UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

FAUSTO MARQUES PINHEIRO JUNIOR

Algoritmo *Branch and Prune* Markoviano em Geometria de Distâncias

Campinas

2025

Fausto Marques Pinheiro Junior

Algoritmo *Branch and Prune* Markoviano em Geometria de Distâncias

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas (UNICAMP) como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada.

Orientador: Carlile Campos Lavor

Este trabalho corresponde à versão final da Dissertação defendida pelo aluno Fausto Marques Pinheiro Junior e orientada pelo Prof. Dr. Carlile Campos Lavor.

Campinas

2025

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

P655a

Pinheiro Junior, Fausto Marques, 1987-
Algoritmo branch and prune markoviano em geometria de distâncias / Fausto Marques Pinheiro Junior. – Campinas, SP : [s.n.], 2025.

Orientador: Carlile Campos Lavor.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Instituto de Matemática, Estatística e Computação Científica.

1. Geometria de distâncias. 2. Modelos markovianos ocultos. 3. Proteínas - Conformação. I. Lavor, Carlile Campos, 1968-. II. Universidade Estadual de Campinas (UNICAMP). Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações complementares

Título em outro idioma: Markovian branch and prune algorithm in distance geometry
Palavras-chave em inglês:
Distance geometry
Hidden Markov models
Proteins - Conformation
Área de concentração: Matemática Aplicada
Titulação: Mestre em Matemática Aplicada
Banca examinadora:
Carlile Campos Lavor [Orientador]
Petra Maria Bartmeyer
Felipe Delfini Caetano Fidalgo
Data de defesa: 07-07-2025
Programa de Pós-Graduação: Matemática Aplicada

Objetivos de Desenvolvimento Sustentável (ODS)
ODS: 3. Saúde e bem-estar

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-9727-8532
- Currículo Lattes do autor: http://lattes.cnpq.br/3810112164535585

Prof. Dr. CARLILE CAMPOS LAVOR

Profa. Dra. PETRA MARIA BARTMEYER

Prof. Dr. FELIPE DELFINI CAETANO FIDALGO

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Aos trabalhadores do mundo. Uni-vos!

Agradecimentos

Primeiramente, agradeço aos meus pais pelo apoio e suporte contínuo, mesmo quando eles não fazem idéia do que eu estou pesquisando. Não obstante, agradeço aos meus irmãos que também me ajudam a manter o pilar fundamental da família para que eu consiga a estabilidade necessária para todas as outras atividades da minha vida.

Agradeço ao meu orientador Prof. Dr. Carlile Lavor pela paciência e orientação ao longo desse curto período, mas que foi intenso em trabalho. Consequentemente, também agradeço a todos os professores, pesquisadores e alunos do grupo de Geometria de Distâncias na Unicamp e demais instituições do Brasil pelo suporte e ajuda nas horas mais inusitadas. Em particular, agradeço ao Prof. Dr. Caio Azevedo, por ter sido uma peça fundamental no começo da pesquisa em que foi brevemente meu co-orientador, sem o qual a idéia fundante da pesquisa não teria surgido, e ao Rômulo Marques, por ter praticamente me co-orientado extra-oficialmente e rendido horas de conversas que conseguiram dar forma à idéia inicial. Em suma, "se vi mais longe, foi por estar de pé sobre os ombros de gigantes".

Por fim, agradeço à UNICAMP por toda o esforço do seu quadro técnico-administrativo e infra-estrutura que viabilizaram cada passo durante o mestrado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"'Science affirms that...' Science is voiceless; it is the scientists who talk."
(Simone Weil)

Resumo

O algoritmo *Branch and Prune* aborda o problema discretizável de geometria de distâncias moleculares explorando a estrutura de árvore binária do espaço de soluções por meio de uma busca em profundidade. Duas limitações desta abordagem são a sua serialidade e a ausência de um critério de preferência entre os vértices que serão explorados em cada iteração. Para sanar essas limitações, propomos a incorporação de modelos probabilísticos gráficos no *Branch and Prune*, dado que essa classe de modelos apresenta alta modularidade e capacidade de tratar com diferentes fontes de incerteza. Em específico, vamos apresentar um caso particular na forma do modelo oculto de Markov autoregressivo que incorpora informações sobre a sequência de aminoácidos e a geometria dos planos peptídicos para auxiliar na exploração probabilística do espaço de soluções. Discutiremos algumas propriedades deste tipo de modelo, o seu ajuste com os dados do *Protein Data Bank*, a sua integração no *Branch and Prune*, e alguns resultados preliminares obtidos na pesquisa. Por fim, arguiremos pela viabilidade de extensões deste tipo de modelo para abordar o problema discretizável de geometria de distâncias moleculares.

Palavras-chave: geometria de distâncias. modelo oculto de Markov. conformação de proteínas.

Abstract

The Branch and Prune algorithm addresses the discretizable problem of molecular distance geometry by exploring the binary tree structure of the solution space through a depth-first search. Two limitations of this approach are its seriality and the absence of a preference criterion between the vertices that will be explored in each iteration. To overcome these limitations, we propose the incorporation of graphical probabilistic models in Branch and Prune, given that this class of models presents high modularity and the ability to deal with different sources of uncertainty. Specifically, we will present a particular case in the form of an autoregressive hidden Markov model that incorporates information about the amino acid sequence and the geometry of the peptide planes to aid in the probabilistic exploration of the solution space. We will discuss some properties of this type of model, its fit with the data from the Protein Data Bank, its integration into Branch and Prune, and some preliminary results obtained in the research. Finally, we will argue for the feasibility of extensions of this type of model to address the discretizable problem of molecular distance geometry.

Keywords: distance geometry. Hidden Markov model. protein conformation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Programming Interface*
AR	Autoregressivo
ARMBP	*Autoregressive (Hidden) Markovian Branch and Prune*
BP	*Branch and Prune*
CM	Cadeia de Markov
DAG	Digrafo Acíclico (do inglês, *Directed Acyclic Graph*)
DDL	*Dictionary Description Language*
DFS	Busca em Profundidade (do inglês, *Depth-First Search*)
GD	Geometria de Distâncias
DNA	Ácido desoxirribonucleico (do inglês, *Deoxyribonucleic Acid*)
FBS	*Frequency Based Search*
f.d.p	Função densidade de probabilidade
f.m.p	Função massa de probabilidade
GB	*Gigabyte* (10^9 bytes)
iBP	*Intervalar Branch and Prune*
iids	Independentes e identicamente distribuídas
IMECC	Instituto de Matemática, Estatística e Computação Científica
IUPAC	*The International Union of Pure and Applied Chemistry*
MB	*Megabyte* (10^6 bytes)
Mbps	*Megabit per second*
MOM	Modelo Oculto de Markov
MOMAR	Modelo Oculto de Markov Autorregressivo

PDGD	Problema Discretizável de Geometria de Distâncias
PDGDM	Problema Discretizável de Geometria de Distâncias Moleculares
PGD	Problema de Geometria de Distâncias
UNICAMP	Universidade Estadual de Campinas
NEF	*NMR Exchange Format*
PDB	*Protein Data Bank*
SSNMR	Ressonância Magnética de Sólidos (do inglês, *Solid-state Nuclear Magnetic Resonance*)
RCBS	*Research Collaboratory for Structural Bioinformatics*
NMR	Ressonância Magnética Nuclear (do inglês, *Nuclear Magnetic Resonance*)
RNA	Ácido Ribonucleico (do inglês, *Ribonucleic Acid*)
wwPDB	*Worldwide Protein Data Bank*

Lista de símbolos

$M_{m\times n}(\mathbb{R})$	Conjunto das matrizes de ordem $m \times n$ com entradas reais
$A \cong B$	Congruência ou relação de equivalência dos conjuntos A e B
$\mathbb{R}^n_+$	Conjunto dos vetores x pertencentes a $\mathbb{R}^n$ que satisfazem $x \geqslant 0$
$\mathbb{R}^{*n}_+$	Conjunto dos vetores x pertencentes a $\mathbb{R}^n$ que satisfazem $x > 0$
$\varnothing$	Conjunto vazio
∞	Infinito
$\mathbf{I}_n$	Matriz identidade de ordem $n \times n$
$\|\cdot\|$	Norma canônica
$\|\cdot\|_p$	Norma-p
$<_A$	Relação de ordem total estrita para um conjunto A
$A \times B$	Produto cartesiano dos conjuntos A e B
$\langle x, y\rangle$	Produto interno dos vetores x e y
$f\|_A$	Restrição da função f ao conjunto A

Lista de Algoritmos

Lista de Códigos-fonte

Sumário

1 INTRODUÇÃO

1.1 Geometria de Distâncias

Suponha uma matriz quadrada D de ordem $|M|$ e uma função de distância ω, em que $D_{i,j} = \omega(i, j)$ se $D_{i,j} \neq 0$. Ou seja, a entrada não-nula $D_{i,j}$ representa uma distância entre o i-ésimo e o j-ésimo elementos, mas uma entrada nula $D_{i,j}$ não implica que $\omega_{i,j} = 0$, de modo que D é uma matriz possivelmente incompleta de distâncias entre os elementos de M. Calcular coordenadas de cada ponto $f(i) \in X$ em espaço métrico $(X, d) \supset (M, q)$ para os $|M|$ elementos, de tal modo que respeite tanto $\omega(i, j) = d(f(i), f(j))$, quanto as propriedades do espaço métrico, não é um problema elementar e nem admite um cálculo direto. Além disso, também não é imediatamente óbvio como calcular tais coordenadas no espaço métrico alvo.

No entanto, esse problema aparece naturalmente em diversas aplicações para uma variedade de atividades humanas, de bioquímica molecular à robótica (LIBERTI et al., 2014). Tal problema é conhecido como **problema fundamental da GD**. A grosso modo, o problema consiste em, se possível, calcular as coordenadas dos pontos em um espaço métrico (X, d) para um conjunto de $|M|$ elementos, cujas distâncias ω entre alguns destes elementos são conhecidas (LAVOR; LIBERTI, 2014), e é formalmente apresentado na subseção 2.2.1. Para esse trabalho, vamos fixar o espaço métrico como sendo $\mathbb{R}^3$.

1.1.1 Aplicações da Geometria de Distâncias

A cognição espacial humana é majoritariamente referencial, ou seja, baseada na noção de distância entre o agente e um objeto conhecido (egocêntrica) ou na distância entre objetos conhecidos (alocêntrica) para posicionar espacialmente o mundo percebido (DENIS, 2017). Raramente detemos a informação sobre as coordenadas exatas no espaço (ou uma de suas representações) em que nós ou objetos de nosso interesse se encontram, e, mesmo nestes casos, usualmente sabemos dessas coordenadas como um dado derivado da nossa distância para outros objetos. Na nossa realidade, a distância é o dado primitivo.

Um exemplo disso é a posição dada pelo GPS, onde o posicionamento é obtido pelo cálculo da distância do aparelho usado para acessar o serviço para 3 ou mais satélites que conhecem a distância entre si (e entre posições fixas no solo terrestre) usando trilateração (ABEL; CHAFFEE, 1991). Outro exemplo, talvez ainda mais ilustrativo, é o uso de mapas, onde o usuário normalmente não sabe onde no mapa ele se encontra, mas, pela distância entre ele e algumas características do local que são representadas no mapa, como um rio ou morros, ele consegue se posicionar neste mapa.

A referencialidade dessa forma de posicionar espacialmente é parcimoniosa porque não requer da realidade que exista um sistema de coordenadas universal, ou seja, não requer a existência de um ponto arquimediano, perfeitamente imóvel e certo. Assim, a nossa posição é sempre relativa e diferentes referências são úteis para problemas distintos.

O exemplo anterior do GPS nos dá uma referência com base na distância entre satélites e bases fixadas na Terra, gerando coordenadas tridimensionais para qualquer posição no nosso planeta. Entretanto, essa referência já não serve se queremos identificar o posicionamento de outros planetas ou galáxias. Por outro lado, os satélites *Voyager* carregam discos dourados contendo na capa um diagrama que define o posicionamento do Sol utilizando a interseção de 14 pulsares (estrelas de nêutrons) e suas respectivas direções, frequências de pulso e distância relativa para a interseção (NASA Science, 2024). Não sabemos se existe vida inteligente fora do nosso sistema solar, mas, se houver, estamos contando que eles conheçam GD.

No entanto, apesar da variedade de aplicações possíveis, uma aplicação em específico que recebe muita atenção da comunidade científica é a determinação da estrutura tridimensional de macromoléculas orgânicas (LAVOR et al., 2017). A importância desta aplicação é tão significativa que algumas classes de problemas dentro da GD são estudados já incorporando as restrições típicas desta área, em especial as relacionadas com macromoléculas classificadas como proteínas. A pesquisa apresentada aqui se dá neste contexto de identificar a estrutura tridimensional de uma proteína, também chamada de conformação.

1.2 Conformação de Proteínas

Como visto anteriormente, o problema fundamental da GD consiste em calcular coordenadas no espaço métrico alvo a partir de um conjunto de distâncias entre um número finito de objetos. No caso da conformação de uma proteína, temos um caso particular em que se deseja achar as coordenadas dos átomos de sua espinha-dorsal (em inglês, *backbone*) em $(\mathbb{R}^3, \langle \cdot, \cdot \rangle)$ a partir das distâncias entre alguns átomos desta espinha-dorsal.

O tipo de objeto que desejamos posicionar nesta aplicação contém propriedades que são úteis de se incorporar no modelo matemático e que não estão presentes em pontos abstratos de $\mathbb{R}^3$. Por exemplo, há a impossibilidade de dois átomos distintos estarem arbitrariamente próximos um do outro, em especial a depender de sua carga elétrica, diferente de pontos distintos quaisquer em $\mathbb{R}^3$ que podem estar o quão próximos quanto desejarmos.

Além disso, proteínas possuem regularidades estruturais e propriedades que também podem ser incorporadas ao modelo, restringindo o espaço de busca ou favorecendo a exploração deste de alguma maneira específica. Por exemplo, algumas ligações químicas

especiais entre átomos de uma proteína forçam com que os átomos adjacentes fiquem em um mesmo plano.

Vamos apresentar no restante desta seção um pouco sobre as proteínas, os aminoácidos que as compõem, e suas propriedades geométricas de interesse no contexto da GD. Ao final desta seção, temos também a Figura 3 com uma representação dos principais aminoácidos e seus átomos constitutivos. Todas estas informações serão úteis para compreender o levantamento, processamento e organização dos dados extraídos do PDB na seção 3.1 para a análise empírica da pesquisa.

1.2.1 Estrutura Primária

Uma cadeia polipeptídica que contém mais de 50 aminoácidos é uma proteína. De modo geral, proteínas consistem de cadeias de aminoácidos elencados, chamados também de resíduos, unidos de ponta-a-ponta. As características físicas de cada proteína decorrem dos diferentes aminoácidos que o constituem. Ao todo, são 22 aminoácidos ditos proteinogênicos, ou seja, que são precursores das proteínas codificadas no código genético dos organismos vivos existentes. Note que cada um destes aminoácidos é denotado pela IUPAC por uma letra maiúscula, de modo que uma proteína pode ser representada por uma cadeia de caracteres (*string*). Alternativamente, os aminoácidos do *Protein Data Bank* (PDB) podem ser codificados por um código de 3 letras (BERMAN, 2000). A relação está disposta na Tabela 1, junto com uma codificação numérica usada posteriormente para esta pesquisa.

A Tabela 1 também contém uma entrada para valores desconhecidos, que são usados para representar aminoácidos que não se conseguiu detectar o tipo ou o início/fim de um córdon de transcrição do RNA que codifica biologicamente uma proteína dentro das células orgânicas.

Em todo caso, aminoácidos são compostos de, no máximo, 5 tipos de átomos: carbono (C), hidrogênio (H), nitrogênio (N), oxigênio (O), e enxofre (S). Todos os átomos de um aminoácido compõem algum dos 4 grupos:

- O carbono-α central (C_α ou CA);
- O grupo carboxila ($-COO\ominus$);
- O grupo amino ($-NH_3\oplus$);
- A cadeia lateral, que varia de acordo com o tipo de aminoácido (p.e. $-H$ para G e $-CH_2OH$ para a S).

Na cadeia de aminoácidos, o i-ésimo aminoácido é usualmente ligado ao $(i+1)$-ésimo aminoácido por uma ligação covalente, dita ligação peptídica ($-CO-NH-$), que é

Tabela 1 – Classificação dos Aminoácidos.

Aminoácido	**Código IUPAC**	**Código 3-Letras**	**Codificação**
Alanina	A	Ala	1
Cisteína	C	Cys	2
Ácido Aspártico	D	Asp	3
Ácido Glutamico	E	Glu	4
Fenilalanina	F	Phe	5
Glicina	G	Gly	6
Histidina	H	His	7
Isoleucina	I	Ile	8
Lisine	K	Lys	9
Leucina	L	Leu	10
Metionina	M	Met	11
Asparagina	N	Asn	12
Pirrolisina	O	Pyl	13
Prolina	P	Pro	14
Glutamina	Q	Gln	15
Arginina	R	Arg	16
Serina	S	Ser	17
Treonina	T	Thr	18
Selenocisteína	U	Sec	19
Valina	V	Val	20
Triptofano	W	Trp	21
Tirosina	Y	Tyr	22
Desconhecido	X	–	23

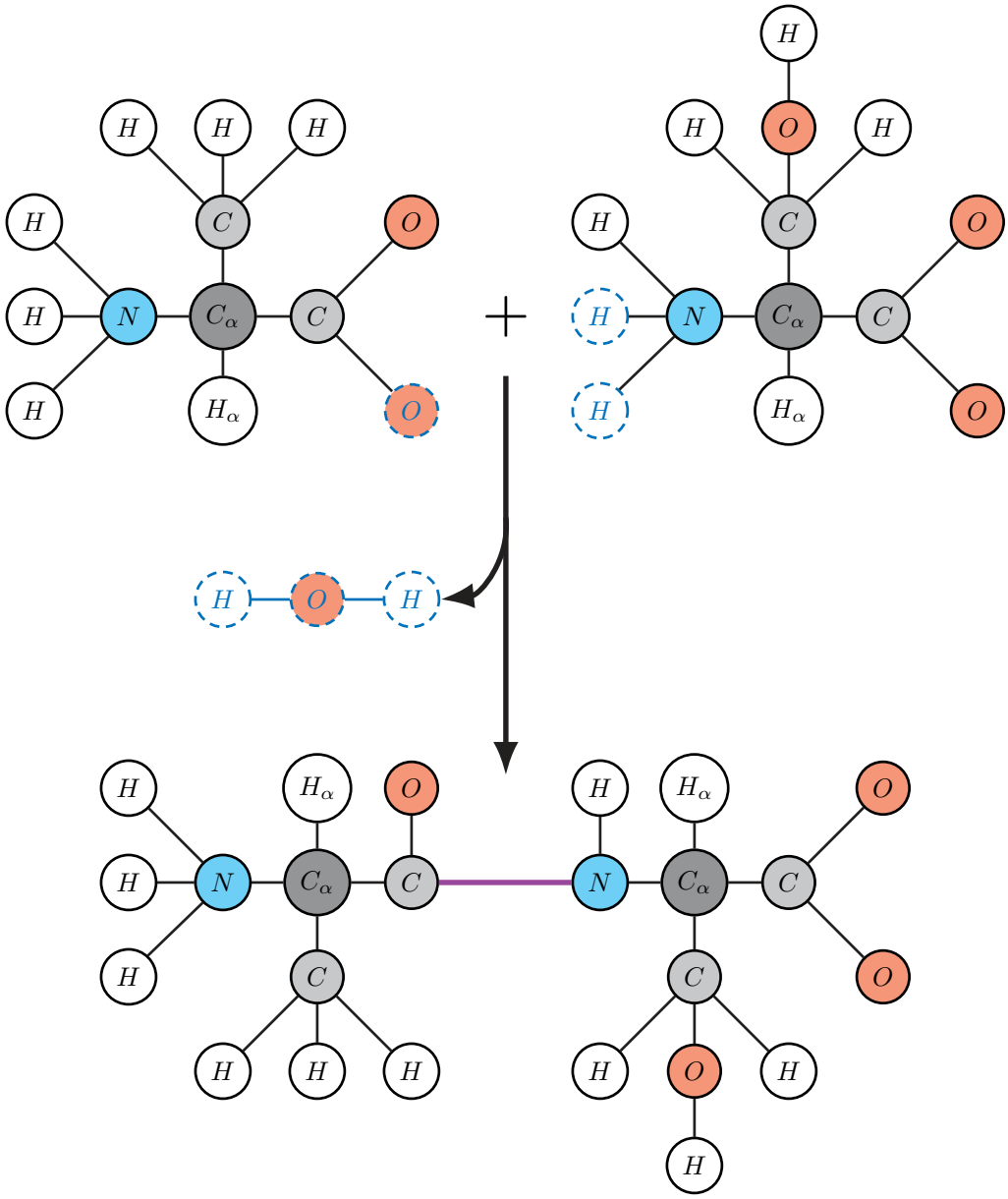
Fonte: IUPAC

formada quando o grupo carboxila do i-ésimo aminoácido e o grupo amina do $(i+1)$-ésimo aminoácido se combinam, eliminando uma molécula de H_2O. As duas extremidades de uma cadeia de n aminoácidos são ditas carboxil-terminal (C-terminal) e amino-terminal (N-terminal). Por convenção, a ordenação da cadeia começa sempre no N-terminal, de modo que o grupo amino do primeiro aminoácido não está envolvido com a ligação peptídica com o segundo, e o grupo carboxila do último aminoácido não está envolvido com a ligação peptídica com o penúltimo. Ou seja, a sequência de aminoácidos em uma proteína é escrita do N-terminal para o C-terminal, começando pelo N no grupo amino do primeiro aminoácido até o C no grupo carboxila do último aminoácido.

Assim, definimos a espinha dorsal de uma proteína com n aminoácidos como sendo a sequência N^i, C^i_α, C^i, com $i \in \{1, \ldots, n\}$. Essa espinha dorsal resultante da sequência de aminoácidos é dita a estrutura primária da proteína. Por conveniência, vamos definir como espinha dorsal estendida o conjunto composto por C_α e seu H_α associado, N e seu H associado do grupo amino, e C e O do grupo carboxila. Dito de outra forma, a espinha dorsal estendida é o resíduo a menos de sua cadeia lateral.

Note que existem exceções, como, por exemplo, P, cujo grupo amino é secundário ($-NH_2\oplus$) e forma um ciclo rígido com a sua cadeia lateral, não apresentando H na espinha dorsal estendida.

Figura 1 – Formação de um peptídio pela ligação de A com S (ligação peptídia em roxo)



1.2.2 Padrão de Nomenclatura

A nomenclatura dos átomos no formato NEF empregado na pesquisa majoritariamente segue o padrão IUPAC, exceto nas cadeias laterais. A Tabela 2 apresenta os nomes para a espinha dorsal estendida da proteína e a codificação usada posteriormente na pesquisa. Nela, empregam-se letras minúsculas gregas (ou equivalentes maiúsculas latinas) no sufixo de cada átomo que se liga a um grupo funcional seguindo a ordem alfabética grega. O grupo funcional de referência é o carboxila. A Figura 3 apresenta a estrutura e

os nomes no padrão NEF de 20 aminoácidos.

Tabela 2 – Espinha dorsal das proteínas.

Átomo	**Código IUPAC**	**Codifição**
Hidrogênio amida	H	1
Nitrogênio amida	N	2
Carbono α	C_α (ou CA)	3
Hidrogênio α	H_α (ou HA)	4
Carbono carbonila	C	5
Oxigênio carbonila	O	6

Fonte: IUPAC

Assim, o C_α é o átomo que está diretamente ligado ao grupo carboxila pelo C da espinha dorsal, o C_β (ou CB) é o átomo que está ligado ao grupo carboxila por meio do C_α, o C_γ ou (CG) é o átomo que está ligado ao grupo carboxila por meio do C_β e assim por diante. Caso exista mais de um átomo na posição, então adiciona-se um número natural após a letra grega, começando a partir do 1 e seguindo a ordem de sucessão. Os átomos que podem estar nestas posições são C, N, O e S.

A única exceção deste formato é o H, que recebe o mesmo sufixo do átomo ao qual está ligado (incluindo o número). Caso exista mais de um hidrogênio na posição, de modo geral, adiciona-se um número após o este sufixo[1]. O número adicionado depende do átomo ao qual H está ligado: nos casos de N, O e S, começando a partir do 1 e seguindo a ordem de sucessão; já no caso de C, depende se C é do grupo metil ($-CH_3$), começando a partir do 1 e seguindo a ordem de sucessão, ou metileno ($-CH_2$), começando a partir do 2 e seguindo a ordem de sucessão.

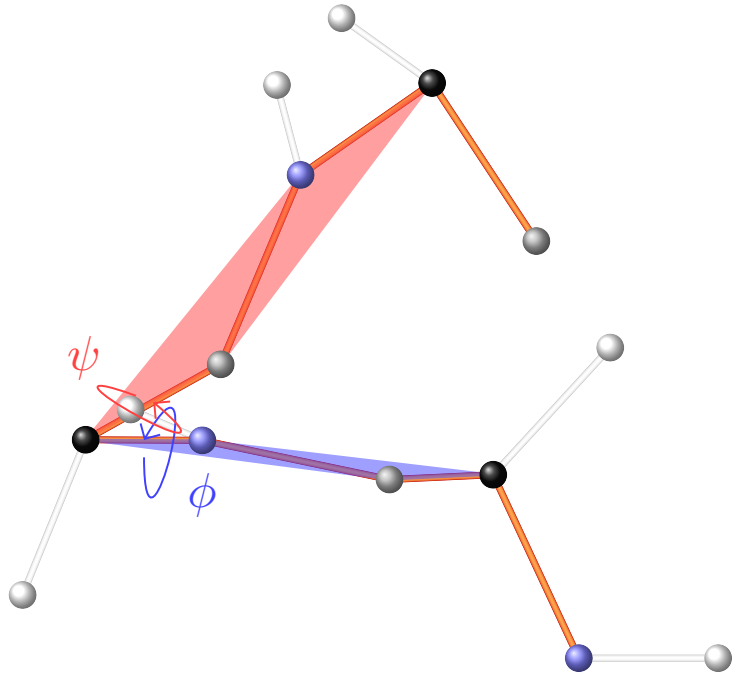
1.2.3 Geometria da Proteína

A ligação peptídica apresenta uma propriedade de dupla ligação parcial. Esta propriedade é atribuída à deslocalização do par de elétrons do átomo de N até grupo carbonil ($C = O$), formando assim uma dupla ligação parcial entre N e C na espinha-dorsal. De fato, os átomos O, C e N têm orbitais moleculares ocupados por elétrons deslocalizados, formando um sistema conjugado. Consequentemente, as três ligações de N nas amidas ($R - C = O$) são planas, ou seja, os átomos $O^{i-1} - C^{i-1} - N^i - H^i$ em torno da ligação peptídica estão dispostos em um plano.

Portanto, a dupla ligação parcial implica na restrição de rotações em torno da ligação do C^i do i-ésimo aminoácido com o N^{i+1} do $(i+1)$-ésimo aminoácido, de modo que C^i_α, C^i, O^i, N^{i+1}, H^{i+1} e C^{i+1}_α devem estar no mesmo plano. Apesar dessa restrição, temos que cada aminoácido pode rotacionar em relação ao próximo aminoácido na cadeia

[1] Em verdade, a depender da presença ou não de estereoespecificidade do próton ou deslocamento químicos degenerados, o carácter adicionado pode ser x, y ou %. No entanto, optamos por descartar estes casos e modelar de forma mais simples esse aspecto do fenômeno.

Figura 2 – Ângulos diedrais ϕ e ψ.



em torno da ligação entre o C_α^{i+1} e o C^{i+1} $(C_\alpha - C)$. Segue que a rotação da cadeia de aminoácidos pode ser descrita como as rotações relativas dos planos criados pelas ligações peptídicas.

Decorre destas condições que a cadeia de aminoácidos é de alguma forma rígida. Ela pode adotar somente um número restrito de conformações que resultam de rotações entre outras ligações da cadeia. Somente duas ligações podem rotacionar devido a estas restrições. Considerando o C_α, podemos ter uma rotação entre a ligação dele com N $(N - C_\alpha)$, dita ϕ, e uma rotação entre a ligação dele com C $(C_\alpha - C)$, dita ψ. Os dois ângulos diedrais ϕ e ψ descrevem a rotação da cadeia em torno das ligações do C_α, como na Figura 2.

O ângulo diedral é definido por 3 segmentos de reta consecutivos entre 4 pontos e descreve a rotação entre o plano composto pelos 3 primeiros pontos e os 3 últimos pontos, tendo como eixo de rotação o segmento de reta entre os 2 pontos que são comuns a ambos os planos. Naturalmente, esse eixo de rotação é a interseção entre os dois planos.

Em proteínas, a espinha-dorsal da cadeia pode ser descrita em termos de ângulos diedrais ϕ, ψ e ω'. Os ângulos ϕ e ψ são ditos ângulos de Ramachandran e descrevem a

rotação em torno das ligações $N - C_\alpha$ e $C_\alpha - C$, respectivamente. Por sua vez, o ângulo ω' é dado pela rotação em torno da ligação peptídica $(C - N)$. No entanto, para cada par de aminoácidos, só precisamos de ϕ e ψ para especificar o formato tridimensional da espinha dorsal.

Note que ϕ é o ângulo de rotação do quarto átomo comparado com o primeiro átomo enquanto observamos o sistema na mesma linha da ligação entre o segundo átomo e o terceiro átomo. De modo a visualizar melhor temos abaixo os pontos que definem cada ângulo diedral e, em vermelho, a ligação cujo segmento de reta é o eixo de rotação:

- O ω é dado por $C_\alpha^{i-1} - C^{i-1} - N^i - C_\alpha^i$;
- O ϕ é dado por $C^{i-1} - N^i - C_\alpha^i - C^i$;
- O ψ é dado por $N^i - C_\alpha^i - C^i - N^{i+1}$.

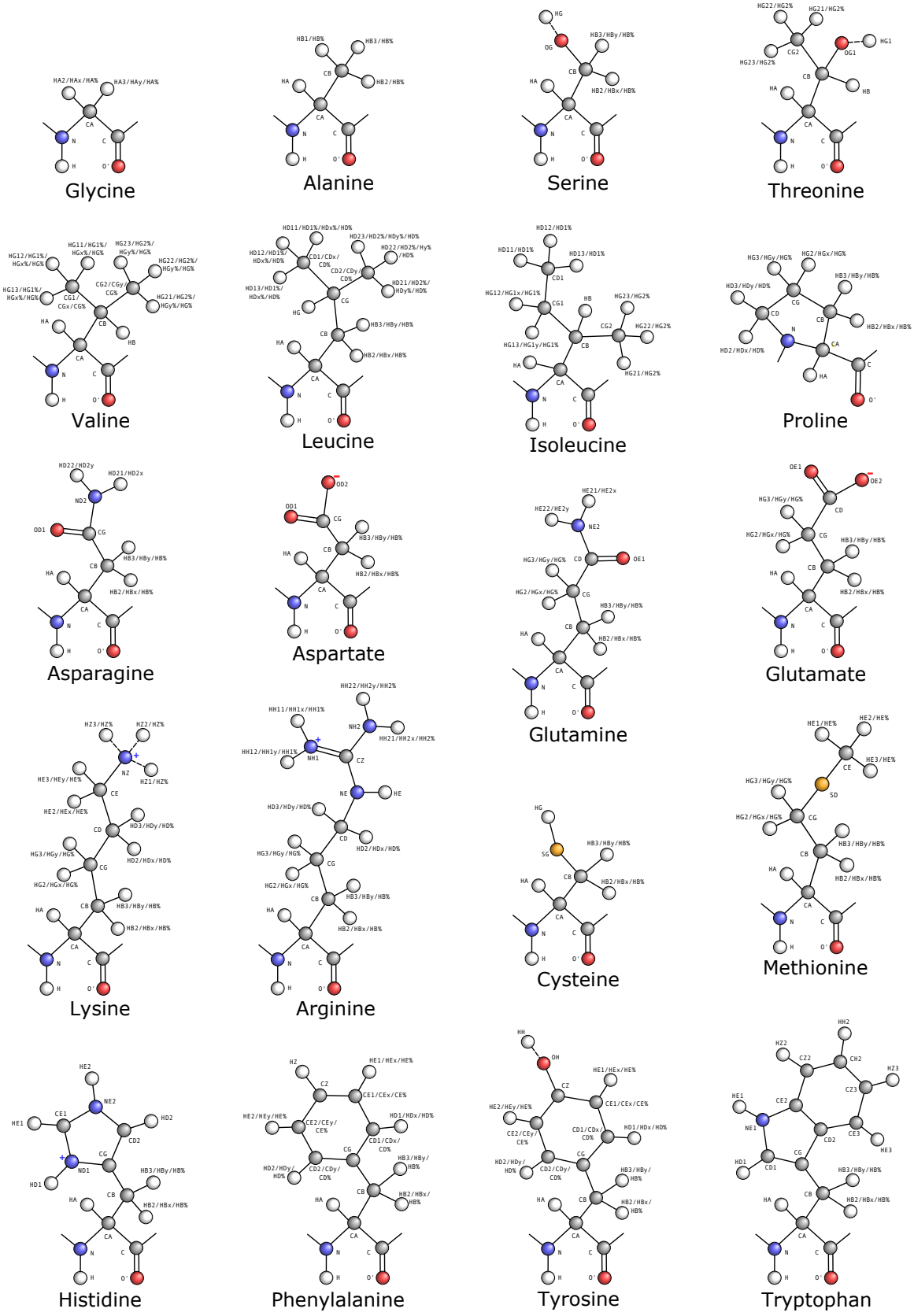
Importante frisar que o ângulo diedral associado ao grupo peptídico, ω', é quase sempre 0 radianos para isômeros *cis* (conformação sin-periplanar) ou π radianos para isômeros *trans* (conformação antiperiplanar). A planaridade da ligação peptídica restringe ω' a essas valores, o que impede a rotação em torno desta ligação na cadeia de aminoácidos. Os aminoácidos adjacentes podem adotar diferentes configurações pela rotação em torno das duas outras ligações da espinha dorsal da cadeia, ϕ e ψ.

Os ângulos diedrais são os mais importantes parâmetros locais que controlam a conformação da proteína, de modo que se conseguimos prever os ângulos ϕ e ψ de uma proteína qualquer, então também podemos prever a sua estrutura tridimensional.

Note que nem todos os ângulos são possíveis. Se dois átomos negativamente carregados de O ficarem próximos, então eles se repelirão pela força magnética. Esse choque é dito choque histérico e também limita o número de possíveis conformações da cadeia de aminoácidos.

O ângulo diedral de cada aminoácido em uma cadeia define a geometria da sua junção com os aminoácidos imediatamente anterior e posterior pelo posicionamento de seu plano peptídico em relação aos planos peptídicos adjacentes. Portanto, o ângulo diedral determina a conformação dos aminoácidos e da cadeia. Muitas das combinações de ângulos não são possíveis pela conjugação da planaridade do plano peptídico e o choque histérico.

Figura 3 – Estrutura e nomes dos átomos dos resíduos (em inglês)



Fonte: CCPN

2 FUNDAMENTOS TEÓRICOS

2.1 Notação e Definições

Definição 1. *Um* ***grafo ponderado nas arestas*** *é uma estrutura* $G = (V, E, \omega)$*, onde* $V = \{v_1, v_2, \ldots, v_n\}$ *é um conjunto finito e não-vazio, cujos elementos são denominados* ***vértices****,* $E \subseteq \{\{v_i, v_j\} : v_i, v_j \in V\}$ *é um conjunto de subconjuntos a dois elementos de* V *denominados* ***arestas****, e* $\omega : V \times V \to \mathbb{R}$ *é uma função que associa a cada par de vértices um valor real denominado* ***peso****.*

Definição 2. *Seja* $G = (V, E)$*. O* ***grau*** *de um vértice* $v_i \in V$*, denotado por* $d(v_i)$ *ou* d_i*, é o número de arestas que nele incidem. Em particular, se* $d(v_i) = 0$*, então* v_i *é dito um* ***vértice isolado****; se* $d(v_i) = 1$*, então* v_i *é dito um* ***vértice-folha****.*

Definição 3. *Seja* $G = (V, E)$*. Se para todo vértice* $v_i \in V$ *temos que* $d(v_i) = k$*, para* $k \leqslant (n-1)$ *e* $k \in \mathbb{N}$*, então* G *é dito um* ***grafo*** k***-regular****. Em particular, se* $k = (n-1)$*, então* G *é dito um* ***grafo completo de ordem*** n *e o denotamos por* K_n*.*

Definição 4. *Seja* $G = (V, E)$*. Um grafo* $H = (V_h, E_h)$ *é dito um* ***subgrafo*** *de* G*, denotado por* $H \subseteq G$*, se* $V_h \subseteq V$ *e* $E_h \subseteq E$*, ou seja, se todos os vértices e arestas de* H *são vértices e arestas de* G*. Em sentido contrário, sob as mesmas condições,* G *é dito um* ***supergrafo*** *de* H*, denotado por* $G \supseteq H$*.*

Definição 5. *Seja* $H = (V_h, E_h)$ *e* $H \subseteq G = (V, E)$*. Se* $E_h = \{\{v_i, v_j\} : v_i, v_j \in V_h, \{v_i, v_j\} \in E\}$*, então* H *é dito* ***subgrafo induzido*** *de* G*. Logo, sendo* H *um subgrafo induzido de* G*, temos que dois vértices quaisquer em* H *são adjacentes se, e somente se, forem adjacentes em* G*. Todos os subgrafos induzidos de* G *podem ser obtidos pela operação de remoção de vértices e o número de subgrafos induzidos de* G *é igual à cardinalidade do conjunto das partes de* V *menos o conjunto vazio, dado por* $2^n - 1$*.*

Definição 6. *Seja* $G = (V, E, \omega)$*. A* ***matriz de adjacência ponderada*** *de* G*,* $A_\omega(G) = (a_{\omega(i,j)})$*, é uma matriz quadrada simétrica de ordem* n *definida da seguinte maneira:*

$$a_{\omega(i,j)} = \omega(v_i, v_j). \tag{2.1}$$

2.2 Problema de Pesquisa

2.2.1 Problema Inverso de Geometria de Distâncias

Suponha um grafo ponderado conexo $G = (V, E, \omega)$ com matriz de adjacência ponderada D. O problema inverso de distância consiste em achar uma posição para

$f(v_i) \in M$, M espaço métrico, com $v_i \in V$, onde $f : V \to M$ é uma imersão tal que, para uma função de distância d,

$$\forall\ \{v_i, v_j\} \in E,\ d(f(v_i), f(v_j)) = \omega(v_i, v_j). \tag{2.2}$$

Em outras palavras, o problema inverso da distância se traduz em posicionar os vértices de V em um espaço métrico M de modo que as distâncias em M respeitem os valores dados por D, satisfazendo a Equação 2.2 (LAVOR; LIBERTI, 2014).

2.2.2 Discretização e o Problema Discretizável de Geometria de Distâncias Moleculares

Dado um grafo ponderado nas arestas arbitrário como entrada, o problema inverso pode ter solução, mas identificar uma realização válida particular pode ser computacionalmente custoso. De fato, o problema inverso no caso geral é NP-Difícil para $k \geqslant 2$ (LAVOR; LIBERTI, 2014). Sob algumas condições, é possível usar a geometria do espaço métrico M para discretizar o procedimento de obtenção da imersão f de modo a diminuir a complexidade de tempo. Em outras palavras, dentro da classe do problema inverso, temos algumas subclasses que são computacionalmente mais tratáveis, como, por exemplo, a subclasse onde o grafo ponderado de entrada é completo.

Além disso, como as entradas do problema inverso são o grafo ponderado e a dimensão do espaço, então é razoável supor que diferentes propriedades topológicas do grafo podem ser de interesse para identificar uma subclasse em que a estrutura adicional possa ser explorada para melhorar o desempenho do procedimento de solução. De modo a restringir a busca pelas propriedades de interesse, vamos fixar o parâmetro de dimensão do espaço com $k = 3$ e determinar que o espaço métrico desejado é $\mathbb{R}^k$. Ao fazermos essa restrição, chegamos no que a literatura denominou problema de geometria de distâncias moleculares (MDGP).

Dado que o grafo ponderado completo é um exemplo de subclasse, então é natural indagar se a esparsidade ou a regularidade do grafo são de interesse. De fato, a depender do padrão de esparsidade, é possível achar uma ordenação parcial $<_M$ para os vértices de $G = (V, E, \omega)$ tal que, para v_i, $i \geqslant 4$, sempre há ao menos 3 valores $\omega(v_i, v_{i-k})$, com $k \in \{1, \ldots, i-1\}$. O conjunto das arestas $\{v_i, v_{i-k}\} \in E$ que atendem essa propriedade é dito conjunto de arestas de discretização, denotado por E_d. A existência desse conjunto faz com que um procedimento iterativo que posiciona uma vértice por iteração seja intuitivamente compreendido como a interseção de esferas dado que o espaço-alvo é $\mathbb{R}^3$.

No entanto, para a interseção de 3 esferas ser finita, é necessário que os centros das 3 esferas $S_{v_i}(v_k)$, $k < i$, não sejam colineares. Dado $v_{(1)}, v_{(2)}, v_{(3)}$, vértices anteriores (mas não necessariamente consecutivos) ao v_i na ordenação $<_M$, garantimos a

não-colinearidade pela desigualdade triangular estrita, tal que $\omega(v_{(1)}, v_{(3)}) < \omega(v_{(1)}, v_{(2)}) + \omega(v_{(2)}, v_{(3)})$.

Sendo satisfeitas a ordenação parcial e a desigualdade triangular estrita, temos que o problema inverso é discretizável. No entanto, a discretização simples tem complexidade exponencial no pior caso, dado que a interseção de 3 esferas com centros não colineares pode ser vazia ou conter, no máximo, 2 pontos. Ou seja, no pior caso, o problema inverso discretizado terá de explorar 2^n possibilidades, onde $|V| = n$. Se não considerarmos a restrição posta no espaço-alvo, então temos que esse é o **problema discretizável de geometria de distâncias** (PDGD).

Considerando a aplicação na conformação de proteínas, as distâncias resultantes de experimentos de NMR apresentam padrões de esparsidade decorrentes das técnicas usadas e do conhecimento físico-químico, fazendo com que algumas distâncias sempre sejam conhecidas. Além disso, a espinha dorsal estendida da proteína fornece uma maneira natural de obter uma ordenação total para os átomos desta estrutura. Segue que a esparsidade do grafo ponderado G não é arbitrária quando se trata de um problema de geometria molecular em proteínas e podemos fazer suposições adicionais em decorrência do conhecimento de domínio específico. Assim, na interseção entre o MDGP e PDGD podemos definir a subclasse do problema discretizável de geometria de distâncias moleculares (PDGDM):

Definição 7. *Seja $G = (V, E, \omega)$ um grafo ponderado e $<_M$ uma ordenação total em V tal que $G[v_{i-3}, v_{i-2}, v_{i-1}, v_i]$, $\forall i \geqslant 4$, é um subgrafo induzido completo (ou uma clique) que satisfaz a desigualdade triangular estrita para os vértices encadeados, $\omega(v_{i-3}, v_{i-1}) < \omega(v_{i-3}, v_{i-2}) + \omega(v_{i-2}, v_{i-1})$. Então o PDGDM consiste em achar, se possível, $f : V \to \mathbb{R}^3$, uma imersão f tal que*

$$\forall\ \{v_i, v_j\} \in E,\ d(f(v_i), f(v_j)) = \omega(v_i, v_j). \tag{2.3}$$

2.2.3 Algoritmo BP

O BP aborda o PDGDM explorando a estrutura de árvore binária do espaço de soluções de modo análogo ao da Busca em Profundidade (DFS) (LIBERTI et al., 2010). Após a sua inicialização, o BP iterativamente posiciona o i-ésimo vértice em uma das 2 posições possíveis $f(v_i)$ (vulgo "esquerda") e $f(v_i)'$ (vulgo "direita"), seguindo a descida até um vértice-folha ou podando um ramo cujos caminhos não sejam viáveis.

Definindo

$$E_d = \{(i, j) : |i - j| < 4\} \tag{2.4}$$

como conjunto de arestas de discretização, e

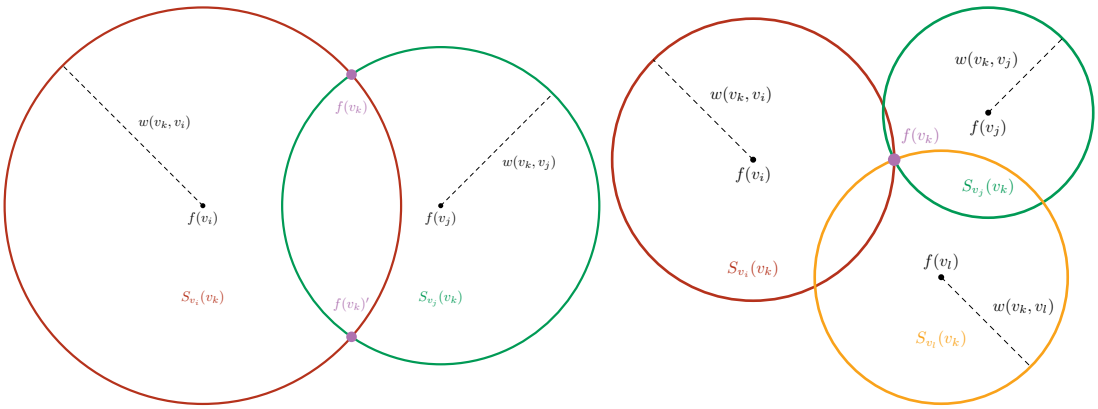
$$E_p = \{(i, i) : |i - j| \geqslant 4\}, \tag{2.5}$$

como conjunto de arestas de poda, tal que $E_d \cup E_p = E$ e $E_d \cap E_p = \varnothing$, temos que, usando E_d, podemos posicionar o i-ésimo vértice, tendo, no máximo, 2 posições possíveis e então, usando E_p, verificar se o posicionamento é válido ou não por meio de um teste de zero. Ou seja, para toda aresta $\{v_i, v_k\} \in E_p$, checamos se $(d(f(v_i), f(v_k))^2 - \omega(v_i, v_k)^2)^2 < \varepsilon$, para $\varepsilon > 0$ dado pelo nível de precisão desejado. Ao final desta iteração, só pode ocorrer um dentre os seguintes resultados:

1. Ambos $f(v_i)$ e $f(v_i)'$ são viáveis. Neste caso, armazenamos ambas as posições e seguimos a DFS da esquerda para a direita.
2. Somente $f(v_i)$ ou $f(v_i)'$, mas não ambos, são viáveis. Neste caso, podamos o caminho do não-viável (removendo seus vértices-filhos do espaço de busca), armazenamos a posição viável e seguimos a DFS.
3. Nem $f(v_i)$ ou $f(v_i)'$ são viáveis. Neste caso, podamos os caminhos não-viáveis, retornamos para o $(i-1)$-ésimo vértice e seguimos o DFS.

Figura 4 – Exemplos de inserções de vértices válidos (em roxo) em $\mathbb{R}^2$ dados 2 vértices de discretização e circunferências $S_{v_i}(v_k)$ com as posições possíveis para v_k com respeito ao $f(v_i)$.

(a) Sem aresta de poda (2 circunferências) (b) Com aresta da poda (3 circunferências)



O algoritmo procede recursivamente até obter uma solução válida ou não ter qualquer caminho válido até seus vértices-folha. Assim, pode-se enxergar o BP como uma DFS com uma fase de inicialização específica e três operações adicionais: **(i)** calcular o $f(v_i)$; **(ii)** checar a viabilidade; e **(iii)** podar um ramo com os vértices-filhos do vértice podado.

Como temos a informação na forma de coordenadas internas, então cada $f(v_i)$ é obtido a partir de $f(v_{i-1})$ e, sequencialmente, uma translação $t_i = [d_{i-1,i}, 0, 0]$, uma

rotação de $\theta = \pi - \theta_{[i-2:i]}$, com $\theta_{[i-2:i]} = 0$ se $0 \leqslant i < 3$, em torno do vetor dado pela transformação do eixo canônico e_3 nas rotações anteriores, e uma rotação $\varphi = \varphi_{[i-3:i]}$ (ou $\varphi = -\varphi_{[i-3:i]}$), com $\varphi_{[i-3:i]} = \pi$, se $i = 2$, ou $\varphi_{[i-3:i]} = \pi$, se $i = 3$, em torno do vetor dado pela transformação do eixo canônico e_1 nas rotações anteriores (THOMPSON, 1967).

$$f(v_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi) & -sen(\varphi) \\ 0 & sen(\varphi) & cos(\varphi) \end{bmatrix} \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \end{bmatrix} + \begin{bmatrix} d_{i-1,i} \\ 0 \\ 0 \end{bmatrix} \right). \quad (2.6)$$

No entanto, essas transformações da Equação 2.6 podem ser representadas no espaço homogêneo em uma única matriz, $B_i \in \mathbb{R}^{4\times4}$, que realiza a composição da translação seguida das duas rotações. Note que, para tal, é necessário que a matriz B_{i-1} seja dada. Logo, o processo é construtivo e parte de $B_1 = I$, matriz identidade de ordem 4.

Para a inicialização, o BP calcula as matrizes associadas aos 3 primeiros vértices e fixa as posições $f(v_1)$, $f(v_2)$, e $f(v_3)$ usando o mesmo procedimento geral para calcular as matrizes B_i achar a posição $f(v_i)$. A distinção da inicialização com relação ao passo geral se dá pelo fato do primeiro vértice ser fixado em $[0, 0, 0, 1]$ e os dois seguintes não necessitarem do ângulo diedral e nem terem aresta de poda. Portanto, a menos de isometria, o estado na inicialização é comum a toda e qualquer conformação válida final. Assim, temos definidas as seguintes matrizes:

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.7)$$

$$B_2 = \begin{bmatrix} -1 & 0 & 0 & -d_{1,2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.8)$$

$$B_3 = \begin{bmatrix} -cos(\theta_{[1:3]}) & -sen(\theta_{[1:3]}) & 0 & -d_{2,3}cos(\theta_{[1:3]}) \\ sen(\theta_{[1:3]}) & -cos(\theta_{[1:3]}) & 0 & d_{2,3}sen(\theta_{[1:3]}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.9)$$

Com o termo geral B_i dado por

$$B_i = \begin{bmatrix} -cos(\theta_{[i-2:i]}) & -sen(\theta_{[i-2:i]}) & 0 & -d_{i-1,i}cos(\theta_{[i-2:i]}) \\ sen(\theta_{[i-2:i]})cos(\varphi_{[i-3:i]}) & -cos(\theta_{[i-2:i]})cos(\varphi_{[i-3:i]}) & -sen(\varphi_{[i-3:i]}) & d_{i-1,i}sen(\theta_{[i-2:i]})cos(\varphi_{[i-3:i]}) \\ sen(\theta_{[i-2:i]})sen(\varphi_{[i-3:i]}) & -cos(\theta_{[i-2:i]})sen(\varphi_{[i-3:i]}) & cos(\varphi_{[i-3:i]}) & d_{i-1,i}sen(\theta_{[i-2:i]})sen(\varphi_{[i-3:i]}) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.10)$$

Note que em B_i temos que conhecer o $sen(\varphi_{[i-3:i]})$. No entanto, temos somente a informação de $[\varphi_{[i-3:i]}]_\sim$, de modo que há somente dois valores possíveis para o seno em decorrência da identidade trigonométrica fundamental[1]: $\pm\sqrt{1-(cos(\varphi_{[i-3:i]}))^2}$. Segue que para a i-ésima iteração, $i \geqslant 4$, temos sempre duas matrizes B_i e B_i', cada um contendo um valor possível para o seno de $[\varphi_{[i-3:i]}]_\sim$. Por conveniência, B_i está associada à $f(v_i)$ e B_i' à $f(v_i)'$, de modo que uma indica um caminho pela esquerda e a outra pela direita no i-ésimo nível da árvore binária.

A partir destas matrizes e sua forma geral B_i podemos obter as posições $f(v_i)$ do seguinte modo:

$$f(v_i) = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = B_i \begin{bmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \\ 1 \end{bmatrix} = B_1 B_2 \dots B_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \forall\ i \in \{1, \dots, n\}. \tag{2.11}$$

A segunda igualdade na Equação 2.11 decorre de querermos expressar a solução em termos da base canônica. Ao expandirmos a $f(v_i) = \left(\prod_{j=1}^{i} B_j\right)[0,0,0,1]$, estamos expressando a origem na base canônica no sistema de coordenadas da i-ésima iteração. Como posicionamos o i-ésimo vértice na origem do sistema de coordenadas associado, então temos que os 3 primeiros elementos de $f(v_i)$ são as suas coordenadas em $\mathbb{R}^3$. Assim, segue que as posições fixadas na inicialização são dadas por

$$f(v_1) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad f(v_2) = \begin{bmatrix} -d_{1,2} \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad f(v_3) = \begin{bmatrix} -d_{1,2} + d_{2,3}cos(\theta_{[1:3]}) \\ d_{2,3}sen(\theta_{[1:3]}) \\ 0 \\ 1 \end{bmatrix}. \tag{2.12}$$

As entradas para o BP consiste dos seguintes parâmetros:

- A árvore ponderada T que representa o espaço de busca já explorado, onde os pesos estão nos vértices v_i e são compostos por:
  - Um vetor V de coordenadas $f(v_i)$ associadas a cada v_i;
  - A matriz de torção cumulada $C = \prod_{j=1}^{i-1} B_j$ associada;
  - O ponteiro para o vértice-pai $P(v_i)$;

[1] Pela Identidade Pitagórica: $sen(x)^2 + cos(x)^2 = 1$.

  - Dois ponteiros para os vértices-filhos $E(v_i), D(v_i)$, respectivamente, à esquerda e à direita.

- O índice i relativo ao i-ésimo vértice a ser posicionado;
- A ordem n da instância;
- O vetor de distâncias encadeadas d;
- O vetor θ dos ângulos planos encadeados;
- O vetor φ dos ângulos diedrais encadeados;
- O valor $\varepsilon > 0$ da tolerância para o teste de zero.

Considerando estes dados, temos que na inicialização T consiste somente de $f(v_1)$, $f(v_2)$, e $f(v_3)$. Temos que o BP simples para obter uma única solução pode ser descrito como no Algoritmo 1. Importante ressaltar que o BP é extremamente competitivo e eficiente para o PDGDM e também é empregado no PDGD, quando não temos a condição da ordem total (LIBERTI; LAVOR; MACULAN, 2008). Além disso, há diversas variações do BP que exploram algumas propriedades do espaço de busca, como, por exemplo, a existência de simetrias.

As principais limitações do BP são a sua serialidade inerente à DFS e o critério de exploração ser fixo, sempre da esquerda para a direita. Com respeito ao problema da serialidade, existem técnicas para paralelizar o BP (MUCHERINO et al., 2010; GRAMACHO et al., 2012). No entanto, a paralelização proposta na literatura apresenta o mesmo problema de uma abordagem ingênua: uma mesma solução pode ter suas partes calculadas múltiplas vezes sem o algoritmo saber que se tratam de trechos da mesma solução. Ou seja, parte da computação é redundante e desperdiçada. O mesmo ocorreria se o BP operasse dois processos, um com o critério de esquerda para direita e outro com o critério da direita para esquerda.

Este fato ressalta a importância da limitação do critério de exploração e sua estreita relação com a serialidade. Como o BP não tem um mecanismo de atualização da preferência entre as alternativas, então a única aprendizagem possível ocorre com relação às podas. Ou seja, a única maneira do BP não explorar inteiramente o espaço de busca como a DFS é podando os ramos com caminhos que não são viáveis. Por este motivo, a limitação do critério de exploração foi enfrentada na construção da Busca Baseada em Frequência (FBS). Iremos expor esse caso na seção 2.2.4.

Por fim, o BP faz uma suposição forte com respeito às mensurações de distância entre átomos. Como visto, é necessária a distância exata entre alguns átomos específicos da espinha dorsal estendida da proteína. No entanto, como essas distâncias são obtidas

Algoritmo 1 – BranchAndPrune

Dados: $T, i, n, d, \theta, \varphi, \varepsilon$
Resultado: T'
se $i = n$ **então**
  retorna T;
fim
$v \leftarrow T[V][i-1]$;
$B_i \leftarrow B(i, d, \theta, \varphi)$;
$C_i \leftarrow T[C][i-1]B_i$;
$x \leftarrow C_i v$;
se $TesteDeZero(v - x, d, \varepsilon)$ **então**
  $T' \leftarrow T \cup \{x\}$;
  $T'[P][i] \leftarrow v$;
  $T'[C][i] \leftarrow C_i$;
  $T'[E][i-1] \leftarrow x$;
  $BranchAndPrune(T', i+1, n, d, \theta, \varphi, \varepsilon)$;
senão
  $T[E][i-1] \leftarrow NULL$;
fim
$B'_i \leftarrow B'(i, d, \theta, \varphi)$;
$C'_i \leftarrow T[C][i-1]B'_i$;
$y \leftarrow C'_i v$;
se $TesteDeZero(v - y, d, \varepsilon)$ **então**
  $T' \leftarrow T \cup \{y\}$;
  $T'[P][i] \leftarrow v$;
  $T'[C][i] \leftarrow C'_i$;
  $T'[D][i-1] \leftarrow y$;
  $BranchAndPrune(T', i+1, n, d, \theta, \varphi, \varepsilon)$;
senão
  $T[D][i-1] \leftarrow NULL$;
fim

experimentalmente, então os valores obtidos possuem um erro associado à mensuração. Além disso, há limitações físico-químicas que nos garantem limitantes inferiores para essas distâncias. Estas condições nos levam a considerar algumas das distâncias como intervalos.

Assim, o BP Intervalar (iBP) é uma adaptação do BP para dar conta do problema com distâncias intervalares (LAVOR; LIBERTI; MUCHERINO, 2011). No entanto, dada a natureza discreta do algoritmo, ao lidar com a incerteza na forma da distância intervalar, o algoritmo assume máxima entropia, ou seja, que o valor verdadeiro para a distância está uniformemente distribuído no intervalo, e tenta discretizar amostrando alguns valores pontuais equidistantes dentro do intervalo. Para cada valor amostrado, o algoritmo volta a proceder como no BP tradicional. No entanto, essa abordagem não garante que uma solução será encontrada e ainda é extremamente custosa. Novamente, isto é uma questão associada ao critério de seleção do caminho, que agora não se limita

somente ao qual vértice-filho escolher, como também do valor de distância associado a este vértice-filho. Em outras palavras, o critério de seleção diz respeito à incerteza, mas não tenta quantificá-la ou investigá-la, assumindo uma suposição forte de máxima entropia.

Como o caminho "esquerdo" e o "direito" são assumidos igualmente possíveis, então uma regra fixa sistemática para escolher da esquerda para a direita faz sentido. Como o intervalo é assumido uniformemente distribuído, então escolher um número de pontos equidistantes é uma estratégia tão boa quanto qualquer outra e também faz sentido. Assim, questionar essa suposição e quantificar a incerteza se apresentam como alternativas naturais para o problema, em especial no caso intervalar.

2.2.4 Busca Probabilística no Espaço de Solução no BP

Ao contrário da DFS, Marques et. al. propuseram uma abordagem *Best-First Search* para o BP, que escolhe um caminho que seja localmente melhor dada a informação empírica prévia sobre os padrões de caminhos mais comuns em proteínas (MARQUES et al., 2024). Para isso, empregaram os dados contidos no *Protein Data Bank* (PDB), de modo análogo ao exposto na seção 3.1, para produzir o FBS.

Partindo de uma ordenação, cada posição associada a um vértice v_i, $i \geqslant 4$, pode ser associada a um *bit* b_i que indica a posição relativa ao plano formado por $f(v_{i-1})$, $f(v_{i-2})$ e $f(v_{i-3})$. Como os 3 primeiros átomos não possuem esse plano de referência, então convenciona-se que $b_1 = b_2 = b_3 = 0$. Este processamento dos dados empíricos é melhor explorado na seção 3.1.2. De toda forma, cada trecho conhecido de proteína contido no PDB pode ter a sua informação geométrica representada por uma sequência finita de binários. Assim, há uma relação entre a sequência finita de binários e uma solução na árvore binária do espaço de busca do PDGDM.

Considerando que o formato das proteínas não é arbitrário e supondo que há certa preferência natural por certos padrões de binários, o FBS explora a informação geométrica disponível nestes trechos e produz um caminho a partir da frequência relativa de subsequências finitas de binários com tamanhos fixados nos múltiplos de 5 até 25 (inclusive). Ao invés de seguir da esquerda para a direita, o critério de seleção opta pelo trecho mais provável a partir dos dados empíricos.

Em outras palavras, o FBS ataca a limitação do BP relativa à escolha de um caminho na árvore binária que compõe o espaço de busca. Mediante a incerteza de qual é o caminho correto, o BP opta por seguir da esquerda para a direita como uma busca em profundidade, ao passo que o FBS utiliza as frequências dos binários em dados reais para estimar um caminho mais provável a partir da concatenação de trechos com tamanho fixo.

No entanto, o FBS não utiliza a informação dos trechos já percorridos até o estado corrente. Assim, o algoritmo testa em cada iteração os trechos mais prováveis

independente do que foi feito anteriormente. Como o procedimento não tem memória do que ocorreu na iteração anterior, então o modelo probabilístico implícito está atribuindo a cada trecho uma variável aleatória independente com mesma distribuição. Essa é uma suposição muito forte do ponto de vista da modelagem.

Por outro lado, essa suposição faz com que a paralelização da computação necessária seja significativamente mais simples do que as propostas de paralelizar o DFS no BP. De fato, dada a independência assumida, múltiplos trechos podem ser testados simultaneamente e não é necessária nenhuma adaptação especial para integrar os resultados de diferentes processamentos paralelos. Além disso, também não há desperdício computacional devido a cálculos redundantes, porque não há como dois processos independentes calcularem uma mesma resposta, a menos de simetria.

De modo ingênuo, ambos BP e FBS são equivalentes em complexidade no pior caso. Dito isto, Marques et. al. mostraram que as distribuições dos trechos não são uniformes e que, em média, o FBS consegue achar uma solução visitando menos vértices do que o BP (MARQUES et al., 2024). Em específico, a diferença de desempenho entre o critério baseado em uma preferência dada pela experiência empírica no FBS e o critério fixo no BP é estatisticamente significante e aumenta com relação ao tamanho da instância.

A limitação do FBS é a forte suposição do seu modelo probabilístico implícito e a limitação da informação que utiliza para determinar o trecho mais provável. Trivialmente, um refinamento possível para o FBS consiste justamente em amenizar as suposições e ampliar a gama de informações passíveis de uso no modelo. Logo, é natural que questionemos se é possível incorporar mais informações, preferencialmente de modo sistemático e não *ad hoc*, e se há indícios que justifiquem o esforço e provável aumento de custo computacional para incorporar mais informações.

2.3 Proposta Teórica

De modo a refinar o FBS e atacar a outra limitação do BP, relativa à serialidade do algoritmo, propomos a utilização de um modelo probabilístico gráfico. Esta classe de modelos emprega a teoria das probabilidades para quantificar a incerteza associada ao fenômeno por meio de variáveis aleatórias, e a teoria dos grafos para estabelecer uma estrutura de independência entre essas variáveis aleatórias que permita a decomposição da distribuição de probabilidade conjunta em módulos mais simples.

Em um modelo probabilístico gráfico, cada variável aleatória é representada por um vértice e as dependências diretas entre as variáveis são representadas por arestas (ou arcos) entre vértices associados. A informação sobre o valor observado de uma variável é propagado pelo grafo para atualizar as distribuições de probabilidade das demais variáveis não-observadas. Como estas distribuições podem ser discretas ou contínuas, então esta

classe de modelo fornece um modo híbrido, envolvendo uma parte discreta e, possivelmente, uma parte contínua, capaz de incorporar a vantagem de ambos para abordar o problema. Para o nosso caso, o BP já fornece *ab initio* uma forma de modelar graficamente o fenômeno usando um digrafo acíclico (DAG, do inglês *Direct Acyclic Graph*) junto com distribuições de probabilidade condicional associadas.

A estratégia proposta é modularizar a nossa abordagem com respeito ao fenômeno, em especial na identificação da sequência finita de binários que é solução de um PDGDM, de modo que possamos aproveitar a arquitetura das CPU modernos e paralelizar diferentes buscas no espaço de soluções. Além disso, esta classe de modelos permite – com alguns ajustes – a incorporação de informações de diversas fontes e que atualizam a sua importância relativa no modelo de acordo com o que é observado em cada instância investigada. Assim, atacamos simultaneamente ambas as limitações do BP e garantimos que o modelo seja extensível por definição.

Embora existam modelos probabilísticos gráficos mais complexos e amplos, vamos abordar somente alguns casos mais simples, como a cadeia de Markov (CM) e o modelo oculto de Markov (MOM). Com base nestes casos, vamos estabelecer um modelo específico para o nosso problema que seja um caso particular de um modelo probabilístico gráfico mais geral. Com isso, o nosso objetivo é avaliar o uso deste modelo no BP para, caso seja razoável, estendê-lo para lidar com problemas mais difíceis, como o caso em que as distâncias são intervalares. Dada a similaridade entre BP e iBP, espera-se que os ajustes sejam diretos e que também permitam lidar com algumas das limitações relativas ao iBP.

Assim, caso a proposta se mostre apta, teremos um alicerce ou um *framework* para desenvolver modelos mais complexos e que incorporem mais informações. No entanto, antes de especificar a nossa proposta, vamos explorar alguns modelos mais simples para construir aos poucos a intuição do porquê da especificação do nosso modelo ser como descrita na seção 3.3.1. Todas as informações abaixo foram extraídas de Rabiner (1989), Norris (1997), Ghahramani (2001), Häggström (2002), Frühwirth-Schnatter (2006), Koller e Friedman (2009), Murphy (2012), Zucchini, MacDonald e Langrock (2017), Zuanetti e Milan (2017).

2.3.1 Modelo Probabilístico Simples

Em um modelo probabilístico simples, temos que uma sequência finita $\{x_t\}_{i=1}^{N}$ é observada e assumimos, para toda variável aleatória X_i, que $X_i \perp X_j$, $\forall\ i \neq j$ e $X_i \sim f$, $\forall\ i \in \{1, \ldots, N\}$. Em outras palavras, que as variáveis aleatórias são iids. Neste caso, o procedimento usual consiste em achar a família de distribuições mais similar aos dados ou escolher uma família em específico por motivos teóricos, e então usar métodos de inferência para estimar os parâmetros que melhor se ajustam aos dados.

Figura 5 – Representação gráfica do modelo probabilístico simples

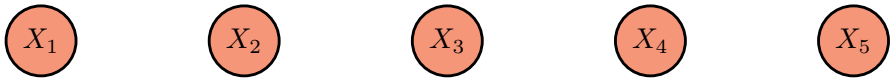
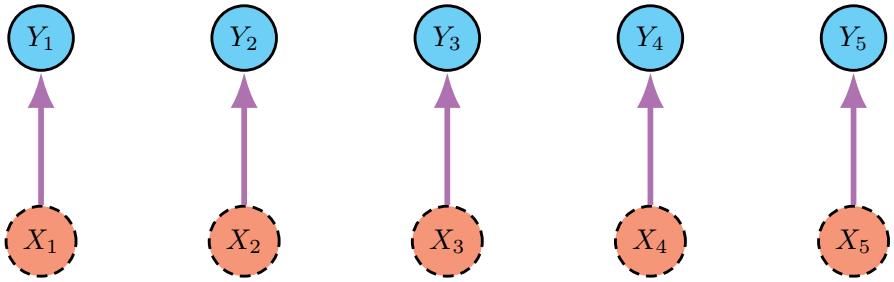


Figura 6 – Representação gráfica do modelo de misturas finitas



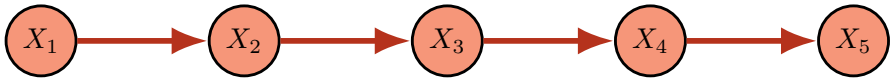
Na Figura 5 temos a representação do modelo simples como um modelo probabilístico gráfico em que o grafo associado é um grafo vazio, ou seja, sem arestas. Como todos os pares de variáveis aleatórias são independentes, então naturalmente não há arestas entre quaisquer dos vértices.

O FBS usa um ajuste deste modelo em que cada variável aleatória é um caminho de tamanho k, $k \in \{5, 10, 15, 20, 25\}$, com distribuição categórica baseada na contagem das subsequência finita de binários. Note que X_t não depende de X_{t-1}, então um novo caminho é selecionado em cada iteração até chegar no vértice-folha, ignorando os caminhos já percorridos ou o valor observado para X_{t-1}.

2.3.2 Modelo de Misturas Finitas

Um modelo mais geral é o de misturas finitas (MF), que permitem abordar fenômenos que apresentam sobredispersão, multimodalidade ou outros indícios da presença de subpopulações com distribuições distintas (componentes). Este modelo é desenhado para lidar com heterogeneidade não-observada na população geral, tratando-a como uma variável aleatória latente que pode assumir um número finito de estados representando cada subpopulação. No caso mais simples, os parâmetros são independentes entre componentes distintas e denominamos o modelo como de misturas finitas independentes.

Figura 7 – Representação gráfica do modelo de cadeia de Markov



A Figura 6 mostra uma representação do modelo de misturas finitas independentes como um modelo probabilístico gráfico. As variáveis latentes X_i estão tracejadas para significar que são não-observáveis, e a dependência das variáveis observáveis Y_i com relação às latentes associadas X_i está representada como uma aresta direcionada. Assim, a distribuição conjunta $P(X_{[1:5]}, Y_{[1:5]})$ pode ser fatorada como $\prod_{i=1}^{5} P(X_i)P(Y_i|X_i)$.

No caso simples, sejam $\delta_1, \ldots, \delta_m$ as probabilidades atribuídas a cada componente e $p_1, \ldots, p_m$ as suas funções de densidade de probabilidade. Então a variável aleatória X tem distribuição de mistura e $p(x) = \sum_{i=1}^{m} \delta_i p_i(x)$. A intuição do modelo gráfico é especialmente útil para enxergar que a fórmula de $p(x)$ é somente a soma das probabilidades $P(X_i)P(Y_i|X_i)$ de cada módulo (ou termo) da nossa fatoração. Como são independentes, então trivialmente a probabilidade da união é a soma das probabilidades dos módulos.

2.3.3 Cadeia de Markov

A Cadeia de Markov é o modelo mais simples para relaxar a suposição de independência e modelar graficamente as relações de dependência entre variáveis aleatórias. Embora os dois modelos anteriores possam ser representados como um modelo probabilístico gráfico, em geral não são considerados como tal, exceto, possivelmente, pelo modelos de misturas finitas dependentes. Dada a importância da CM, daremos um enfoque nas suas principais propriedades e seus parâmetros.

Seja uma sequência de variáveis aleatórias discretas $\{X_t\}$. Então $\{X_t\}$ é dita uma CM se, para todo $t \in \mathbb{N}$, satisfaz a propriedade Markoviana, ou seja,

$$P(X_{t+1}|X_1, \ldots, X_t) = P(X_{t+1}|X_t). \tag{2.13}$$

Na Figura 7 temos a representação gráfica de uma CM. As variáveis aleatórias X_i apresentam a relação de dependência com respeito à X_{i-1} por meio de uma aresta direcionada. Logo, a distribuição conjunta $P(X_{[1:5]})$ pode ser fatorada como $P(X_1)\prod_{i=2}^{5} P(X_i|X_{i-1})$.

Os parâmetros que especificam uma CM são a matriz de transição e o vetor de distribuição inicial. Dado estes parâmetros, conseguimos inferir sobre o estado da CM em um tempo t qualquer. Portanto, achar estas quantidades para um conjunto de dados e entender as suas propriedades analíticas e computacionais é fundamental para a compreensão da utilidade deste modelo.

Primeiramente, se a CM finita $\{X_t\}$ é tal que $|Im(X_t)| = M$, então dizemos que a CM tem M estados e temos que a matriz de transição $T(t)$ de 1-passo associada é uma matriz estocástica de ordem M tal que

$$(t_{i,j})(t) = P(X_{t+1} = j | X_t = i). \tag{2.14}$$

Se as probabilidades das entradas de $(t_{i,j})$ não dependem de t, a CM é dita homogênea e podemos omitir o argumento de T. Neste caso, pela Equação de Chapman-Kolmogorov, podemos construir uma matriz Γ de m-passos, $m = u + w$, tal que

$$\Gamma(m) = T^m \implies \Gamma(u + w) = T^u T^w \tag{2.15}$$

$$(\gamma_{i,j}(m)) = \sum_{k=1}^{M} t_{i,k}^u t_{k,j}^w = (t_{i,j}^m). \tag{2.16}$$

Intuitivamente, isto se traduz em dizer que dar m passos é aplicar m vezes a operação que dá 1 passo. Além disso, como $m = u + w$, então dar m passos é equivalente a dar u passos e então dar mais w passos. Assim, dado vetor-linha representando um estado da CM, podemos obter a sua configuração no passo seguinte pela multiplicação à direita por T. Essa propriedade permite identificar de modo direto a periodicidade da CM, o tempo esperado de primeira passagem em determinado vértice *etc.* O problema com CM não-homogênea também é tratável, mas exige algumas adaptações e perde parte da intuitividade dos passos. Por isso, vamos assumir que a CM abordada neste trabalho é homogênea.

Considerando agora as probabilidades incondicionais $P(X_t = j)$ como quantidades de interesse, tal que, para todo $t \in \mathbb{N}$, temos o vetor-linha

$$u(t) = \begin{bmatrix} P(X_t = 1) & \dots & P(X_t = M) \end{bmatrix},$$

então, tomando $t = 1$, temos que o vetor-linha $u(1)$ é dito vetor de distribuição inicial da CM. Caso a CM seja homogênea, temos que

$$u(t + 1) = u(t)T = u(1)T^t. \tag{2.17}$$

Sob algumas condições, temos também que existe um vetor de distribuição estacionária π, ou seja, um vetor-linha tal que

$$\pi T = \pi, \quad \sum_i \pi_i = 1, \pi_i \geqslant 0. \tag{2.18}$$

Uma CM finita sempre possui ao menos um vetor estacionário. Se T é irredutível, então este vetor é único; se a CM for aperiódica, então as linhas de $\lim_{n\to\infty} T^n$ convergem para este vetor. Pela natureza do que estamos modelando no nosso trabalho em específico, como exposto na seção 3.3.1, temos que a CM de interesse é aperiódica e ergódica, ou seja, contém somente uma classe de estados que é recorrente e aperiódica. Essas suposições são fortes e fazem com que a CM empregada aqui seja simples e eficiente.

Suponha que temos uma sequência finita binária $\{x_t\}_{t=1}^N$ com $N = 24$ valores observados, então é possível identificar as passagens entre os estados possíveis que ocorreram, de modo que

$$\begin{aligned}
\{x_t\} &= (1,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,0,1,1)\\
[0 \to 0] &= (1,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,0,1,1)\\
[0 \to 1] &= (1,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,0,1,1)\\
[1 \to 0] &= (1,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,0,1,1)\\
[1 \to 1] &= (1,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,0,1,1),
\end{aligned}$$

onde as linhas $[x \to y]$ contém as ocorrências em que o estado mudou de x para y. A Figura 8 apresenta uma representação da matriz T da CM que modela essa sequência finita de binários. Note que $P(X_{t+1} = j | X_t = i)$ é abreviado por $p_{i,j}$, dado que não precisamos do t pela suposição de homogeneidade.

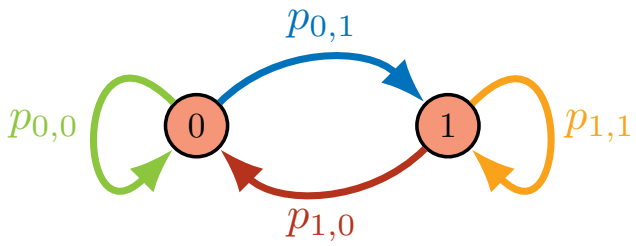
Seja $f_{i,j}$ o número de vezes que ocorreu uma transição do estado i para o estado j em $\{x_t\}_{t=1}^N$. Então,

$$\begin{aligned}
f_{0,0} &= 2 \qquad f_{0,1} = 7\\
f_{1,0} &= 7 \qquad f_{1,1} = 7.
\end{aligned}$$

Essa contagem de transições é a informação central para ajustarmos o modelo da CM aos dados. Há várias formas de estimar T, no entanto, supondo que sabemos qual é o estado inicial, a mais intuitiva é a estimação por verossimilhança.

Suponha que queremos estimar os $M^2 - M$ parâmetros $(t_{i,j})$ da matriz de transição T de uma CM $\{X_t\}$ a partir de observações $x_1, \ldots, x_N$, com $f_{i,j}$ denotando o

Figura 8 – Gráfico de estados de uma CM binária



número de transições observadas do estado i para o estado j. O estimador de máxima verossimilhança condicionado na primeira observação x_1 é dado por

$$L(T|\mathbf{x}) = \prod_{i=1}^{M}\prod_{j=1}^{M}(t_{i,j})^{f_{i,j}}. \tag{2.19}$$

Aplicando o logaritmo, temos a log-verossimilhança dada por

$$l(T|\mathbf{x}) = \sum_{i=1}^{M}\left(\sum_{j=1}^{M} f_{i,j} ln(t_{i,j})\right), \tag{2.20}$$

onde o máximo global de $l(\theta|\mathbf{x})$ obtido maximizando cada termo dentro dos parênteses separadamente. No entanto, não é necessário percorrer todos os termos.

Como T é matriz estocástica, sabemos que $t_{i,i} = 1 - \sum_{j\neq i} t_{i,j}$. Além disso, podemos diferenciar cada termo dentro dos parênteses com respeito a $t_{i,j}$ e igualar a 0, tal que

$$\frac{\partial}{\partial t_{i,j}}\left(\sum_{j=1}^{M} f_{i,j} ln(t_{i,j})\right) = \frac{-f_{i,i}}{1-\sum_{j\neq i} t_{i,j}} + \frac{f_{i,j}}{t_{i,j}} = \frac{-f_{i,i}}{t_{i,i}} + \frac{f_{i,j}}{t_{i,j}} = 0. \tag{2.21}$$

Assim, a menos que algum denominador seja 0[2], podemos usar o fato da matriz T ser estocástica para fazer a soma em j. Como $\sum_j t_{i,j} = 1$, então temos que

$$\sum_j f_{i,j} t_{i,i} = \sum_j f_{i,i} t_{i,j} \implies t_{i,i} = \frac{f_{i,i}}{\sum_j f_{i,j}} \tag{2.22}$$

[2] Note que pela topologia do grafo isto pode ocorrer se, e somente se, o estado i for um sorvedouro ou não tiver um laço. Nenhum dos dois casos pode ocorrer no caso de interesse dado o número de estados e a suposição de ergodicidade da cadeia.

Usando o valor de $t_{i,i}$ obtido acima na equação original, chegamos aos estimadores de máxima verossimilhança condicionados à observação x_1 dados por

$$\hat{t}_{i,i} = \frac{f_{i,i}}{\sum_{j=1}^{M} f_{i,j}} \quad \text{e} \quad \hat{t}_{i,j} = \frac{f_{i,j}}{\sum_{j=1}^{M} f_{i,j}}, \tag{2.23}$$

ou seja, os estimadores são simplesmente as probabilidades empíricas de transição utilizando a contagem. Outro método razoável seria usar a conjugação Dirichlet-Categórica para estimar os parâmetros em uma abordagem Bayesiana. Para dados suficientemente grandes, ambos os estimadores ficam arbitrariamente próximos. Esta abordagem alternativa é especialmente útil se tivéssemos uma amostra pequena ou se alguma transição de estados não ocorresse na amostra, quando sabemos que empiricamente essa transição ocorre no fenômeno estudado.

Por fim, a estimação do $u(1)$ é, no caso mais simples, a média aritmética dos valores observados na primeira posição em uma amostra composta por um conjunto de K realizações independente de $\{X_t\}$. Assim, especificamos o modelo CM estabelecendo $\{X_t\} = (T, u(1))$ para os valores obtidos a partir das observações.

2.3.4 Cadeia de Markov de Ordem Superior

Uma generalização da cadeia de Markov é obtida se relaxarmos a propriedade Markoviana de modo que, para $X_{[a:b]} = (X_a, \ldots, X_b)$, $a < b$, e $l \geqslant 2$,

$$P(X_{t+1}|X_{[1:t]}) = P(X_{t+1}|X_{[t-l+1:t]}). \tag{2.24}$$

Para $\mathbf{Y}_t = X_{[t-l+1:t]}$, temos que $\{\mathbf{Y}_t\}$ é uma cadeia de Markov em M^l. Embora exija-se alguns ajustes para obter algumas das propriedades vistas na seção anterior, não é necessário novo arcabouço teórico algum.

Supondo $l = 2$, uma cadeia de Markov de ordem 2 estacionária é caracterizada pelas probabilidades de transição

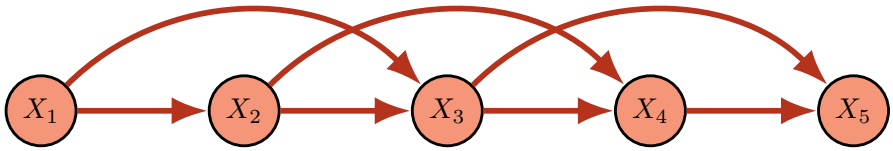
$$(t_{i,j,k}) = P(X_{t+1} = k|X_t = j, X_{t-1} = i), \tag{2.25}$$

e pela matriz estocástica da distribuição estacionária dada, para todo $t \in \mathbb{N}$, por

$$u(j, k) = P(X_{t-1} = j, X_t = k), \tag{2.26}$$

que satisfaz

Figura 9 – Representação gráfica do modelo de cadeia de Markov de ordem 2



$$u(j,k) = \sum_{i=1}^{M} u(i,j)t_{i,j,k} \quad \text{e} \quad \sum_{j=1}^{M}\sum_{k=1}^{M} u(j,k) = 1. \tag{2.27}$$

Suponha uma CM $\{X_t\}$ binária de ordem 2 como na Figura 9, com $X_t \in \{0,1\}$ definida pelas seguintes probabilidades de transição

$$a = P(X_t = 1 | X_{t-1} = 0, C_{t-2} = 0), \tag{2.28}$$
$$b = P(X_t = 0 | X_{t-1} = 1, C_{t-2} = 1), \tag{2.29}$$
$$c = P(X_t = 0 | X_{t-1} = 1, C_{t-2} = 0), \tag{2.30}$$
$$d = P(X_t = 1 | X_{t-1} = 0, C_{t-2} = 1). \tag{2.31}$$

A CM $\{Y_t = (X_{t-1}, X_t)\}$ é uma CM de ordem 1 com $Y_t \in \{(0,0),(0,1),(1,0),(1,1)\}$ e matriz de transição

$$T = \begin{bmatrix} 1-a & a & 0 & 0 \\ 0 & 0 & c & 1-c \\ 1-d & d & 0 & 0 \\ 0 & 0 & b & 1-b \end{bmatrix}. \tag{2.32}$$

Note que a matriz T neste caso possui uma esparsidade estrutural, dado que é impossível a transição direta de $(1,0)$ para $(1,1)$. O último passo do primeiro elemento deve ser igual ao primeiro passo do segundo elemento para formar um caminho válido. De toda forma, vemos que a intuição da CM simples permanece útil e as operações matemáticas seguem essa noção de passos em um caminho.

A distribuição estacionária de $\{Y_t\}$ é proporcional ao vetor

$$(b(1-d), ab, ab, a(1-c)), \tag{2.33}$$

do que segue que a matriz $u(j,k)$ de distribuição bivariada estacionária para $\{X_t\}$ é

$$\frac{1}{b(1-d)+2ab+a(1-c)}\begin{bmatrix} b(1-d) & ab \\ ab & a(1-c) \end{bmatrix}. \tag{2.34}$$

O uso de CM de ordem maior aumenta o número de parâmetros do modelo, de modo que uma abordagem ingênua de um modelo de ordem l com M estados possui $M^l(M-1)$ parâmetros independentes. Algumas técnicas mais avançadas de modelagem permitem reduzir significativamente o número de parâmetros. De fato, é possível modelar uma CM de ordem superior de modo que o aumento da complexidade seja linear com respeito à ordem l (RAFTERY, 1985).

2.3.5 Séries Temporais

Uma série temporal é uma coleção $\{y_t\}_{t=-\infty}^{\infty}$ tal que uma amostra observada $(y_1,\ldots,y_T)$ é um segmento finito desta coleção.

Um operador de série temporal é uma função que transforma uma ou mais séries temporais $\{x_t\}_{t=-\infty}^{\infty}$ e $\{w_t\}_{t=-\infty}^{\infty}$ em uma nova sequência $\{y_t\}_{t=-\infty}^{\infty}$. Em particular, o operador de *lag* L é definido por $y_t = Lx_t = x_{t-1}$ e pode ser aplicado recursivamente de modo que $y_t = L^n x_t = x_{t-n}$.

Uma equação de diferenças é uma expressão que relaciona uma variável y_t com seus valores prévios. Para o caso onde queremos escrever y_t em função do termo anterior e um coeficiente ϕ, temos uma equação de diferenças de primeira ordem e podemos escrever

$$y_t = \phi y_{t-1} + w_t = \phi L y_t + w_t \implies (1-\phi L)y_t = w_t. \tag{2.35}$$

Se $|\phi| < 1$ e $\{y_t\}_{t=-\infty}^{\infty}$ é limitada, $|y_t| < M, \forall t \in \mathbb{R}$, então $\lim_{k\to\infty}(1+\phi L+\ldots+\phi^k L^k) = (1-\phi L)^{-1}$ é uma progressão geométrica no operador L. Portanto,

$$\lim_{k\to\infty}(1+\phi L+\ldots+\phi^k L^k) = (1-\phi L)^{-1}. \tag{2.36}$$

Isto implica que

$$y_t = (1-\phi L)^{-1} w_t = w_t + \phi w_{t-1} + \phi^2 w_{t-2} + \ldots. \tag{2.37}$$

Ou seja, para um valor suficientemente grande, y_t é aproximadamente uma soma ponderada de k termos $w_t,\ldots,w_{t-k+1}$.

Suponha que $w_t = c + \varepsilon_t$, com $E[\varepsilon_t] = 0$ e $Var(\varepsilon_t) < \infty$, $\varepsilon_i \perp \varepsilon_j, \forall i \neq j$. Então, para uma variável aleatória Y_t, temos

$$Y_t = c + \phi Y_{t-1} + \varepsilon_t. \tag{2.38}$$

Quando $|\phi| < 1$, temos que a soma das potências do valor absoluto de ϕ formam uma série geométrica

$$\sum_{j=0}^{\infty} |\phi|^j = \frac{1}{1 - |\phi|}. \tag{2.39}$$

Tomando a expectância e considerando a sua linearidade, temos que

$$E[Y_t] = \frac{c}{1 - \phi} = \mu, \tag{2.40}$$

e dizemos que a média do processo é μ. Isto é dito modelo autoregressivo de ordem 1, denotado AR(1), e pode ser estendido para p arbitrário, denotado AR(p). Importante ressaltar que o AR(1) segue a propriedade Markoviana, ou seja, o futuro é independente do passado dado o presente. Isso não é necessariamente verdadeiro para $p > 1$.

No entanto, os modelos autoregressivos estão interessados nos valores observados. Os componentes estocásticos ε_t aparecem como o erro associado à t-ésima observação, sendo independente e identicamente distribuídos com relação aos demais erros. A relação de dependência se dá entre os componentes determinísticos. Em contraposição, na CM, o interesse é a incerteza e há uma relação de dependência nos componentes estocásticos.

Há diversos outros modelos de séries temporais que podem ser de interesse, como ARMA, ARIMA, SARIMA, GARCH, VARMA *etc.* O AR é um dos casos mais simples e é facilmente incorporado em outros modelos, como, por exemplo, o modelo autoregressivo de misturas.

2.3.6 Modelo Oculto de Markov

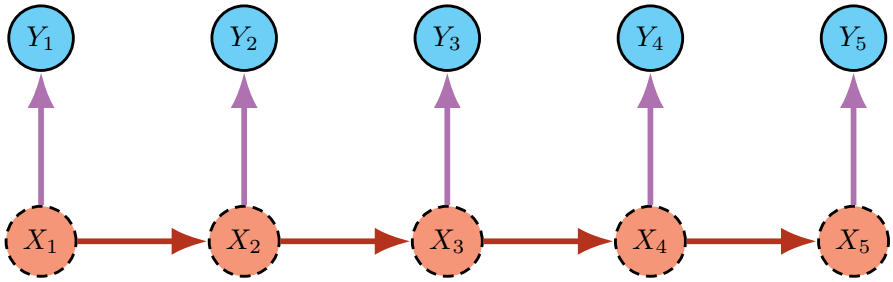
Um modelo oculto de Markov (MOM) $\{Y_t\}$ consiste de duas partes: **(a)** uma cadeia de Markov não-observável $\{X_t\}$, dita processo de parâmetro; e **(b)** uma sequência de variáveis aleatórias $\{Y_t\}_{t=1}^{\infty}$, dita processo de estado, onde a distribuição de Y_t depende apenas de X_t. Ou seja,

$$P(X_t | X_{[1:t-1]}) = P(X_t | X_{t-1}), \tag{2.41}$$

$$P(Y_t | Y_{[1:t-1]}, X_{[1:t]}) = P(Y_t | X_t). \tag{2.42}$$

Por essa propriedade, também é chamado de modelo de misturas dependentes.

Figura 10 – Representação gráfica do modelo oculto de Markov



Se o processo de parâmetros $\{X_t\}$ tem M estados, então a MOM $\{Y_t\}$ tem M estados. Para $i \in \{1, \ldots, M\}$, temos que

$$p_i(y) = P(Y_t = y | X_t = i), \tag{2.43}$$

onde p_i é a f.m.p de $Y_t|X_t = i$, se discreta, ou f.d.p de $Y_t|X_t = i$, se contínua. Dados os ajustes necessários, vamos assumir o caso discreto sem perda de generalidade.

Seja o vetor $u_i(t) = P(X_t = i)$, $\forall t \in \mathbb{N}$, então temos que

$$\begin{aligned} P(Y_t = y) &= \sum_{i=1}^{M} P(X_t = i) P(Y_t = y | X_t = i) \\ &= \sum_{i=1}^{M} u_i(t) p_i(y). \end{aligned} \tag{2.44}$$

Sejam $D(y)$ uma matriz diagonal com o i-ésimo elemento dado por $p_i(y)$ e $u(t)$ vetor-linha com i-ésimo elemento dado por $u_i(t)$. Se $\{X_t\}$ é homogênea, então podemos escrever $P(Y_t = y)$ matricialmente na forma

$$P(Y_t = y) = u(1) T^{t-1} D(y) \mathbf{1}^T. \tag{2.45}$$

Em qualquer modelo probabilístico gráfico direcionado $G = (V, E)$, a distribuição conjunta dos vértices $V = \{V_1, \ldots, V_n\}$ é dado por

$$P(V_1, \ldots, V_n) = \prod_{i=1}^{n} P\left(V_i | \Pi(V_i)\right), \tag{2.46}$$

onde $\Pi(V_i) = \{V_j \in V : (V_j, V_i) \in E\}$ é o conjunto de pais de v_i. Portanto, para qualquer MOM, segue que

$$P(Y_t, Y_{t+k}, X_t, X_{t+k}) = P(X_t)P(Y_t|X_t)P(X_{t+k}|X_t)P(Y_{t+k}|X_{t+k}). \tag{2.47}$$

Dada essa fatoração, podemos também escrever

$$\begin{aligned} P(Y_t = v, Y_{t+k} = w) &= \sum_{i=1}^{M}\sum_{j=1}^{M} P(Y_t = v, Y_{t+k} = w, X_t = i, X_{t+k} = j) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} \underbrace{P(X_t = i)}_{u_i(t)} p_i(v) \underbrace{P(X_{t+k} = j|X_t = i)}_{t^k_{i,j}} p_j(w) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} u_i(t)p_i(v)t^k_{i,j}p_j(w). \end{aligned} \tag{2.48}$$

Matricialmente, $P(Y_t = v, Y_{t+k} = w) = u(t)D(v)T^kD(w)\mathbf{1}^T$.

A expectância da variável observável Y_t é dada por

$$E[Y_t] = \sum_{i=1}^{M} P(X_t = i)E[Y_t|X_t = i] = \sum_{i=1}^{M} u_i(t)E[Y_t|X_t = i]. \tag{2.49}$$

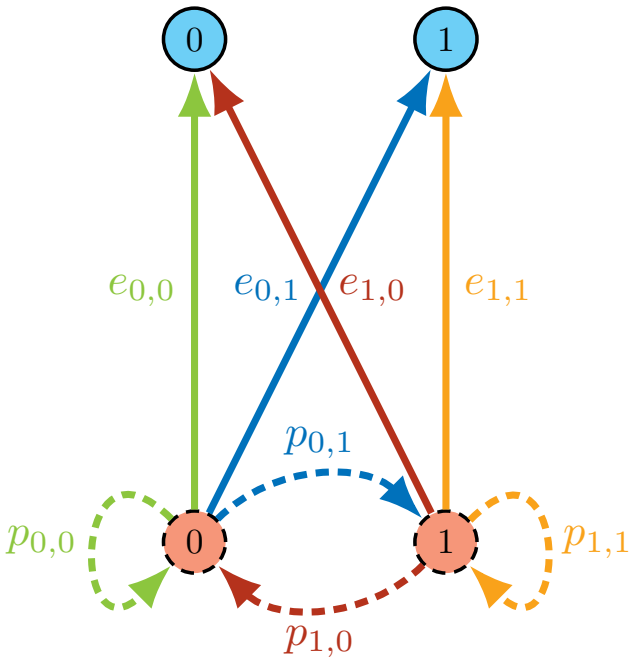
Suponha que observamos tanto uma sequência finita binária $\{y_t\}_{t=1}^N$ quanto uma sequência finita binária associada $\{x_t\}_{t=1}^N$, com $N = 24$. Dada uma nova observação y_{N+1}, é possível estimar um valor para X_{N+1} usando a informação sobre as dependências modeladas e os valores no tempo anterior. Para o processo de parâmetro, o procedimento é igual ao visto na seção 2.3.3. Já para o processo de estados, vamos usar novamente a contagem, mas dessa vez das ocorrências simultâneas de x_t e y_t. Assim, temos que:

$$\begin{aligned} \{x_t\} &= (1,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,0,0,1,1) \\ \{y_t\} &= (1,1,1,0,1,0,0,1,0,0,1,0,1,1,0,0,1,1,1,0,0,0,1,1), \end{aligned}$$

onde as cores indicam uma passagem dos valores binários de x para y como no exemplo anterior relativo à Figura 8. No entanto, agora a passagem dos valores de interesse é inter-linhas $\{x_t\}$ e $\{y_t\}$, e não só intra-linha $\{x_t\}$. A Figura 11 mostra os valores associados aos dois tipos de transição, onde os vértices vermelhos representam o processo de parâmetros e os vértices azuis representam o processo de estados.

Seja $n_{i,j}$ o número de vezes que ocorreu simultaneamente $x_t = i$ e $y_t = j$. Então,

Figura 11 – MOM binário com matrizes T (pontilhado) e E (sólido)



$$n_{0,0} = 7 \qquad n_{0,1} = 2$$
$$n_{1,0} = 4 \qquad n_{1,1} = 11.$$

Os valores de contagem $n_{i,j}$ e $f_{i,j}$ são as principais informações para o MOM. Com base nestes valores podemos achar os parâmetros que especificam o modelo.

Seja uma sequência finita de observações $y_1, \ldots, y_N$. A máxima verossimilhança L_N da MOM é equivalente à probabilidade de observar essa sequência finita de observações, dado uma MOM de M estados com distribuição inicial δ, matriz de transição T e funções de probabilidade p_i, e é dada por

$$P(Y_{[1:N]} = y_{[1:N]}) = L_N = \delta D(y_1) T D(y_2) T \ldots D(y_{n-1}) T D(y_n) \mathbf{1}^T. \tag{2.50}$$

A estimação da máxima verossimilhança é feita pelo algoritmo EM. Quando a MOM é homogênea, o algoritmo especializado é o BaumWelch (RABINER, 1989).

Seja α_t, para $t \in \{1, \ldots, N\}$, o vetor-linha de *probabilidades de avanço* dado por

$$\begin{aligned}\alpha_t &= \left[\begin{array}{ccc} P(Y_{[1:t]} = y_{[1:t]}, X_t = 1) & \ldots & P(Y_{[1:t]} = y_{[1:t]}, X_t = M) \end{array} \right] \\ &= \delta D(y_1) \prod_{s=2}^{t} TD(y_s), \end{aligned} \tag{2.51}$$

onde δ é a distribuição inicial e o j-ésimo elemento de α_t, denotado $\alpha_t(j)$, é a probabilidade conjunta de $(Y_{[1:t]}, X_t)$ ser $(y_{[1:t]}, j)$. Segue imediatamente que, para $t \in \{1, \ldots, N-1\}$,

$$\alpha_{t+1} = \alpha_t TD(y_{t+1}) = \left(\sum_{i=1}^{M} \alpha_t(i) t_{i,j} \right) p_j(y_{t+1}). \tag{2.52}$$

Seja β_t, para $t \in \{1, \ldots, N\}$, o vetor-linha de *probabilidades de retrocesso* dado por

$$\beta_t = \left[\begin{array}{c} P(Y_{[t+1:N]} = y_{[t+1:N]} | X_t = 1) \\ \vdots \\ P(Y_{[t+1:N]} = y_{[t+1:N]} | X_t = M) \end{array} \right]^T = \left(\prod_{s=t+1}^{N} TD(y_s) \right) \mathbf{1}^T, \tag{2.53}$$

onde, por convenção, qualquer produto vazio é igual à matriz identidade, com $\beta_N = \mathbf{1}$, e o j-ésimo elemento de β_t, denotado $\beta_t(j)$, é a probabilidade condicional de $Y_{[t+1:N]}|X_t$ ser $y_{[t+1:N]}$ dado $x_t = j$. Segue que, para todo $t \in \{1, \ldots, N-1\}$,

$$\beta_t^T = TD(y_{t+1}) \beta_{t+1}^T. \tag{2.54}$$

Para todo $t \in \{1, \ldots, N\}$,

$$\alpha_t(j) \beta_t(j) = P(Y_{[1:N]} = y_{[1:N]}, X_t = j), \tag{2.55}$$

logo, segue que

$$\alpha_t \beta_t^T = P(Y_{[1:N]} = y_{[1:N]}) = L_N. \tag{2.56}$$

Note que $Y_{[1:t]} \perp Y_{[t+1:N]} | X_t$.

Para abreviar, tomemos $B_t = TD(y_t)$.

Como não estamos interessados em Y_t e sim em X_t, então usamos a regra de Bayes para inverter as probabilidades condicionais. Assim, para $t \in \{1, \ldots, N\}$, temos que

$$P(X_t = j | Y_{[1:N]} = y_{[1:N]}) = \frac{\alpha_t(j) \beta_t(j)}{L_N}, \tag{2.57}$$

dito decodificador local, e, para $t \in \{2, \ldots, N\}$, que

$$P(X_{t-1} = j, X_t = k | Y_{[1:N]} = y_{[1:N]}) = \frac{\alpha_{t-1}(j) t_{j,k} p_k(y_t) \beta_t(k)}{L_N}. \tag{2.58}$$

Este é o problema da inferência. Neste caso, o algoritmo EM é uma escolha natural dado que os estados da CM não são observados. Entretanto, não é esse o nosso foco, queremos ajustar um MOM a partir dos dados obtidos, que incluem ambos Y_t e X_t.

O problema da decodificação envolve determinar os estados X_t da CM que são mais prováveis no modelo ajustado dada uma sequência finita observada $\{y_t\}_{t=1}^N$. Há duas formas de decodificação.

Decodificação local $P(X_t | Y_{[1:N]})$. Determinar qual estado x é o mais provável de X_t, no tempo t, dado $\{y_t\}_{t=1}^N$ observado. Obtemos o resultado maximizando a expressão $\alpha_t(i)\beta_t(i)$ com respeito à variável i.

Decodificação global $P(X_{k:N} | Y_{[1:N]})$. Determinar qual sequência finita de $\{x_t\}_{t=k}^N$, $N > k > 1$ mais provável de $\{X_t\}$, dado $\{y_t\}_{t=1}^N$ observado. Obtemos o resultado usando o algoritmo de Viterbi.

Não há garantia de que uma sequência de decodificações locais leve à decodificação global.

Na decodificação global, queremos achar a sequência finita $\{x_t\}_{t=1}^N$ que maximiza a probabilidade condicional $P(X_{[1:N]} | Y_{[1:N]} = y_{[1:N]})$, ou, equivalentemente, que maximiza a probabilidade conjunta dada por

$$P(X_{[1:N]}, Y_{[1:N]}) = \delta_{x_1} \prod_{t=2}^{N} t_{x_{t-1}, x_t} \prod_{t=1}^{N} p_{x_t}(y_t). \tag{2.59}$$

Por força bruta, seriam necessárias M^N avaliações de funções que estabelecemos anteriormente. O algoritmo de Viterbi oferece uma abordagem de programação dinâmica que reduz drasticamente o custo computacional. Sejam

$$\xi_{1,i} = P(X_1 = i, Y_1 = y_1) = \delta_i p_i(y_1), \text{ e,} \tag{2.60}$$

$$\xi_{t,i} = \max_{x_{[1:t-1]}} \left\{ P(X_{[1:t-1]} = x_{[1:t-1]}, X_t = i, Y_{[1:t]} = y_{[1:t]}) \right\}. \tag{2.61}$$

Pode-se mostrar que $\xi_{t,j}$ satisfaz, para $t \in \{2, \ldots, N\}$ e $j \in \{1, \ldots, M\}$, a propriedade recursiva

$$\xi_{t,j} = \left(\max_i \{\xi_{t-1,i} t_{i,j}\} \right) p_j(y_t). \tag{2.62}$$

Isso fornece um modo eficiente de computar a matriz $N \times M$ dos valores de $\xi_{t,j}$ em tempo linear com respeito a T. A maximização da sequência de estados pode ser determinada recursivamente a partir de

$$i_T = \underset{i \in 1,\ldots,M}{\text{argmax}}\{\xi_{T,i}\}, \text{ e,} \tag{2.63}$$

$$i_t = \underset{i \in 1,\ldots,M}{\text{argmax}}\{\xi_{t,i} t_{i,i_{t+1}}\}. \tag{2.64}$$

No MOM, as observações não são o objeto de interesse e servem apenas para fornecer informação sobre os estados não-observados. Uma aplicação do MOM envolve pegar uma amostra com observações $\{x_t\}_{t=1}^N$ e $\{y_t\}_{t=1}^N$ e ajustar o modelo pela estimação dos parâmetros do MOM usando a maximização da log-verossimilhaça dos dados completos. O modelo ajustado pode ser usado para determinar os estados não-observados $\{X_t\}$ mais prováveis para uma nova sequência finita de $\{y_t\}_{t=N+1}^K$, seja por decodificação local ou global. Além disso, o que foi dito sobre CM de ordens $l > 1$ também se aplica aqui. A extensão do modelo para ordens maiores não exige nenhum novo arcabouço teórico.

2.3.7 Modelo Autoregressivo Oculto de Markov

O MOM(1)-AR(1) é composto pelo processo de parâmetro $\{X_t\}$, não-observável, com $|Im(X_t)| = N$, e o processo de estados $\{Y_t\}$, observável, com $|Im(Y_t)| = M$, tal que $Y_t \perp X_t$, $Y_{t-1} \sim p_{Y_t|X_t,Y_{t-1}}(y_t)$, para $t \in \{2,\ldots,T\}$. Assim, temos as seguintes relações de independência condicional

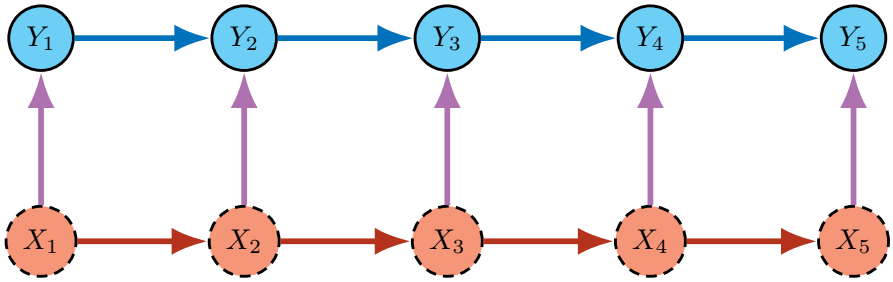
$$P(X_t|X_{[1:t-1]}, Y_{[1:t]}) = P(X_t|X_{t-1}), \tag{2.65}$$

$$P(Y_t|Y_{[1:t-1]}, X_{[1:t]}) = P(Y_t|Y_{t-1}, X_t). \tag{2.66}$$

Podemos especificar o modelo por meio da matriz estocástica de transição $T = (t_{i,j})$, onde $t_{i,j} = P(X_t = j|X_{t-1} = i)$, e o conjunto das matrizes de emissão $E = \{E^{(1)},\ldots,E^{(N)}\}$, onde $E^{(k)} = (e_{i,j}^{(k)})$, matriz de emissão entre estados observáveis condicionada pelo estado não-observável $X_t = k$, $e_{i,j}^{(k)} = P(Y_t = j|X_t = k, Y_{t-1} = i)$, para $k \in \{1,\ldots,N\}$ e $i,j \in \{1,\ldots,M\}$.

A função de verossimilhança para o modelo é dado por

Figura 12 – Representação gráfica do modelo autoregressivo oculto de Markov



$$
\begin{aligned}
&L(T, E|x_{[1:T]}, y_{[1:T]}) = P(Y_1 = y_1, X_1 = x_1|T, E) \\
&\times \prod_{t=2}^{T} P(X_t = x_t|X_{t-1} = x_{t-1}, T) P(Y_t = y_t|X_t = x_t, Y_{t-1} = y_{t-1}, E) \\
&= \frac{1}{NM} \left(\prod_{k=1}^{N} \prod_{l=1}^{N} t_{k,l}^{m_{k,l}} \right) \left(\prod_{k=1}^{N} \prod_{i=1}^{M} \prod_{j=1}^{M} (e_{i,j}^{(k)})^{n_{i,j}^{(k)}} \right) \\
&= \frac{1}{NM} \prod_{i=1}^{T} t_{x_{i-1}, x_i} e_{y_{i-1}, y_i}^{(x_i)},
\end{aligned} \tag{2.67}
$$

onde, para $\mathbb{I}$ função indicadora, $m_{k,l} = \sum_{t=1}^{T-1} \mathbb{I}(X_t = k, X_{t+1} = l)$ e $n_{i,j}^{(k)} = \sum_{i=2}^{T} \mathbb{I}(Y_{t-1} = i, Y_t = j, X_t = k)$.

Para o MOM(1)-AR(1) ainda é razoável usar o Baum-Welch adaptado. No entanto, para casos de ordens maiores ou para casos em que queremos incorporar informações *a priori*, é possível usar uma abordagem Bayesiana no EM usando a conjugação Dirichlet-Categórica.

O MOMAR permite dependência serial entre as variáveis observáveis Y_t e também é chamado de *Markov-switching model*. Além disso, possui variações permitindo dependências adicionais entre o processo de parâmetros ou o processo de estados. De modo geral, é um modelo versátil que permite incorporar diferentes fontes de informação nos seus processos, respeitadas algumas limitações com respeito à estrutura do grafo direcionado subjacente.

3 EXPERIMENTOS COMPUTACIONAIS

3.1 Base de Dados

O PDB é uma de base de dados digital, aberta e de domínio público, que hospeda entradas com informações estruturais tri-dimensionais de estruturas biológicas submetidas por pesquisadores e revisadas por pares de diversas partes do mundo. O acesso ao PDB é fornecido por institutos membros do wwPDB nos EUA, UE, Reino Unido e Japão, entre os quais está o RCSB PDB (EUA) (BERMAN; HENRICK; NAKAMURA, 2003). Todos os dados coletados para a presente pesquisa e procedimentos descritos dizem respeito ao RCSB PDB[1].

A base de dados é atualizada semanalmente em um processo de curadoria que envolve não só a adição de novas entradas, como também a modificação de entradas já existentes, obsolescência de entradas que foram melhor investigadas, e atualização de meta-dados associados. Devido a esta característica, optamos por estabelecer 2024-03-05T00:00:00-03:00 como a data de referência para o estado do PDB durante a pesquisa. Neste ponto no tempo, o PDB consistia de 223166 entradas relativas a 6 grupos de estruturas biológicas, ocupando um espaço de armazenamento de aproximadamente 24 GB no formato MMTF. Os grupos de entradas são:

- Apenas proteínas;
- Apenas ácidos nucleicos;
- Apenas oligossacarídeos;
- Proteínas com ácidos nucleicos;
- Proteínas com oligossacarídeos;
- Outros.

Acima de 80% das entradas pertencem ao grupo de apenas proteínas. No segundo lugar em tamanho, o grupo de proteínas com oligossacarídeos compõe aproximadamente 6% das entradas, dando dimensão do quão concentrado são os dados em torno de proteínas. Veremos na seção 3.1.1 que o nome dado destes grupos pode ser enganoso se não entendido no contexto da unidade biológica. De fato, dentro de entradas do grupo de apenas proteínas

[1] Embora todos os membros do wwPDB tenham os mesmos dados e todos os diferentes acessos resultem na mesma informação bruta, é importante frisar que cada membro oferece um conjunto de ferramentas e formatos distintos para a obtenção e análise dos dados contidos na base de dados.

podem existir entidades que não são proteínas, como ligantes. A restrição no nome do grupo se refere somente a outros tipos de polímeros.

Cada entrada também está relacionada a um grupo relativo ao método experimental que foi empregado para a obtenção da informação. Cada um destes grupos contém 1 ou mais métodos específicos distintos, como, por exemplo, SSNMR dentro do grupo NMR. Os grupos dos métodos são:

- Cristalografia de raios X;
- NMR;
- Crio-microscopia eletrônica;
- Neutron;
- Híbrido/Múltiplos;
- Outros.

Acima de 80% das entradas pertencem ao grupo de cristalografia de raios X. No segundo e terceiro lugares em tamanho, os grupo de crio-microscopia e NMR compõem, respectivamente, cerca de 9% e 6% das entradas. É notável o crescimento das entradas de crio-microscopia, superando recentemente o NMR como segundo grupo de métodos em tamanho. O contexto deste crescimento é evidenciado pelo prêmio Nobel em Química de 2017, dado aos criadores da crio-microscopia. Apesar desta tendência, a conformação da proteína com os dados obtidos pelo NMR é uma das aplicações mais célebres da DG e será o enfoque das próximas secções.

3.1.1 Estrutura dos Dados

O dado bruto de uma entrada do PDB tem como formato padrão o mmCIF. Dada a necessidade de armazenar e transmitir eficientemente esses dados, usualmente são comprimidos em outros formatos, como o MMTF. Diferentes formatos vão exigir *parsers* distintos, mas sempre é possível traduzir o conteúdo destes formatos para o mmCIF e vice-versa. Para fins de exposição, vamos tratar somente da estrutura do dado bruto.

O arquivo mmCIF de uma entrada é organizado em blocos de dados que particionam o arquivo, sendo cada partição delimitada por um comando `data_`, seguida ou não de uma cadeia de caracteres que nomeia o bloco. Os dados daquele bloco são dispostos nas linhas seguintes ao comando `data_`. Dados de diferentes categorias são separados visualmente por linhas contendo somente o caractere de cerquilha (`#`), que designa também um comentário. Para valores simples, cada linha de dado é composta por um par chave-valor. A chave é um nome composto pelos seguintes elementos em ordem e

sem espaços em branco: **(i)** início com o caractere de subtraço (_); **(ii)** nome da categoria em *snake_case*; **(iii)** caractere de ponto (.); e **(iv)** nome do atributo. Por sua vez, o valor encontra-se depois da chave e de um ou mais espaços em branco ou entre caracteres de ponto-e-vírgula (;) na linha seguinte. O Código-fonte 1 apresenta as primeiras linhas do arquivo mmCIF da entrada 4HHB.

Código-fonte 1 – Exemplo do cabeçalho de 4HHB.

```
data_4HHB
#
_entry.id          4HHB
#
_audit_conform.dict_name          mmcif_pdbx.dic
_audit_conform.dict_version       5.367
_audit_conform.dict_location      http://mmcif.pdb.org/dictionaries/ascii/mmcif_pdbx.dic
#
```

Os valores admitidos nos dados são uma sequência finita de caracteres ASCII, não sendo admitido espaço em branco a menos que todo o valor esteja envolto por aspas simples. Algumas chaves apresentam restrições adicionais e permitem somente combinações válidas de caracteres ASCII que possam ser interpretadas nos tipos de dados primários usuais (inteiro, ponto flutuante, texto). Além disso, há caracteres com funções especiais, como o caractere de interrogação (?) para dado faltante, ou o caractere de ponto para ausência de valor apropriado.

De modo a distinguir visualmente a informação sobre atributos e categorias no corpo do texto, convencionamos que os nomes dos atributos são monoespaçados e os nomes das categorias são monoespaçados e capitalizados. Assim, o Código-fonte 1 contém 2 categorias, denominadas `ENTRY` e `AUDIT_CONFORM`. Além disso, a menção de um valor não-numérico será envolto por aspas simples. Caso o valor já tenha aspas simples, então será envolto somente por aspas duplas. Portanto, o valor de `_entry.id` é '4HHB'.

Para valores compostos, o conjunto dos dados é composto por uma tabela. A construção da tabela é feita por uma linha com o comando `_loop`, seguida de um número n de chaves, uma em cada linha, e então m linhas de n valores simples separados por espaços em branco. O Código-fonte 2 apresenta a primeira linha da categoria `ENTITY` da entrada 4HHB. Note que 'A,C' é um valor simples associado à chave `_entity_poly.pdbx_strand_id`.

O formato especificado somado aos caracteres especiais de dado faltante ou não aplicável faz com que a organização dos dados seja versátil, permitindo criar vetores de tamanho arbitrário e armazenar informações sem precisar especificar um limite fixo. Isto contrasta com uma tabela usual que, para armazenar uma informação de que há 3 valores, precisa ou ter 3 colunas do valor, ou armazenar os 3 valores como se fosse um valor simples, violando a atomicidade. Além disso, este formato permite que se estruture uma hierarquia por meio de redundância. Sob algumas condições, duas categorias com

Código-fonte 2 – Exemplo dos dados na categoria `ENTITY` de 4HHB.

```
loop_
_entity_poly.entity_id
_entity_poly.type
_entity_poly.nstd_linkage
_entity_poly.nstd_monomer
_entity_poly.pdbx_seq_one_letter_code
_entity_poly.pdbx_seq_one_letter_code_can
_entity_poly.pdbx_strand_id
_entity_poly.pdbx_target_identifier
1 'polypeptide (L)' no no
;VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD
LHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR
;
;VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD
LHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR
;
A,C ?
```

chaves distintas e garantidamente de mesmo valor podem se comunicar, de modo análogo ao como funciona a *Foreign Key* em bases de dados relacionais.

A vasta comunicação entre categorias (e a redundância decorrente) viabiliza que dois tipos de hierarquia organizem os dados de uma entrada: a estrutural, estritamente relacionada com a organização biológica; e a operacional, relativa ao formato do dado e como ele é computacionalmente armazenado e acessado. A hierarquia estrutural facilita a manipulação semântica dos dados, permitindo que se procure, identifique e analise partes de interesse do pesquisador na estrutura biológica contida na entrada. Por outro lado, a estrutura operacional permite a manipulação programática dos dados, atribuindo identificadores e relações que viabilizam o uso de uma álgebra relacional.

Assim, uma entrada do PDB pode ser acessada usando uma ou outra hierarquia, de acordo com a conveniência. No entanto, a compreensão de ambas é fundamental para evitar erros no uso e interpretação dos dados.

3.1.1.1 Hierarquia Estrutural

Como abordado na seção 1.2, proteínas são compostas por cadeias lineares de aminoácidos. Além disso, proteínas podem se associar entre si ou com ligantes, solventes, ácidos nucleicos e outras moléculas em uma estrutura biológica (ROSE et al., 2021). Dado que cada entrada no PDB contém uma estrutura biológica, então é necessário organizar esses diferentes elementos, seus níveis e suas relações de modo significativo. Assim, cada entrada no PDB é composta por uma hierarquia de 4 níveis da estrutura biomolecular:

- **Entrada**. A entrada é uma estrutura biológica depositada no PDB. Toda entrada contém pelo menos uma entidade polimérica.
- **Entidade**. A entidade é uma molécula quimicamente única que pode ser um polímero, como proteínas ou ácidos nucleicos, ou não, como ligantes. Toda entidade pertence

ao menos a uma montagem.

- **Montagem**. A montagem é um agrupamento biologicamente relevante de uma ou mais entidades associadas entre si para formar um complexo estável ou exercer uma função. Toda montagem contém pelo menos uma instância de uma entidade polimérica.
- **Instância**. A instância é uma ocorrência particular de uma entidade. Toda instância pertence a uma única entidade.

Por exemplo, a entrada 1VDF[2] contém uma estrutura homo-oligomérica, um complexo composto por múltiplas instâncias de uma mesma entidade polimérica. Observa-se, neste caso, que a estrutura possui uma única montagem formada pela associação de 5 glicoproteínas quimicamente idênticas e nomeadas em ordem alfabética de 'A' até 'E'.

Uma entidade pode ser desde um único íon até uma macromolécula que tem seus próprio elementos constitutivos hierarquicamente organizados. Temos que cada entidade é composta por uma hierarquia de 3 níveis da sua estrutura atômica:

- **Entidade**. A entidade é uma molécula pertencente a uma entrada. Toda entidade contém pelo menos um resíduo.
- **Resíduo**. O resíduo é um bloco discretizável e ordenado da entidade em uma posição t, dita **número do resíduo**. Em macromoléculas do tipo proteína (ou ácido nucleico), cada resíduo é um aminoácido (ou nucleotídeo) incorporado na t-ésima posição da cadeia principal. Todo resíduo contém pelo menos um átomo.
- **Átomo**. O átomo é a menor unidade discreta que compõe um resíduo. Os átomos seguem um padrão de nomenclatura descrito na seção 1.2.2.

Portanto, a estrutura atômica está contida dentro da estrutura biomolecular, criando uma subestrutura no nível da entidade. Essa disposição permite especificar com clareza cada elemento de uma estrutura biológica. Assim, podemos buscar semanticamente no PDB a coordenada cartesiana do átomo C_α de um resíduo A, com número de resíduo 4, da instância 'B' da única entidade na entrada 1VDF, por exemplo.

3.1.1.2 Hierarquia Operacional

A estrutura do dado de uma entrada segue uma hierarquia operacional com diferentes níveis definido por uma DDL. A DDL define alguns elementos no dado bruto como identificadores (ou chaves) que podem ser usadas para agrupar ou descrever outra

[2] URL: <https://www.rcsb.org/3d-view/1VDF> (Acesso em 03/05/2024).

parte do mesmo dado, combinando-as em categorias. Portanto, uma categoria funciona como uma tabela relacional, onde cada linha é uma observação única dos dados agrupados com relação aos identificadores desta categoria. Como os identificadores de uma categoria podem estar em outras categorias, então formam-se relações entre as tabelas que assumem um caráter hierárquico do tipo pai-filho. No entanto, é importante frisar que o formato mmCIF tradicional não garante a validade dos dados, então um arquivo pode violar a especificação e armazenar dados em que uma categoria tenha mais de uma linha com o mesmo identificador único[3].

Para o funcionamento desta hierarquia, primeiro é necessário definir os identificadores, de modo a permitir o acesso programático a cada elemento de determinado nível na estrutura biomolecular. Ou seja, é necessário garantir que todo elemento da estrutura biológica seja univocamente determinado a partir de identificadores. Além disso, uma entrada pode conter um ou mais experimentos que mensuraram as posições e conformações de uma estrutura biológica, então é preciso identificar cada experimento para não juntar dados de diferentes conjuntos de mensurações. O PDB define cada experimento como um modelo, incorporando-o como o primeiro nível abaixo da entrada.

A DDL do PDB conta com 648 categorias, cada uma com ao menos um identificador[4]. Essas categorias podem conter desde os dados da estrutura biológica até meta-dados, como, por exemplo, o nome do autor da entrada, o DOI da publicação associada e o método experimental empregado. Dada a quantidade de categorias, vamos nos restringir a descrever somente as que são imediatamente necessárias para a pesquisa. Assim, os principais identificadores e suas respectivas categorias são:

1. **ID PDB**. Uma entrada é identificada unicamente pelo ID PDB, um código alfanumérico de 4 caracteres dado por `_entry.id`, que define unicamente a categoria `ENTRY`.

2. **ID Modelo**. Um modelo é identificado unicamente pelo ID Modelo, um código numérico sequencial dado por `_atom_site.pdbx_PDB_model_num`, que não define uma categoria.

3. **ID Montagem**. Uma montagem é identificada unicamente pelo ID Montagem, um código numérico sequencial dado por `_pdbx_struct_assembly.id`, que define unicamente a categoria `PDBX_STRUCT_ASSEMBLY`.

[3] No entanto, o novo formato PDBx/mmCIF tem validação e testes de auto-consistência e está em uso, de modo opcional, desde 2014. A previsão é que todas as entradas submetidas empreguem necessariamente este novo formato a partir da adoção do novo padrão de identificador de entrada, o que está sem data certa para ocorrer até o momento da elaboração deste texto.

[4] URL: <https://mmcif.wwpdb.org/dictionaries/mmcif_pdbx_v50.dic/Categories/index.html> (Acesso em 03/05/2024).

4. **ID Entidade**. Uma entidade é identificada unicamente pelo ID Entidade, um código alfanumérico de 1 ou mais caracteres dado por `_entity.id`, que define unicamente a categoria `ENTITY`.

5. **ID Químico**. Um composto químico é identificado unicamente pelo ID Químico, um código alfanumérico de 3 (se polímero) ou 5 (caso contrário) caracteres dado por `_chem_comp.id`, que define unicamente a categoria `CHEM_COMP`.

6. **ID Sequência**. Um número do resíduo é identificado pelo ID Sequência, um código numérico sequencial dado por `_entity_poly_seq.num` que, junto com `_entity.id` e `_chem_comp.id`, define unicamente a categoria `ENTITY_POLY_SEQ`.

7. **ID Instância**. Uma instância de uma entidade é identificada unicamente pelo ID Instância, um código alfanumérico de 1 ou mais caracteres dado por `_struct_asym.id`, que define unicamente a categoria `STRUCT_ASYM`.

8. **ID Átomo**. Um átomo é identificado pelo ID Átomo, um código alfanumérico de 1 a 4 caracteres dado por `_chem_comp_atom.atom_id`, que, junto com `_chem_comp.id`, define unicamente a categoria `CHEM_COMP_ATOM`.

9. **ID Tipo-Átomo**. Um tipo de átomo é identificado pelo ID Tipo-Átomo, um código alfanumérico de 1 ou mais caracteres dado por `_atom_type.symbol`, que define unicamente a categoria `ATOM_TYPE`.

10. **ID Alternativo**. Um átomo que tenha múltiplas localizações em um mesmo experimento terá cada localização identificada por um ID Alternativo, um código alfabético de 0 ou mais caracteres dado por `_atom_site.label_alt_id`, que não define uma categoria[5].

11. **ID Local**. Um local (ou posição) de um átomo é unicamente identificada por um ID Local, um código alfanumérico de 1 ou mais caracteres dado por `_atom_site.id`, que define unicamente a categoria `ATOM_SITE`.

A hierarquia operacional tem na sua ponta, no último nível, os dados da categoria `ATOM_SITE`, onde há uma linha para cada átomo (a menos de múltiplas localizações) de cada modelo da entrada, como exemplificado no Código-fonte 3. Essa é a categoria de interesse central da pesquisa. A Figura 13 apresenta um diagrama em que é possível ver quais são os dados identificadores de `ATOM_SITE` e as categorias que eles definem. A seta no diagrama indica que a categoria do destino é categoria-pai da categoria de origem, ou seja, temos o conjunto de pais de `ATOM_SITE`. Além disso, pela definição da

[5] Quando ocorrer esta situação, a informação de `_atom_site.occupancy` associada à entrada dará a frequência relativa em que o átomo foi mensurado naquela posição. Ver Figura 13.

DDL, sabemos que o conjunto dos identificadores das categorias-pai equivale ao identificador na categoria de referência. Ou seja, `_atom_site.id` é equivalente ao conjunto composto por `_entity.id`, `_chem_comp.id`, `_entity_poly_seq.num`, `_struct_asym.id`, `_chem_comp_atom.atom_id`, `_atom_type.symbol` e `_atom_site.label_alt_id` (se existir), de modo que ambos podem ser chamados de ID Local.

Código-fonte 3 – Exemplo dos dados na categoria `ATOM_SITE` de 4HHB.

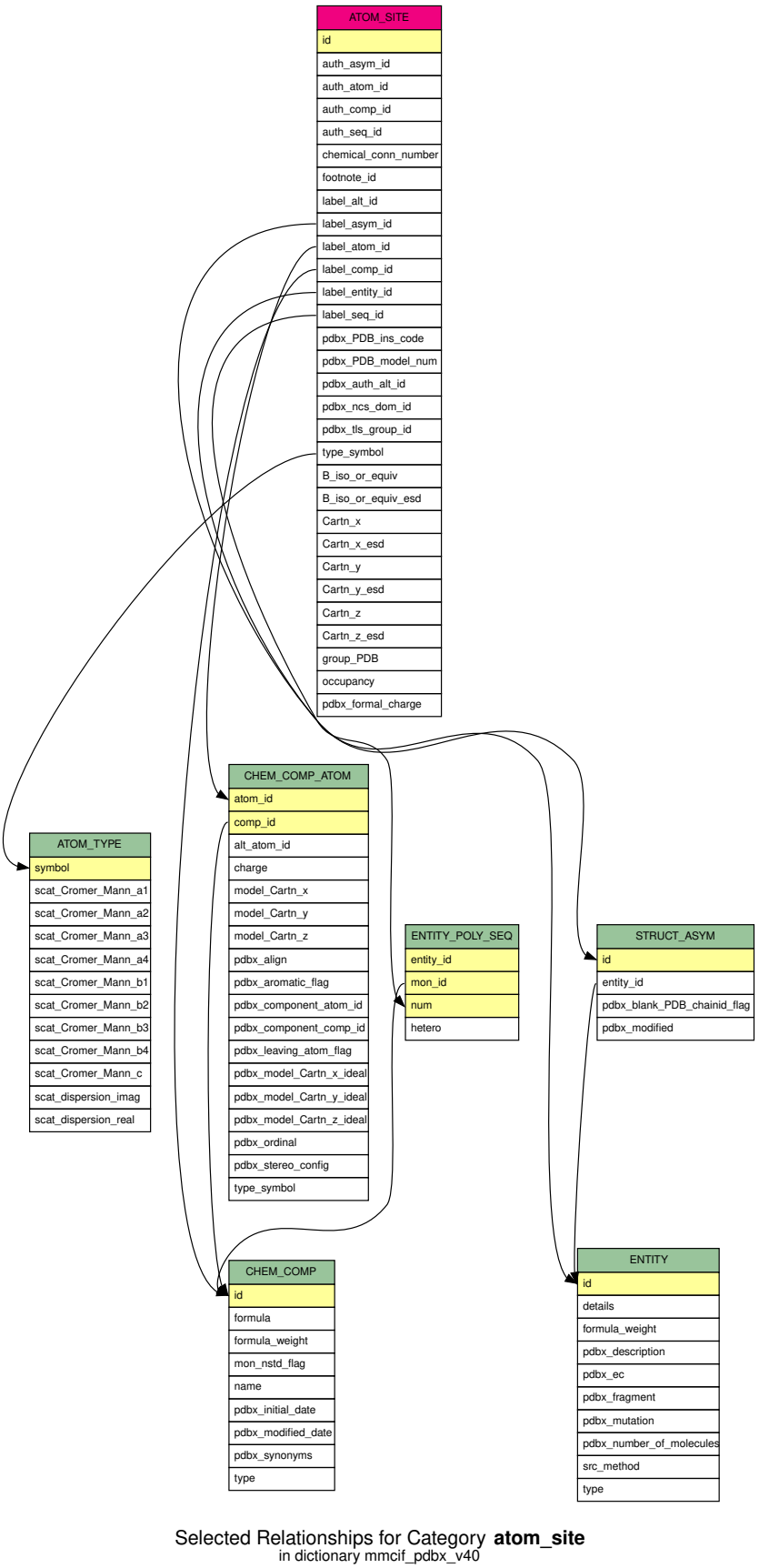
```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.pdbx_formal_charge
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM   1    N  N   . VAL A 1 1   ? 6.204   16.869  4.854   1.00 49.05 ? 1   VAL A N   1
ATOM   2    C  CA  . VAL A 1 1   ? 6.913   17.759  4.607   1.00 43.14 ? 1   VAL A CA  1
ATOM   3    C  C   . VAL A 1 1   ? 8.504   17.378  4.797   1.00 24.80 ? 1   VAL A C   1
ATOM   4    O  O   . VAL A 1 1   ? 8.805   17.011  5.943   1.00 37.68 ? 1   VAL A O   1
ATOM   5    C  CB  . VAL A 1 1   ? 6.369   19.044  5.810   1.00 72.12 ? 1   VAL A CB  1
ATOM   6    C  CG1 . VAL A 1 1   ? 7.009   20.127  5.418   1.00 61.79 ? 1   VAL A CG1 1
ATOM   7    C  CG2 . VAL A 1 1   ? 5.246   18.533  5.681   1.00 80.12 ? 1   VAL A CG2 1
```

A mesma lógica se aplica a qualquer dos níveis superiores e permite uma correspondência com a hierarquia estrutural. Assim, todo resíduo propriamente dito é identificado unicamente pelo conjunto composto pelo ID Químico, ID Sequência e ID Entidade. Em tese, seria possível identificar unicamente usando somente os dois últimos identificadores, no entanto, há situações de mutações pontuais, em que um resíduo extra é adicionado entre os t-ésimo e $(t + 1)$-ésimo resíduos de uma instância específica. Para preservar a sequência geral, nesta situação é adicionado um código alfabético de inserção `_atom_site.pdbx_PDB_ins_code` ao ID Sequência do t-ésimo resíduo e da mutação.

Importante notar que vários dos identificadores não são padronizados ao longo do PDB como, por exemplo, o ID Instância. Dado que não há um padrão para atribuir um valor para o ID Instância no PDB, temos que duas entradas distintas que tenham uma única instância de uma mesma entidade polimérica podem atribuir valores de ID Instância distintos para a instância da entidade em comum.

A situação é ainda mais complexa no caso de entidades não-poliméricas, onde cada instância é identificada pelo ID Instância da instância polimérica mais próxima. Porém,

Figura 13 – Diagrama da tabela `ATOM_SITE` e seus principais relacionamentos no PDB.



Fonte: wwPDB

isto faz com que o ID Instância deixe de ser parte do conjunto de identificadores únicos para a categoria `ATOM_SITE`. Neste casos, essas instâncias de entidades não-poliméricas são usualmente marcados como hétero-resíduos no valor `_chem_comp_atom.hetero`.

Além disso, a codificação da informação biológica também não é padronizada como, por exemplo, os resíduos. Apesar do amplo uso do código de 3-Letras, em múltiplas entradas é empregada a tabela de códon, em que um aminoácido é representado por uma sequência de 3 bases nitrogenadas do RNA mensageiro que o codifica. Múltiplas trincas podem levar a um mesmo aminoácido, como CUU, CUC, CUA e CUG que levam ao aminoácido L. Não obstante, alguns D-aminoácidos também estão presentes no PDB, com nomes distintos dos seus aminoácidos associados, como DSN para a D-Serina (ou D-Ser, D-S). Somando estes casos, o uso de 3 letras para hétero-resíduos como SEP (dexfosfoserina) e os casos da Tabela 1, tivemos mais de 120 tipos diferentes de elementos que seriam indicados como resíduos pertencentes a alguma cadeia de aminoácidos pelo ID Instância se não ocorresse uma inspeção e filtragem dos dados de cada entrada investigada.

3.1.2 Filtragem e Processamento

A complexidade da estrutura dos dados no PDB decorre, ao menos em parte, das adequações feitas para permitir interpretação semântica de acordo com a estrutura biomolecular. Como exposto anteriormente, isto se traduz na existência de múltiplos caminhos entre níveis da hierarquia operacional e em diferenças tanto nas categorias quanto nos identificadores que dependem do conteúdo do dado. Ou seja, um código que programaticamente gere respostas corretas para entradas em que só há entidade poliméricas não necessariamente gerará uma resposta correta para uma entrada em que exista uma entidade não-polimérica.

Para dar conta desta complexidade, vamos primeiro selecionar somente um subconjunto homogêneo das entradas disponíveis no PDB e então empregar este subconjunto para extrair as informações, estatísticas e dados derivados que desejamos para criar e testar um modelo para a busca probabilística. Segue que, para tal, deve-se primeiro estabelecer os critérios para selecionar este subconjunto – filtragem – e depois descrever as operações realizadas com o dado bruto até chegar no formato empregado na pesquisa – processamento.

Para fazer estas etapas, empregamos 2 *softwares*, respectivamente: *rcsbsearchapi*, e *BioStructures.jl*.

As entradas de interesse são as de apenas proteínas cujo método experimental é o NMR. Embora tenhamos uma cópia do estado do PDB no tempo de referência, ler cada uma das entradas localmente para identificar a informação seria custoso. Portanto, para obtê-las, primeiro usamos o *schema* da ferramenta de busca do PDB para identificar

onde estão as informações e quais os valores que correspondem ao filtro desejado[6]. Pela disposição dos dados na ferramenta, identificamos que checando somente dois conjuntos chave-valor podemos determinar se uma entrada atende ou não os nossos critérios:

- `properties.rcsb_entry_info.properties.experimental_method`, 'NMR';
- `properties.rcsb_entry_info.properties.selected_polymer_entity_types`, "Protein (only)".

Para obter programaticamente uma lista dos ID PDB que atendem ao filtro, empregamos o *rcsbsearchapi* como no Código-fonte 4. A lista destas entradas também pode ser obtida pela ferramenta de busca avançada do PDB. Ambas as alternativas usam o mesmo API de busca desenvolvido pela RCSB (ROSE et al., 2021). O resultado da busca está disponível *online*[7].

Código-fonte 4 – Busca com filtro no PDB (*Python*)

```
import rcsbsearchapi as sapi
q1 = sapi.rcsb_attributes.rcsb_entry_info \
        .selected_polymer_entity_types == "Protein (only)"
q2 = sapi.rcsb_attributes.rcsb_entry_info \
        .experimental_method == "NMR"
query = q1 & q2
ret = open("Resultado_Busca.txt", "w")
for entry_id in query("entry"):
        ret.write(f"{entry_id}\n")
ret.close()
```

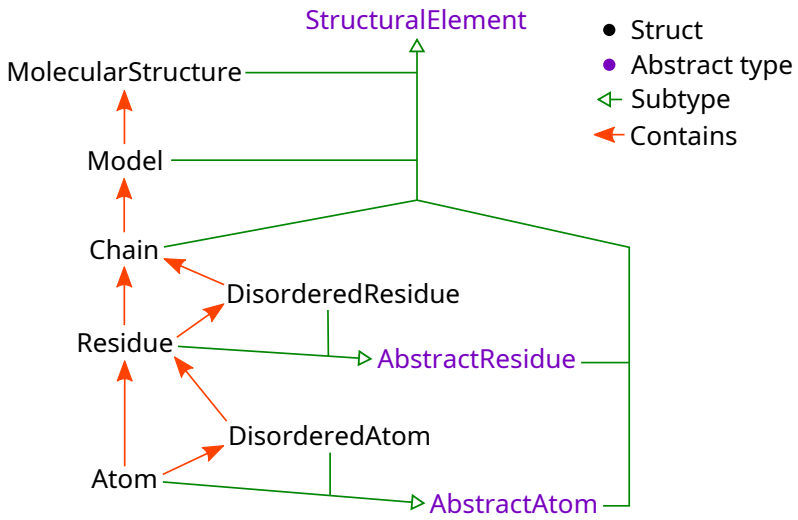
No total, 12351 entradas atenderam os nossos critérios e estão listadas de acordo com o seu ID PDB. O próximo passo consiste de acessar e processar os dados destas entradas usando o *BioStructures.jl*.

O desenho do *software* de processamento escolhido é similar ao do pacote Biopython e operações análogas em ambos devem produzir uma mesma resposta com base nos dados brutos. No entanto, é importante frisar que a abordagem em cada *software* é distinta. O *BioStructures.jl* emprega o sistema de tipagem da linguagem de programação *Julia* para emular parte da hierarquia dos dados, como apresentado na Figura 14. De modo geral, a hierarquia é parecida com visto anteriormente, sendo *Chain* o nível da instância, e *Residue* o nível do componente químico, usualmente a unidade monomérica (p.ex., aminoácidos e nucleotídeos). Em específico, os tipos *DisorderedAtom* e *DisorderedResidue* representam átomos e resíduos com locais alternativos, como, por exemplo, em mutações.

Assim, o *software* não está interessado somente em fazer a leitura das linhas do dado bruto, mas também em representar hierarquicamente as moléculas contida no dado, de modo que abordagem semântica funcione também para o processamento destes

[6] URL: <https://search.rcsb.org/rcsbsearch/v2/metadata/schema>.
[7] URL: <https://bit.ly/3WvwQee> (Acesso em 03/05/2024).

Figura 14 – Diagrama dos tipos concretos e abstratos no *BioStructures.jl*.



Fonte: BioStructures.jl

dados. Essa particularidade do *software* traz também algumas desvantagens. Em especial, o custo computacional inicial é maior do que o de um *parser* mais simples e impõe algumas limitações na capacidade de leitura de arquivos que violem a estrita especificação do formato (GREENER; SELVARAJ; WARD, 2020). Dependendo do tipo de informações que fuja do formato, o *software* pode ler a informação e ignorar os erros silenciosamente ao invés de retornar um erro explícito e interromper a operação.

Considerando essas limitações, aplicamos dois testes de validação para cada entrada, de modo a evitar trechos dos dados que tivessem algumas violações grosseiras à especificação ou às suposições básicas sobre o dado. De modo geral, estes testes operam no nível hierárquico-estrutural dos resíduos e permitem criar subsequências contíguas de resíduos com informação completa válida, denominados como **segmentos**. Assim, cada entrada é investigada quanto a viabilidade de qualquer de suas partes, de modo que uma entrada pode ter 0 ou mais segmentos.

O primeiro teste de validação é o de identificabilidade do resíduo. Definimos, para tal, que um resíduo conhecido é aquele que tem ID Químico distinto de '?', e que resíduo válido é aquele em que o ID Químico corresponde a algum dos 22 aminoácidos da Tabela 1 (excetuando o X) e que não é uma mutação. Operacionalmente, isto se traduz em existir a informação do resíduo, não ser um hétero-resíduo e não ser uma mutação.

O segundo teste de validação é o de satisfatibilidade das suposições da ordenação do PDGDM. Para determinar a ordem do PDGDM para os átomos na espinha-dorsal estendida temos as seguintes suposições: (i) as distâncias entre átomos conectados por uma ou duas ligações covalentes é conhecida; (ii) como a ligação peptídica conecta o

carbono C^i do grupo carboxila do i-ésimo resíduo ao nitrogênio N^{i+1} do grupo amino do $(i+1)$-ésimo resíduo, então os átomos no i-ésimo plano peptídico são $C^i_\alpha, C^i, N^{i+1}, C^{i+1}_\alpha$, então a distância entre estes átomos é também conhecida; (iii) o NMR pode fornecer as distâncias entre H^i, H^i_α e H^i_α, H^{i+1}. Definimos, para tal, que um resíduo ordenável é aquele em que todos os átomos descritos acima possuem ID Local. Operacionalmente, isto se traduz em existir a informação destes átomos em cada par de resíduos e não serem hétero-átomos.

Como ambos os testes têm a mesma estrutura básica de percorrer os resíduos sequencialmente e segmentá-los de acordo com seus respectivos critérios, então podemos juntá-los em um processo único. Assim, o processo consiste em, para $1 \leqslant i \leqslant t \leqslant n$, n comprimento da sequência de resíduos, partirmos da i-ésima posição da sequência de resíduos e verificarmos se o t-ésimo resíduo desta sequência é conhecido, válido e ordenável. Caso positivo, incrementamos o valor de t em 1, se possível, e repetimos o processo. Caso negativo, então definimos $[i, t-1]$, se $i \neq t$, como um segmento, incrementamos o valor de t em 1, se possível, atribuímos à variável i o valor de t e repetimos o processo. O processo termina quando não for possível incrementar o valor de t em qualquer um dos casos. Ao final, temos todos os segmentos de uma sequência de resíduos.

Isto posto, percorremos a lista dos ID PDB que passaram no filtro, acessando cada entrada associada nos dados locais e extraindo as informações de interesse na forma de segmentos pela validação. De cada segmento extraímos os seguintes dados:

- Átomo;
- Resíduo;
- Coordenadas em $\mathbb{R}^3$ do átomo.

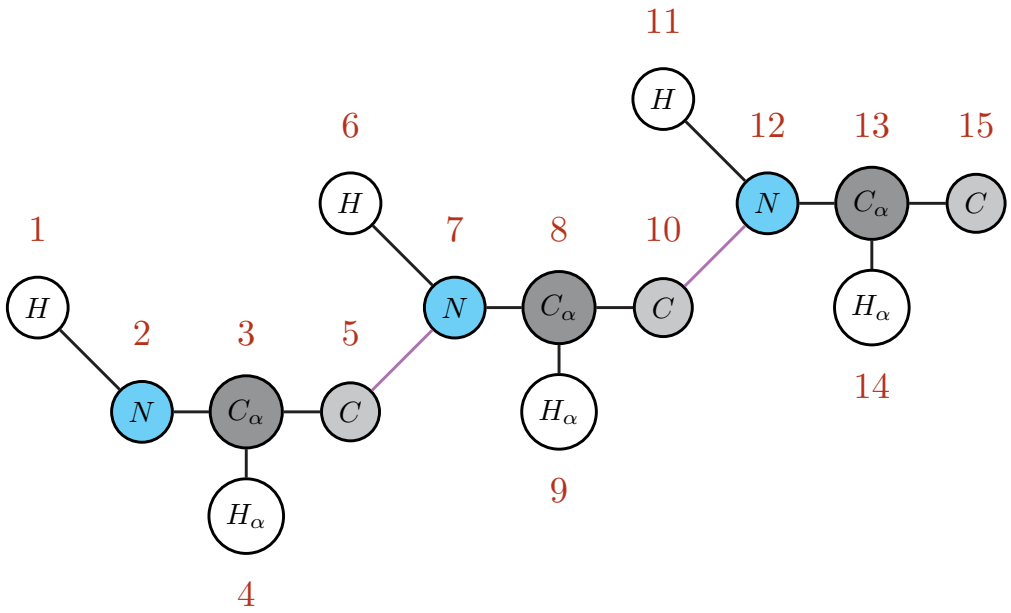
Para cada resíduo, somente interessam os átomos relacionados à espinha dorsal estendida como descrito nas suposições básicas acima. Baseado nisto, Marques et. al. determinam a seguinte ordenação dos átomos de resíduos sequenciais (MARQUES et al., 2024):

$$\sigma = \left(H^1, N^1, C^1_\alpha, H^1_\alpha, C^1, \ldots, H^i, N^i, C^i_\alpha, H^i_\alpha, C^i, \ldots, H^n, N^n, C^n_\alpha, H^n_\alpha, C^n\right). \tag{3.1}$$

A informação extraída então será ordenada de modo que os átomos sigam sempre esta ordem, como exposto na Figura 15. A única exceção é com respeito ao P e G, onde o H escolhido pode ser distinto dos demais por questões relativas à geometria destes dois resíduos, mas ainda assim permanece na mesma posição da ordenação. Como os segmentos foram construídos seguindo a sequência de resíduos, então os dados dentro

de cada segmento estão totalmente ordenados e sabemos que o i-ésimo resíduo conterá uma 5-upla de átomos dada por $(H^i, N^i, C^i_\alpha, H^i_\alpha, C^i)$.

Figura 15 – Ordenação dos átomos da espinha dorsal estendida (ligação peptídia em roxo).

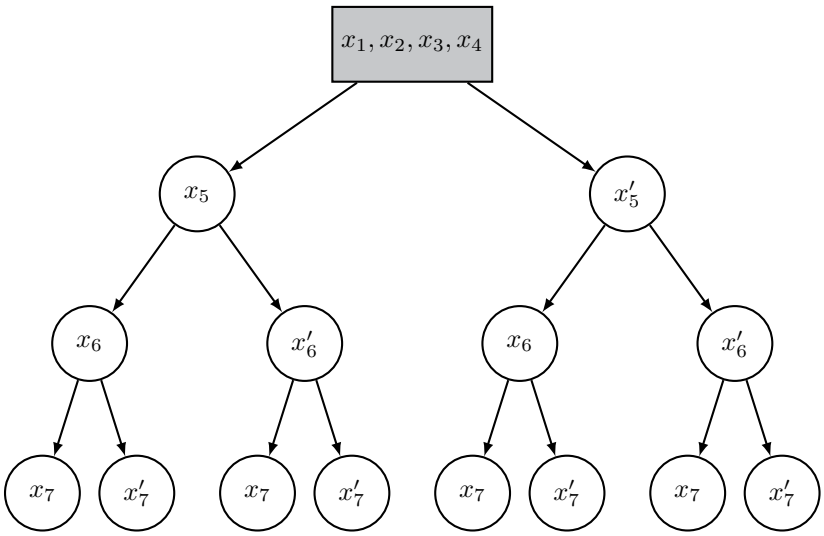


Esta ordenação é importante porque o espaço de busca de um PDGDM pode ser representado como uma árvore binária. A Figura 16 apresenta um pequeno exemplo desta representação com somente 7 átomos. A árvore binária incorpora a informação geométrica dado uma ordenação, pois cada ponto na posição x_i está associado a um *bit* b_i que indica a posição relativa com respeito ao plano formado pelos pontos anteriores de referência, assumindo 0 se abaixo do plano ou 1 se acima do plano. No entanto, cumpre definir o que queremos dizer com estas orientações acima e abaixo.

Para isto, seja x_k um ponto com pontos de referência associados x_i, x_j, x_l e sejam os vetores $x_k - x_i$, $x_j - x_i$ e $x_l - x_i$. Consideramos que a orientação de x_k com relação ao plano é determinada pelo sinal do coeficiente da componente ortogonal de $x_k - x_i$ dada pelo vetor normal $x_j - x_i \times x_l - x_i$. Se positivo, 1; se não-positivo, 0.

Os átomos de referência utilizados em cada caso são os 3 imediatamente anteriores na ordenação, com a exceção dos N^i, em que os antecessores, se existirem, são H^i, C^{i-1} e C^{i-1}_α. Portanto, as coordenadas cartesianas de cada segmento de tamanho n podem ser colocadas por um procedimento iterativo em bijeção com uma sequência finita binária $b = (b_1, \ldots, b_{5n})$, formando uma solução para aquela instância do PDGDM.

Figura 16 – Árvore binária do espaço de busca de um PDGDM.



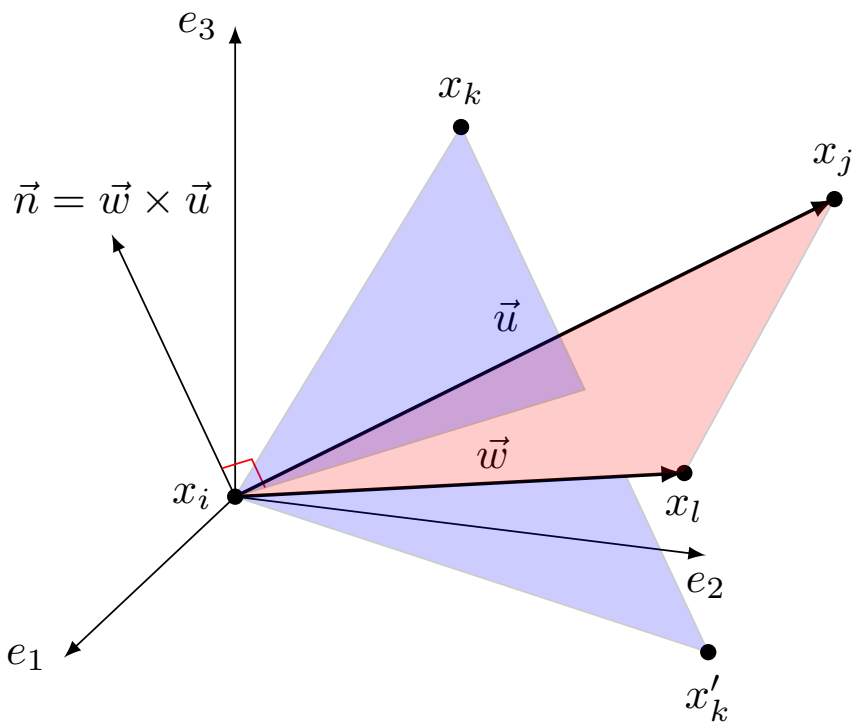
Como os 3 primeiros átomos da ordenação são facilmente fixados em $\mathbb{R}^3$ e sabemos que H, N, C e O estão no mesmo plano peptídico, então tomamos $b_1 = b_2 = b_3 = b_4 = 0$ sem perda de generalidade. Além disso, o quinto átomo da ordenação tem sempre duas posições viáveis possíveis que são simétricas. Por convenção, fixamos $b_5 = 0$ e sabemos que se temos uma solução x associada a uma sequência finita b, então também temos uma solução x' associada a uma sequência finita b' onde realizamos um *flip* em b_5 e em diante.

Procedemos para os demais átomos de modo análogo, identificando a posição com relação ao plano formado pelos antecessores de referência e atribuindo o valor do binário de acordo com a sua orientação. A Figura 17 mostra um exemplo em que vamos fixar o k-ésimo ponto e temos um plano composto pelos pontos de referência já fixados x_i, x_j, x_l, de modo que se o k-ésimo ponto estiver em x_k, então $b_k = 1$, e se estiver em x'_k, então $b_k = 0$.

Assim, temos que qualquer sequência finita b associada a uma solução é da forma $b = (0, 0, 0, 0, 0, b_6, \ldots, b_{5n})$. Adotamos uma representação reduzida $s = (b_6, \ldots, b_{5n}) \in \{0, 1\}^{n-5}$, removendo os *bits* fixados. A partir de b podemos também obter a subsequência finita associada ao i-ésimo resíduo, bastando selecionar os elementos $k + (i - 1) * 5$ de s, para $k \in \{1, \ldots, 5\}$.

Dado o procedimento acima, produzimos o dado derivado de interesse central para a pesquisa: a sequência finita de binários associada a uma sequência finita de aminoácidos e seus átomos ordenados. Como a unidade principal de análise é o segmento, então vamos tratar cada segmento como uma estrutura em separado. Portanto, temos como identificar dentro de cada segmento qual é o resíduo anterior e qual o binário anterior de cada um dos átomos, convencionando que os antecessores do primeiro resíduo e do

Figura 17 – Plano dos vértices x_i, x_j, x_l e a posição relativa que x_k pode assumir.



primeiro binário recebem valor `nothing`. Assim, temos para cada segmento as seguintes informações:

- Átomo;
- Resíduo;
- Resíduo anterior;
- Binário;
- Binário anterior.

Por fim, dado que temos para cada segmento as coordenadas em $\mathbb{R}^3$ dos seus átomos e sua ordenação, construímos a matriz com o conjunto minimal das distâncias que garantem as suposições da ordenação do PDGDM como dado de suporte. Cada uma destas matrizes pode ser usada como entrada para o BP de modo que necessariamente exista ao menos uma conformação válida. Esse dado de suporte será usado na seção 3.4 para os experimentos computacionais.

3.2 Análise Preliminar

A primeira investigação feita foi uma análise exploratória em uma parte dos dados filtrados e validados. O objetivo desta análise é determinar se a informação da sequência finita dos aminoácidos de uma proteína potencialmente traz algum ganho no processo de determinar uma solução do PDGDM ao ser incorporada no FBS. Para isso, as informações principais a serem extraídas dos dados do PDB são as sequências finitas de aminoácidos que compõem as proteínas e as sequências finitas binárias associadas à anterior.

Sejam X_t uma variável aleatória binária associada ao t-ésimo elemento da sequência finita binária, e Y_t uma variável aleatória discreta associada ao elemento correspondente da sequência finita de resíduos. Estabelecemos como hipótese nula da existência de ganho potencial na incorporação da informação o seguinte: a distância da f.m.p empírica de X_t para a f.m.p empírica de $X_t|Y_t$ é, em média, aproximadamente 0.

$$\begin{cases} H_0: & \|p(x) - p(x|y)\|_2 = 0 \\ H_1: & \|p(x) - p(x|y)\|_2 \neq 0 \end{cases} \tag{3.2}$$

Isso é equivalente a um teste de independência.

Para manter a comparabilidade com o trabalho de Marques et al., utilizamos nesta seção somente a primeira cadeia do primeiro modelo de cada uma das entradas (MARQUES et al., 2024). Também incluímos, sem qualquer prejuízo, os aminoácidos G e P na nossa análise exploratória. Como agregamos os dados por tipo de aminoácido no argumento principal desta seção, basta ignorar G e P que todas as demais relações serão proporcionais ao que obteríamos se não os tivéssemos incluído.

Cada subsequência finita extraída de s é relacionada aos 5 *bits* imediatamente precedentes. Como somente a primeira subsequência finita é garantida de ter como *bit* precedente um 0, então pode ser necessário inverter todos os *bits* da subsequência finita analisada para manter o padrão convencionado. Portanto, em todos os casos, checamos se o valor binário do átomo imediatamente anterior ao primeiro da subsequência é 1. Caso seja, todos os valores binários da subsequência são alvos de um *flip*.

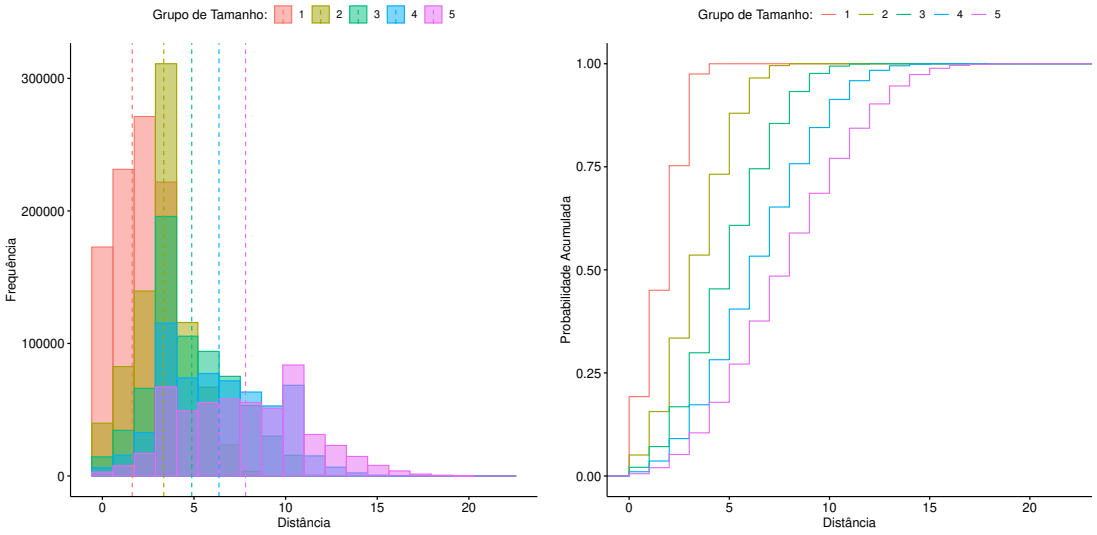
As subsequências finitas binárias obtidas são então agrupadas por comprimento e calculamos a frequência de cada ocorrência, ordenando-as de modo decrescente dentro de cada grupo. Estabelecemos a subsequência finita binária mais comum de cada grupo como o vetor binário de referência.

Para ter uma visão mais geral dos dados binários, calculamos a distância de Hamming de cada binário para o vetor binário de referência de seu respectivo grupo. Seja x, y dois *strings* binários de n elementos. Definimos como distância de Hamming entre

x, y o valor dado por $d(x,y) = \sum_{i=1}^{n} \delta_i$, onde $\delta_i = 0$, se $x_i = y_i$, e $\delta_i = 1$, caso contrário. Contabilizamos a frequência absoluta e calculamos a probabilidade acumulada empírica da distância para cada grupo de 1 até 5 resíduos. A Figura 18 apresenta o panorama mais amplo das subsequências finitas de binários.

Figura 18 – Distância de Hamming entre subsequências de binários.

(a) Histograma. (b) Probabilidade Acumulada Empírica.



Na Subfigura 18(a) temos um histograma com a frequência absoluta em cada um dos 5 grupos, com a média de grupo marcada com uma barra vertical pontilhada. Note que no k-ésimo grupo há 2^{5k} binários possíveis, então o crescimento do número de binários possíveis é exponencial. Adicionalmente, o número de binários possíveis no k-ésimo grupo com distância q de uma referência arbitrária é C_q^{5k}, onde a maior distância possível no grupo é $5k$. Ou seja, se considerarmos que a distribuição dos binários é uniforme, então a distribuição da distância destes binários para uma referência qualquer deve ser simétrica, com média próxima de $5k/2$ e caudas pesadas. Ainda assim, podemos ver no histograma que a média empírica da distância parece aumentar linearmente à medida em que aumenta o tamanho do grupo, sempre menor do que $5k/2$. Além disso, é esperado pelo princípio da casa dos pombos que quanto menor o grupo, mais concentrado esteja. Afinal, o número de possíveis binários cresce exponencialmente e o número de distâncias possíveis cresce linearmente. No entanto, os grupos de 4 e 5 resíduos parecem mais concentrados do que deveriam pelo seu tamanho se os binários fossem uniformemente distribuídos, apresentando uma forte assimetria positiva e uma cauda longa à direita. Isto tudo parece indicar que há uma estrutura para os binários, ou seja, que há certa preferência por certas subsequências e não é razoável supor que são uniformemente distribuídos.

Na Subfigura 18(b) temos um gráfico com a probabilidade acumulada empírica da distância em cada um dos 5 grupos. Note que à medida em que aumenta o tamanho do

grupo, a frequência relativa do vetor binário de referência diminui significativamente. No grupo 1 ele representa 20% das observações, ao passo que no grupo 5 não chega a 5%. Ou seja, os grupos partem progressivamente de valores menores de probabilidade acumulada. Apesar disso, todos os grupos convergem rapidamente para a ordenada 1 e a taxa dessa convergência parece aumentar na medida em que aumenta o tamanho do grupo. Pode-se verificar por inspeção visual que a distância entre as curvas dos grupos 1 e 2 é maior do que a distância entre as curvas dos grupos 4 e 5, bastando contar os pontos de contato entre as respectivas curvas de cada dupla de grupos. Além disso, também podemos notar que os maiores saltos ocorrem próximos de $5k/2$, mas que são em média maiores à esquerda de $5k/2$ em todos os grupos de tamanho k. No entanto, o quanto são maiores depende do tamanho do grupo, sendo a maior diferença no grupo 5. Isto reitera a análise feita no gráfico anterior, mostrando que à medida em que aumenta o número de resíduos, uma forte preferência se manifesta com respeito aos binários e as suas possíveis configurações.

Dado o cenário em que temos suspeitas para acreditar que os binários apresentam alguma estrutura que é exacerbada quando aumentamos o tamanho do grupo, é natural cogitar que os resíduos tenham alguma relação com esse fenômeno. Afinal, o k-ésimo grupo é composto por conjunto de $5k$ átomos associados a k resíduos. Não é difícil imaginar que os átomos de um resíduo A, precedido por L e sucedido por C, podem apresentar certas preferências do seu binário associado em decorrência da geometria de L e C. Em tese, quanto mais resíduos antecedendo ou sucedendo, mais restrito ficam os átomos desse resíduo. No entanto, também pode-se imaginar que essa influência tem um retorno marginal a partir de determinada quantidade de resíduos.

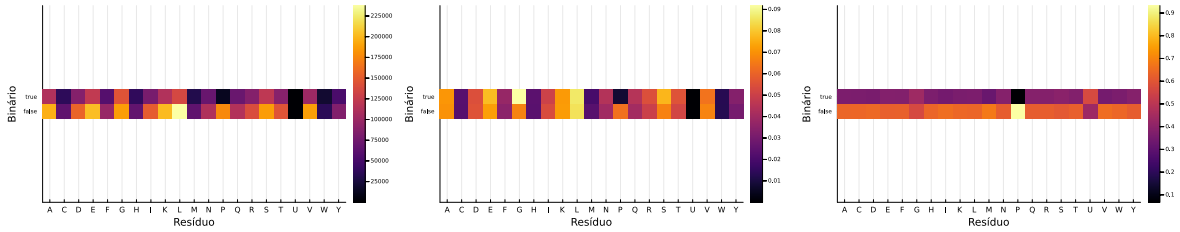
De toda forma, queremos verificar se existe essa relação entre os binários e os resíduos que se manifesta como uma preferência na configuração dos binários. Como vimos que o grupo 1 apresentou a menor preferência, então é razoável pensar que se existe uma relação deste tipo neste grupo, é muito provável que exista nos outros. Além disso, é mais fácil explorar visualmente o grupo 1 do que os demais, pois aqueles exigiriam um crescente número de ilustrações em altas dimensões.

Assim, tomamos as subsequências finitas de 1 resíduo e enumeramos a quantidade de ocorrências de cada par resíduo-binário. Note que, como todo resíduo tem 5 binários, então para cada resíduo temos 5 pares resíduo-binários. Além disso, como não controlamos por tipo de átomo, então a enumeração perde parte importante da informação geométrica, dado que duas subsequências `[true, true, true, false, false]` e `[true, false, true, false, true]` resultam em uma mesma enumeração. A despeito dessa limitação, a Figura 19 mostra as frequências (absoluta e relativas) das ocorrências dos resíduo-binários e algumas informações extras sobre os resíduos e os binários em separado.

Na Subfigura 19(a) temos um mapa de calor com a frequência absoluta dos resíduo-binários. Visualmente é possível notar que os valores `false` são os mais comuns,

Figura 19 – Mapa de calor dos resíduos pelos binários.

(a) Enumeração (b) Proporcional aos binários (c) Proporcional aos resíduos



com a linha associada apresentando uma coloração em média mais clara. Também é notável como o resíduo L é o mais comum, desta vez usando a coluna associada. Comparando cada resíduo nas colunas, é visível que embora na maioria dos resíduos exista uma preferência pelo `false`, essa preferência não é homogênea para todos os resíduos. O contraste entre os binários de cada resíduo parecem diferentes. Além disso, é importante ressaltar que, como não houve qualquer ocorrência do resíduo O, então ele não aparece neste gráfico ou nos gráficos vizinhos.

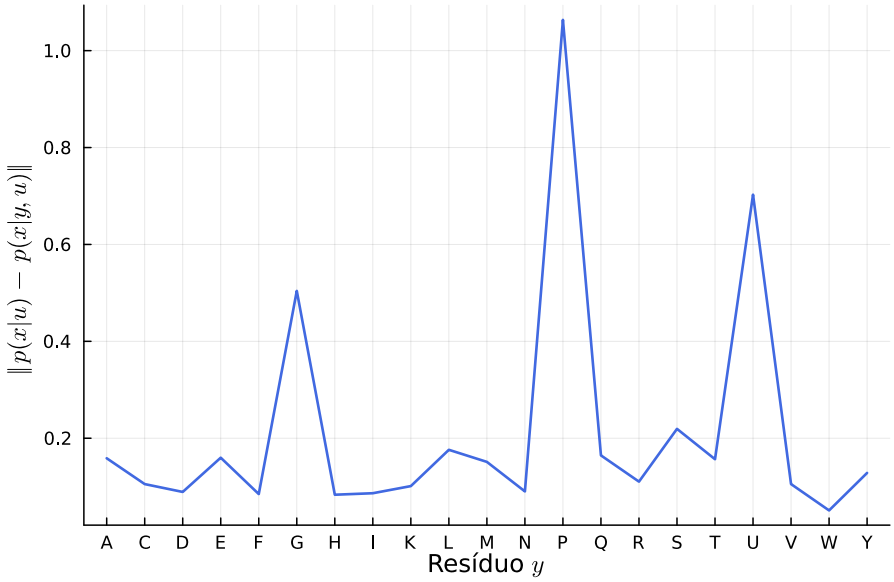
Na Subfigura 19(b) temos a frequência relativa de cada par resíduo-binário com respeito ao binário associado. Em outras palavras, dividimos os valores de cada linha pelo total da linha. Assim, se somarmos os valores de cada linha, obtemos 1. Isto é, o elemento na i-ésima linha e j-ésima coluna é equivalente à probabilidade estimada do resíduo ser j dado que o binário é i. Por outro lado, na Subfigura 19(c) temos a frequência relativa de cada par resíduo-binário com respeito ao resíduo associado. Analogamente, dividimos os valores de cada coluna pelo total da coluna. Assim, o elemento na i-ésima linha e j-ésima coluna é equivalente à probabilidade estimada do binário ser j dado que o resíduo é i. Ambos os gráficos reforçam que a preferência não é homogênea para todos os resíduos, mesmo tendo desconsiderado a informação sobre cada átomo. Entretanto, as diferenças marcantes entre as probabilidades condicionais naturalmente levam a crer que a informação do átomo pode ser a chave para distinguir melhor os casos, dado que parte da homogeneização se deve ao fato de, para quaisquer 5 binários consecutivos, termos no máximo 2 resíduos associados a estes binários.

Seja U_t o átomo associado ao t-ésimo elemento da subsequência finita de binários. Deixaremos implícito o condicionamento ao U_t de agora em diante, então sempre que escrevermos X_t ou $X_t|Y_t$, entenda-se $X_t|U_t$ e $X_t|\{Y_t, U_t\}$, respectivamente. Dado o nosso modelo de contagem, calculamos as funções de massa de probabilidade de X_t e $X_t|Y_t$, implicitamente condicionados ao U_t, e as distância euclidianas entre as duas funções.

$$\begin{cases} H_0: & \|p(x|u) - p(x|y,u)\|_2 = 0 \\ H_1: & \|p(x|u) - p(x|y,u)\|_2 \neq 0 \end{cases} \tag{3.3}$$

Note que ao condicionarmos ao U_t, as funções de massa de probabilidade não são mais vetores de comprimento $|X_t|$, e sim matrizes $|U_t| \times |X_t|$. No caso de $X_t|\{Y_t, U_t\}$, temos $|Y_t| = 21$ matrizes que representam a sua f.m.p para cada um dos resíduos, exceto O. Então a distância foi calculada vetorizando as matrizes, o que é equivalente à norma de Frobenius. Importante notar que, dado $|U_t| = 5$ e $|X_t| = 2$, a maior distância possível entre duas funções de massa de probabilidade é $\sqrt{10}$, o que ocorre se uma matriz tiver a primeira coluna só com 1 e a segunda só com 0, ao passo que na outra matriz ocorre o contrário. A Figura 20 apresenta o gráfico das distâncias calculadas para cada um dos resíduos.

Figura 20 – Distância da f.m.p. de X_t para a f.m.p. de $X_t|Y_t$.



Como dito anteriormente, se resíduos e binários são independentes, então a distância $\|p(x|u) - p(x|y,u)\|$ deve estar bem próxima de zero para cada um dos resíduos. No entanto, é possível ver que dado quase qualquer resíduo, exceto W, temos que a distância entre as funções é maior do que zero. Em particular, os resíduos G, P e U possuem uma diferença acentuada entre as duas funções. Nos casos de G e P, tal diferença pode ser explicada pelas características específicas da geometria molecular de ambos. No entanto, quanto aos resíduos U e W, devemos lembrar que as ocorrências do primeiro praticamente não figuram nos dados e o segundo não possui qualquer ocorrência. Ainda assim, os resultados reiteram a suspeita de que os resíduos fornecem informação sobre os binários e que faz sentido tentar incorporar essa informação no BP.

Por fim, aplicamos um teste χ^2 de independência para $X_t|U_t$ e $Y_t|U_t$. Para isso construímos uma tabela de contingência para cada átomo em que as linhas são os resíduos e as colunas são os binários. Com base nesta tabela, calculamos a estatística de teste χ^2 e obtivemos os valores $[24469.7, 32751.4, 43394.4, 66016.2, 34.3726]$ para os átomos na ordenação que estipulamos anteriormente. Para um $\alpha = 0.05$ usual e 20 graus

de liberdade, temos que $p \ll 0.001$, para os 4 primeiros átomos, e $p \approx 0.02371$, no quinto átomo. Portanto, como todos são menores que α, temos evidências para rejeitar a hipótese nula de que $X_t|U_t$ e $Y_t|U_t$ são independentes, o que está de acordo com a análise feita até este ponto.

3.3 BP Autoregressivo Oculto de Markov

Considerando que é possível usar a informação sobre os resíduos para refinar a proposta do FBS, propomos a incorporação de um modelo MOM(n)-AR(m) no algoritmo BP para o PDGDM. Nomeamos a alteração resultante de *Autoregressive (Hidden) Markovian Branch and Prune* (ARMBP) e estabelecemos que ele deve seguir algumas condições essenciais:

- A análise e o processamento dos dados devem ser prévios à execução do BP, de modo que o BP somente tem acesso ao modelo já ajustado no seu *runtime*;
- O modelo ajustado tem acesso somente aos dados da instância de entrada do BP, que incluem a identificação dos resíduos e dos átomos da espinha dorsal estendida;
- A incorporação deve ser de modo que somente seja necessário fazer consultas ou cálculos simples durante a execução do BP;
- A incorporação deve reduzir o número médio de vértices visitados na árvore binária do espaço de soluções do PDGDM, quando comparada com o BP tradicional.

Se o ARMBP conseguir satisfazer estas 4 condições, então consideramos que o refino foi bem-sucedido. Em especial, o último item é o nosso principal critério de comparação para a pesquisa e sua satisfatibilidade depende de experimentos computacionais descritos na seção 3.4. No entanto, para satisfazer as demais condições, precisamos somente especificar o modelo e a forma de integração ao BP. Caso ambos estejam bem-definidos e fundamentados, temos que estas condições são facilmente verificáveis.

3.3.1 Especificação

Primeiramente, vamos estipular que t é o valor de uma posição. Apesar de usualmente esta variável aparecer como um indexador de tempo, não é incomum o seu uso como indexador de posição ou local. Dada uma ordenação para um segmento de proteína, a t-ésima posição diz respeito ao t-ésimo binário associado a um átomo da espinha dorsal estendida de um resíduo. Logo, qualquer que seja a variável aleatória sendo indexada por t, é importante lembrar que está dizendo a respeito ao binário.

Dito isto, para cada resíduo ou átomo, vamos codificar o seu nome como um inteiro positivo, como exposto nas Tabela 1 e Tabela 2. Assim, tanto o resíduo A quanto o átomo C correspondem ao natural 1. Não há risco de confusão porque átomos e resíduos são dados por variáveis aleatórias distintas. Por sua vez, convencionamos para os binários que `false` é o natural 1 e `true` é natural 2. Estas codificações e convenções são importantes, pois permitem uma associação intuitiva entre o i-ésimo valor de uma variável aleatória e a i-ésima linha ou coluna de alguma das matrizes associadas ao modelo. Assim, definimos as seguintes variáveis aleatórias:

X_t : Valor codificado do binário na posição t, com $|X_t| = 2$;

Y_t : Valor codificado do resíduo na posição t, com $|Y_t| = 23$;

U_t : Valor codificado do átomo na posição t, com $|U_t| = 5$.

Portanto, os dados processados descritos na seção 3.1 podem ser vistos como sequências finitas $\{x_t\}_{t=1}^N$, $\{y_t\}_{t=1}^N$ e $\{u_t\}_{t=1}^N$, com $t, N \in \mathbb{N}$, que formam uma realização das sequências finitas de variáveis aleatórias $\{X_t\}_{t=1}^N$, $\{Y_t\}_{t=1}^N$ e $\{U_t\}_{t=1}^N$. Analogamente ao estabelecido no seção 2.3.7, temos que o ARMBP(m, n) é dado pelas relações de independência condicional

$$P(X_t|X_{[1:t-1]}, Y_{[1:t]}, U_{[1:t]}) = P(X_t|X_{[t-n:t-1]}, U_t), \tag{3.4}$$

$$P(Y_t|X_{[1:t]}, Y_{[1:t-1]}, U_{[1:t]}) = P(Y_t|X_t, Y_{[t-m:t-1]}, U_t). \tag{3.5}$$

Sem perda de generalidade, vamos estabelecer que $n = m = 1$. Como mencionado anteriormente na capítulo 2, não é necessário nenhum arcabouço teórico distinto para ordens maiores. Caso o modelo esteja bem-definido para estes parâmetros e desempenhe de modo satisfatório, então a extensão é direta para valores maiores. Além disso, vamos supor que a CM é homogênea, ou seja, as matriz de transição são independentes de t.

O conjunto de matrizes de transição $T = \{T^{(1)}, \ldots, T^{(5)}\}$ do modelo contém $|U_t| = 5$ matrizes estocásticas, uma para cada valor possível de U_t. A k-ésima matriz de transição $T^{(k)}$ é dada por

$$(t_{i,j}^{(k)}) = P(X_t = j|X_{t-1} = i, U_t = k), \tag{3.6}$$

$$T^{(k)} = \begin{bmatrix} p_{1,1}^{(k)} & p_{1,2}^{(k)} \\ p_{2,1}^{(k)} & p_{2,2}^{(k)} \end{bmatrix}. \tag{3.7}$$

Equivalentemente, podemos fazer uma única matriz de transição T de ordem $|X_t \times U_t|$, onde o elemento da i-ésima linha e k-ésima coluna está associada às duplas

ordenadas $(p,q),(r,s) \in X_t \times U_t$, de modo que $p,r \in X_t$ e $q,r \in U_t$. Neste caso, como descrito na seção 2.3.3, teríamos uma significativa esparsidade estrutural, agravada pelo fato que a ordenação dos átomos impossibilita qualquer transição para uma dupla ordenada cujo átomo esteja fora da ordem. A Equação 3.8 mostra a T neste caso. Optamos por usar o conjunto de matrizes como na Equação 3.7 porque parece mais adequado e intuitivo pedagogicamente[8]. Importante notar que a adoção desta representação significa que o mesmo será feito para a matriz de emissão, de modo que as matrizes permaneçam compatíveis para o produto matricial usual.

$$T = \begin{array}{c} \\ (1,1) \\ (1,2) \\ (1,3) \\ (1,4) \\ (1,5) \\ (2,1) \\ (2,2) \\ (2,3) \\ (2,4) \\ (2,5) \end{array} \begin{array}{c} \begin{array}{cccccccccc} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (2,1) & (2,2) & (2,3) & (2,4) & (2,5) \end{array} \\ \begin{bmatrix} 0 & p_{1,1}^{(1)} & 0 & 0 & 0 & 0 & p_{1,2}^{(1)} & 0 & 0 & 0 \\ 0 & 0 & p_{1,1}^{(2)} & 0 & 0 & 0 & 0 & p_{1,2}^{(2)} & 0 & 0 \\ 0 & 0 & 0 & p_{1,1}^{(3)} & 0 & 0 & 0 & 0 & p_{1,2}^{(3)} & 0 \\ 0 & 0 & 0 & 0 & p_{1,1}^{(4)} & 0 & 0 & 0 & 0 & p_{1,2}^{(4)} \\ p_{1,1}^{(5)} & 0 & 0 & 0 & 0 & p_{1,2}^{(5)} & 0 & 0 & 0 & 0 \\ 0 & p_{2,1}^{(1)} & 0 & 0 & 0 & 0 & p_{2,2}^{(1)} & 0 & 0 & 0 \\ 0 & 0 & p_{2,1}^{(2)} & 0 & 0 & 0 & 0 & p_{2,2}^{(2)} & 0 & 0 \\ 0 & 0 & 0 & p_{2,1}^{(3)} & 0 & 0 & 0 & 0 & p_{2,2}^{(3)} & 0 \\ 0 & 0 & 0 & 0 & p_{2,1}^{(4)} & 0 & 0 & 0 & 0 & p_{2,2}^{(4)} \\ p_{2,1}^{(5)} & 0 & 0 & 0 & 0 & p_{2,2}^{(5)} & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}. \tag{3.8}$$

Note que como os átomos estão ordenados sempre da mesma forma, então uma matriz de 5-passos que comece no primeiro átomo de qualquer resíduo é sempre dada pela composição $\prod_{i=1}^{5} T^{(i)}$. Ou seja, quando afirmamos que a matriz de transição é independente de t, não estamos falando das $T^{(i)}$ individualmente, e sim da matriz $\prod_{i=1}^{5} T^{(i)}$. Claramente que cada $T^{(i)}$ depende do i, que por sua vez está estritamente relacionado ao t, mas os 5-passos completos não:

$$\prod_{i=1}^{5} T^{(i)} = \prod_{i=k}^{k+4} T^{(i)}, \forall\ k \in \mathbb{N} : k \equiv 1 \bmod 5. \tag{3.9}$$

Ainda que se deseje começar de outra posição, sempre é possível adaptar qual será a composição ou, analogamente, completar o número de passos até o primeiro átomo do próximo resíduo. É essencial frisar que isto não contradiz a nossa suposição de que a CM é homogênea. Deste modo, temos que T é primeiro parâmetro do modelo.

[8] No entanto, com respeito a implementação computacional, é arguível que o uso de matrizes esparsas pode ser vantajoso em termos de desempenho, mas isto está fora do escopo desta seção.

A probabilidade incondicional com respeito ao Y_t de $X_t|U_t$, $P(X_t = j|U_t = k)$, é dada por uma matriz de distribuição com dimensão $|U_t| \times |X_t|$ ou, equivalentemente, por $k = 5$ vetores-linha

$$u(t,k) = \begin{bmatrix} P(X_t = 1|U_t = k) & P(X_t = 2|U_t = k) \end{bmatrix}, \tag{3.10}$$

$$P(X_t = j|U_t = k) = \sum_{i=1}^{|X_t|} P(X_t = j|X_{t-1} = i, U_t = k)P(X_{t-1} = i|U_{t-1} = k-1). \tag{3.11}$$

Note que, necessariamente, $k \equiv t \bmod 5$. Como o indexador t dá a posição do t-ésimo binário associado a cada átomo que ocorre seguindo a mesma ordenação, então naturalmente para $t \in \{1, 6, 11, \ldots\}$ temos que k só pode ser 1, ou seja H. Assim, podemos omitir o segundo parâmetro de $u(t,k)$ e escrever somente $u(t)$. De toda forma, todo $u(t)$ deve satisfazer Equação 2.27. Como na seção 2.3.3, vamos denotar por $u_i(t)$ o i-ésimo elemento do vetor $u(t)$.

Além disso, é importante ressaltar que a matriz de distribuição inicial $u(1)$ é efetivamente um único vetor-linha não-nulo em decorrência da ordenação. Isto decorre do fato de que o único valor de U_t que os elementos do vetor podem assumir é 1, porque todo resíduo está ordenado de modo que o primeiro átomo seja o hidrogênio amida H. Em outras palavras, a representação matricial tem uma única linha com elementos possivelmente distintos de 0 e todas as demais entradas nulas.

Por fim, a Equação 3.11 define um procedimento recursivo para achar $P(X_t = j|U_t = k)$ com um critério bem determinado de parada, sempre parando quando chega no $P(X_1 = j|U_1 = 1)$. De fato, este vetor de distribuição para os valores $j \in supp(X_t)$ é um dos principais parâmetros do modelo. Como vimos anteriormente, ele é dito vetor distribuição inicial $u(1)$. Considerando a homogeneidade de T e que $0 \equiv 5 \bmod 5$, temos que, para qualquer $t, j \in \mathbb{N}$,

$$u(t+j) = u(t)\left(\prod_{i=t}^{t+j-1} T^{(i \bmod 5)}\right) = u(1)\left(\prod_{i=1}^{t+j-1} T^{(i \bmod 5)}\right), \tag{3.12}$$

onde convencionamos que $T^{(0)} = T^{(5)}$, com a identidade anterior permitindo que se expresse a relação acima de forma mais compacta. Note que se $t+j \equiv 1 \bmod 5$, então a Equação 3.12 tem uma forma próxima da usual do vetor de distribuição inicial:

$$u(t+j) = u(1)\left(\prod_{i=1}^{5} T^{(i)}\right)^{\left\lfloor \frac{t+j-1}{2} \right\rfloor}. \tag{3.13}$$

Evidentemente, a representação da matriz T como na Equação 3.8 não tem o mesmo problema de incluir módulos e a preocupação de saber em qual átomo se está em cada instante, seguindo as propriedades exatamente como descritas na Equação 2.16 e Equação 2.17. Como dito, ambas as formas de expressar T são equivalentes, somente sendo necessário alguns ajustes em decorrência da comodidade adicional da representação como conjunto de matrizes de transição $T^{(k)}$.

De toda forma, a intuição é a mesma apresentada na seção 2.3.3: Multiplicar à direita um vetor de distribuição da posição t pela matriz $T^{(t \bmod 5)}$ resulta no vetor de distribuição da posição $t+1$. Em outras palavras: a matriz de transição avança o vetor de distribuição em um passo. Com isso, o processo de parâmetro do MOM(1)-AR(1) está bem-definido e temos que $u(1)$ é o segundo parâmetro do modelo.

O conjunto das matrizes de emissão $E = \{E^{(1)}, \ldots, E^{(115)}\}$ do modelo contém $|Y_t \times U_t| = 115$ matrizes de dimensão $|X_t| \times |Y_t|$, uma para cada valor possível do produto cartesiano de Y_t com U_t. De modo análogo à matriz de transição, há outras representações possíveis para a matriz de emissão E. No entanto, a forma escolhida faz com que as matrizes em E tenham o número de linhas igual ao número de colunas das matrizes em T. A (r,s)-ésima matriz de emissão $E^{(r,s)}$ é dada por

$$(e_{j,k}^{(r,s)}) = P(Y_t = k | X_t = j, Y_{t-1} = r, U_t = s), \tag{3.14}$$

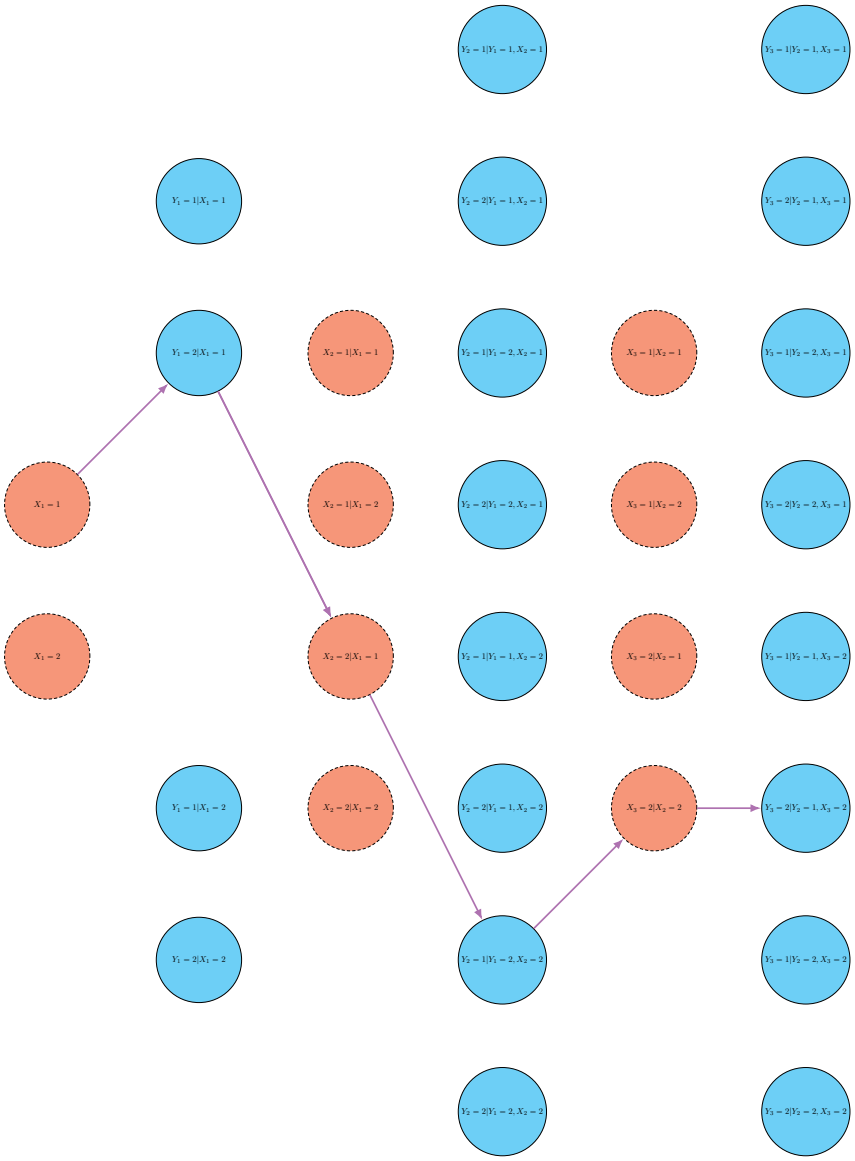
$$E^{(r,s)} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & \ldots & 21 & 22 \end{array} \\ \begin{bmatrix} e_{1,1}^{(r,s)} & e_{1,2}^{(r,s)} & \ldots & e_{1,21}^{(r,s)} & e_{1,22}^{(r,s)} \\ e_{2,1}^{(r,s)} & e_{2,2}^{(r,s)} & \ldots & e_{2,21}^{(r,s)} & e_{2,22}^{(r,s)} \end{bmatrix} \end{array}. \tag{3.15}$$

Com a matriz de emissão, podemos escrever um vetor $p_j^{(r,s)}$, cujo k-ésimo elemento é $p_j^{(r)}(k) = P(Y_t = k | X_t = j, Y_{t-1} = r, U_t = s)$. Dado esse vetor, seja $D^{(r,s)}(k)$ uma matriz diagonal tal que $(d^{(r,s)}(k))_{j,j} = p_j^{(r,s)}(k)$. O uso das matrizes na forma $D^{(r,s)}(k)$ é equivalente ao uso das matrizes $E^{(r,s)}$ dado alguns ajustes. O uso de uma ou outra depende principalmente da priorização do uso de memória ou de processamento. De toda forma, essas matrizes permitem calcular matricialmente o vetor das probabilidades incondicionais com respeito ao Y_t, $P(Y_t|U_t)$, e as matrizes de probabilidade conjunta de X_t, Y_t, $P(X_t, Y_t|U_t)$. Assim, temos

$$\begin{aligned} P(X_t, Y_t | U_t) = &\underbrace{P(X_1|U_1)}_{=u(1)} \underbrace{P(Y_1|X_1, U_1)}_{=\frac{P(X_1,Y_1|U_1)}{P(X_1|U_1)}} \\ &\times \prod_{h=2}^{t} \underbrace{P(X_h|X_{h-1}, U_h)}_{=T^{(h \bmod 5)}} \underbrace{P(Y_h|X_h, Y_{h-1}, U_h)}_{=E^{(Y_{h-1}, h \bmod 5)}}. \end{aligned} \tag{3.16}$$

A Equação 3.16 é uma maneira compacta de dizer que a probabilidade de uma sequência é o produto dos fatores que correspondem a cada passo em um grafo como o Figura 21. Assim, o processo de estados do MOM(1)-AR(1) está bem-definido e temos o terceiro e último parâmetro para o modelo.

Figura 21 – Sequência de 3 passos que fatora a probabilidade de ocorrer $x_{[1:3]} = (1, 2, 2)$ e $y_{[1:3]} = (2, 1, 2)$ em um MOM em que X_t e Y_t são binários.



Dado que as variáveis são discretas e os dados são de contagem, então o cálculo da máxima verossimilhança para os parâmetros é simples, como visto anteriormente na seção 2.3.7. Apesar disso, é importante frisar que o modelo lida bem com dados censurados ou observações faltantes. Isso faz com que não seja necessário segmentar as proteínas. No entanto, a calibração do modelo neste caso exigiria o uso de uma modificação do algoritmo de Baum-Welch. De modo a simplificar a pesquisa, optamos por manter o uso dos segmentos por 2 motivos: **(i)** manter a comparabilidade com a primeira versão do FBS

e a pesquisa iniciada por Marques et al.; e **(ii)** tratar os segmentos como sendo os dados completos para fins de ajuste do modelo. De toda forma, não precisamos mais dos dados uma vez que o modelo esteja ajustado, o que satisfaz a primeira condição para o ARMBP.

Por fim, para o ajuste do modelo, usamos 90% dos segmentos obtidos pelo processamento dos dados do PDB descritos na seção 3.1.2 como dados de treinamento. O restante dos segmentos foi separado como dados de teste para a avaliação e validação do ARMBP. Importante ressaltar que a partição não foi completamente aleatória: se dois ou mais segmentos provenientes de uma mesma entrada ou de entradas distintas do PDB possuem sequência finita de binários e resíduos idênticas, então não permitimos que fiquem separadas e são alocadas necessariamente para os dados de treinamento. Isso se traduz como uma desvantagem induzida no desenho do nosso experimento para evitar sobreajuste e atuar no caso mais difícil, que é quando a instância investigada não figura nos dados de treinamento.

3.3.2 Integração

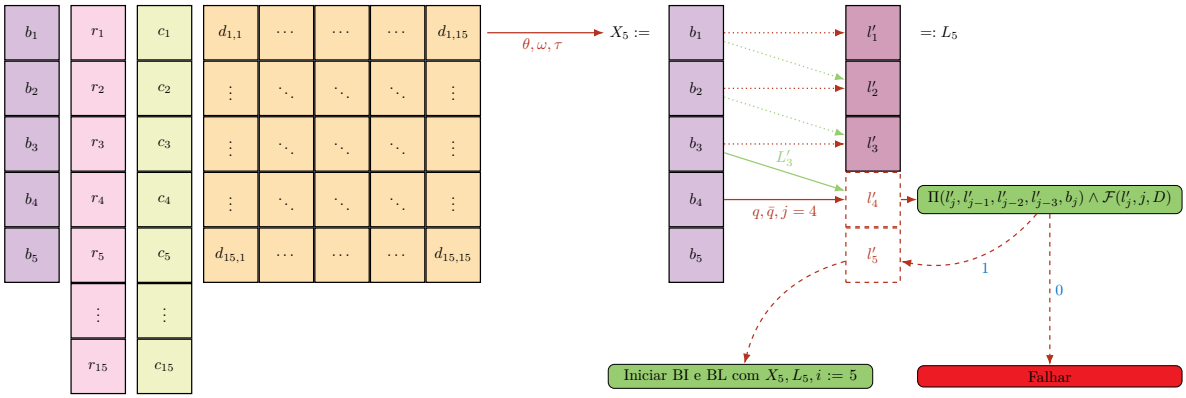
Na literatura, há artigos sobre uso do MOM para conformação de proteína como um método de otimização contínua (BOOMSMA et al., 2008; LENNOX et al., 2010). Nossa proposta difere destas não apenas pela especificação do modelo, mas especialmente na sua integração com o BP. A capacidade de operar dentro do BP e sanar algumas das principais limitações do algoritmo são os pontos mais fortes da proposta, em especial porque o BP já tem um desempenho extremamente competitivo. De fato, o BP está sendo adaptado para tratar de problemas mais complexos, como no caso em que as distâncias são, ao menos em parte, intervalares (LAVOR; LIBERTI; MUCHERINO, 2011). Então desenhar uma integração no BP que seja simples e fácil de adaptar para o caso intervalar é um diferencial almejado na pesquisa.

Como visto na seção 2.2.3, o BP é um algoritmo baseado na busca em profundidade. Cada iteração envolve uma entre duas ações com respeito ao vértice corrente: explorar um vértice-filho; ou retornar para um vértice-pai. Anteriormente, descrevemos dois problemas desta abordagem no PDGDM: **(a)** o algoritmo é naturalmente serial, com a paralelização incorrendo em computação desnecessária; **(b)** o algoritmo não exibe preferência na seleção de qual vértice-filho explorar primeiro. O FBS aborda em específico o segundo item, mostrando que é possível incorporar informação de modo que o BP apresente uma preferência dependendo dos dados, empregando um modelo probabilístico simples com respeito à sequência finita de binários (MARQUES et al., 2024).

Na seção anterior, especificamos um modelo probabilístico gráfico mais complexo e incorporamos uma informação adicional com respeito à sequência finita de resíduos, vejamos agora como esse modelo pode interagir com o BP para sanar ambos os problemas.

Suponha que temos o modelo descrito bem-definido e ajustado para os dados disponíveis do PDB com a tripla $(T, E, u(1))$. Este modelo é um novo parâmetro de entrada para o BP, junto à instância que se deseja achar uma conformação. Primeiramente, na fase de inicialização, o BP coleta a informação da sequência finita de resíduos da instância de entrada e ordena os átomos de acordo com o estipulado anteriormente, passando essa informação para o modelo e satisfazendo a segunda condição para o ARMBP. Nesta fase também ocorrem o pré-processamento de alguns dos dados, como o cálculo de X_5, o vetor coordenadas cartesianas dos 5 primeiros átomos que estão fixados no vetor binário $(0, 0, 0, 0, 0)$ ou, alternativamente, no vetor binário $(b_1, b_2, b_3, b_4, b_5)$ dado pelo algoritmo de avanço-retrocesso, e a definição inicial do conjunto de poda $\mathcal{P} = \varnothing$. A Figura 22 apresenta uma visão panorâmica desta fase.

Figura 22 – Diagrama da fase inicial de captura da entrada e pré-processamento com uma instância de tamanho $n = 15$.



Tanto na fase de inicialização quanto nas buscas posteriores, usamos as funções Π e $\mathcal{F}$ para a validação dos resultados intermediários. Sejam $\Pi : \{\mathbb{R}^3\}^4 \times \{0, 1\} \to \{0, 1\}$, função que verifica se a posição relativa do primeiro termo com relação ao plano gerado pelos 3 demais está de acordo com o seu binário associado, tal que

$$\Pi(x_1, x_2, x_3, x_4, b) = \begin{cases} 0, & \text{se } \pi(x_1, x_2, x_3, x_4) \neq b, \\ 1, & \text{se } \pi(x_1, x_2, x_3, x_4) = b. \end{cases}, \tag{3.17}$$

onde $\pi : \{\mathbb{R}^3\}^4 \to \{0, 1\}$ é a função que indica essa posição relativa, dada por

$$\pi(x_1, x_2, x_3, x_4) = \begin{cases} 0, & \text{se } (x_1 - x_2) \cdot \dfrac{(x_3 - x_2) \times (x_4 - x_2)}{\|(x_3 - x_2) \times (x_4 - x_2)\|} < 0, \\ 1, & \text{se } (x_1 - x_2) \cdot \dfrac{(x_3 - x_2) \times (x_4 - x_2)}{\|(x_3 - x_2) \times (x_4 - x_2)\|} \geqslant 0. \end{cases}, \tag{3.18}$$

e $\mathcal{F} : \mathbb{R}^3 \times \mathbb{N} \times \mathbb{R}^{n \times n} \to \{0, 1\}$, função que verifica se há aresta de poda associada à coordenada e, caso exista, se a posição respeita a distância dada pela matriz D para as

coordenadas obtidas anteriormente, dada por

$$\mathcal{F}(x_j, j, D) = \begin{cases} 1, & \text{se } \forall\ i \leqslant n, j-i > 3 : d_{i,j} \neq 0 \wedge |\|x_j - x_i\| - d_{i,j}| < \varepsilon^*, \\ 0, & \text{c.c.} \end{cases}, \tag{3.19}$$

onde $\varepsilon^* = 10^{-6}$ é estipulado como a tolerância permitida.

A computação mais intensiva da fase de inicialização é a produção de uma resposta parcial viável, denotada por L_5, a partir do vetor binário inicial $X_5 = (0, 0, 0, 0, 0)$. Para construir essa resposta parcial, procedemos fixando a primeira coordenada l'_1 na origem $(0, 0, 0)$, a segunda coordenada l'_2 em $(-d_{1,2}, 0, 0)$ e a terceira coordenada l'_3 em $(d_{1,2} + d_{2,3}\cos(\theta_{1,3}), d_{2,3}\text{sen}(\theta_{1,3}, 0))$, como dito anteriormente (LAVOR; LIBERTI, 2014). No entanto, para l'_4 temos duas opções relacionadas com o ângulo diedral entre a primeira e quarta coordenadas. Como queremos somente coordenadas que estejam de acordo com o binário b_4, então verificamos se a posição candidata satisfaz o binário usando Π e se atendem as distâncias dadas pelas arestas de poda com $\mathcal{F}$. Se l'_4 satisfaz ambas as funções, então aceitamos e passamos para a seguinte, l'_5. Analogamente para a escolha de $(b_1, b_2, b_3, b_4, b_5)$ usando avanço-retrocesso, salvo a exceção de, caso ocorra poda, o algoritmo ajustar a resposta parcial viável para os vetores binários mais prováveis, excluídos os anteriormente produzidos.

Esse procedimento de pegar um vetor de binários X'_k e achar iterativamente um L'_k associado que respeite as condições desejadas será usado também nos demais processos do ARMBP. Para a fase de inicialização, no entanto, o objetivo é chegar a um L_5 válido e, se obtido, definimos o número i da iteração como 5 e iniciamos a Busca Irrestrita e a Busca Local com X_5 e L_5. Deste ponto em diante, temos um procedimento com três processos distintos das buscas. Em comum e com permissão de leitura a todo o procedimento, temos a informação de qual é a iteração i corrente, de quais vértices foram explorados e quais caminhos foram podados.

3.3.2.1 Busca Irrestrita

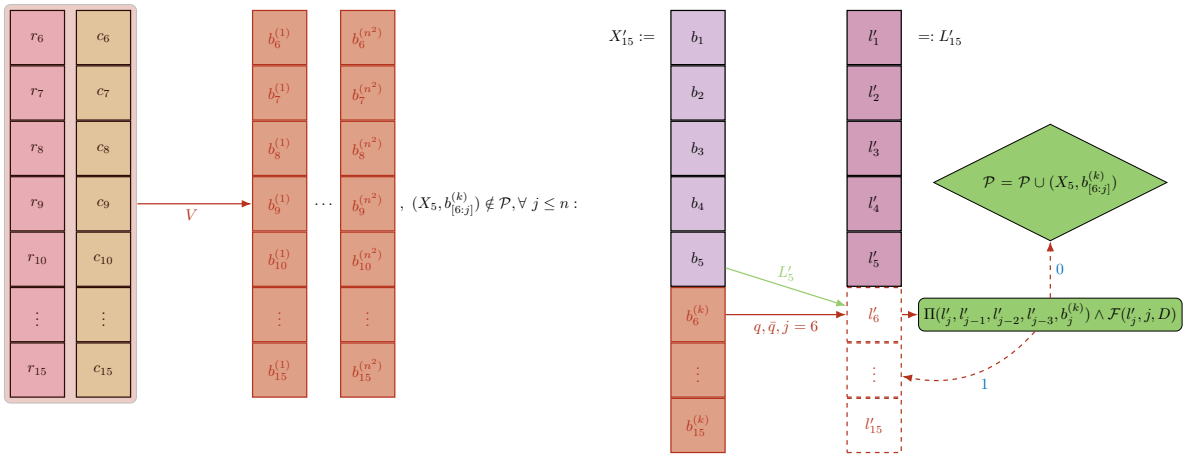
O primeiro processo de busca é global e não depende da iteração corrente ou dos vértices que foram explorados, sendo denominado **busca irrestrita** (BI) e representado na Figura 23. Neste caso, emprega-se a decodificação global para determinar a sequência finita de binários $\{x_t\}_{t=1}^N$ mais provável dada a sequência finita de resíduos $\{y_t\}_{t=1}^N$, ou seja, $P(X_{[1:N]}|Y_{[1:N]} = y_{[1:N]})$, pelo algoritmo de Viterbi de Lista.

Para isso, o algoritmo acha conjuntamente o caminho globalmente mais provável e um número de outros caminhos em ordem decrescente quanto a sua probabilidade, totalizando n caminhos. Após identificar um candidato, a busca irá testar para ver se a sequência finita de binários não está nos caminhos que foram podados e é uma conformação

válida. Se positivo, retorna a conformação; se negativo, exclui a sequência finita obtida e obtém-se a próxima mais provável. Note que isto é trivial pelo desenho do algoritmo de Viterbi, conceitualmente bem semelhante à Figura 21 em que cada par (X_t, Y_t) é um vértice no passo t, dado que a próxima sequência mais provável é necessariamente idêntica até um passo atrás da anteriormente obtida[9]. Além disso, este processo tem permissão de escrita para modificar os caminhos que foram podados.

O único parâmetro a ser estabelecido pela BI é o número n de caminhos a serem gerados pelo Viterbi de Lista. Por padrão, estabelecemos este valor como N^2, onde N é o tamanho da instância.

Figura 23 – Diagrama da Busca Irrestrita com uma instância de tamanho $n = 15$.



3.3.2.2 Busca Local

O segundo processo de busca é local e depende de todas as informações do procedimento, sendo denominado **busca local** (BL) e representado na Figura 24. Neste caso, no início da k-ésima iteração, emprega-se a decodificação local para determinar o binário x_{k+1} mais provável dada a sequência finita de resíduos $\{y_t\}_{t=1}^{N}$, ou seja, $P(X_{k+1}|Y_{[1:N]} = y_{[1:N]})$, pelo algoritmo de avanço-retrocesso. No caso do binário inicial $(0, 0, 0, 0, 0)$, essa etapa é realizada em cada iteração conjuntamente com um novo cálculo de avanço-retrocesso. Por outro lado, no caso de $(b_1, b_2, b_3, b_4, b_5)$, podemos calcular uma única vez, de modo que o avanço-retrocesso retorna $(b_1, \ldots, b_N)$ e temos um indicador de até qual índice sabemos que é uma resposta parcial válida. Neste segundo caso, usamos uma *max heap* para armazenar as possibilidades em ordem de probabilidade local.

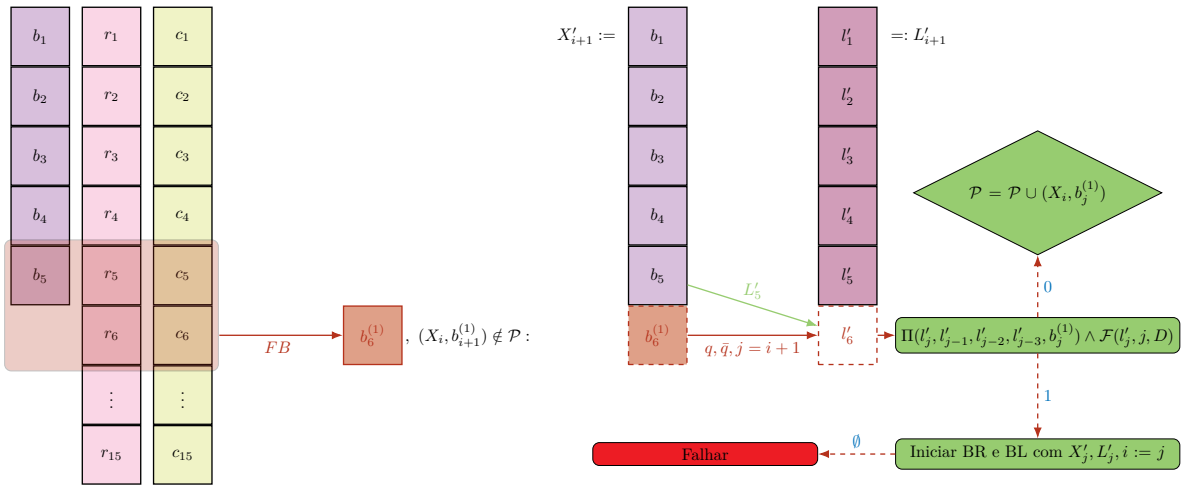
Após inferir o binário, a busca irá testar para ver se concatenação do caminho já percorrido $\{x_t\}_{t=1}^{k}$ com x_{k+1} não está nos caminhos que foram podados e é uma conformação

[9] Uma comparação apta é pensar no algoritmo de Dijkstra para o Problema do Caminho Mínimo. De fato, Viterbi é um algoritmo de caminho mínimo em que os pesos estão nas arestas direcionadas e são probabilidades.

válida. Se positivo, adiciona o vértice como explorado e segue para a próxima iteração ou, se $k = N$, retorna a conformação; se negativo, exclui a sequência finita obtida e testa-se a próxima mais provável. Este processo é próximo do FBS, na concepção de ser um BP guiado para os vértices-filhos mais prováveis dado o modelo probabilístico empregado. Além disso, este processo tem permissão de escrita para modificar a iteração corrente, os vértices que foram explorados e os caminhos que foram podados.

Importante frisar que, no caso do uso de um binário $(b_1, b_2, b_3, b_4, b_5)$ obtido pelo avanço-retrocesso, temos a vantagem adicional do *backtracking* do algoritmo ser naturalmente indicada pela *max heap*. Ao invés de voltar para a iteração imediatamente anterior, podemos voltar para a iteração com a maior diferença de probabilidade ao se mudar algum dos binários já percorridos. Assim, não só o avanço é guiado para os vértices-filhos mais prováveis, como também o retrocesso é guiado para a combinação de vértices antecessores mais provável. De toda forma, o algoritmo tem a garantia de que irá explorar todo o espaço de busca eventualmente. Em outras palavras, o algoritmo tem a garantia de que termina. Isso é importante porque esse retorno probabilístico poderia incorrer na possibilidade de um laço em que o programa entre em uma repetição infinita, no entanto a estrutura da *max heap* impede que isso ocorra.

Figura 24 – Diagrama da Busca Local com uma instância de tamanho $n = 15$ na iteração $i = 5$.



3.3.2.3 Busca Restrita

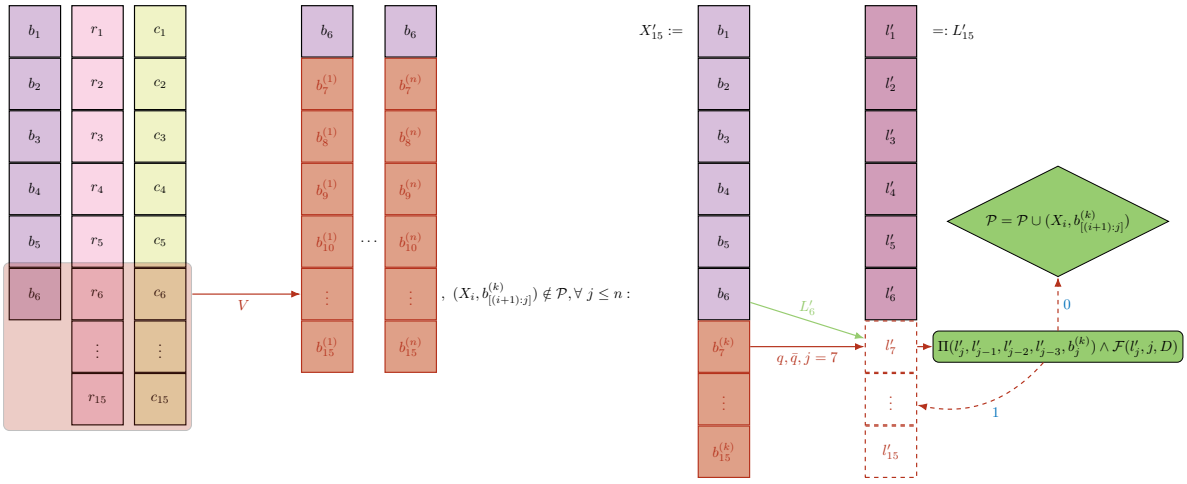
O terceiro processo de busca é global, depende de todas as informações do procedimento e de um parâmetro n, sendo denominado **busca restrita** (BR) e representado na Figura 25. Neste caso, no início da k-ésima iteração, emprega-se a decodificação global para determinar as n subsequências finitas de binários $\{x_t\}_{t=k+1}^{N}$ mais prováveis dada a sequência finita de resíduos $\{y_t\}_{t=1}^{N}$, ou seja, $P(X_{[k:N]}|Y_{[1:N]} = y_{[1:N]})$, com $k < N$, pelo

algoritmo de Viterbi de Lista. A BR é muito próxima da BI, tendo como diferença o ponto de início.

Após identificar um candidato, a busca irá testar para ver se concatenação do caminho já percorrido $\{x_t\}_{t=1}^{k}$ com a subsequência finita de binários inferida não está nos caminhos que foram podados e é uma conformação válida. Se positivo, retorna a conformação; se negativo, exclui a sequência finita obtida e testa-se a próxima mais provável. Como este processo depende da iteração, então ele tem que terminar após um número n de tentativas, mas não é necessário que o processo principal do BP termine a k-ésima iteração concorrentemente. Como o processo para cada k é autônomo, então não há condição de corrida possível e se o processo principal entrar em uma nova iteração antes da busca restrita terminar a anterior, então inicia-se uma nova busca restrita em paralelo sem qualquer prejuízo. Além disso, este processo tem permissão de escrita para modificar os caminhos que foram podados.

O único parâmetro a ser estabelecido pela BR é o número n de caminhos a serem gerados pelo Viterbi de Lista. Por padrão, estabelecemos este valor como $\left\lceil \frac{N-k+1}{2} \right\rceil$, onde N é o tamanho da instância. Note que aqui temos a preocupação de não pedir para o Viterbi de Lista produzir mais caminhos do que os existentes em determinado nível da árvore binária do espaço de soluções.

Figura 25 – Diagrama da Busca Restrita com uma instância de tamanho $n = 15$ na iteração $i = 6$.



3.3.2.4 Propriedades das Buscas

Importante notar que todos os processos são independentes uns dos outros e diferentes iterações da busca restrita são independentes entre si. Dada a tendência contemporânea de maior disponibilidade de núcleos de processamento, temos que esse desenho permite um uso tanto intensivo quanto extensivo dos recursos computacionais. Além disso,

o uso de memória não escala linearmente com o número de processos simultâneos, dado que não é necessário copiar múltiplas vezes a informação que é compartilhada entre os processos. Como a única informação em que é permitido escrita para qualquer processo é a de caminhos podados e esta informação é usada somente como teste para avaliar ou não um candidato, então uma condição de corrida não levaria a resultados diferentes. Fora isso, não há risco de criar condições de corrida em qualquer uma das outras informações.

Ainda assim, é possível aumentar o uso de memória em troca de um desempenho melhor em tempo computacional. Como tanto Viterbi quanto avanço-retrocesso são algoritmos de programação dinâmica, então é possível usar técnica de *memoize* para armazenar resultados intermediários em comum que seriam calculados diversas vezes nos diferentes processos. Isto faz com que uma série de resultados de cálculos sejam armazenados e somente consultados posteriormente. Como todas as operações realizadas pelas buscas são, no máximo, sequências de multiplicações de matrizes, então em casos de alta regularidade ou que vão se repetir várias vezes, o ganho pode ser significativo. Por exemplo, temos na Equação 3.16 que sucessivos produtos de matrizes T e E são necessários para achar a probabilidade conjunta, mas pela natureza cíclica dos átomos da espinha dorsal estendida temos que alguns produtos aparecem reiteradamente, como $T^{(2)}E^{(r,2)}$, para cada resíduo r no passo anterior. Portanto, o uso de *memoization* permitiria que esses produtos fossem calculados somente uma vez, evitando desperdício de computação. Isto naturalmente satisfaz a terceira condição do ARMBP.

Desta forma, temos que a integração atende 3 das condições estabelecidas no início da seção 3.3. Resta assegurar a última condição, para a qual é necessária a realização de experimentos computacionais.

3.4 Método

Para avaliar a última condição essencial para o ARMBP e julgar o seu desempenho, estabelecemos como critério primário o número de vértices explorados na árvore binária do espaço de soluções no momento em que é identificada a primeira conformação válida. Naturalmente, também temos interesse em mensurar o tempo de execução em segundos (*time*) e a quantidade de memória usada em *bytes* (*space*) em cada instância e suas medidas resumo. Entretanto, é trivialmente verdadeiro que o ARMBP utiliza mais memória do que o BP, então por mais que seja uma medida de interesse, não é um bom critério. A questão é se o acréscimo de memória é relevante ou não para a melhora do desempenho. Assim, definimos como critério secundário somente o tempo de execução.

Adicionalmente, a partir dos dados pré-processados, também podemos verificar a correção do algoritmo ao dar a matriz completa e ver se o resultado gerado é capaz de reproduzi-la com uma tolerância de 10^{-16} na diferença entre as normas. Uma instância

só entrará para a análise se for capaz de reproduzir a matriz completa dada. Caso não seja capaz, também queremos saber se é por algum defeito da matriz ou se por falha do programa. Para tal, em caso de falha, iremos verificar se a matriz completa é uma matriz de distância válida e se a instância apresenta todos os dados de coordenadas dos átomos de interesse. Em todo caso, a matriz reduzida é uma matriz de distâncias parciais em que permaneceram as entrada entre um átomo e seus 3 predecessores na ordenação dada, e, fora essas, aproximadamente 25% das entradas menores ou iguais a 5.

O critério primário é uma informação fácil de obter no BP. Para fins desta análise, vamos utilizar o algoritmo *Symmetry-based Build-up* (SBBU). O SBBU é uma evolução do BP clássico que emprega relações de simetria para identificar uma solução (GONçALVES et al., 2021). O tempo de execução para uma série de instâncias está descrito na literatura usando um computador com poder de processamento relativamente próximo ao dos usados para os experimentos computacionais do ARMBP [10].

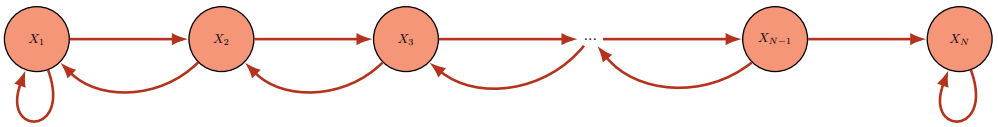
Para o BP, a entrada é uma matriz de distâncias com alguns elementos possivelmente faltantes ou, equivalentemente, uma lista de distância entre os vértices. No caso do ARMBP, a entrada é uma matriz de distâncias com alguns elementos faltantes e a sequência finita dos resíduos. Para manter a paridade de informação sobre as distâncias que os métodos utilizam, vamos manter na matriz somente as distâncias descritas na seção 3.1.2, garantindo as suposições da ordenação do PDGDM. Assim, todos teriam como entrada a mesma matriz em cada instância. Convenientemente, as matrizes já foram produzidas durante o processamento como dados de suporte.

No entanto, não conseguimos reproduzir os segmentos e entradas descritos por Gonçalves et al. e não ficou claro que tipo de pré-processamento foi realizado (GONçALVES et al., 2021). Por exemplo, a entrada do PDB de código 1FW5 é descrita como tendo 60 vértices. Entretanto, tanto no nosso pré-processamento quanto nos próprio meta-dados do PDB, há a informação de que o 1FW5 possui 100 vértices, pois é composta por 20 resíduos[11]. O mesmo problema se apresentou em várias outras entradas, muitas das quais sequer foram selecionadas pelo filtro empregado para obter as entradas da PDB, como descrito em seção 3.1. Devido a este problema, a comparação do tempo de processamento será indireta. Apesar disso, não é um problema significativo, porque o maior interesse é saber se o tempo de ambas está suficientemente próximo. Operacionalmente, estabelecemos que "suficientemente próximo" significa ter, no máximo, uma ordem de grandeza na diferença dos tempos de processamento.

Considere os dados de teste descritos na seção 3.3.1. Selecionamos 45 segmentos, sendo 15 escolhidos aleatoriamente entre instâncias de tamanho 15, entre 20 até 75, e entre 80 até 250. Note que a escolha do tamanho fixo 15 se dá porque é funcionalmente

[10] <https://github.com/michaelsouza/sbbu>
[11] <https://www.rcsb.org/structure/1FW5>

Figura 26 – Representação da cadeia de Markov associada ao tempo de primeira passagem



a menor instância que podemos analisar usando informação do resíduo antecessor. Os demais foram selecionados por disponibilidade dos dados e facilidade de realizar inspeção visual do processo. Adicionalmente, realizamos testes para além de 250 vértices, mas sem acompanhamento e somente para verificar o estresse causado na máquina.

Assim, Para cada segmento e matriz associada, realizamos 100 ensaios com a aplicação do ARMBP. De cada um colhemos o número de vértices explorados, tempo de execução e memória usada. Após percorrer todos os segmentos dos dados de teste, calculamos as medidas resumos e as estatísticas derivadas. Para a comparação do número de vértices visitados, vamos utilizar uma estimativa da média do número de vértices visitados do BP. Para realizar essa estimação, usamos dois métodos. O primeiro é descrito por Marques et. al. e consiste no número de vértices visitados até chegar a uma solução binária específica $\mathbf{b} = (b_1, \ldots, b_n)$ (MARQUES et al., 2024), dada por

$$EDFS(\mathbf{b}) = n + \sum_{i=0}^{n-1} b_{n-i}(2^{i+1} - 1). \tag{3.20}$$

No entanto, este método não considera que existem mais soluções do que a dada por $\mathbf{b}$ e que poderiam ser obtidas antes de visitar essa quantidade de vértices. Assim, consideramos que não seria um bom estimador para o número médio de vértices visitados pelo BP. Alternativamente, desenvolvemos um estimador dado pela média dos tempos de primeira passagem (MTPP) de um sorvedouro em uma cadeia de Markov que simula a estrutura do BP, ou seja, em que este sorvedouro equivale a um vértice-folha na árvore binária do espaço de busca, como na Figura 26. Para levar em conta a incerteza sobre o número de soluções, não simulamos a árvore binária inteira e excluímos os 3 primeiros vértices, fazendo com que o tempo de primeira passagem seja calculado por uma relação de recorrência entre os vértices que realmente "arvoram" e um único sorvedouro, representando a primeira solução dentro do processo recursivo feito pelo BP.

Por construção, a cadeia de Markov que simula o BP é finita com apenas uma classe recorrente e aperiódica, contendo somente o vértice sorvedouro, e uma classe transiente, contendo todos os demais vértices. Além disso, supomos uma probabilidade igual de avanço e retrocesso, embora esse parâmetro deva ser melhor analisado para ajustar a qualidade da estimativa. Dado que o número esperado de vértices visitados no BP

depende da quantidade de arestas, é natural que, para ser bom estimador, a probabilidade de avanço e retrocesso deva depender também do número de arestas. Isto posto, por força do tempo, calculamos de modo mais simples a matriz fundamental N e a multiplicamos por um vetor $\mathbf{1}$, obtendo na primeira entrada do vetor resultante o MTPP partindo do vértice 1 para o sorvedouro.

Para comparar o BP e o ARMBP com respeito ao critério primário, empregamos um teste não-paramétrico pareado. Caso a diferença entre as mensurações dos pares de observação seja aproximadamente simétrica, então aplicamos o teste de postos sinalizados de Wilcoxon. Caso contrário, aplicamos o teste do sinal. De toda forma, a hipótese nula H_0 é de que não há diferença entre as medianas das observações do critério associado. A hipótese alternativa H_1 é de que há diferença significativa entre as medianas das observações do critério associado e é positiva. A hipótese alternativa H_2 é de que há diferença significativa entre as medianas das observações do critério associado e é negativa. Dado o contexto, estabelecemos um nível de significância $\alpha = 0.01$.

O código-fonte do algoritmo e de todas as etapas da análise está disponível no repositório <https://github.com/fausto-mpj/ARMBP>. O hardware empregado foi um Intel i7-7700K com 32GB de RAM no sistema operacional GNU/Linux de 64-bit.

3.5 Resultados

Os dados das 45 instâncias selecionadas e os resultados dos experimentos computacionais estão resumidos nas Tabela 3 e Tabela 4. Em todas as instâncias sorteadas conseguimos achar uma solução e reproduzir com sucesso a matriz de distâncias completa usando o ARMBP. Depois dessa verificação, fizemos 100 ensaios para cada instância, usando tanto a matriz completa quanto a incompleta. Nestes ensaios usamos o ARMBP com as 3 Buscas e um ARMBP somente com a Busca Local.

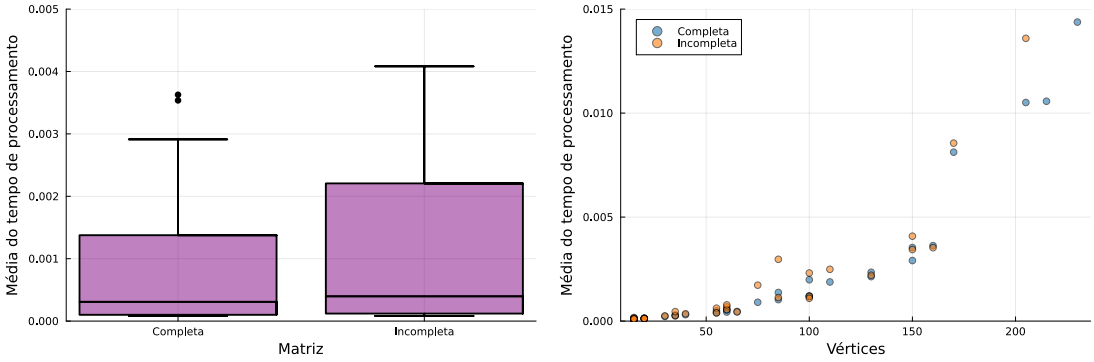
No entanto, omitiremos a análise do ARMBP com a 3 Buscas, porque **todas** as estatísticas ficaram piores que a do ARMBP somente com a Busca Local. Segue que, no restante dessa seção, por ARMBP queremos dizer o ARMBP Local. Abordaremos mais sobre isso na seção 3.6.

Para fins da pesquisa, o principal enfoque é no desempenho do ARMBP com a matriz incompleta. Entretanto, antes de relatar os resultados principais, temos algumas observações interessantes na comparação do ARMBP com as matrizes completa e incompleta. Essas observações auxiliarão na construção do quadro de análise geral da implementação do algoritmo. Uma síntese dessa comparação está apresentada na Figura 27.

Primeiramente, em nenhuma instância o ARMBP visitou mais vértices com a matriz completa em relação à incompleta. No pior caso, ambas visitaram o mesmo número

Figura 27 – Comparação do desempenho usando matrizes completa e incompleta.

(a) *Boxplot.* (b) Gráfico de dispersão.



de vértices, o que ocorreu somente 5 vezes. Dada uma tolerância de 15%, em 26 instâncias a soma dessa tolerância com a razão entre os número de vértices visitados dados a matriz completa e a incompleta foi menor do que 1. Ou seja, em mais da metade das instâncias temos que o número de vértices visitados até a primeira solução é significativamente menor ao usarmos a matriz completa. Isso é totalmente esperado.

Apesar disso, em 11 instâncias o ARMBP com matriz incompleta obteve tempo médio inferior ao do ARMBP com a matriz completa. Dada uma tolerância de 15%, em 15 instâncias a soma dessa tolerância com a razão entre os número de vértices visitados dados a matriz completa e a incompleta foi menor do que 1. Por sua vez, isso indica que somente em um terço dos casos há uma melhora significativa do tempo médio ao usar a matriz completa.

Dito de outra forma, mesmo um aumento significativo do número de vértices visitados no ARMBP não implica que ocorrerá um aumento significativo do tempo médio. Isso está de acordo com o nosso objetivo inicial, dado que idealmente queremos um algoritmo que faça ao mesmo tempo uma busca inteligente e uma busca massiva no espaço de busca. Considerando a presença de *cache* para os cálculos de posicionamento dos vértices, temos que se as buscas retornarem trechos próximos aos já percorridos, então o custo de visitar esses vizinhos é muito baixo.

Por outro lado, se o aumento de vértices visitados estiver associado a mudanças significativas da solução parcial binária, retornando trechos distantes, então o constante cálculo e armazenamento de novos trechos no *cache* naturalmente irão aumentar o tempo de processamento. Isso provavelmente é o que ocorreu na instância 1WQC, como veremos abaixo.

Com respeito ao resultado principal, temos na Tabela 3 as seguintes colunas: **(i)** “Nome” com o ID PDB da entrada; **(ii)** “Ini” com o índice do átomo inicial; **(iii)** “Fin” com o índice do átomo final; **(iv)** “Vis” com o número de vértices visitados pelo ARMBP até obter uma solução; **(v)** “EDFS” com o número de vértices visitados pelo BP para obter

Tabela 3 – Instâncias e resultados computacionais (Parte 1).

Nome	Ini	Fin	Vis	EDFS	MTPP	Aloc	Mem	TGC	Vert	Arst
2DN4	3626	3640	14	6351	156	338	1389320	0.0	15	100
2EKF	1696	1710	14	14804	156	338	1389320	0.0	15	98
2JND	2776	2790	24	6606	156	459	1396312	0.0	15	98
2L6K	461	475	14	6613	156	338	1389320	0.0	15	94
2L6K	7886	7900	70	64715	156	614	1529576	0.0	15	100
2L6N	4411	4425	15	57297	156	360	1390408	0.0	15	90
2MF6	1	15	40	8147	156	583	1408128	0.0	15	102
2QL0	1976	1990	17	14828	156	378	1391672	0.0	15	98
2RLL	16	30	21	6351	156	409	1394152	0.0	15	98
2RR4	5416	5430	26	2561	156	498	1398072	0.0	15	92
5M9Y	156	170	18	6606	156	400	1392712	0.0	15	96
6ZOM	7131	7145	97	14625	156	1032	1552848	0.0	15	96
7JU9	131	145	50	7992	156	612	1410480	0.0	15	94
7TV7	461	475	48	6358	156	638	1414016	0.0	15	92
8DIJ	376	390	22	6351	156	405	1394296	0.0	15	106
2JUY	3221	3240	34	80344	306	607	1800224	0.0	20	130
2L6N	11356	11375	29	$> 10^5$	306	538	1790968	0.0	20	144
2N5W	41	60	24	73726	306	481	1787368	0.0	20	142
2N5W	601	620	26	$> 10^5$	306	493	1789016	0.0	20	136
2G46	12381	12410	75	$> 10^8$	756	1036	3042080	0.0	30	220
2BBU	10916	10950	52	$> 10^9$	1056	891	4188952	0.0	35	256
2NPV	71	105	121	$> 10^9$	1056	1182	4488008	0.0	35	332
1JLP	1206	1245	71	$> 10^{11}$	1406	1137	5404816	0.0	40	330
2N08	1046	1100	140	$> 10^{13}$	2756	1717	11532680	0.0	55	506
2N9A	1046	1100	79	$> 10^{15}$	2756	1305	11439336	0.0	55	482
6N68	1021	1080	128	$> 10^{17}$	3306	1954	14898624	0.0	60	552
7TVQ	301	360	97	$> 10^{17}$	3306	1577	14814288	0.0	60	488
8DGH	2741	2800	151	$> 10^{17}$	3306	1966	14889360	0.0	60	558
1QKL	1621	1685	83	$> 10^{18}$	3906	1491	18803176	0.0	65	658
6M19	676	750	254	*Overflow*	5256	3021	27199128	0.0	75	752
1NPQ	5266	5350	176	*Overflow*	6806	2553	37094456	0.0	85	752
1PAO	2041	2125	405	*Overflow*	6806	4094	39896920	0.0	85	722
1F0D	301	400	141	*Overflow*	9506	2313	58487536	0.0	100	926
1FW5	1	100	261	$> 10^{18}$	9506	3216	58730944	0.0	100	1004
5ZYX	1601	1700	146	*Overflow*	9506	2435	58550960	0.0	100	968
5JTP	70361	70470	290	*Overflow*	11556	3685	78596872	0.0	110	854
1WQC	1561	1690	33098	$> 10^{18}$	16256	265946	216098048	0.0	130	1220
2K6T	571	700	212	*Overflow*	16256	3226	124907000	0.0	130	1198
2AIY	15151	15300	304	$> 10^{18}$	21756	4565	187858824	0.0	150	1400
6YHI	1801	1950	255	*Overflow*	21756	4072	183755976	0.0	150	1456
6OFA	5921	6080	282	*Overflow*	24806	4195	223219720	0.0	160	1562
2DCO	681	850	434	$> 10^{18}$	28056	5840	275512312	0.0	170	1656
1HY9	1026	1230	461	*Overflow*	41006	6302	465315440	$> 10^{-3}$	205	1910
2N4K	1	215	1466	*Overflow*	45156	13982	534080504	$> 10^{-3}$	215	2010
2W9O	461	690	697	$> 10^{18}$	51756	8902	649036088	$> 10^{-3}$	230	2104

a mesma solução; **(vi)** "MTPP" com a estimativa para o número de vértices visitados pelo BP para obter uma solução qualquer; **(vii)** "Aloc" com a quantidade de alocações na *heap* para obter a solução; **(viii)** "Mem" com a quantidade de memória usada em *bytes*; **(ix)** "TGC" com o tempo médio usado pelo *garbage collector* em segundos; **(x)** "Vert" com o número de vértices; e **(xi)** "Arst" com o número de arestas.

Subsidiariamente, é interessante notar que o *garbage collection* só foi utilizado

Tabela 4 – Instâncias e resultados computacionais (Parte 2).

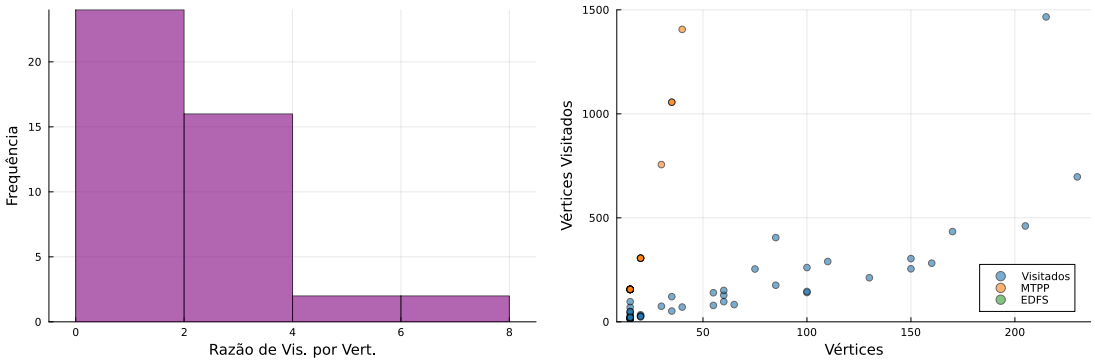
Nome	Média	Mediana	Mínimo	Máximo	Desvio-padrão	Vert	Arst
2DN4	0.00008331	0.00007851	0.00007370	0.00012078	0.00001077	15	100
2EKF	0.00008327	0.00007900	0.00007372	0.00011732	0.00001018	15	98
2JND	0.00010148	0.00009551	0.00008971	0.00014599	0.00001300	15	98
2L6K	0.00008327	0.00007824	0.00007412	0.00011641	0.00000962	15	94
2L6K	0.00013331	0.00012483	0.00011950	0.00020243	0.00001843	15	100
2L6N	0.00009160	0.00008556	0.00008037	0.00013229	0.00001197	15	90
2MF6	0.00012648	0.00011592	0.00010706	0.00019700	0.00001977	15	102
2QL0	0.00009265	0.00008613	0.00008202	0.00012897	0.00001288	15	98
2RLL	0.00009821	0.00009097	0.00008615	0.00014235	0.00001412	15	98
2RR4	0.00010291	0.00009709	0.00009387	0.00014364	0.00001292	15	92
5M9Y	0.00009629	0.00008937	0.00008506	0.00013141	0.00001249	15	96
6ZOM	0.00017249	0.00016239	0.00015455	0.00024319	0.00002158	15	96
7JU9	0.00012008	0.00011395	0.00010941	0.00017309	0.00001322	15	94
7TV7	0.00012192	0.00011336	0.00010879	0.00019197	0.00001736	15	92
8DIJ	0.00009707	0.00009106	0.00008609	0.00013236	0.00001189	15	106
2JUY	0.00014175	0.00013021	0.00012384	0.00021747	0.00002230	20	130
2L6N	0.00012767	0.00011784	0.00011043	0.00018295	0.00001795	20	144
2N5W	0.00012103	0.00011273	0.00010641	0.00017272	0.00001740	20	142
2N5W	0.00012087	0.00011281	0.00010747	0.00016712	0.00001465	20	136
2G46	0.00024915	0.00023404	0.00022488	0.00037038	0.00003388	30	220
2BBU	0.00025311	0.00024099	0.00022856	0.00033035	0.00002574	35	256
2NPV	0.00045501	0.00042769	0.00040899	0.00061529	0.00005496	35	332
1JLP	0.00035172	0.00033382	0.00031383	0.00049814	0.00004155	40	330
2N08	0.00062049	0.00059901	0.00057967	0.00080742	0.00004690	55	506
2N9A	0.00039675	0.00037332	0.00035854	0.00052912	0.00004376	55	482
6N68	0.00067992	0.00064735	0.00062600	0.00091611	0.00006462	60	552
7TVQ	0.00055223	0.00052482	0.00049565	0.00070173	0.00005584	60	488
8DGH	0.00077960	0.00074184	0.00071434	0.00110496	0.00008294	60	558
1QKL	0.00044116	0.00041733	0.00039492	0.00059334	0.00004979	65	658
6M19	0.00172757	0.00168052	0.00152767	0.00261731	0.00019588	75	752
1NPQ	0.00113107	0.00109272	0.00104730	0.00140345	0.00008908	85	752
1PAO	0.00297053	0.00296496	0.00277824	0.00330810	0.00013111	85	722
1F0D	0.00118877	0.00116691	0.00110637	0.00138117	0.00007436	100	926
1FW5	0.00231322	0.00230380	0.00214165	0.00265519	0.00011873	100	1004
5ZYX	0.00109601	0.00106126	0.00102203	0.00133511	0.00007865	100	968
5JTP	0.00248965	0.00247138	0.00227858	0.00293414	0.00014348	110	854
1WQC	0.60486486	0.60335915	0.58496582	0.63636592	0.00922196	130	1220
2K6T	0.00220642	0.00218065	0.00204106	0.00256336	0.00011037	130	1198
2AIY	0.00408229	0.00402784	0.00379719	0.00500197	0.00019272	150	1400
6YHI	0.00343929	0.00341221	0.00317669	0.00393744	0.00016528	150	1456
6OFA	0.00352938	0.00350276	0.00331503	0.00390148	0.00012966	160	1562
2DCO	0.00855641	0.00855722	0.00812069	0.00899320	0.00019824	170	1656
1HY9	0.01359557	0.01354797	0.01283399	0.01439992	0.00034090	205	1910
2N4K	0.09153397	0.09141914	0.08770423	0.09521918	0.00162872	215	2010
2W9O	0.02822686	0.02825941	0.02669507	0.02948377	0.00055877	230	2104

em 1HY9, 2N4K, e 2W9O. Ou seja, nas instâncias acima de 170 vértices. Não é imediatamente óbvio o motivo do *garbage collector* ter ficado mais agressivo nestas instâncias maiores, ainda mais dado que somente na 2N4K as alocações ficaram muito acima do esperado. Por outro lado, 1WQC apresentou o maior número de alocações e ainda assim não teve a ativação de *garbage collection*. Na seção 3.6 abordamos um pouco o que isso pode significar para essa implementação do ARMBP.

De início, podemos notar que o EDFS não parece oferecer uma boa estimativa para o número médio de vértices visitados pelo BP até achar uma solução. Apesar de ser um número relacionado, não parece justo usá-lo para fins de comparação entre ARMBP e BP. De fato, o EDFS cresce tão rápido que em mais da metade das instâncias com 75 vértices ou mais tivemos *overflow* e não é possível nem ver os pontos associados na Figura 28. Mesmo se uma iteração custasse somente uma operação de ponto flutuante, esses números são tão grandes que nenhum processador seria capaz de fazer essa quantidade de operações em uma fração de um segundo. Por outro lado, o MTPP pareceu ao menos razoável, feitas as ressalvas anteriores expostas no seção 3.4. Segue que iremos usar o MTPP para avaliar o primeiro critério.

Figura 28 – Comparação do número de vértices visitados.

(a) Histograma. (b) Gráfico de dispersão.



É arguível que, por mera inspeção visual da Tabela 3 e Figura 28, a execução de um teste estatístico para a diferença das medidas dos visitados para os esperados seria desnecessário. No entanto, vamos proceder como fora descrito no seção 3.4. Sejam X_i a i-ésima entrada de "Vis" e Y_i a i-ésima entrada de "MTPP", então definimos $Z_i = X_i - Y_i$. Como Z definitivamente não é aproximadamente simétrica, bastando notar que a média está muito longe da mediana, então vamos usar o teste do sinal (aproximado) pareado para verificar se X e Y apresentam uma diferença significativa. Para tal, estabelecemos como H_0 a mediana de Z ser igual a 0. Equivalentemente, isso significa que $P(X < Y) = \frac{1}{2}$, ou seja, que X e Y têm igual probabilidade de serem maiores um que outro. Como hipótese alternativa H_1 temos a mediana de Z ser diferente que 0.

Dado o que foi observado na Tabela 3 e na Figura 28, sabemos que ser diferente de 0 neste caso é, para todos os fins, equivalente a ser menor do que 0. Para o nível de significância α escolhemos 0.01. A Tabela 5 apresenta as conclusões do teste do sinal. Assim, rejeitamos a hipótese nula e temos bons motivos para acreditar que o número de vértices visitados pelo ARMBP é menor do que o número esperado de vértices visitados no BP, se for o caso de MTPP for uma boa estimativa.

Por fim de justiça e para nos certificar de que estamos fazendo algo razoável,

Tabela 5 – Teste do Sinal (Aproximado).

Observações	Est. Pontual	IC 99%	p-valor (bilateral)
45	-1004	$(-5002, -142)$	$< 10^{-10}$

também consideramos que o "MTPP" está errado por um fator de 2, então dividimos o "MTPP" por 2 e testamos novamente. Isso não mudou o resultado do teste: o intervalo de confiança permaneceu estritamente negativo, alcançando $(-2374.0, -64.0)$, e o p-valor continuou na mesma escala de grandeza.

Com respeito ao segundo critério, a Tabela 4 apresenta as mesmas instâncias na mesma ordem da Tabela 3, mas com algumas colunas adicionais com os resultados relacionados ao tempo de processamento. Importante frisar que, nas instâncias testadas até 120 vértices, o SBBU apresentou um tempo de execução na escala de 10^{-5} segundos. Já acima de 120 e até 1400 vértices, o tempo de execução ficou na escala de 10^{-4} segundos. Os resultados do SBBU estão expostos na Tabela 6. Estes testes foram feitos em um computador equiparável ao usado para testar o ARMBP em termos de capacidade de computação. No caso do SBBU, também incluímos na tabela duas medidas que não empregamos nas nossas análises, mas que foram originalmente calculadas pelos autores: *Mean Distance Error* (MDE) e *Largest Distance Error* (LDE).

Tabela 6 – Instâncias e resultados computacionais do SBBU.

Nome	Vert	Arst	Tempo	MDE	LDE
1N6T	30	176	0.0001355	5.64447e-12	3.52769e-10
1FW5	60	417	0.0003331	2.47589e-12	1.05505e-10
1ADX	120	659	0.0004194	2.92542e-12	5.90388e-10
1BDO	241	1345	0.0008558	8.48612e-12	6.95429e-10
1ALL	480	3443	0.0026793	1.23433e-12	2.98227e-09
6S61	522	3699	0.0028595	2.96239e-12	1.33815e-09
1FHL	1002	6378	0.0045314	1.02105e-11	3.98329e-09
4WUA	1033	6506	0.0045645	2.91208e-12	1.03046e-09
6CZF	1494	9223	0.0063527	2.35117e-12	1.18549e-09
5IJN	1950	11981	0.0083134	7.24299e-12	4.47862e-09
6RN2	2052	13710	0.0101728	6.21599e-12	2.63864e-09
1CZA	2694	17451	0.0126060	3.21992e-11	1.96634e-08
6BCO	2856	18604	0.0135202	8.94579e-12	2.50133e-08
1EPW	3861	23191	0.0157351	9.52218e-11	4.81396e-08
5NUG	8760	56979	0.0413242	1.77840e-10	1.79760e-06
4RH7	9015	59346	0.0433663	8.69409e-11	6.17915e-07
5NP0	7584	59478	0.0473714	4.37042e-11	2.25869e-07
3VKH	9126	59592	0.0676699	1.71725e-09	1.22824e-05

Fonte: <https://github.com/michaelsouza/sbbu>

Comparadamente, o ARMBP conseguiu manter o tempo na escala de 10^{-5} segundos em instâncias até 65 vértices, incluindo algumas na escala de grandeza de 10^{-6}. A presença destes casos extremos foi uma surpresa, porque não tínhamos certeza se seria possível competir com o BP em instâncias pequenas, dado que o ARMBP tem um custo inicial elevado quando comparado à simplicidade elegante do BP. Acima de 65 e abaixo de

200, exceto em um caso patológico envolvendo a instância 1WQC, conseguimos manter o tempo na escala de 10^{-4} segundos, ainda dentro da diferença na escala de grandezas menor ou igual a 1. No entanto, acima de 200 vértices, vemos um aumento significativo na quantidade de memória, tempo de *garbage collection* e alocações na *heap*, elevando o tempo de processamento para além de 10^{-3} segundos. Essas alterações nos casos acima de 200 vértices indicam problemas na implementação que serão discutidos a seguir.

3.6 Discussão

Apesar das limitações de caráter técnico no desenvolvimento da capacidade de paralelismo do *software* desenvolvido, é possível arguir que a incorporação de informação no BP foi bem-sucedida. Na seção 3.5 vimos que o tempo de processamento das instâncias analisadas se manteve majoritariamente dentro da escala de magnitude do BP, ao mesmo tempo em que detectamos uma redução no número esperado de vértices visitados até encontrar a primeira solução na Busca Local. Isto é, a incorporação da informação sobre a sequência de aminoácidos que compõe cada instância ao modelo não implicou em uma deterioração do tempo necessário para obter uma solução, ao menos no caso da Busca Local.

Para as duas outras buscas propostas, tivemos um resultado positivo em um aspecto específico: ambas funcionaram para aumentar significativamente o número de vértices processados por unidade de tempo. Ao contrário da Busca Local, em que queremos visitar o menor número de vértices por instância, as demais buscas tinham como objetivo explorar de modo inteligente o espaço de busca para eliminar por meio de podas os caminhos globalmente mais prováveis que não são viáveis. A expectativa era que isso acelerasse o programa ao produzir muitos caminhos que levassem a podas e tivessem como guiar a Busca Local para caminhos com alta probabilidade local e global de acordo com o modelo.

No entanto, ao colocar os 3 processos rodando simultaneamente, tivemos um *overhead* devido à necessidade de coordenação e manejo da memória compartilhada entre os processos para não causar condições de corrida. Dada a escala de tempo, o *overhead* foi significativo, aumentando a ordem de magnitude do tempo de processamento e do espaço em memória das instâncias analisadas. Devido a esse aumento, encontramos efeitos deletérios em instâncias de tamanho maior ou igual a 1000, como, por exemplo, a ocorrência de erros do tipo *out of memory* (OOM). Isso inviabilizou a aplicação do programa às instâncias destas ordens maiores.

Assim, a incorporação das duas buscas globais não foi bem-sucedida. De fato, nas instâncias testadas, somente em um único caso a Busca Irrestrita conseguiu processar um número significativo de caminhos antes da Busca Local retornar uma solução, e mesmo

assim o número de vértices visitados pela Busca Local não foi inferior ao do obtido quando usado somente a Busca Local – sem a Busca Irrestrita ou a Busca Restrita. Em outras palavras, as duas buscas demoraram tanto devido ao *overhead* que não ajudaram a Busca Local ao reduzir o número de vértices visitados e ainda causaram uma maior lentidão para evitar conflitos de acesso à memória compartilhada.

Apesar disso, ainda há boas razões para acreditar que é tecnicamente viável fazer com que as buscas funcionem para atender o objetivo proposto, mas por outra arquitetura ou implementação. Não identificamos nas análises qualquer motivo para pensar que a abordagem multi-processamento não é viável, mas humildemente reconhecemos que programação multi-processamento é mais difícil e complexa. Por exemplo, encontramos inicialmente problemas de sobreatribuição de tarefas devido ao modo como o *Julia* interage com bibliotecas de sub-rotinas de álgebra linear, como BLAS e LAPACK. Mesmo quando resolvido, o resultado incorreu em perdas de desempenho em algumas tarefas. Como apontamos anteriormente, a abordagem serial do BP é um dos seus pontos fracos. Entretanto, não basta saber disso para conseguir explorar essa fraqueza. Como apresentado, uma aplicação multi-processamento ingênua pode apresentar pior desempenho do que uma serial.

Os resultados computacionais também apresentam algumas anomalias no uso de memória que sugerem a necessidade de mudar a arquitetura do programa ou, ao menos, a estrutura de dados de alguns trechos que são responsáveis por parte significativa do tempo de processamento. Como apontado anteriormente, a o *garbage collection* agressivo em instâncias maiores, em especial quando há poucas arestas, indica que há um excesso de alocações na *heap* que são derreferenciadas à medida em que o programa itera sobre os posicionamentos dos vértices e os seus binários associados. Idealmente, queremos colocar no *stack* ou pré-alocar estes objetos na *heap* e evitar que exijam mais espaço do que o já previamente alocado, minimizando a quantidade de *syscall* necessárias.

Como boa parte do código foi produzido pensando nisto, então o mais provável é que exista no código alguma instabilidade do tipo, de modo que uma variável inicialmente com o tipo e tamanho conhecidos pelo compilador para ser alocada no *stack* perde essa propriedade sob algumas condições e acaba sendo alocada na *heap*, ou aumento inesperado do tamanho dos objetos em memória para além do que foi pré-alocado. Fora isso, não tivemos qualquer motivo para suspeitar de *memory leak* ou outro erro no manejo da memória.

De modo central, os resultados obtidos apontam para a possibilidade de uma maior incorporação de informações no modelo. É possível usar informação para explorar melhor o espaço de busca ao mesmo tempo que se mantém ou ganha desempenho. Em especial, reforçamos que é viável a concepção de um modelo treinado externamente ao programa e incorporado no tempo de execução.

Em contrapartida, temos como lado negativo o aumento do uso de memória. O BP é extremamente elegante no aspecto de uso da memória, tendo uma baixa pegada deste recurso. A proposta que apresentamos mesmo no seu aspecto teórico envolve um uso mais intensivo de memória, e empiricamente isso foi confirmado. No entanto, esse uso de memória mais intensivo não é para crescer descontroladamente. Ou seja, em tese, é escalável. Quando operando de forma correta, o programa estabelece no início a quantidade de memória que será necessária e isso é configurável no código-fonte. Dado o mantra contemporâneo "processamento é caro, espaço é barato", então enxergamos esse caminho como adequado, restando somente a necessidade de uma implementação multi-processamento para adequar um algoritmo de solução do PDGDM ao desenvolvimento atual dos recursos computacionais.

4 CONSIDERAÇÕES FINAIS

A pesquisa se iniciou com a pretensão de verificar se é possível incorporar mais informações sobre a molécula de proteína a ser conformada de modo sistemático e sem prejuízo ao desempenho do BP no PDGDM. Para isso, detectamos alguns pontos em que a estrutura do algoritmo BP pode ser melhorado e elaboramos um modelo probabilístico explícito para quantificar as incertezas associadas à informação que queremos incorporar. Inicialmente, nos restringimos aos dados sobre a sequência de aminoácidos que compõe a proteína alvo. A partir destes elementos, tentamos produzir um *framework* que fosse facilmente expansível e que apresentasse bom desempenho inicial, segundo alguns critérios estabelecidos na seção 3.3 e na seção 3.4.

Os resultados foram apresentados na seção 3.5 e apontam algumas considerações importantes. Primeiramente, entendemos que os resultados nos dão boas razões para acreditar que mais informações podem ser incorporadas em modelos mais sofisticados dentro do BP. De início, a dúvida era se a exploração determinística do espaço de busca pelo BP não seria muito mais rápida que o uso de algum modelo probabilístico pelo fato de ser tão simples e computacionalmente eficiente em calcular cada iteração, que, mesmo dado crescimento exponencial do espaço de busca, não compensaria perder tempo com alternativas mais agressivas de limitar ou guiar a exploração nesse espaço. No entanto, como exposto, conseguimos incorporar um modelo probabilístico que guiou a exploração no espaço de busca, diminuindo o número de vértices visitados e mantendo uma desempenho compatível com o BP, mesmo com uma implementação longe da otimalidade e produzida em uma linguagem de programação cujo desempenho bruto em tempo de processamento é, em média, inferior ao da usada no BP (*Julia versus C*). Neste sentido, consideramos o resultado satisfatório e que o modelo proposto é um candidato *bona fide* a ser um sucessor do BP.

Por outro lado, os resultados também apontaram para uma série de dificuldades em superar o BP. Quando tentamos atacar todas as fraquezas ao mesmo tempo, por meio de um programa multi-processamento com as 3 buscas descritas na subseção 3.3.2, obtivemos um resultado bem abaixo do desempenho do BP e até do que foi obtido quando usamos somente uma das buscas. Múltiplos pontos de falha tornaram a análise dessas falhas em algo substancialmente complexo. Alguns indícios apresentados na seção 3.6 dão caminhos possíveis de serem explorados. Ao todo, além de usarmos múltiplas buscas baseadas nos algoritmos de Viterbi e de avanço-retrocesso, também empregamos *caches* para algumas computações repetitivas e uma nova abordagem para o *backtracking* que utiliza *heaps* e a menor diferença entre as probabilidades locais produzidas pelo algoritmo de avanço-

retrocesso. Não verificamos antes de proceder a análise do programa o quanto algumas destas abordagens não refletiram negativamente até umas com as outras, ou seja, com efeito deletério na interação entre essas múltiplas técnicas. O que se mostrou um erro grosseiro pelos resultados apresentados.

Para o ARMBP ser de fato um sucessor válido ao BP, algumas investigações adicionais são necessárias. A primeira é com respeito à ordem da cadeia de Markov e da autoregressão, porque a abordagem ingênua de aumentar a dimensão das matrizes de transição e emissão como apresentado na subseção 2.3.4 definitivamente não é escalável e apresentaria um gargalo ainda pior do que o observado nas buscas globais pelo Viterbi de Lista. No entanto, ao mesmo tempo, temos a suspeita de que a ordem ideal no caso de conformação de proteínas não seria maior do que 3 ou 4, tendo pouco ganho explicativo para ordens maiores do que isso. É improvável que resíduos tão distantes ainda exerçam tanta influência que não as exercidas por resíduos mais próximos. A segunda é com respeito à comparação com o número esperado de vértices visitados pelo BP, porque nossa estimativa usando MTPP com probabilidade fixa ainda parece prejudicar o BP. Obter critérios e instrumentos para mensurar diferenças consideradas significantes entre algoritmos é sempre algo essencial e gostaríamos que isso estivesse mais padronizado na literatura. A dificuldade de se replicar e comparar os programas se mostrou excessivamente custoso ao longo da pesquisa. Sem dúvida, para termos um sucessor válido ao BP, seja qual for, precisamos primeiro ter modos comuns e acessíveis de comparar entre candidatos as quantidades consideradas de interesse.

Por fim, ressaltamos que o emprego de modelos probabilísticos gráficos de modo mais geral podem ser vias razoáveis de avanço mesmo no caso intervalar, dado o conjunto de restrições topológicas associadas ao problema. Essa classe de modelos é notoriamente complexa e intensiva em uso de recursos computacionais, mas também pode ser extremamente eficiente quando estão presentes certas restrições, como no caso MOM. Assim, considerando o exposto, esperamos que não só o ARMBP seja melhor explorado no futuro, como também que o seu potencial de generalização seja avaliado quanto a viabilidade *de facto* para os problemas de conformação de proteína.

Referências

ABEL, J.; CHAFFEE, J. Existence and uniqueness of gps solutions. *IEEE Transactions on Aerospace and Electronic Systems*, v. 27, n. 6, p. 952–956, 1991. Citado na página 20.

BERMAN, H.; HENRICK, K.; NAKAMURA, H. Announcing the worldwide protein data bank. *Nature Structural & Molecular Biology*, Springer Science and Business Media LLC, v. 10, n. 12, p. 980980, December 2003. ISSN 1545-9985. Disponível em: <http://dx.doi.org/10.1038/nsb1203-980>. Citado na página 56.

BERMAN, H. M. The protein data bank. *Nucleic Acids Research*, Oxford University Press (OUP), v. 28, n. 1, p. 235242, January 2000. ISSN 1362-4962. Disponível em: <http://dx.doi.org/10.1093/nar/28.1.235>. Citado na página 22.

BOOMSMA, W.; MARDIA, K. V.; TAYLOR, C. C.; FERKINGHOFF-BORG, J.; KROGH, A.; HAMELRYCK, T. A generative, probabilistic model of local protein structure. *Proceedings of the National Academy of Sciences*, Proceedings of the National Academy of Sciences, v. 105, n. 26, p. 89328937, July 2008. ISSN 1091-6490. Disponível em: <http://dx.doi.org/10.1073/pnas.0801715105>. Citado na página 83.

DENIS, M. *Space and Spatial Cognition: A Multidisciplinary Perspective*. Routledge, 2017. ISBN 9781315103808. Disponível em: <http://dx.doi.org/10.4324/9781315103808>. Citado na página 20.

FRÜHWIRTH-SCHNATTER, S. *Finite Mixture and Markov Switching Models*. Springer New York, 2006. (Springer Series in Statistics). ISBN 9780387357683. Disponível em: <https://books.google.com.br/books?id=f8KiI7eRjYoC>. Citado na página 39.

GHAHRAMANI, Z. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, World Scientific Pub Co Pte Lt, v. 15, n. 01, p. 942, February 2001. ISSN 1793-6381. Disponível em: <http://dx.doi.org/10.1142/S0218001401000836>. Citado na página 39.

GONçALVES, D. S.; LAVOR, C.; LIBERTI, L.; SOUZA, M. A new algorithm for the ^{K}dmdgp subclass of distance geometry problems with exact distances. *Algorithmica*, Springer Science and Business Media LLC, v. 83, n. 8, p. 24002426, May 2021. ISSN 1432-0541. Disponível em: <http://dx.doi.org/10.1007/s00453-021-00835-6>. Citado na página 90.

GRAMACHO, W.; MUCHERINO, A.; LAVOR, C.; MACULAN, N. A parallel bp algorithm for the discretizable distance geometry problem. In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 2012. v. 5526, p. 17621768. Disponível em: <http://dx.doi.org/10.1109/IPDPSW.2012.218>. Citado na página 35.

GREENER, J. G.; SELVARAJ, J.; WARD, B. J. Biostructures.jl: read, write and manipulate macromolecular structures in julia. *Bioinformatics*, Oxford University Press (OUP), v. 36, n. 14, p. 42064207, May 2020. ISSN 1367-4811. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btaa502>. Citado na página 67.

HäGGSTRöM, O. *Finite Markov Chains and Algorithmic Applications*. [S.l.]: Cambridge University Press, 2002. (London Mathematical Society Student Texts). Citado na página 39.

KOLLER, D.; FRIEDMAN, N. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. [S.l.]: The MIT Press, 2009. ISBN 0262013193. Citado na página 39.

LAVOR, C.; LIBERTI, L. *Um Convite à Geometria de Distâncias*. São Carlos, SP: SBMAC, 2014. v. 71. 57 p. (Notas em Matemática Aplicada, v. 71). ISBN 97885-8215-057-3. Citado 3 vezes nas páginas 20, 30 e 85.

LAVOR, C.; LIBERTI, L.; LODWICK, W. A.; COSTA, T. Mendonça da. Distance geometry and molecular geometry. In: _____. *An Introduction to Distance Geometry applied to Molecular Geometry*. Springer International Publishing, 2017. p. 4147. ISBN 9783319571836. Disponível em: <http://dx.doi.org/10.1007/978-3-319-57183-6_6>. Citado na página 21.

LAVOR, C.; LIBERTI, L.; MUCHERINO, A. The interval branch-and-prune algorithm for the discretizable molecular distance geometry problem with inexact distances. *Journal of Global Optimization*, Springer Science and Business Media LLC, v. 56, n. 3, p. 855871, November 2011. ISSN 1573-2916. Disponível em: <http://dx.doi.org/10.1007/s10898-011-9799-6>. Citado 2 vezes nas páginas 36 e 83.

LENNOX, K. P.; DAHL, D. B.; VANNUCCI, M.; DAY, R.; TSAI, J. W. A dirichlet process mixture of hidden markov models for protein structure prediction. *The Annals of Applied Statistics*, Institute of Mathematical Statistics, v. 4, n. 2, June 2010. ISSN 1932-6157. Disponível em: <http://dx.doi.org/10.1214/09-AOAS296>. Citado na página 83.

LIBERTI, L.; LAVOR, C.; MACULAN, N. A branchandprune algorithm for the molecular distance geometry problem. *International Transactions in Operational Research*, Wiley, v. 15, n. 1, p. 117, January 2008. ISSN 1475-3995. Disponível em: <http://dx.doi.org/10.1111/j.1475-3995.2007.00622.x>. Citado na página 35.

LIBERTI, L.; LAVOR, C.; MACULAN, N.; MUCHERINO, A. Euclidean distance geometry and applications. *SIAM Review*, Society for Industrial & Applied Mathematics (SIAM), v. 56, n. 1, p. 369, January 2014. ISSN 1095-7200. Disponível em: <http://dx.doi.org/10.1137/120875909>. Citado na página 20.

LIBERTI, L.; LAVOR, C.; MUCHERINO, A.; MACULAN, N. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, Wiley, v. 18, n. 1, p. 3351, August 2010. ISSN 0969-6016. Disponível em: <http://dx.doi.org/10.1111/j.1475-3995.2009.00757.x>. Citado na página 31.

MARQUES, R. S.; SOUZA, M.; BATISTA, F.; GONçALVES, M.; LAVOR, C. A probabilistic approach in the search space of the molecular distance geometry problem. *Journal of Chemical Information and Modeling*, American Chemical Society (ACS), v. 65, n. 1, p. 427434, November 2024. ISSN 1549-960X. Disponível em: <http://dx.doi.org/10.1021/acs.jcim.4c00427>. Citado 6 vezes nas páginas 37, 38, 68, 72, 83 e 91.

MUCHERINO, A.; LAVOR, C.; LIBERTI, L.; TALBI, E.-G. A parallel version of the branch & prune algorithm for the molecular distance geometry problem. In: *ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010*. IEEE, 2010. v. 22, p. 16. Disponível em: <http://dx.doi.org/10.1109/AICCSA.2010.5586983>. Citado na página 35.

MURPHY, K. *Machine Learning: A Probabilistic Perspective*. [S.l.]: MIT Press, 2012. (Adaptive Computation and Machine Learning series). ISBN 9780262304320. Citado na página 39.

NASA Science. *The Golden Record Cover*. 2024. Disponível em: <https://science.nasa.gov/mission/voyager/golden-record-cover/>. Acesso em: 02 abr 2025. Citado na página 21.

NORRIS, J. R. *Markov Chains*. [S.l.]: Cambridge University Press, 1997. (Cambridge Series in Statistical and Probabilistic Mathematics). Citado na página 39.

RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, v. 77, n. 2, p. 257–286, 1989. Citado 2 vezes nas páginas 39 e 51.

RAFTERY, A. E. A model for high-order markov chains. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, Oxford University Press (OUP), v. 47, n. 3, p. 528539, July 1985. ISSN 1467-9868. Disponível em: <http://dx.doi.org/10.1111/j.2517-6161.1985.tb01383.x>. Citado na página 47.

ROSE, Y.; DUARTE, J. M.; LOWE, R.; SEGURA, J.; BI, C.; BHIKADIYA, C.; CHEN, L.; ROSE, A. S.; BITTRICH, S.; BURLEY, S. K.; WESTBROOK, J. D. Rcsb protein data bank: Architectural advances towards integrated searching and efficient access to macromolecular structure data from the pdb archive. *Journal of Molecular Biology*, Elsevier BV, v. 433, n. 11, p. 166704, May 2021. ISSN 0022-2836. Disponível em: <http://dx.doi.org/10.1016/j.jmb.2020.11.003>. Citado 2 vezes nas páginas 59 e 66.

THOMPSON, H. B. Calculation of cartesian coordinates and their derivatives from internal molecular coordinates. *The Journal of Chemical Physics*, AIP Publishing, v. 47, n. 9, p. 34073410, November 1967. ISSN 1089-7690. Disponível em: <http://dx.doi.org/10.1063/1.1712406>. Citado na página 33.

ZUANETTI, D. A.; MILAN, L. A. Second-order autoregressive hidden markov model. *Brazilian Journal of Probability and Statistics*, Institute of Mathematical Statistics, v. 31, n. 3, August 2017. ISSN 0103-0752. Disponível em: <http://dx.doi.org/10.1214/16-BJPS328>. Citado na página 39.

ZUCCHINI, W.; MACDONALD, I. L.; LANGROCK, R. *Hidden Markov Models for Time Series: An Introduction Using R*. Chapman and Hall/CRC, 2017. ISBN 9781315372488. Disponível em: <http://dx.doi.org/10.1201/b20790>. Citado na página 39.