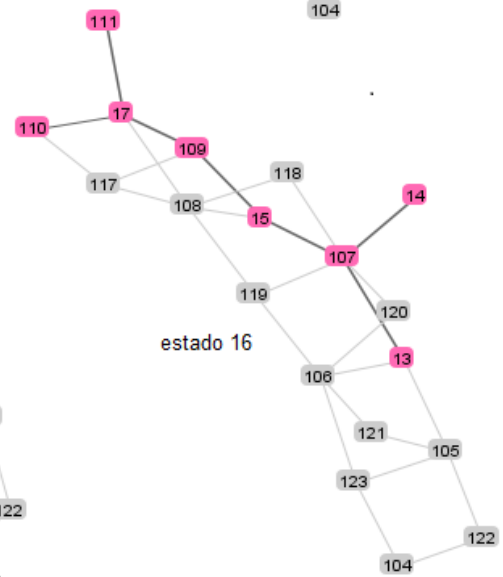
estado 13



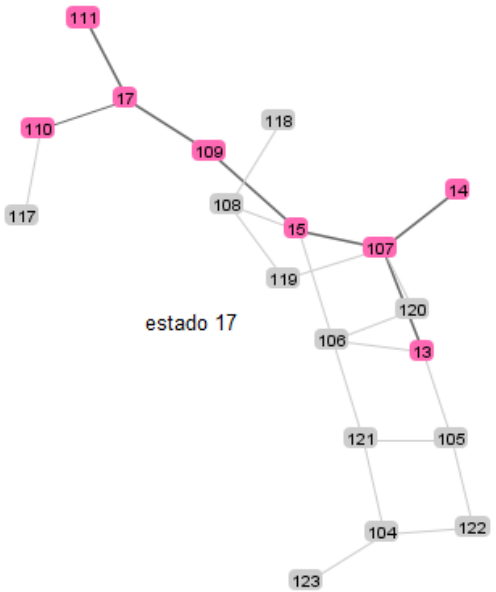
estado 14



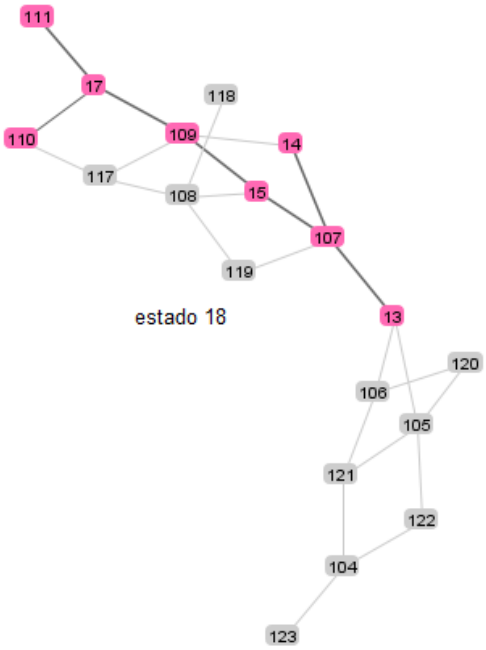
estado 15



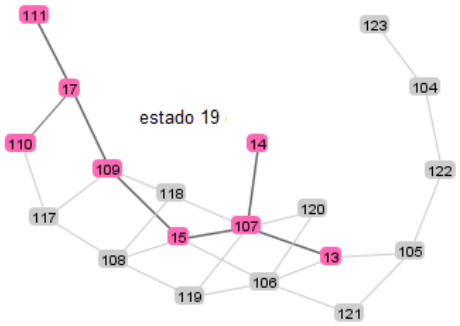
estado 16

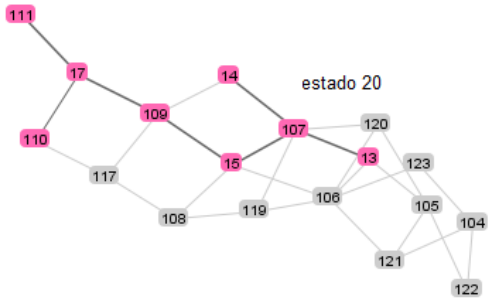


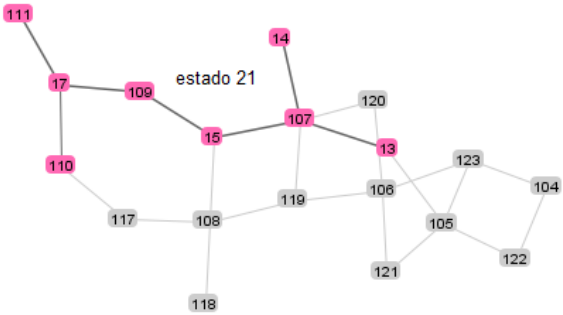
estado 17

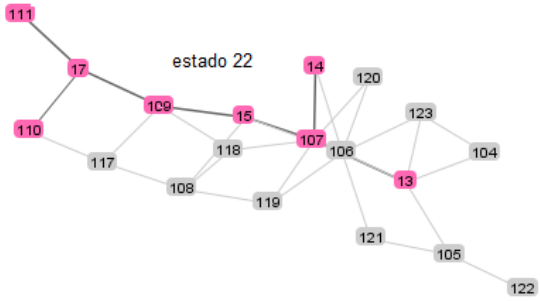


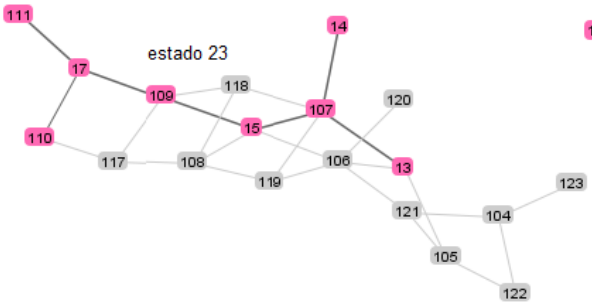
estado 18

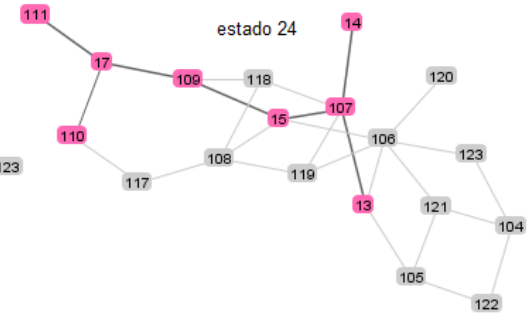

estado 19
111
17
110
109
14
118
117
120
107
15
13
123
104
122
105
108
106
119
121

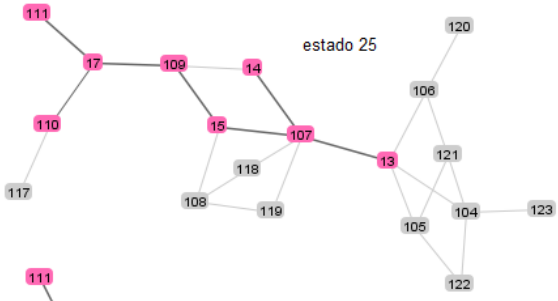

estado 20
111
17
14
109
110
107
120
15
13
123
117
106
105
104
108
119
121
122

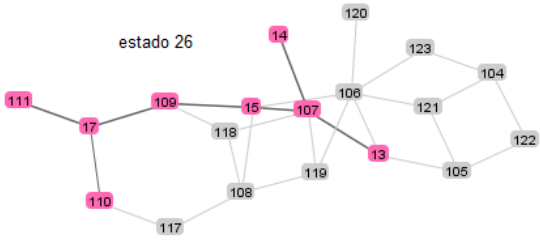

estado 21
111
14
17
109
120
107
15
13
110
123
104
106
119
117
108
105
121
122
118

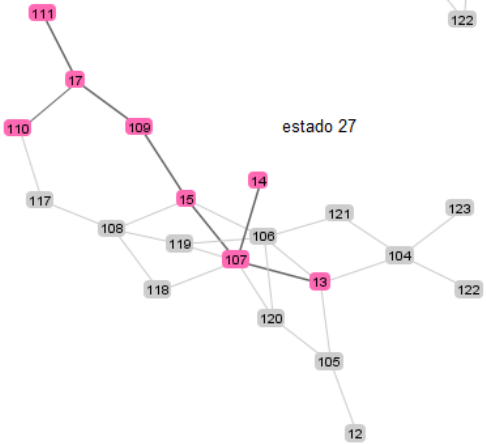

estado 22
111
17
14
120
109
15
123
110
107
118
106
104
117
108
13
119
121
105
122

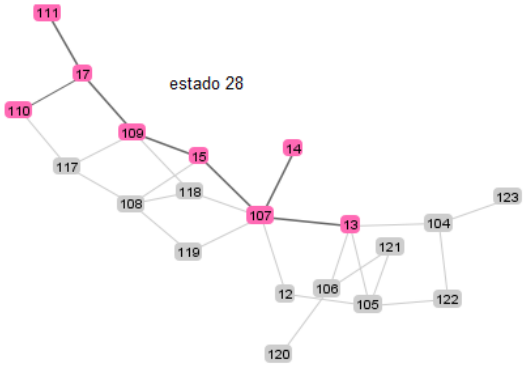

estado 23
111
14
17
118
109
120
107
15
110
117
108
106
13
119
123
121
104
105
122

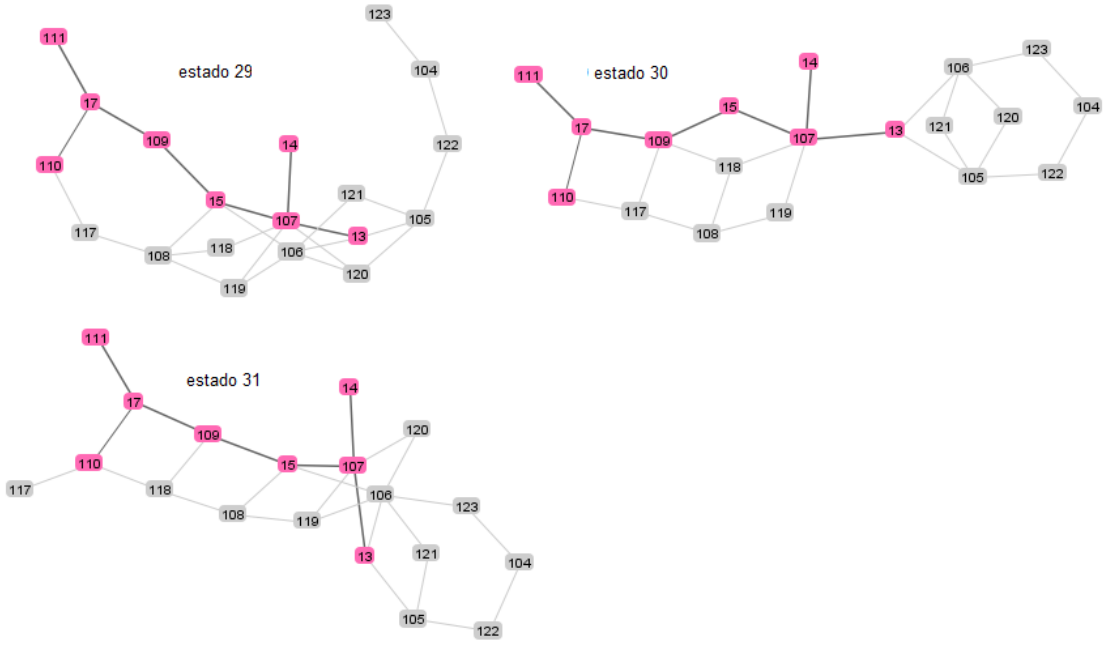

estado 24
111
17
14
109
118
120
15
107
110
106
123
108
119
117
13
121
104
105
122


estado 25
111
17
109
14
120
106
110
15
107
13
121
118
117
108
119
104
123
105
122

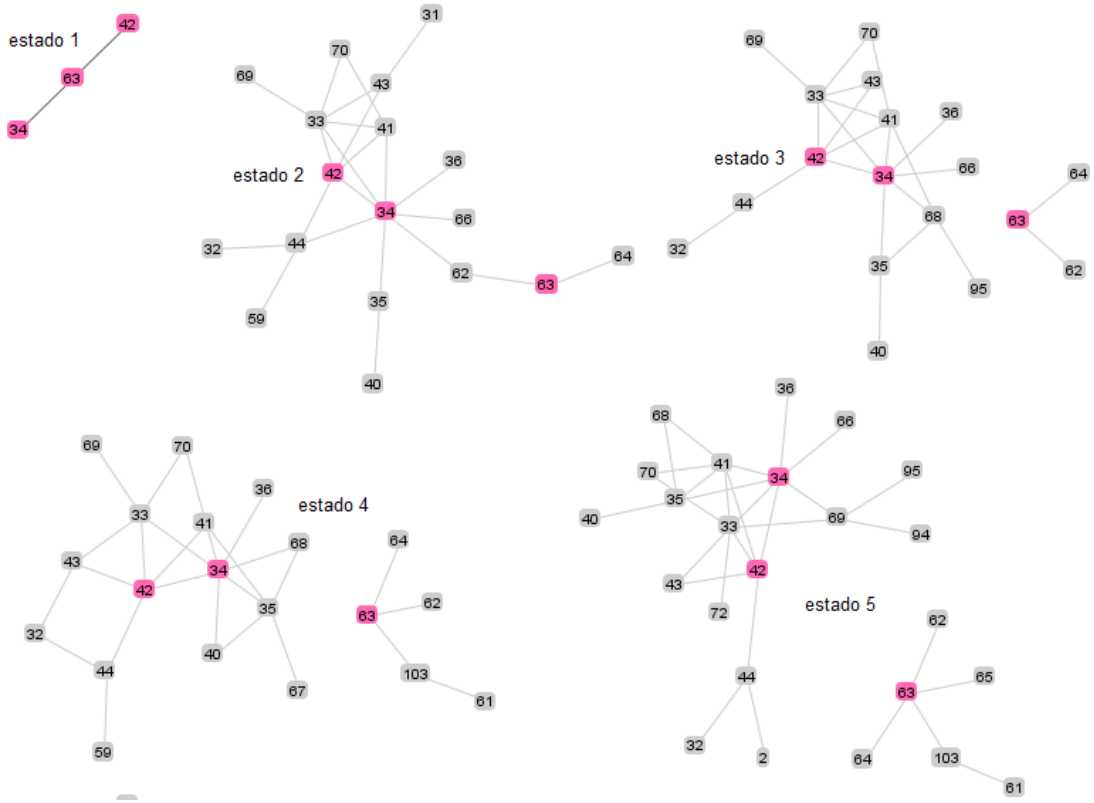

estado 26
120
14
123
104
111
109
15
107
106
121
17
118
122
13
119
105
108
110
117

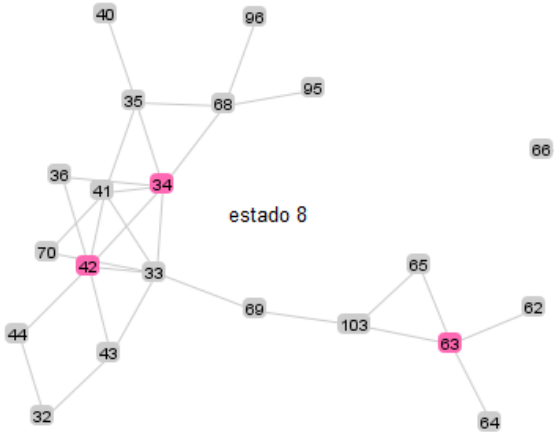

estado 27
111
17
110
109
14
117
15
121
123
108
119
106
104
107
13
118
122
120
105
12

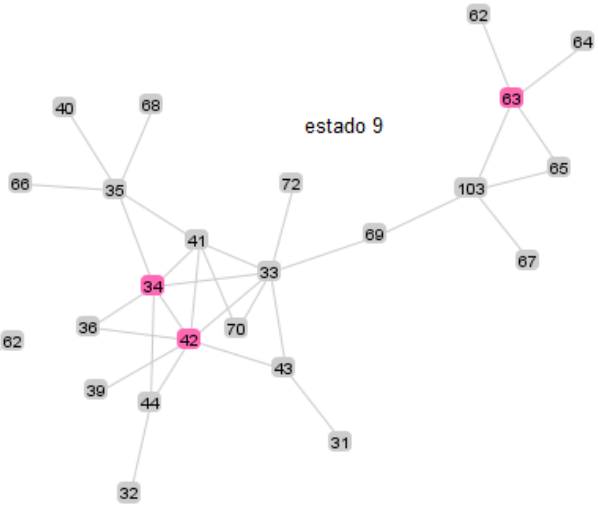

estado 28
111
17
110
109
15
14
117
118
123
108
107
13
104
119
121
12
106
105
122
120

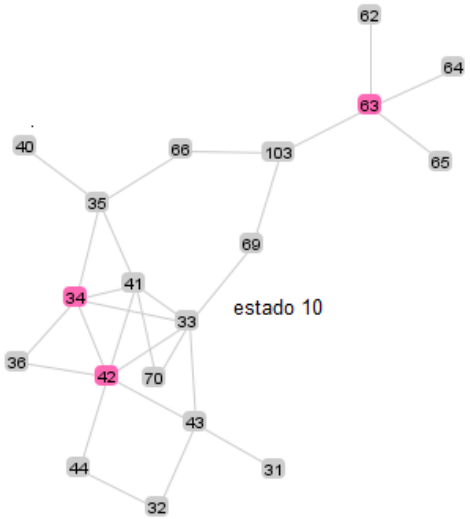


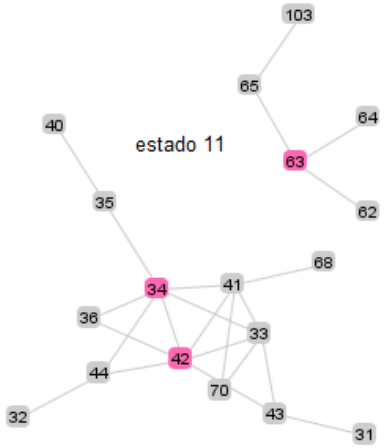
B. Segundo Exemplo

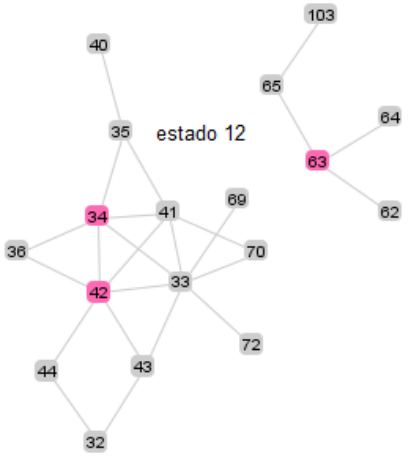


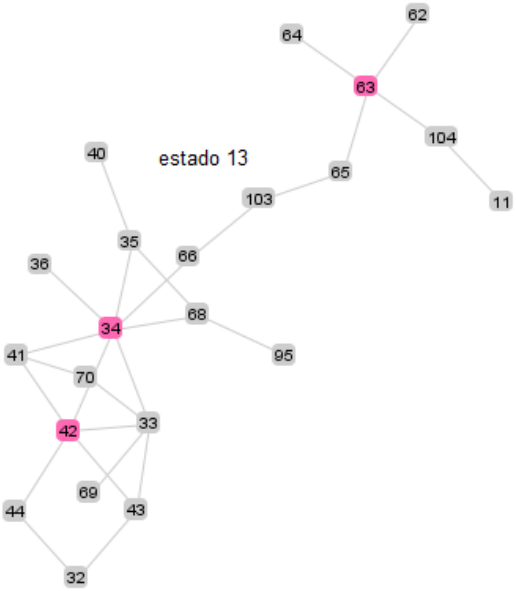

estado 8
40
96
35
68
95
36
41
34
70
42
33
69
65
103
62
44
43
63
32
64

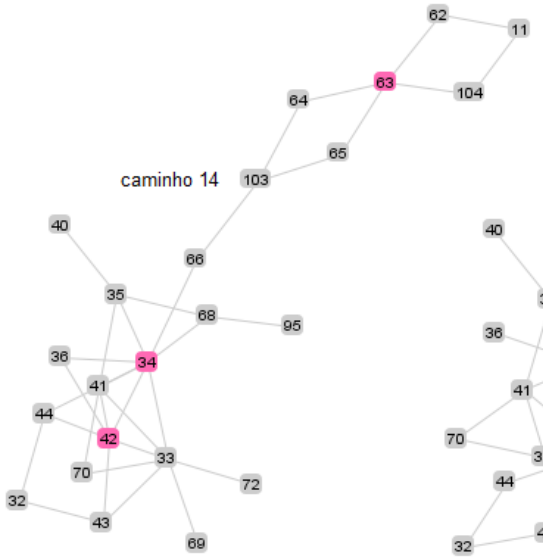

estado 9
62
64
63
40
68
65
66
35
72
103
41
69
67
34
33
36
70
42
43
39
44
31
32

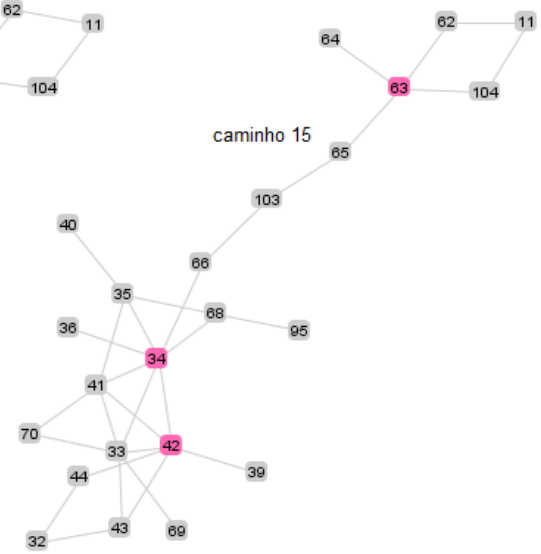

estado 10
62
64
63
40
66
103
65
35
69
34
41
33
36
42
70
43
44
31
32

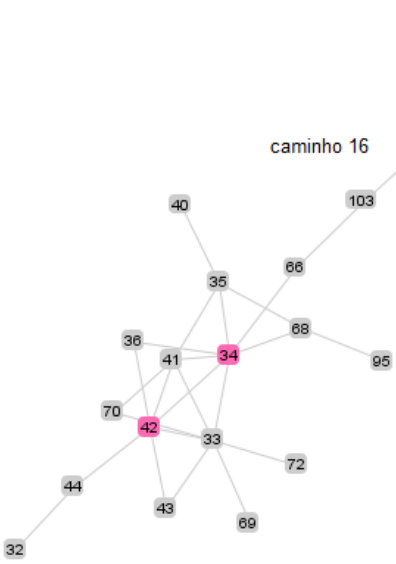

estado 11
103
65
64
40
63
35
62
68
34
41
36
33
42
44
70
32
43
31

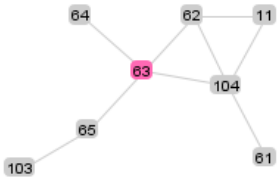

estado 12
103
40
65
64
35
63
62
34
41
69
36
70
42
33
72
44
43
32

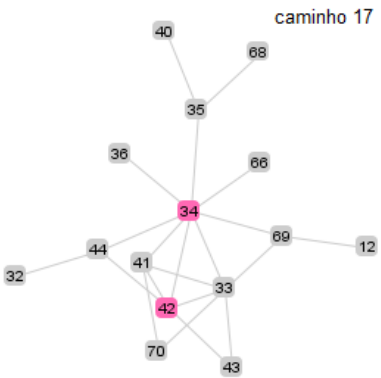

estado 13
62
64
63
104
40
65
11
103
35
66
36
68
34
95
41
70
42
33
69
44
43
32

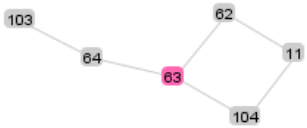

caminho 14
62
11
63
64
104
65
103
40
66
35
68
95
36
34
41
44
42
33
70
72
32
43
69

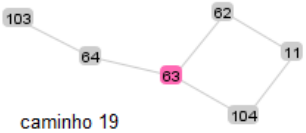

caminho 15
64
62
11
63
104
65
103
40
66
35
68
95
36
34
41
70
33
42
39
44
32
43
69

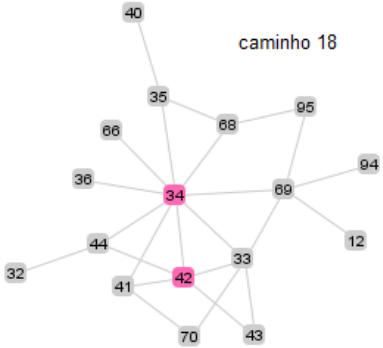

caminho 16
64
62
11
63
104
65
103
40
66
35
68
36
41
34
95
70
42
33
72
44
43
69
32

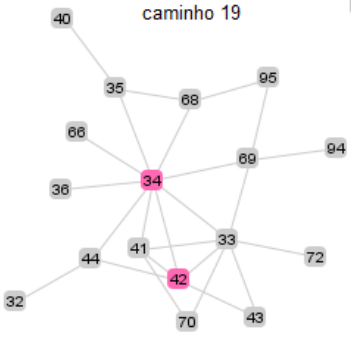

64
62
11
63
104
65
103
61


caminho 17
40
68
35
36
66
34
69
12
44
41
32
33
42
70
43


103
62
64
11
63
104


103
62
64
11
63
104


caminho 18
40
35
68
95
66
36
34
69
94
12
44
33
32
42
41
70
43


caminho 19
40
35
68
95
66
69
94
36
34
33
41
72
44
42
32
70
43

Anexo IV- Mensagens de Erro da Subgraph Paths

Erro	Origem
No fragment selected	Utilizador não selecionou qual o fragmento inicial do caminho.
No file selected	Utilizador não seleccionou, no campo *file,* o nome de um ficheiro para guardar os caminhos a produzir.
No option selected	Utilizador solicitou a produção de caminhos sem indicar que tipo de expansão usar.
Fragmentation path not possible	O fragmento selecionado ocorre na primeira *frame* da corrida.
Any residue chosen	Utilizador selecionou a opção *Directed path* mas não escolheu resíduos.
No final fragment chosen	Utilizador selecionou a opção *Use fragment* mas não escolheu o fragmento.
The fragments are equal	Utilizador escolheu um fragmento final igual ou é um subfragmentos do fragmento inicial.
Please select a slide value and a threshold value	Faltam informações relativas ao tamanho de salto e de *threshold*.
Chosen slide value invalid	O tamanho de salto escolhido pelo utilizador é menor que uma *frame* ou maior que o tamanho de observação escolhido dos caminhos.
Chosen observation value invalid	O tamanho da observação escolhido pelo utilizador é igual ou inferior a zero *frames* ou superior à diferença entre o tamanho total das corridas e a primeira *frame*.
Chosen threshold value invalid	O *threshold* escolhido pelo utilizador é menor que zero ou maior do que 100.

Tabela 12 - Erros tratados na Subgraph Paths.

Anexo V – Modelação

A. Modelo de Domínio

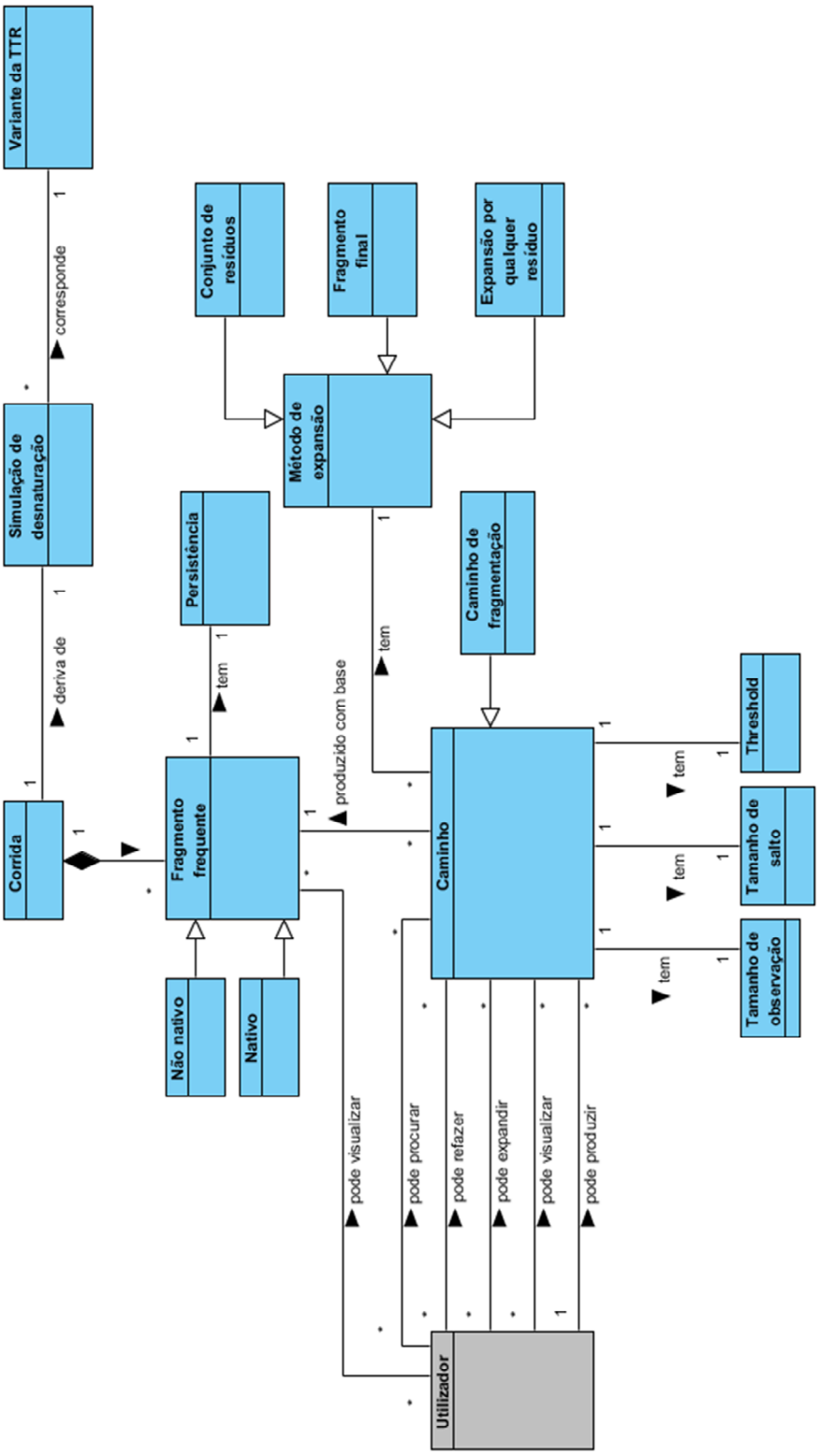


Figura 46 - Modelo de domínio.

B. Casos de Uso

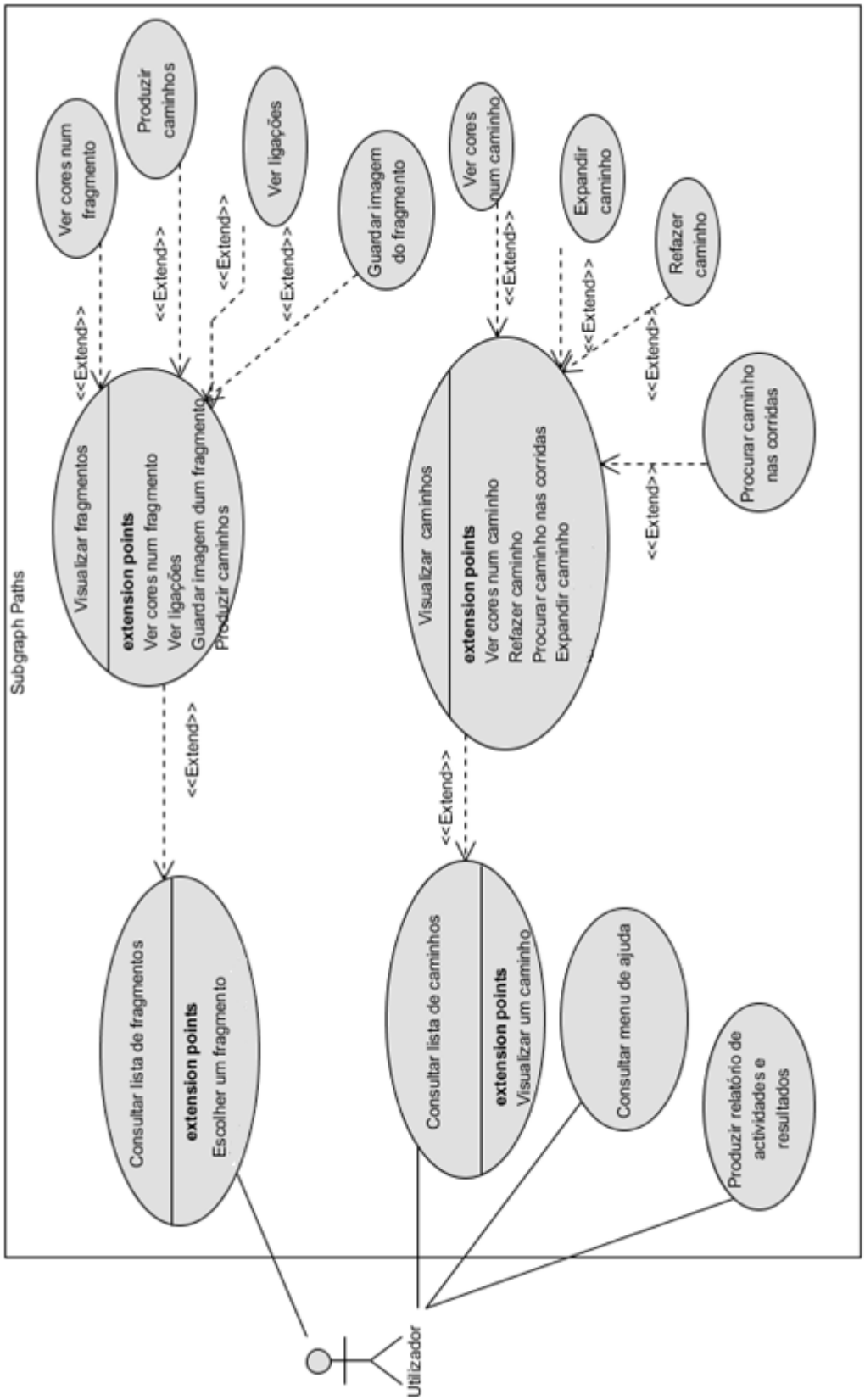


Figura 47 - Modelo de casos de uso.

C. Platform-Independent Model

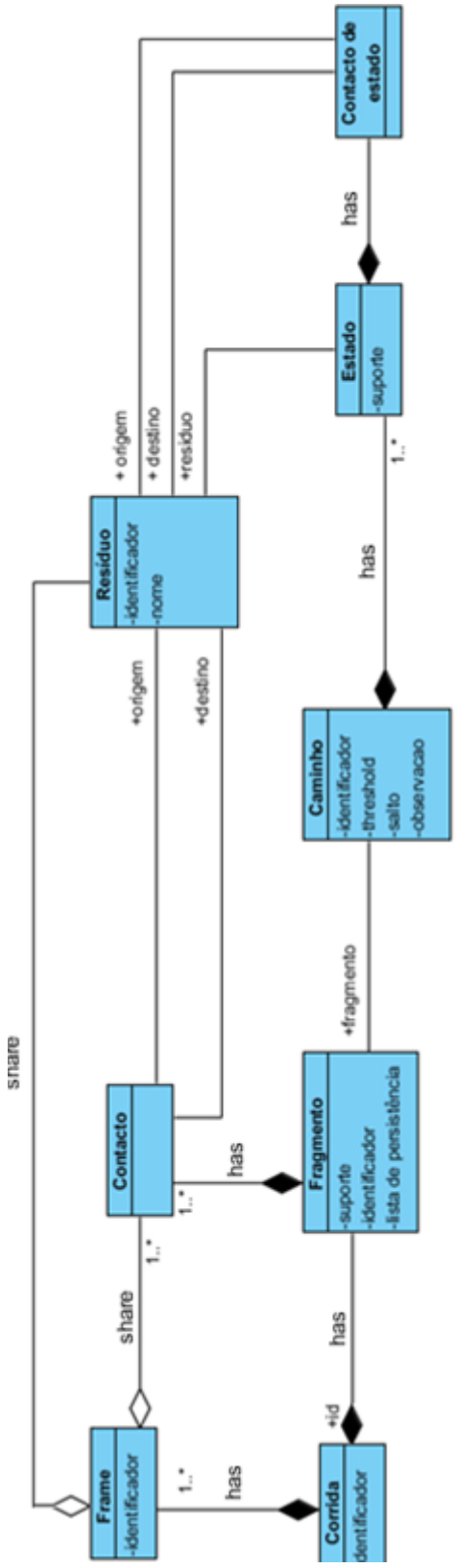


Figura 48 - *Platform-Independent Model.*

Bibliografia

[Adams 2000] D. Adams. The course and prognostic factors of familial amyloid polyneuropathy after liver transplantation. Em *Brain: a journal of neurology,* volume 123, páginas 1495-1504, 2000.

[Ando 2005] Y. Ando, M. Nakamura e S. Araki. Transthyretin – Related Familial Amyloidotic Polyneuropathy. Em *Arch Neurology,* volume 62, páginas 1057-1062, 2005.

[Andrade 1952] C. Andrade. A peculiar form of peripheral neuropathy; familiar atypical generalized amyloidosis with special involvement of the peripheral nerves. Em *Brain: a journal of neurology,* volume 75, páginas 408-427, 1952.

[Azevedo 2005] P. J. Azevedo, C. Silva, J. Rodrigues, N. Loureiro-Ferreira e R. Brito. Detection of hydrophobic clusters in molecular dynamics protein unfolding simulations using association rules. Em *International Symposium on Biological and Medical Data Analysis,* volume 3745*,* páginas 329-337, 2005.

[Azevedo 2007] P. J. Azevedo e A. M. Jorge. Comparing rule measures for predictive association rules. Em *Lecture Notes in Computer Science*, volume 4701, páginas 510-217, 2007.

[Berrar 2005] D. Berrar, F. Stahl, C. Silva, J. R. Rodrigues, R. M. Brito e W. Dubitzky. Towards Data Warehousing and Mining of Protein

Unfolding Simulation Data. Em *Journal of Clinical Monitoring and Computing*, volume 19, páginas 307-317, 2005.

[Borgelt 2002] C. Borgelt, e M. R. Berthold. Mining Molecular Fragments: Finding Relevant Substructures of Molecules. Em *Proc. IEEE Int'l Conf. on Data Mining*, páginas 51–58, 2002.

[Borgelt 2009] C. Borgelt. Graph Mining: An Overview. Em *Proc. 19th GMA/GI Workshop Computational Intelligence*, páginas 189-203, 2009.

[Bray 1994] D. Bray, J.Lewis, M. Radd, K. Roberts, B. Alberts e J. D. Watson. *Molecular Biology of The Cell Third Edition*, 1994.

[Brito 2003] R. M. Brito, A. Damas e M. J. Saraiva. Amyloid formation by transthyretin: From protein stability to protein aggregation. Em *Current Medicinal Chemistry-Immunology, Endocrine & Metabolic Agents,* volume 3, páginas 349-360, 2003.

[Brito 2004] R. M. Brito, W. Dubitzky e J. Rodrigues. Protein folding and unfolding simulations: a new challenge for data mining. Em *OMICS: A Journal of Integrative Biology*, páginas 153-166, 2004.

[Brito 2006] R. M. Brito, C. Silva, N. Loureiro-Ferreira, R. Rodrigues e P. Azevedo. *Identification of hydrophobic clusters on MD unfolding simulations of TTR*, 2006.

[Brooks 1983] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan e M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Em Journal of Computational Chemistry,* volume 4*,* páginas 187–217, 1983.

[Castro 2010] N. Castro e P. Azevedo. Multiresolution Motif Discovery in Time Series. Em *Proceedings of the SIAM International Conference on Data Mining*, páginas 665–676, 2010.

[Chi 1999] E. Huai-hsin Chi e S. K. Card. Sensemaking of Evolving Web Sites Using Visualization Spreadsheets. Em *IEEE Transactions*

on Visualization and Computer Graphics, volume 14, páginas 18-25, 1999.

[Coutinho 1976] P. Coutinho, A. R. Barbosa. Aspetos Neurológicos da PAF. Em *Boletim do Hospital Geral de Santo António*, volume 1, páginas 27-34, 1976.

[Coutinho 1989] P. Coutinho. Em *As viagens de um Gene*. Rocha/Artes Gráficas Lda, Porto, 1989.

[Dickson 1985] P. Dickson, A. Aldred, P. Marley, G. Tu, G. Howlett e G. Schreiber. High prealbumin and transferrin mRNA levels in the choroid plexus of rat brain. Em *Biochemical and biophysical research communications*, volume 127, páginas 890-895, 1985.

[Hamilton 1993] J. A. Hamilton, L. K. Steinrauf, L, B. C. Braden, J. Liepnicks, M. D. Benson, G. Holmgren, O. Sandgreen e L. Steen. The X-ray crystal structure refinements of normal human transthyretin and the amyloidogenic Val-30-Met variant to 170 pm resolution, Em *The Journal of biological chemistry,* volume 268, páginas 2416-2424, 1993.

[Fayyad 1996] U. Fayyad, G. Piatetsky-shapiro, P. Smyth e T. Widener. The KDD process for extracting useful knowledge from volumes of data. Em *Communications of the ACM,* volume 39, páginas 27-34, 1996.

[Felding 1982] P. Felding e G. Fex. Cellular origin of prealbumin in the rat. Em *Biochimica et biophysica acta*, volume 716, páginas 446-449, 1982.

[Fernandes 2008] E. Fernandes. *Extração de conhecimento de dados obtidos por simulação da desnaturação proteica.* Tese de Mestrado, Universidade do Porto, Faculdade de Economia. Orientação: Dr. Alípio Mário Jorge e Dr. Rui Brito, 2008.

[Fernandes 2009] E. Fernandes, A. M. Jorge, C. Silva e Rui M. Brito. A Knowledge Discovery Method for the Characterization of Protein Unfolding

Processes. Em *Advances in Soft Computing*, volume 49, páginas 180-188, 2009.

[Ferreira 2006] P. G. Ferreira, P. J. Azevedo, C. G. Silva e R. M. Brito. Mining approximate motifs in time series. In *Discovery Science*. páginas 89-101, 2006.

[Ferreira 2007] P. G. Ferreira, C. G. Silva, R. M. Brito, e P. J. Azevedo. A closer look on protein unfolding simulations through hierarchical clustering. Em *Computational Intelligence Methods for Bioinformatics and Biostatistics*, páginas 461-468, 2007.

[Ferreira 2009] P. G. Ferreira, C. G. Silva, P. J. Azevedo e R. M. Brito. Spatial Clustering of Molecular Dynamics Trajectories in Protein Unfolding Simulations. Em *Computational Intelligence Methods for Bioinformatics and Biostatistics,* páginas 156-166, 2009.

[Fischer 2004] I. Fischer e T. Meinl. Graph based molecular data mining - an overview. Em *IEEE SMC 2004 Conference Proceedings*, páginas 4578-4582, 2004.

[Freitas 1976] A. F. Freitas. Aspetos Clínicos da Polineuropatia Amiloidótica Familiar (tipo Andrade), Em *Boletim do Hospital Geral de Santo António,* volume 1, páginas 17-25, 1976.

[Hardy 2002] J. Hardy e D. Selkoe. The amyloid hypothesis of Alzheimer's disease: progress and problems on the road to therapeutics. Em *Science*, volume 297, páginas 353-356, 2002.

[Hoing 1999] B. Honig. Protein folding: from the levinthal paradox to structure prediction. Em *Journal of molecular biology,* volume 293, páginas 283–293, 1999.

[Huan 2003] J. Huan, W. Wang, e J. Prins. Efficient mining of frequent subgraphs in the presenceof isomorphism. Em *Proceedings of the Third IEEE International Conference on Data Mining*, páginas 549–552, 2003.

[Hubbard 1993] S. J. Hubbard e J. M. Thornton. *'NACCESS', computer program. - technical report*, Department of Biochemistry Molecular Biology, University College London, 1993.

[Jankovic 2008] J. Jankovic. Parkinson's disease: clinical features and diagnosis, Em *Journal of Neurol Neurosurg Psychiatry*, volume 79, páginas 368-376, 2008.

[Jiang 2001] X. Jiang, J. Buxbaum, e J. Kelly. The V122I cardiomyopathy variant of transthyretin increases the velocity of rate-limiting tetramer dissociation, resulting in accelerated amyloidosis. Em *Proceedings of the National Academy of Sciences of the United States of America*, volume 98, páginas 14943-14948, 2001.

[Kalé 1999] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Philips, A. Shinozaki, K. Varadarajan, e K. Schulten. NAMD2: Greater Scability for Parallel Molecular Dynamics. Em *Journal of Computational Physics,* volume 151, páginas 283-312, 1999.

[Lew 2000] K. Lewis. *Creating effective JavaHelp*. O'Reilly & Associates, Inc., 2000.

[Liu 2000] K. Liu, H. Cho, D. Hoyt, T. Nguyen, P. Olds, J. Kelly e D. Wemmer. Deuterium-proton exchange on the native wild-type transthyretin tetramer identifies the stable core of the individual subunits and indicates mobility at the subunit interface. Em *Journal of Molecular Biology*, volume 303, páginas 555-565, 2000.

[Liu 2002] K. Liu, J. Kelly e D. Wemmer. Native State Hydrogen Exchange Study of Suppressor and Pathogenic Variants of Transthyretin. Em *Journal of Molecular Biology,* volume 320, páginas 821–832, 2002.

[Lobato 2003] L. Lobato, I. Beirão, M. Silva, F. Bravo, F. Silvestre, S. Guimarães, A. Sousa, L. H. Noël e J. Sequeiros. Familial ATTR Amyloidosis: microalbuminuria as a predictor of symptomatic disease and clinical nephropathy, Em *Nephrology, dialysis, transplantation: official publication of the European Dialysis and Transplant Association - European Renal Association,* volume 18, páginas 532-538, 2003.

[Luis 2006] M. Luis. 2006. Polineuropatia Amiloidótica Familiar do Tipo Português: do artigo original ao futuro. Em *Sinapse -Sociedade Portuguesa de Neurologia,* volume 6, páginas 40-42, 2006.

[Meinl 2006] T. Meinl, M. Wörlein, O. Urzova, I. Fischer e M. Philippsen. The parmol package for frequent subgraph mining. Em *Third International Workshop on Graph Based Tools (GraBaTs)*, páginas 94–105, 2006,

[Menza 2005] M. Menza e L. Marsh. Psychiatric Issues in Parkinson's Disease: A Practical Guide. Em *The American Journal of Psychiatry*, volume 163, página 356, 2005.

[Morrison 1992] R. Morrison e R. Boyd. *Química Orgânica*. Fundação Calouste Gulbenkian 6ª Edição Cap. 36, 1992.

[Nijssen 2004a] S. Nijssen e J. N. Kok.. Frequent Graph Mining and its Application to Molecular Databases. Em *Proceedings of Systems, Man and Cybernetics, SMC 2004*, páginas 4571–4577, 2004.

[Nijssen 2004b] S. Nijssen e J. N. Kok. The Gaston Tool for Frequent Subgraph Mining. Em *Electronic Notes in Theoretical Computer Science*, volume 127, páginas 77–87, 2004.

[Pearson 1900] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. Em *Philosophical Magazine*, volume 50, páginas 157–175, 1900.

[Pettersen 2004] E. Pettersen, T. Goddard, C. Huang, G. Couch, D. Greenblatt, E. Meng e T. Ferrin. Chimera – a visualization system for exploratory research and analysis. Em *Journal of Computational Chemistry*, volume 25, páginas 1605-1612, 2004.

[Quintas 1999] A. Quintas, M. Saraiva, e R. M. Brito. The tetrameric protein transthyretin dissociates to a non-native monomer in solution. Em *The Journal of Biological Chemistry*, volume 274, páginas 32943-32949, 1999.

[Quintas 2001] A. Quintas, D. Vaz, I. Cardoso, M. Saraiva e R. M. Brito. Tetramer Dissociation and Monomer Partial Unfolding Precedes Protofibril Formation in Amyloidogenic Transthyretin Variants. Em *The Journal of Biological Chemistry*, volume 276, páginas 27207–27213, 2001.

[Raskind 1995] M. A. Raskind. Alzheimer's disease: treatment of noncognitive behavioural abnormalities. *Em Psychopharmacology: the fourth generation of progress*, páginas 1427-1435, 1995.

[Richardson 2007] S. J. Richardson. Cell and molecular biology of transthyretin and thyroid hormones. Em *International Review of Cytology,* volume 258, páginas 137–193, 2007.

[Rodrigues 2010] J. R. Rodrigues, C. Simões, C. Silva e Rui M. Brito. Potentially amyloidogenic conformational intermediates populate the unfolding landscape of transthyretin: insights from molecular dynamics simulations. Em *Protein science: a publication of the Protein Society,* volume 19, páginas 202-219, 2010.

[Sayan 2009] S. Ranu e A. K. Singh. GraphSig: A Scalable Approach to Mining Significant Subgraphs in Large Graph Databases. Em *International Conference on Data Engineering*, páginas 844-855, 2009.

[Sereniki 2008] A. Sereniki e M. Vital. A doença de Alzheimer: aspetos fisiopatológicos e Farmacológicos. Em *Revista Psiquiátrica*, volume 30, suplemento 1, 2008.

[Sgarbieri 1996] V. C. Sgarbieri. *Proteínas em alimentos protéicos: propriedades, degradações, modificações*, 1996.

[Sharon 2010] I. Sharon, R. Sharon, J.Wilkens e T. Ersan. Huntington Disease Dementia. Em *emedicine, WebMD. Medscape*, 2010.

[Shmygelska 05] A. Shmygelska. Search for folding nuclei in native protein structures. *Em Bioinformatics,* volume 21, páginas 394–402, 2005.

[Shortle 1996] D. Shortle. The denatured state (the other half of the folding equation) and its role in protein stability. Em *FASEB journal: official publication of the Federation of American Societies for Experimental Biology*, volume 10, páginas 27-34, 1996.

[Silva 2007] C. Silva, R. M. Brito, N. Loureiro-Ferreira, R. Rodrigues e P. Azevedo. *Studying solvent exposure of individual amino acid residues across multiple MD unfolding simulations of transthyretin*, 2007.

[Silva 2008] C.Silva e R. M. Brito. *Graph Mining*. No âmbito do projeto PTDC/PRO-BIA/72838/2006, 2008.

[Stefani 2004] M. Stefani. Protein misfolding and aggregation: new examples in medicine and biology of the dark side of the protein world. Em *Biochimica et biophysica acta*, volume 1739, páginas 5-25, 2004.

[Venugopal 09] K. Venugopal, K. Srinivasa, e L. Patnaik. Soft Computing for Data Mining Applications. Em *Studies in computational intelligence,* volume 268, páginas 2416-2424, 2009.

[Vieira 2008] B. Vieira. *Polineuropatia Amiloidótica Familiar (PAF) Tipo I: Uma Visão Atual, de um Problema de Saúde Antigo*. Tese de

Mestrado, Universidade da Beira Interior, Faculdade de Ciências da Saúde. Orientação: Dr. António Jorge Santos Silva, 2008.

[Vilaça 2008] M. Vilaça. *Extração e Análise de Fragmentos Frequentes de Desnaturação Proteica*. Tese de Mestrado, Universidade do Minho. Orientação Dr. Paulo Azevedo, 2008.

[Yan 2002] X. Yan e J. Han. gSpan: Graph–Based Substructure Pattern Mining. Em *Proceedings of the 2002 IEEE International Conference on Data Mining*, páginas 721–723, 2002.

[Walker 2007] F. O. Walker. Huntington's disease. Em *The Lancer*, volume 369, páginas 218-228, 2007.

[Washio 2003] T. Washio e H. Motoda. State of the art of graph-based data mining. Em *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter*, volume 5, páginas 59–68, 2003.

[Wei 2004] C. Wei, J. Li e R. E. Bumgarner. Sample size for detecting differentially expressed genes in microarray experiments. Em *BioMed Central Genomics*, 2004.

[Wörlein 2005] M. Wörlein, T. Meinl, I. Fischer, e M. Philippse. A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. Em *Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases*, páginas 392–403, 2005.