UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

RÔMULO DA SILVA MARQUES

Um algoritmo polinomial para ordenação de vértices em grafos de proteínas

Campinas

2020

Rômulo da Silva Marques

Um algoritmo polinomial para ordenação de vértices em grafos de proteínas

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

Orientador: Carlile Campos Lavor

Este exemplar corresponde à versão final da Dissertação defendida pelo aluno Rômulo da Silva Marques e orientada pelo Prof. Dr. Carlile Campos Lavor.

Campinas

2020

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

M348a

Marques, Rômulo da Silva, 1993-
Um algoritmo polinomial para ordenação de vértices em grafos de proteínas / Rômulo da Silva Marques. – Campinas, SP : [s.n.], 2020.

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

1. Geometria de distâncias. 2. Prolina. I. Lavor, Carlile Campos, 1968-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: A polynomial algorithm for vertex order in protein graphs
Palavras-chave em inglês:
Distance geometry
Proline
Área de concentração: Matemática Aplicada
Titulação: Mestre em Matemática Aplicada
Banca examinadora:
Carlile Campos Lavor [Orientador]
Paulo José da Silva e Silva
Jayme Luiz Szwarcfiter
Data de defesa: 15-09-2020
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-4321-0393
- Currículo Lattes do autor: http://lattes.cnpq.br/7923789778633214

Dissertação de Mestrado defendida em 15 de setembro de 2020 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). CARLILE CAMPOS LAVOR

Prof(a). Dr(a). PAULO JOSÉ DA SILVA E SILVA

Prof(a). Dr(a). JAYME LUIZ SZWARCFITER

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.

Agradecimentos

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

Aos meus pais, **Selmar** e **Rui**, por me ensinarem a ser forte, mesmo quando de encontro a todas as probabilidades.

À minha grande amiga, **Amanda Dias**, por existir e por compartilhar comigo a benção que é a sua vida.

Ao professor **Airton Cleyton**, grande amigo, por mostrar-me o quão interessante a ciência pode ser, especialmente a matemática. Por ajudar-me a levantar quando caí, por ver bondade em mim.

Ao meu orientador, **Carlile Lavor**, por guiar e lapidar este trabalho. Por mostrar-me uma nova faceta do poder da escrita e a influência da imaginação humana na "criação"da matemática.

Ao meu mais novo amigo, **Gerson Gaeta Rosa**, por receber-me em sua casa antes mesmo de conhecer-me, e por viver comigo tantos momentos calorosos, sejam simples, como um jantar de segunda-feira, ou sejam especiais, como o aniversário de seus pais e a defesa desta dissertação.

Ao professor **Michael Souza**, também um amigo, por diversas discussões que levaram a melhoria e expansão deste trabalho.

A **Francisco Bruno**, amigo do peito, por apontar-me o dedo e dizer-me: "bora pra Unicamp!". Por dar-me um lar enquanto cursava a minha primeira disciplina de verão.

Resumo

Seja $G = (V, A, d)$ um grafo simples, não direcionado e de pesos não negativos nas arestas, e que possui uma ordenação dos vértices cujos quartetos de elementos consecutivos são 4-cliques. O *Problema Discretizável de Geometria de Distâncias Moleculares Tridimensional* (^3PDGDM) consiste em associar coordenadas do $\mathbb{R}^3$ a cada um dos vértices de G que respeitem os valores de d. Uma ^3PDGDM *re-ordem* é uma sequência de vértices similar à ordenação anterior, mas que permite a repetição de elementos sob certas circunstâncias. Encontrar uma ^3PDGDM re-ordem é um problema que pode ser resolvido com custo computacional polinomial de $O(|V|^6)$. Esta dissertação apresenta um método polinomial, fazendo uso da Busca em Profundidade, para determinar uma ^3PDGDM re-ordem com custo de $O(|V|^4 \log_2^{|V|})$. Obtido um grafo $G_3 = (V_3, A_3)$ derivado de $G = (V, A)$, constrói-se uma re-ordem de G a partir de um passeio γ em G_3 que atinge todos os vértices de G. Se uma re-ordem não é encontrada, significa que não existem re-ordens para o problema. O algoritmo é implementado na linguagem de programação *C++* e testado em instâncias de moléculas de proteínas presentes no *RCSB Protein Data Bank* (RCSB PDB). Os arquivos do PDB são lidos por meio de funções da biblioteca *Protein Dynamics and Sequence Analysis* (ProDy). Com o objetivo de diminuir o tempo de execução do método, propõe-se uma modificação no algoritmo que leva em consideração características particulares das proteínas. Os resultados analisados são de dois tipos: tempo de execução e obtenção ou não de re-ordem. Ambos os algoritmos foram testados. Quanto a performance de tempo, a primeira versão do algoritmo não terminou a execução dos casos que possuem mais de 180 resíduos. Por outro lado, a versão modificada resolveu todos os casos, gastando não mais que alguns segundos. Quanto a existência de re-ordens, em apenas duas das instâncias houve resultado positivo. Observaram-se evidências de que um tipo específico de aminoácido pode estar causando a inexistência de re-ordens: a prolina.

Palavras-chave: re-ordem. geometria de distâncias. prolina.

Abstract

Let $G = (V, E, d)$ be a simple, non-directed and non-negatively weighted graph, which has a vertex order whose quartets of consecutive vertices are 4-cliques. The *Discretable Molecular Distance Geometry Problem in* $\mathbb{R}^3$ (^3DMDGP) consists of associating a point of $\mathbb{R}^3$ to each vertice of G in a way that all the values of d are respected. A ^3DMDGP *re-order* is a vertex sequence similar to the earlier order, but it allows the repetition of vertices under certain conditions. Finding a ^3DMDGP re-order is a problem that can be solved with worst-case time complexity of $O(|V|^6)$. This dissertation presents a method, that utilizes *Depth First Search*, to determine a ^3DMDGP re-order with the cost of $O(|V|^4 \log_2^{|V|})$. Firstly, it generates a graph $G_3 = (V_3, E_3)$ derived from $G = (V, E)$, then constructs a re-order of G from a walk γ in G_3 which reaches all vertices of G. If a re-order is not found, it means that there are no re-orders in that problem. The algorithm is implemented in the *C++* programming language and tested on instances of protein molecules of *RCSB Protein Data Bank* (RCSB PDB). PDB files are processed using functions from the *Protein Dynamics and Sequence Analysis* (ProDy) library. In order to decrease the execution time, a modification in the algorithm is proposed, taking into account the particular characteristics of the proteins. The execution time and the achievement or not of a re-order are the parameters taken as results. Both algorithms have been tested. As for time performance, the first version of the algorithm did not end execution in cases that have more than 180 amino acid residues. On the other hand, the modified version solved all cases, spending no more than a few seconds. As for the existence of re-orders, only two of the instances had a *True* output. It was observed some evidence that a specific type of amino acid may be causing the absence of re-orders: the proline.

Keywords: re-order. distance geometry. proline.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BP	Branch-and-Prune.
GD	Geometria de Distâncias.
^{3}PGD	Problema de Geometria de Distâncias Tridimensional.
^{K}PGD	Problema de Geometria de Distâncias K-Dimensional.
$^{3}PDGDM$	Problema Discretizável de Geometria de Distâncias Moleculares Tridimensional.
$^{K}PDGDM$	Problema Discretizável de Geometria de Distâncias Moleculares K-Dimensional.

Lista de Algoritmos

Sumário

Introdução

A *Geometria de Distâncias* (GD) é uma parte da matemática que investiga as propriedades de um espaço que são oriundas do conceito de distância. A GD passa a ser considerada como uma nova área do conhecimento, contemporaneamente, a partir dos resultados de Blumenthal (1970). No entanto, os estudos a seu respeito remontam à Grécia Antiga. Lá, o foco estava nos objetos geométricos clássicos: pontos, retas, planos, círculos, poliedros. Herão de Alexandria (por volta de 50 d.C.), por exemplo, desenvolveu uma maneira de calcular a área de um triângulo por meio dos comprimentos de seus lados (LIBERTI; LAVOR, 2016).

Saltando cerca de 2000 anos para frente, Arthur Cayley publica em 1841, ainda como graduando, um artigo que serviria de base para os trabalhos de um outro importante geômetra: Karl Menger. Cayley provou que o volume 4-dimensional de um *4-simplex* imerso no espaço tridimensional é zero (LIBERTI; LAVOR, 2016). O *K-simplex* é o sólido do espaço K-dimensional que corresponde ao triângulo, no espaço bidimensional, e ao tetraedro, no espaço tridimensional. Um K-simplex pode ser formalmente definido da seguinte forma (BLUMENTHAL, 1970; BAZARAA; JARVIS; SHERALI, 2010):

Definição 1. *Sejam x_0, x_1, ..., x_K K+1 pontos do espaço K-dimensional que são não-coplanares neste mesmo espaço. Então, o conjunto S_k é chamado de K-simplex:*

$$S_k = \left\{ \alpha_0 x_0 + \alpha_1 x_1 + \ldots \alpha_K x_K \;\middle|\; x_i \geqslant 0, i = 1, 2, \ldots, K, \text{ e } \sum_{i=0}^{K} \alpha i = 1 \right\} \tag{1}$$

O teorema que apresenta aquele resultado faz uso do determinante que viria a ser conhecido como o *determinante de Cayley-Menger* (LIBERTI; LAVOR, 2016).

Em 1928, Karl Menger expande a ideia por trás do determinante de Cayley e contribui para a generalização do cálculo do volume de um K-simplex, utilizando somente as distâncias entre seus vértices (LIBERTI; LAVOR, 2016). A construção da fórmula para computar este volume está presente em Blumenthal (1970, p. 97-98):

Definição 2. *Sejam x_0, x_1, ..., x_K K+1 pontos do $\mathbb{R}^K$ e D a matriz simétrica de dimensão $K+1$, cuja componente (i, j) guarda a distância entre x_i e x_j ao quadrado, ou seja, $\|x_i - x_j\|^2$. Seja D' a matriz simétrica de dimensão K+2 que corresponde à matriz D cercada pelos vetores $[0, 1, \ldots, 1]^T$ e $[0, 1, \ldots, 1]$, respectivamente, pela esquerda e por*

cima. Explicitamente:

$$D' = \begin{bmatrix} 0 & 1 & 1 & 1 & \dots & 1 & \dots & 1 & \dots & 1 \\ 1 & 0 & d_{x_0,x_1} & d_{x_0,x_2} & \dots & d_{x_0,x_i} & \dots & d_{x_0,x_j} & \dots & d_{x_0,x_K} \\ 1 & d_{x_1,x_0} & 0 & d_{x_1,x_2} & \dots & d_{x_1,x_i} & \dots & d_{x_1,x_j} & \dots & d_{x_1,x_K} \\ 1 & d_{x_2,x_0} & d_{x_2,x_1} & 0 & \dots & d_{x_2,x_i} & \dots & d_{x_2,x_j} & \dots & d_{x_2,x_K} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & d_{x_i,x_0} & d_{x_i,x_1} & d_{x_i,x_2} & \dots & 0 & \dots & d_{x_i,x_j} & \dots & d_{x_i,x_K} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & d_{x_j,x_0} & d_{x_j,x_1} & d_{x_j,x_2} & \dots & d_{x_j,x_i} & \dots & 0 & \dots & d_{x_j,x_K} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & d_{x_K,x_0} & d_{x_K,x_1} & d_{x_K,x_2} & \dots & d_{x_K,x_i} & \dots & d_{x_K,x_j} & \dots & 0 \end{bmatrix}. \tag{2}$$

O volume K-dimensional do K-simplex, cujos vértices são x_0, x_1, ..., x_K, *é dado por:*

$$V_K(x_0, x_1, \dots, x_K) = \sqrt{\frac{(-1)^{K+1}}{2^K (K!)^2} |D'|}. \tag{3}$$

O símbolo $|D'|$ *representa o determinante de* D', *que, por sua vez, corresponde ao determinante de Cayley-Menger para o caso K-dimensional.*

Atualmente, a GD está presente nos mais variados campos da ciência (LIBERTI et al., 2014a): astronomia (LINDEGREN et al., 2012; SANTIAGO et al., 2018), robótica (NIELSEN; ROTH, 1999), telecomunicações (ANDERSON et al., 2009), bioquímica (DONALD, 2011). O último campo é de fundamental importância, já que o desenvolvimento deste trabalho é motivado por aplicações diretas à bioquímica. Aqui, o problema de GD consiste em determinar a estrutura 3D de proteínas, quando são conhecidas apenas algumas distâncias entre pares de átomos.

Mais rigorosamente, pode-se descrever esse problema da seguinte maneira: seja $\{d_{ij} : i, j \in I\}$, $I \subset \{1, \dots, n\} \times \{1, \dots, n\}$, um conjunto de distâncias conhecidas entre n objetos. O **Problema de Geometria de Distâncias Tridimensional** (^3PGD) consiste em determinar um ponto x_i do $\mathbb{R}^3$ para cada objeto de forma que as distâncias conhecidas sejam respeitadas, ou seja:

$$\|x_i - x_j\| = d_{ij}, \forall \{i, j\} \in I. \tag{4}$$

Com o advento da Teoria dos Grafos, área de estudos que se encontra na interseção entre a Matemática e a Computação, torna-se possível redefinir os problemas de Geometria de Distâncias em termos de grafos. Para isso, alguns conceitos devem ser levados em consideração:

- Um *grafo direcionado* $G = (V, A)$ é um par ordenado formado por um conjunto V de objetos, que são chamados de *vértices*, e por um conjunto A de pares ordenados de vértices, que são chamados de *arcos* (BONDY; MURTY, 2008, p. 31).

- Um arco (u, v) retrata uma relação direcionada do vértice u ao vértice v. O grafo $G = (V, A)$ é dito *não direcionado* quando uma relação entre pares de vértices de V é dada sempre em ambas as direções (BONDY; MURTY, 2008, p. 1-2,31). Precisamente,

$$(u, v) \in A \Rightarrow (v, u) \in A. \tag{5}$$

  Neste caso, é conveniente retratar os arcos (u, v) e (v, u) como um único objeto que expresse a relação bidirecional entre u e v. Este objeto é chamado de *aresta* e representado pelo conjunto $\{u, v\}$. Consequentemente, o objeto A em grafos simples e não direcionados é um conjunto de arestas (BONDY; MURTY, 2008, p. 1-2,31).

- G é dito ser *simples* quando não possui arcos formados por um mesmo vértice (BONDY; MURTY, 2008, p. 3-4), ou seja,

$$\forall\, v \in V,\ (v, v) \notin A. \tag{6}$$

- Um grafo é dito *ponderado* quando existe uma função $d : A \to \mathbb{R}_+$ que atribui valores não-negativos às arestas do grafo (BONDY; MURTY, 2008, p. 50).

- Graficamente, os vértices são representados como pontos, enquanto as arestas são representadas como segmentos de reta que conectam vértices (BONDY; MURTY, 2008, p. 1-2).

Estabelecidos os conceitos básicos de grafos, redefine-se o ${}^3PGD$ da seguinte maneira:

Definição 3 (${}^3PGD$)**.** *Seja $G = (V, A, d)$ um grafo simples, não direcionado e ponderado. O* ***Problema de Geometria de Distâncias Tridimensional*** *(${}^3PGD$) consiste em determinar uma função $x : V \to \mathbb{R}^3$ que respeite as seguintes equações:*

$$\|x(u) - x(v)\| = d(u, v),\ \forall \{u, v\} \in A. \tag{7}$$

A função x é dita uma realização do grafo $G = (V, A, d)$ no $\mathbb{R}^3$ (LAVOR; LIBERTI, 2014, p. 7). Visando reduzir a notação, pode-se retratar $x(u)$ por x_u e $d(u, v)$ por d_{uv}.

Deve-se salientar que o ${}^3DGP$ é um problema *NP-difícil* (SAXE, 1979), o que estimula o desenvolvimento de diversas estratégias para resolvê-lo. Tradicionalmente, o

^3PGD é abordado como um problema de otimização global, podendo ser representado pela seguinte formulação matemática (LIBERTI et al., 2014a):

$$\min_{x_u \in \mathbb{R}^3,\, u \in V} \sum_{\{u,v\} \in A} (\|x_u - x_v\|^2 - d_{uv}^2)^2. \tag{8}$$

A Função (8) constitui um problema de programação não-linear sem restrições. Como cada parcela do somatório remete a uma das Equações (7), resolver o problema significa determinar pontos do $\mathbb{R}^3$ que levem a função a zero (LAVOR; LIBERTI, 2014).

Naturalmente, essa maneira de descrever um ^3PGD encoraja a utilização de métodos contínuos para atingir um mínimo global. No entanto, uma vez que as abordagens contínuas dispõem apenas de informações locais (LIBERTI et al., 2014a), não há garantias de se obter mínimos globais.

Os métodos *Multi-Level Single Linkage* (KUCHERENKO; SYTSKO, 2005) e *Variable Neighbourhood Search* (LIBERTI; DRAŽIC, 2005) são exemplos de algoritmos contínuos empregados para resolver ^3PGD associados a moléculas de proteínas (LAVOR; LIBERTI; MACULAN, 2006).

Ainda no contexto de proteínas, deve-se destacar também os métodos contínuos de suavização de funções de várias variáveis baseados no operador *Transformada Gaussiana* (KOSTROWICKI; PIELA, 1991; MORÉ; WU, 1997), visto que foram implementados e testados com sucesso em instâncias do tipo *cubical grid* (LIBERTI et al., 2014a). Esta implementação é denominada *DGSOL* e está disponível gratuitamente em http://www.mcs.anl.gov/ more/dgsol/.

Na contramão dessas abordagens contínuas, Liberti, Lavor e Maculan (2008) apresentam um método discreto para resolver ^3PGD atrelados a proteínas. Este método é chamado de *Branch and Prune* (BP).

O funcionamento do BP está assentado numa ideia simples: é possível *realizar* um vértice **z** de um grafo $G = (V, A, d)$, contanto que se conheçam as realizações de três vértices **u**, **v** e **w** cujas distâncias de **z** a **u**, de **z** a **v** e de **z** a **w** sejam também conhecidas. Esta circunstância é idêntica ao problema de determinar um ponto $x_z = [x_{z1}, x_{z2}, x_{z3}]^T$ do $\mathbb{R}^3$ que pertença à interseção entre as esferas $S(x_u, d_{uz})$, $S(x_v, d_{vz})$ e $S(x_w, d_{wz})$. Em outras palavras, x_z deve ser solução para o seguinte sistema de equações associado a essas três esferas:

$$\begin{cases} \|x_z - x_u\|^2 = d_{uz}^2; \\ \|x_z - x_v\|^2 = d_{vz}^2; \\ \|x_z - x_w\|^2 = d_{wz}^2. \end{cases} \tag{9}$$

A quantidade máxima de soluções para o sistema acima é finita e pequena:

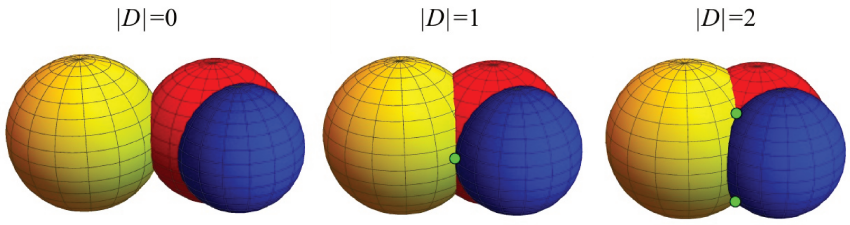
2 soluções. Primeiramente, considere o caso de duas esferas. A interseção destes objetos pode resultar em:

- nenhum ponto, quando as esferas não se tocam;
- um ponto, quando as esferas são tangentes;
- infinitos pontos, quando as esferas são secantes. Neste caso, os pontos da interseção constituem um círculo.

Com a terceira esfera, o número de pontos resultantes da interseção pode ser zero, um ou dois, mas não infinitos. Mais precisamente, a interseção da terceira esfera com as duas primeiras pode resultar em (Figura 1):

- nenhum ponto, se a terceira esfera não intersecta o círculo oriundo da interseção das primeira e segunda esferas;
- um ponto, se a terceira esfera tangencia o círculo;
- dois pontos, se a terceira esfera é secante ao círculo.

Figura 1 – Interseção entre as esferas $S(x_u, d_{uz})$, $S(x_v, d_{vz})$ e $S(x_w, d_{wz})$.



Fonte – Lavor (2019).

Nota – D é o conjunto de pontos presentes na interseção entre as esferas $S(x_u, d_{uz})$, $S(x_v, d_{vz})$ e $S(x_w, d_{wz})$, ou seja, $D = S(x_u, d_{uz}) \cap S(x_v, d_{vz}) \cap S(x_w, d_{wz})$.

Ao se manipular algebricamente as Equações (9), obtém-se um sistema equivalente cujas soluções podem ser observadas mais facilmente (LAVOR; LIBERTI, 2014, cap. 3.2):

$$\begin{cases} \begin{bmatrix} x_{z1} \\ x_{z2} \end{bmatrix} = A - B[x_{z3}]; \\ \\ \|x_z - x_u\|^2 = d_{uz}^2, \end{cases} \tag{10}$$

tal que

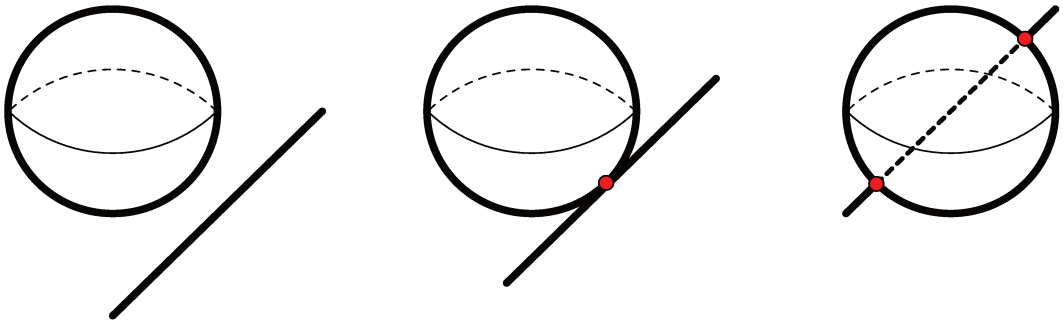
$$A = \frac{1}{2}\begin{bmatrix} x_{v1} - x_{u1} & x_{v2} - x_{u2} \\ x_{w1} - x_{u1} & x_{w2} - x_{u2} \end{bmatrix}^{-1} \begin{bmatrix} \|x_v\|^2 - \|x_u\|^2 + d_{uz}^2 - d_{vz}^2 \\ \|x_w\|^2 - \|x_u\|^2 + d_{uz}^2 - d_{wz}^2 \end{bmatrix},$$

$$B = \begin{bmatrix} x_{v1} - x_{u1} & x_{v2} - x_{u2} \\ x_{w1} - x_{u1} & x_{w2} - x_{u2} \end{bmatrix}^{-1} \begin{bmatrix} x_{v3} - x_{u3} \\ x_{w3} - x_{u3} \end{bmatrix}. \tag{11}$$

Nota-se que, geometricamente, a primeira equação do Sistema (10) corresponde à equação de uma reta. Já se sabe que a segunda equação descreve uma esfera centrada em x_u e de raio d_{uz}. Consequentemente, a solução do Sistema (10) consiste na interseção entre uma reta e uma esfera, que por sua vez corresponde a (Figura 2):

- nenhum ponto, se a reta não intersecta a esfera;
- um ponto, se a reta tangencia a esfera;
- dois pontos, se a reta "perfura"a esfera.

Figura 2 – Interseção entre uma esfera e uma reta.



Fonte – Lavor e Liberti (2014).

A ideia de realizar um vértice fazendo uso de informações de outros três vértices já realizados é expandida no BP, a fim de realizar um grafo $G = (V, A, d)$ integralmente. É por esta razão que o BP tem como pré-requisito uma ordenação dos vértices de G, na qual cada vértice esteja conectado a três vértices anteriores. Além disto, pode-se exigir que estes vértices sejam os três imediatamente anteriores. Assim, o BP procede da seguinte maneira (LIBERTI; LAVOR; MACULAN, 2008):

1. É sempre possível realizar os três primeiros vértices, desde que cada par de vértices esteja conectado por uma aresta e as distâncias d_{v_1,v_2}, d_{v_2,v_3} e d_{v_1,v_3} respeitem a desigualdade triangular estrita;

2. Do quarto em diante, os vértices v_i são realizados em sequência, um de cada vez. De fato, para cada v_i, existem até 2 pontos que podem servir como realização: aqueles provindos da interseção entre as esferas $S(x_{v_{i-3}}, d_{v_{i-3},v_i})$, $S(x_{v_{i-2}}, d_{v_{i-2},v_i})$ e $S(x_{v_{i-1}}, d_{v_{i-1},v_i})$;

3. Se um dado vértice v_i não possui pontos que lhe sirvam como realização, retrocede-se ao mais recente vértice v_j, $j < i$, que possua uma segunda realização. Este outro ponto é escolhido como realização de v_j e o método continua de v_{j+1}.

O problema resolvido pelo BP é, na verdade, um sub-caso do ${}^3PGD$, uma vez que o BP requer uma ordenação para os vértices de $G = (V, A, d)$. Este novo problema recebe o nome de *Problema Discretizável de Geometria de Distâncias Moleculares Tridimensional* (${}^3PDGDM$) e é formalmente definido da seguinte maneira (LAVOR et al., 2012):

Definição 4 (${}^3PDGDM$)**.** *Sejam $G = (V, A, d)$ um grafo simples, não direcionado e ponderado de um ${}^3PGD$, e $v_1, v_2, \ldots, v_{|V|}$ uma ordenação dos vértices de G, tal que*

1. *existe uma realização para v_1, v_2 e v_3;*

2. *para todo $i = 4, \ldots, |V|$, os três vértices **imediatamente** anteriores a v_i são adjacentes a v_i, ou seja,*

$$\{v_{i-3}, v_i\}, \{v_{i-2}, v_i\}, \{v_{i-1}, v_i\} \in A. \tag{12}$$

Além disto, a seguinte desigualdade triangular estrita é satisfeita:

$$d_{v_{i-3},v_{i-2}} + d_{v_{i-2},v_{i-1}} > d_{v_{i-3},v_{i-1}}. \tag{13}$$

*O **Problema Discretizável de Geometria de Distâncias Moleculares Tridimensional** (${}^3PDGDM$) consiste em determinar uma função $x : V \to \mathbb{R}^3$ que respeite:*

$$\|x_{v_i} - x_{v_j}\| = d_{v_i,v_j}, \forall \{v_i, v_j\} \in A.$$

A ordenação de vértices presente na Definição 4 é chamada de ***${}^3PDGDM$ ordem*** (CASSIOLI et al., 2015).

É válido ressaltar que tanto o ${}^3PGD$ quanto o ${}^3PDGDM$ podem ser generalizados para além do caso tridimensional. A seguir, são apresentadas, respectivamente, as definições destes problemas para um inteiro positivo K arbitrário (LIBERTI et al., 2014b):

Definição 5 (${}^KPGD$)**.** *Seja $G = (V, A, d)$ um grafo simples, não direcionado e ponderado. O **Problema de Geometria de Distâncias K-dimensional** (${}^KPGD$) consiste em determinar uma função $x : V \to \mathbb{R}^K$ que respeite as seguintes equações:*

$$\|x(u) - x(v)\| = d(u, v), \forall \{u, v\} \in A. \tag{14}$$

Definição 6 ($^{K}PDGDM$)**.** *Sejam $G = (V, A, d)$ um grafo simples, não direcionado e ponderado de um ^{K}PGD, e $v_1, v_2, \ldots, v_{|V|}$ uma ordenação dos vértices de G, tal que*

1. *existe uma realização para v_1, v_2, ..., v_K;*
2. *para todo $i = K + 1, \ldots, |V|$, os K vértices **imediatamente** anteriores a v_i são adjacentes a v_i, ou seja,*

$$\{v_{i-K}, v_i\}, \{v_{i-(K-1)}, v_i\}, \ldots, \{v_{i-1}, v_i\} \in A. \tag{15}$$

Além disto, a seguinte desigualdade estrita de (K-1)-simplex é satisfeita:

$$V_{K-1}(v_{i-K}, v_{i-(K-1)}, \ldots, v_{i-1}) > 0. \tag{16}$$

*O **Problema Discretizável de Geometria de Distâncias Moleculares K-dimensional** ($^{K}PDGDM$) consiste em determinar uma função $x : V \to \mathbb{R}^K$ que respeite:*

$$\|x_{v_i} - x_{v_j}\| = d_{v_i,v_j}, \forall \{v_i, v_j\} \in A.$$

O advento do BP traz vantagens importantes em comparação aos métodos contínuos. O seu próprio funcionamento evidencia uma delas: como há sempre a determinação de todas as realizações possíveis para cada vértice v_i, $i \in \{4, 5, \ldots, |V|\}$, o algoritmo pode encontrar todas as soluções para o problema. Além disto, o BP tem facilidade em lidar com instâncias de grande porte e as soluções obtidas são mais precisas (LIBERTI; LAVOR; MACULAN, 2008).

A despeito dessas características, resolver um $^{3}PDGDM$ é, assim como um ^{3}PDG, um problema NP-difícil (LAVOR et al., 2012). Há ainda um outro fator agravante: obter uma $^{3}PDGDM$ ordem é também NP-difícil (CASSIOLI et al., 2015).

Com a finalidade de minimizar esta dificuldade, Lavor, Liberti e Mucherino (2013) propõem uma nova ordenação dos vértices de $G = (V, A)$ a partir do relaxamento da definição de $^{3}PDGDM$ ordem: permite-se a *repetição* de vértices sob certas condições. Esta ordenação é denominada *re-ordem*. De fato, Lavor et al. (2019) mostram que é possível achar uma re-ordem num grafo de um $^{3}PDGDM$ em tempo polinomial.

Esta dissertação concentra-se em elaborar um algoritmo exato, e de custo polinomial, para encontrar uma $^{3}PDGDM$ re-ordem, de acordo com o explorado em Lavor et al. (2019). Ademais, o algoritmo é testado em instâncias de moléculas de proteínas presentes no *Protein Data Bank*, o maior repositório de estruturas tridimensionais de proteínas. Como principais parâmetros, são avaliados a existência ou não de re-ordens e o tempo de execução.

1 Re-ordem

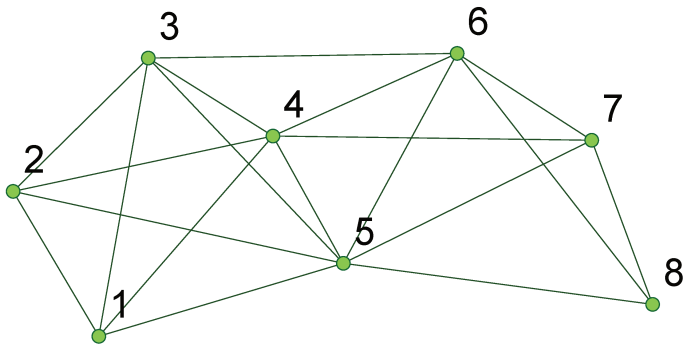
Uma *re-ordem* em um grafo $G = (V, A)$ simples e não direcionado é uma sequência de vértices na qual cada elemento do quarto em diante conecta-se com os dois elementos imediatamente anteriores, e conecta-se ao terceiro antecessor ou é igual ao terceiro antecessor. Além disto, os três primeiros termos estão conectados entre si dois-a-dois (ou seja, é uma *3-clique*). Mais precisamente, como apresentada em Lavor et al. (2019), pode-se defini-la do seguinte modo:

Definição 7 (Re-ordem)**.** *Seja $G = (V, A)$ um grafo simples e não direcionado. Uma* ^3PDGDM re-ordem *de* G *é uma função sobrejetiva $r : \{1, 2, \ldots, m\} \to V$, $m \in \mathbb{N}$, que satisfaz as seguintes propriedades (para simplificar a notação, escrevemos $r(i)$ como r_i):*

1. $\{r_1, r_2, r_3\}$ *é uma 3-clique;*
2. $\forall\ i = 4, 5, \ldots, m,\ \{r_{i-2}, r_i\},\ \{r_{i-1}, r_i\} \in A;$
3. $\forall\ i = 4, 5, \ldots, m,\ \{r_{i-3}, r_i\} \in A$ *ou* $r_i = r_{i-3}$.

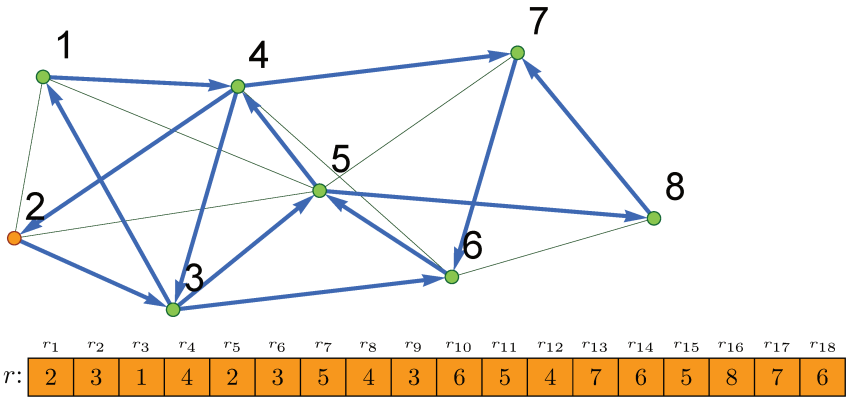
A Figura 3 ilustra um grafo simples e não direcionado e a Figura 4 mostra uma re-ordem deste grafo.

Figura 3 – Grafo simples e não direcionado.



O nome *re-ordem* deriva de uma das propriedades que a define: uma sequência de elementos que pode conter *repetição* de vértices. Sua definição carrega, propositalmente, informações geométricas de um PGD necessárias para encará-lo como um ^3PDGDM. Por exemplo, as propriedades 2 e 3 garantem que conseguimos calcular a interseção entre as esferas $S(x_{r_{i-1}}, d_{r_i,r_{i-1}})$, $S(x_{r_{i-2}}, d_{r_i,r_{i-2}})$ e $S(x_{r_{i-3}}, d_{r_i,r_{i-3}})$, se o vértice r_i está sendo "descoberto"naquele momento. Se r_i é igual ao vértice r_{i-3}, então significa que r_i já foi realizado em algum momento anterior.

Figura 4 – Re-ordem de $G = (V, A)$.



Nota – O primeiro vértice da re-ordem r está colorido em *laranja* (elemento 2). Os arcos em *azul* representam a sucessão, em r, de um vértice a outro consecutivo.

Por outro lado, examinando a re-ordem apenas como um ente matemático, novas propriedades podem levar a outras compreensões sobre o ^3PDGDM. Isto dito, definamos *passeio* e *k-clique* de um grafo.

Chama-se de *passeio* em $G = (V, A)$ uma sequência de vértices na qual todo par de elementos consecutivos corresponde a uma aresta de A. Em outras palavras, $\alpha = (u_1, u_2, \ldots, u_p)$ é um passeio em G se, e somente se, $\alpha \subseteq V$ e $\{u_i, u_{i+1}\} \in A$, para todo $i \in \{1, 2, \ldots, p-1\}$.

Uma *k-clique* de $G = (V, A)$ é um subgrafo completo que possui k vértices.

Então, uma re-ordem pode ser vista como um passeio em $G = (V, A)$ no qual cada quarteto de elementos consecutivos representa: (a) ou uma 4-clique de G; (b) ou uma 3-clique, e o primeiro termo é igual ao quarto. Isto pode ser observado na Figura 4.

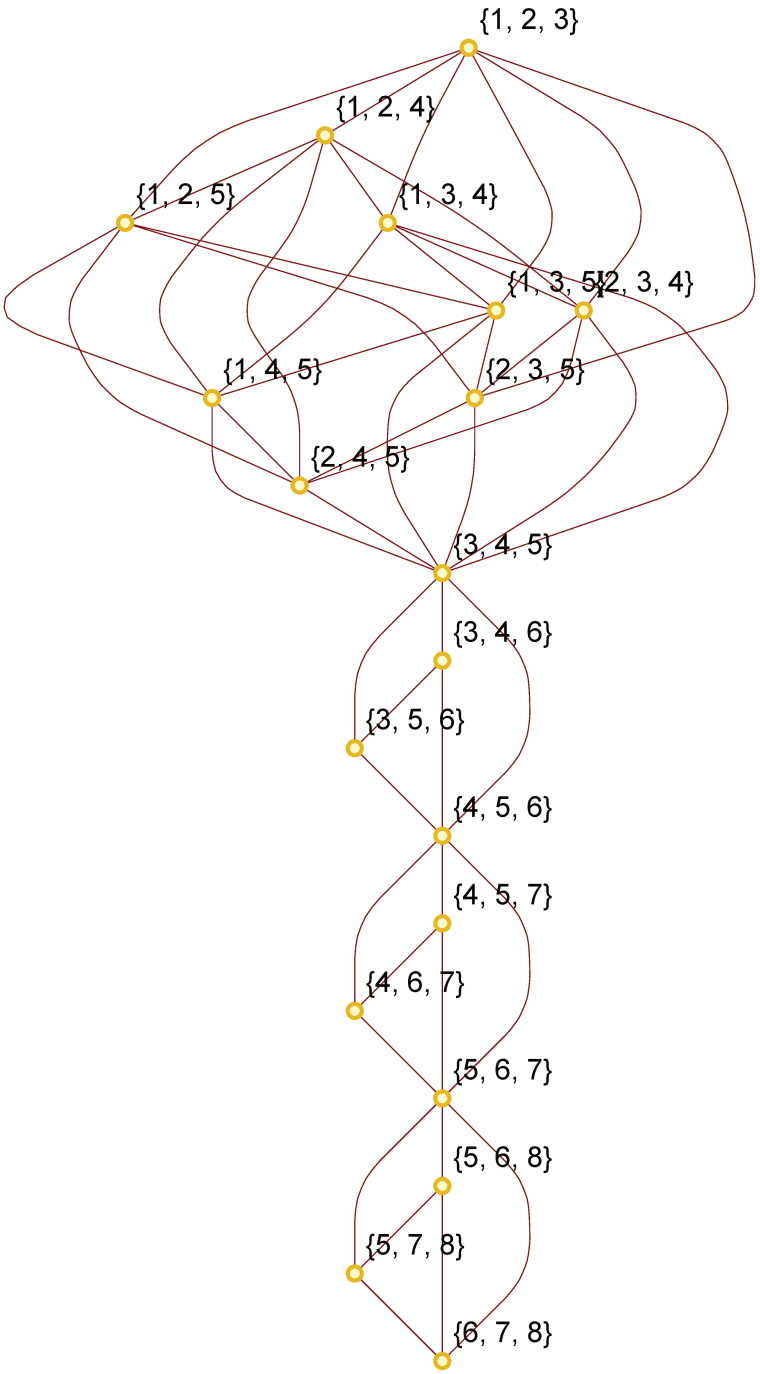
Lavor et al. (2019) constroem um grafo auxiliar que relaciona re-ordens e cliques. Este grafo é chamado de $G_3 = (V_3, A_3)$ e é formalmente definido abaixo:

Definição 8 (Grafo G_3)**.** *Dado um grafo simples e não direcionado $G = (V, A)$, define-se $G_3 = (V_3, A_3)$ como um grafo cujos vértices e arestas respeitam, respectivamente:*

1. *$\tau_i \in V_3 \Leftrightarrow \tau_i$ representa uma 3-clique em G;*
2. *$\{\tau_i, \tau_j\} \in A_3 \Leftrightarrow \tau_i \cup \tau_j$ representa uma 4-clique em G.*

A Figura 5 ilustra o grafo G_3 atrelado ao grafo da Figura 3. Cada vértice τ_i de G_3 recebe como rótulo o conjunto $\{t_{i1}, t_{i2}, t_{i3}\}$, cujos elementos t_{i1}, t_{i2} e t_{i3} correspondem aos rótulos dos vértices de G atrelados a τ_i, em ordem crescente.

Figura 5 – Grafo G_3.



O vínculo entre re-ordem e G_3 sugere uma relação entre a existência de re-ordens em $G = (V, A)$ e a existência de algum tipo de passeio em $G_3 = (V_3, A_3)$. Isto é comprovado por Lavor et al. (2019). De fato, os passeios $\gamma = (\gamma_i)_{i=1}^{p}$ em $G_3 = (V_3, A_3)$ cogitados anteriormente são aqueles cujo o conjunto dos vértices de G atrelados aos vértices γ_i é igual ao próprio V. Os Teorema 1 e Teorema 2 a seguir reapresentam o *Theorem 1* de Lavor et al. (2019) para os casos particulares de ${}^{K}PDGDM$ nos quais $K = 3$.

Teorema 1. *Seja $G = (V, A)$ um grafo simples e não direcionado. Se existe uma* ^3PGDMD re-ordem *em* G*, então existe um passeio γ em $G_3 = (V_3, A_3)$ tal que $\bigcup_i \gamma_i = V$.*

Demonstração. Suponhamos que existe uma re-ordem r, de tamanho m, em G. Tomamos a seguinte sequência de subconjuntos de V:

$$\begin{aligned}\gamma_1 &= \{r_1, r_2, r_3\}\\ \gamma_2 &= \{r_2, r_3, r_4\}\\ &\vdots\\ \gamma_{m-2} &= \{r_{m-2}, r_{m-1}, r_m\}.\end{aligned}$$

Consideraremos dois casos:

Caso 1: $\gamma_i \neq \gamma_{i+1}, \forall\, i = 1, 2, \ldots, m-3$.

Dos itens 1 e 2 da definição de re-ordem, γ_i representa uma 3-clique em G, para $i = 1, 2, \ldots, m-2$. Daí, $\gamma_i \in V_3$.

Pelos itens 2 e 3 da mesma definição, $\gamma_j \cup \gamma_{j+1} = \{r_j, r_{j+1}, r_{j+2}, r_{j+3}\}$ é 4-clique em G, para $j = 1, 2, \ldots, m-3$. Pela definição de G_3, isto implica que $\{\gamma_j, \gamma_{j+1}\} \in A_3$.

Como r é sobrejetiva, concluímos que a sequência $(\gamma_i)_{i=1,2,\ldots,m-2}$ é um passeio em G_3 tal que $\bigcup_i \gamma_i = V$.

Caso 2: Existe γ_k tal que $\gamma_{k-1} \neq \gamma_k = \gamma_{k+1} = \cdots = \gamma_{k+(l_k-1)} \neq \gamma_{k+l_k}$ $(l_k \in \mathbb{N},\, l_k > 1)$.

A sequência $(\gamma)_{i=1}^{m-2}$ não é passeio em $G_3 = (V_3, A_3)$, uma vez que $\{\gamma_k, \gamma_{k+1}\} = \{\gamma_k, \gamma_k\} \notin A_3$.

Seja $(\overline{\gamma}_i)_{i=1}^{m'}$, $m' < m-2$, a sequência de trios de vértices de $G = (V, A)$ obtida de $(\gamma_i)_{i=1}^{m-2}$ ao transformar cada subsequência de elementos consecutivos idênticos $(\gamma_j, \ldots, \gamma_{j+l_j})$ em um único elemento γ_j. $(\overline{\gamma}_i)_{i=1}^{m'}$ adéqua-se ao **Caso 1**. Portanto, $(\overline{\gamma}_i)_{i=1}^{m'}$ é um passeio em $G_3 = (V_3, A_3)$ tal que $\bigcup_i \overline{\gamma'}_i = V$.

□

O Teorema 1 mostra que toda re-ordem possui associada a si pelo menos um passeio em $G_3 = (V_3, A_3)$ que atinge todos os vértices de $G = (V, A)$.

O Teorema 2, em contrapartida, anuncia o inverso: de cada passeio em G_3 que atinge todo o conjunto V, pode-se construir uma re-ordem de $G = (V, A)$. Esta

possibilidade é "insinuada"por algumas características do passeio γ em $G_3 = (V_3, A_3)$. Por exemplo, consideremos a sequência $s = (\gamma_{i,1}, \gamma_{i,2}, \gamma_{i,3})_{i=1}^{p}$ de vértices de $G = (V, A)$. Tal sequência é formada pela sucessão das trincas de vértices de V de cada elemento de γ, na ordem em que são representados como vértices de $G_3 = (V_3, A_3)$. Nota-se que os três primeiros elementos de s respeitam o item 1 da definição de re-ordem (Definição 7) e que o terceiro elemento de cada trinca respeita os itens 2 e 3 da mesma definição. Ademais, como os elementos de s associados a γ_i e γ_{i+1} consecutivos formam uma 4-clique de G, parece razoável pensar que pode existir um ajuste da ordem destes elementos de forma que a sequência satisfaça todos os itens da Definição 7.

A Figura 6 ilustra um passeio no grafo $G_3 = (V_3, A_3)$ da Figura 5. A Figura 7 ilustra no passeio da Figura 6 as "insinuações"elencadas acima.

Portanto, assim como em Lavor et al. (2019), a demonstração do Teorema 2 é feita por meio da construção de uma sequência particular de vértices de G extraída do passeio γ. Tal sequência deverá ser uma re-ordem.

A fim de ajudar nesta construção, apresentamos algumas novas notações.

Seja γ um passeio de p vértices de G_3. Para todo γ_i de γ, atribuímos até dois rótulos a cada um de seus elementos, um rótulo positivo e um rótulo negativo. Então, estabelecemos

$$\{\gamma_i^{*-}, \gamma_i^{1-}, \gamma_i^{2-}\} = \gamma_i = \{\gamma_i^{*+}, \gamma_i^{1+}, \gamma_i^{2+}\} \tag{1.1}$$

tais que, $\forall i \in \{1, 2, \ldots, p-1\}$,

$$\begin{cases} \gamma_i^{*+} = \gamma_i \backslash \gamma_{i+1}; \\ \gamma_i^{1+} = \min(\gamma_i \cap \gamma_{i+1}); \\ \gamma_i^{2+} = \max(\gamma_i \cap \gamma_{i+1}); \end{cases} \tag{1.2}$$

e, $\forall i \in \{2, \ldots, p\}$,

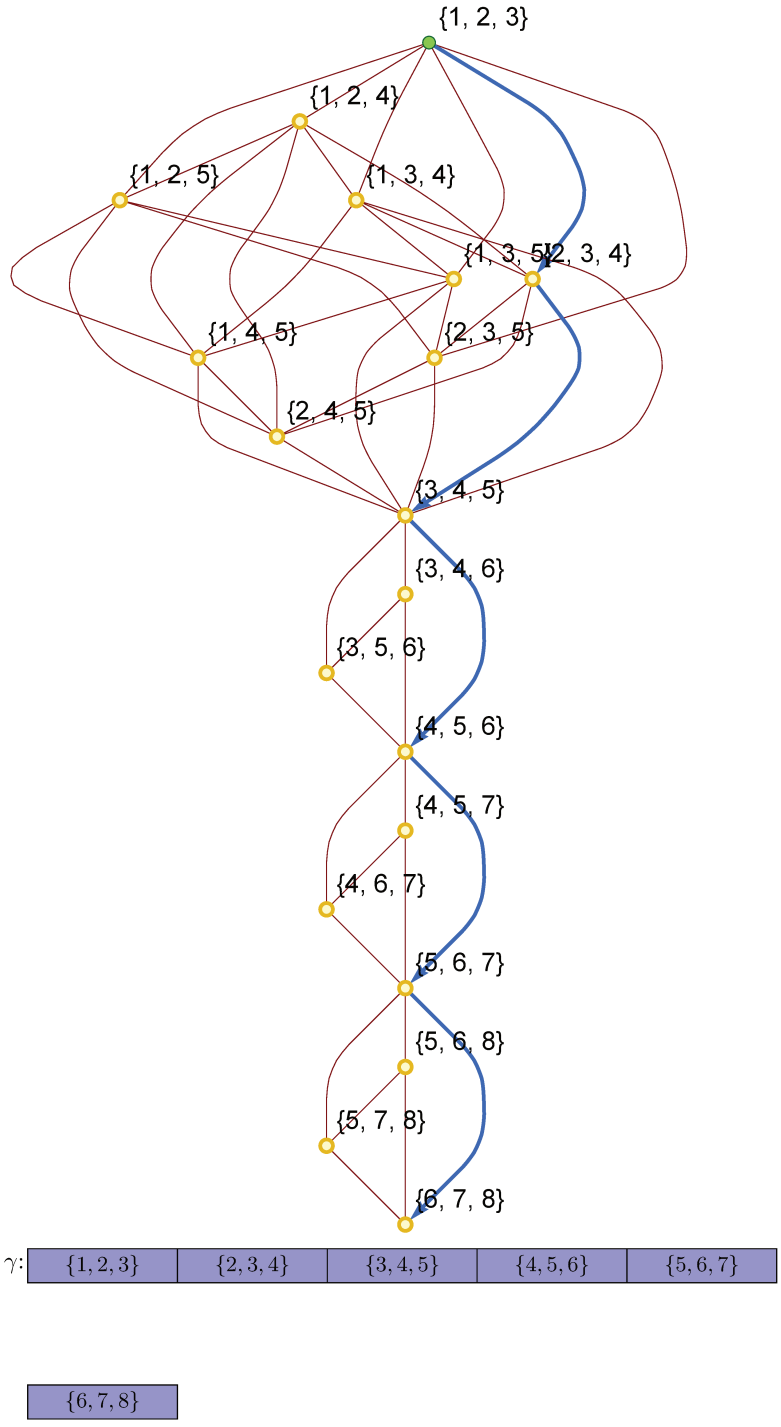
$$\begin{cases} \gamma_i^{*-} = \gamma_i \backslash \gamma_{i-1}; \\ \gamma_i^{1-} = \min(\gamma_i \cap \gamma_{i-1}); \\ \gamma_i^{2-} = \max(\gamma_i \cap \gamma_{i-1}). \end{cases} \tag{1.3}$$

A parte superior da Figura 8 ilustra a atribuição de rótulos γ^+ e γ^- ao passeio γ do grafo $G_3 = (V_3, A_3)$ da Figura 6.

Os rótulos positivos relacionam os vértices de γ_i com os vértices de γ_{i+1}, o seu sucessor. O elemento que recebe o símbolo $*+$ é aquele que *está* em γ_i e que *não está* no sucessor. Aos dois elementos restantes, aquele que for menor recebe o símbolo $1+$, enquanto o maior recebe $2+$.

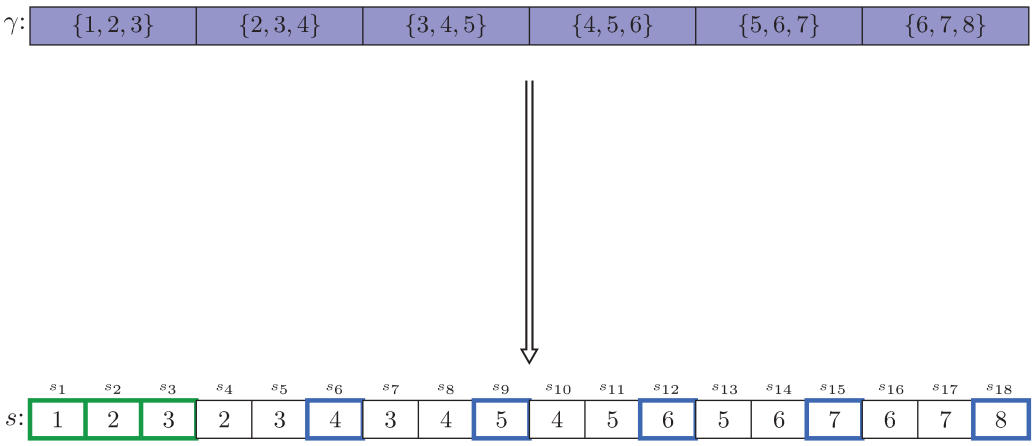
Os rótulos negativos carregam informação idêntica a dos rótulos positivos, porém relacionam γ_i ao seu antecessor, γ_{i-1}.

Figura 6 – Passeio γ de $G_3 = (V_3, A_3)$.



Nota – O primeiro vértice do passeio γ está colorido em *verde* (elemento $\{1, 2, 3\}$). Os arcos em *azul* representam a sucessão entre os vértice de γ.

Figura 7 – Sequência s.



Nota – A sequência s do grafo $G = (V, A)$ da Figura 3 é obtida por meio da justaposição dos inteiros do passeio γ no grafo $G_3 = (V_3, A_3)$ (ver Figura 6). Os três primeiros vértices (em *verde*) da sequência s formam uma 3-clique. Dos vértices restantes, cada um daqueles que se encontram em posição i múltipla de *três* (em azul) respeitam os itens 2 e 3 da definição de re-ordem: está conectado por arestas aos *(i-1)*-ésimo e *(i-2)*-ésimo elementos; ou está também conectado ao *(i-3)*-ésimo vértice ou é igual a ele.

Uma vez que γ_i está ligado a γ_{i+1} e a γ_{i-1} por arestas, pela definição de G_3, há exatamente dois elementos de γ_i que também pertencem a γ_{i+1}, e há dois elementos de γ_i que pertencem a γ_{i-1}. Portanto, a atribuição das rotulações γ^+ e γ^- representadas nas Equações (1.2) e (1.3) estão *bem-definidas*. Consequentemente, $\forall i \in \{1, 2, \ldots, p-1\}$,

$$\begin{cases} \gamma_i^{1+} = \gamma_{i+1}^{1-}; \\ \gamma_i^{2+} = \gamma_{i+1}^{2-}; \end{cases} \tag{1.4}$$

e, $\forall i \in \{2, \ldots, p\}$,

$$\begin{cases} \gamma_i^{1-} = \gamma_{i-1}^{1+}; \\ \gamma_i^{2-} = \gamma_{i-1}^{2+}. \end{cases} \tag{1.5}$$

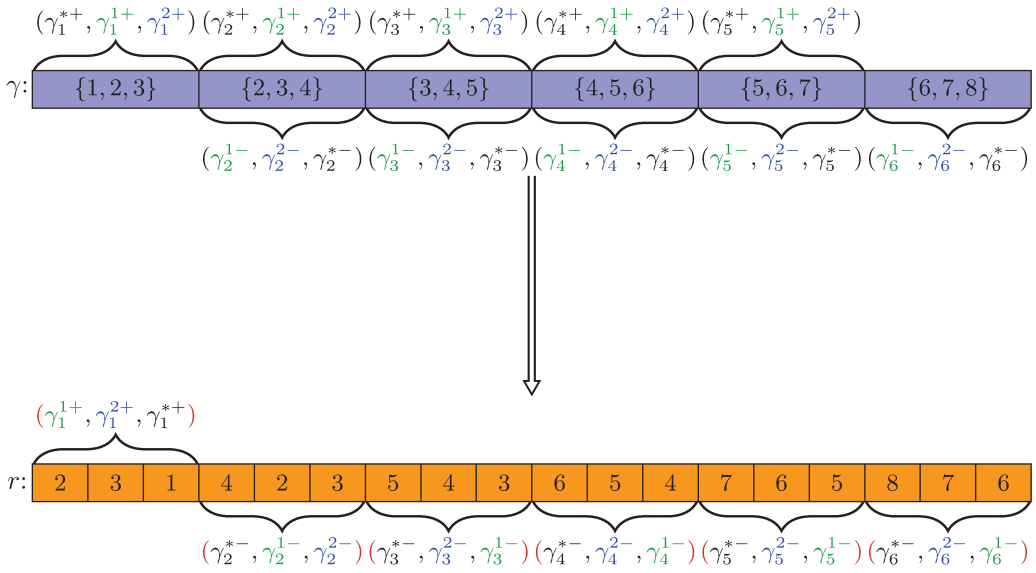
Por fim, construímos $r : \{1, 2, \ldots, 3p\} \to V$, do elemento r_1 ao elemento r_{3p}, da seguinte maneira:

$$
\begin{aligned}
r_1 &= \gamma_1^{1+};\\
r_2 &= \gamma_1^{2+};\\
r_3 &= \gamma_1^{*+};
\end{aligned}
$$

$$
\forall\, j \in \{4,\ldots,3p\},\ \begin{cases}
\text{se } j = 3\lceil\frac{j}{3}\rceil - 2, & r_j = \gamma^{*-}_{\lceil\frac{j}{3}\rceil};\\[2ex]
\text{se } j = 3\lceil\frac{j}{3}\rceil - 1, & \begin{cases} r_j = r_{j-4}, & \text{se } r_{j-4} = \gamma^{1-}_{\lceil\frac{j}{3}\rceil} \text{ ou } r_{j-4} = \gamma^{2-}_{\lceil\frac{j}{3}\rceil};\\ r_j = r_{j-3}, & \text{se } r_{j-4} = \gamma^{*-}_{\lceil\frac{j}{3}\rceil}; \end{cases}\\[2ex]
\text{se } j = 3\lceil\frac{j}{3}\rceil, & r_j = (\gamma_{\lceil\frac{j}{3}\rceil-1} \cap \gamma_{\lceil\frac{j}{3}\rceil})\backslash\{r_{j-1}\}.
\end{cases}
\tag{1.6}
$$

O cerne da demonstração do Teorema 2 revela que a sequência de vértices determinada por r respeita os três itens da Definição 7. A Figura 8 ilustra a re-ordem r obtida a partir do passeio γ da Figura 6, por meio das Equações (1.6).

Figura 8 – Re-ordem encontrada a partir de um passeio em $G_3 = (V_3, A_3)$.



Nota – Acima dos vértices do passeio γ do grafo $G_3 = (V_3, A_3)$ (ver Figura 6), estão as rotulações positivas definidas nas Equações (1.2); abaixo, as rotulações negativas definidas nas Equações (1.3). Ao se organizar as 3-cliques de cada vértice γ_i de acordo com as Equações (1.6), obtém-se a re-ordem r.

Teorema 2. *Seja $G = (V, A)$ um grafo simples e não direcionado. Se existe um passeio γ em $G_3 = (V_3, A_3)$ tal que $\bigcup_i \gamma_i = V$, então existe uma ${}^3PGDMD$* re-ordem *em* G.

Demonstração. Suponhamos que existe um passeio γ em $G_3 = (V_3, A_3)$ tal que $\bigcup_{i \leqslant p} \gamma_i = V$, cujo $p \in \mathbb{N}$, $p > 0$. Construímos a função $r : \{1, 2, \ldots, 3p\} \to V$ como aquela definida nas Equações (1.6). Como $\bigcup_{j \leqslant 3p} \{r_j\} = \bigcup_{i \leqslant p} \gamma_i = V$, r é sobrejetiva. Observamos que

$$\begin{cases} \{r_1, r_2, r_3\} = \gamma_1 \\ \gamma_1 \in V_3 \end{cases} \quad \Rightarrow \quad \{r_1, r_2, r_3\} \text{ é 3-clique em } G.$$

Logo, r respeita a propriedade *1* da definição de re-ordem (Definição 7).

Chamamos de $r^{[j]}$ a subsequência $\{r_1, \ldots, r_j\}$ e $G^{[j]}$ o subgrafo de G cujo conjunto de vértices é igual a $\bigcup_{l \leqslant j} \{r_l\}$. É trivial mostrar que $r^{[3]}$ respeita as propriedades 2 e 3 da definição de re-ordem. Logo, $r^{[3]}$ é re-ordem para $G^{[3]}$.

Agora, para todo $j \in \{4, \ldots, 3p\}$, clamamos que a subsequência $r^{[j]} = \{r_1, \ldots, r_j\}$ é re-ordem do subgrafo $G^{[j]}$. Por indução matemática, suponhamos que $r^{[l]}$ é re-ordem de $G^{[l]}$, para todo $l \in \{3, \ldots, k-1\}$, $k \geqslant 4$. Por contradição, admitamos que $r^{[k]}$ não é re-ordem de $G^{[k]}$. Ou seja,

$$\begin{cases} \{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} \text{ nem é 4-clique e nem é 3-clique;} \\ ou \\ r_k = r_{k-1} \text{ ou } r_k = r_{k-2}. \end{cases} \tag{1.7}$$

Da construção de r, observamos que o elemento r_k é definido de acordo com o resto de k por 3. Desta forma, k pode assumir três possíveis valores:

Caso 1: $k = 3\lceil \frac{k}{3} \rceil - 2$ (resto 2)

Temos que:

$$\begin{cases} \{r_{k-3}, r_{k-2}, r_{k-1}\} = \gamma_{\lceil \frac{k}{3} \rceil - 1} \\ r_k = \gamma^{*-}_{\lceil \frac{k}{3} \rceil} \end{cases} \quad \Rightarrow \{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} \text{ é 4-clique} \tag{1.8}$$

$$\Rightarrow r_k \neq r_{k-1} \text{ e } r_k \neq r_{k-2}.$$

Caso 2: $k = 3\lceil \frac{k}{3} \rceil - 1$ (resto 1)

Há duas possibilidades:

(a) $r_{k-4} = \gamma^{1+}_{\lceil \frac{k}{3} \rceil - 1}$ ou $\gamma^{2+}_{\lceil \frac{k}{3} \rceil - 1}$:

$$\begin{cases} \{r_{k-3}, r_{k-2}\} = \gamma_{\lceil \frac{k}{3} \rceil - 1} \backslash \{r_{k-4}\} \\ \{r_{k-1}, r_k\} = \{\gamma^{*-}_{\lceil \frac{k}{3} \rceil}, r_k\} \\ r_k = r_{k-4} \end{cases} \Rightarrow \{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} \text{ é 4-clique} \tag{1.9}$$

$$\Rightarrow r_k \neq r_{k-1} \text{ e } r_k \neq r_{k-2}.$$

(b) $r_{k-4} = \gamma^{*+}_{\lceil \frac{k}{3} \rceil - 1}$:

Em primeiro lugar,

$$\begin{cases} \{r_{k-3}, r_{k-2}\} = \{\gamma^{1+}_{\lceil \frac{k}{3} \rceil - 1}, \gamma^{2+}_{\lceil \frac{k}{3} \rceil - 1}\} \\ \{r_{k-1}, r_k\} = \{\gamma^{*-}_{\lceil \frac{k}{3} \rceil}, r_k\} \\ r_k = r_{k-3} \end{cases} \Rightarrow \{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} \text{ é 3-clique.} \tag{1.10}$$

Então,

$$\begin{cases} r_k = r_{k-3} \\ \{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} \text{ é 3-clique} \end{cases} \Rightarrow r_k \neq r_{k-1} \text{ e } r_k \neq r_{k-2}. \tag{1.11}$$

Caso 3: $k = 3\lceil \dfrac{k}{3} \rceil$ (resto 0)

Temos que:

$$\{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} = \{r_{k-3}\} \cup \gamma_{\lceil \frac{k}{3} \rceil} \Rightarrow \begin{cases} \{r_{k-3}, r_{k-2}, r_{k-1}, r_k\} \text{ é 3-clique ou 4-clique} \\ \{r_{k-2}, r_{k-1}, r_k\} = \gamma_{\lceil \frac{k}{3} \rceil} \Rightarrow r_k \neq r_{k-1} \text{ e } r_k \neq r_{k-2} \end{cases} \tag{1.12}$$

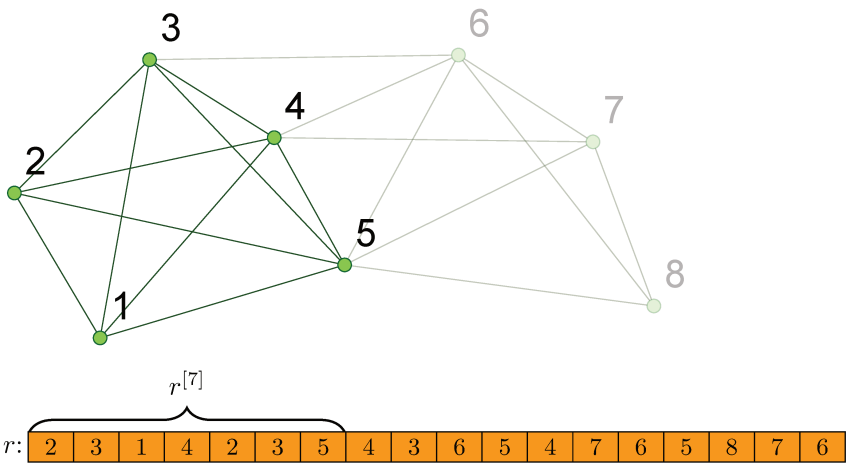
Portanto, como ambas as sentenças (1.7) não acontecem, $r^{[k]}$ é re-ordem de $G^{[k]}$. Então, por indução, $r^{[j]}$ é re-ordem de $G^{[j]}$, para todo $j \in \{3, 4, \ldots, 3p\}$.

Por fim, como $\bigcup_{i \leqslant p} \gamma_i = V$ por hipótese, concluímos que r é re-ordem de $G = (V, A)$.

□

No Teorema 2, conseguimos provar que toda subsequência $r^{[i]}$ de r é uma re-ordem para o subgrafo $G^{[i]}$ de G. Consequentemente, r é re-ordem de G.

Figura 9 – Subgrafo $G^{[7]}$ e subsequência $r^{[7]}$.



Nota – A subsequência $r^{[7]}$ da re-ordem r é aquela composta pelos 7 primeiros elementos. O subgrafo $G^{[7]}$ do grafo $G = (V, A)$ da Figura 3 é aquele formado pelos vértices de $r^{[7]}$ e por todas as arestas entre estes elementos que pertencem a A.

A Figura 9 representa os objetos $r^{[7]}$ e $G^{[7]}$ atrelados à re-ordem da Figura 4.

Vale ressaltar que a re-ordem criada neste trabalho para demonstrar o Teorema 2 difere daquela empregada em Lavor et al. (2019) (*Theorem 1*). De fato, os seis primeiros termos da sequência r retratada nas Equações (1.6) são $(\gamma_1^{1+}, \gamma_1^{2+}, \gamma_1^{*+}, \gamma_2^{*-}, \gamma_2^{1-}, \gamma_2^{2-})$, enquanto $(\gamma_1^{1+}, \gamma_1^{2+}, \gamma_1^{*+}, \gamma_2^{1-}, \gamma_2^{2-}, \gamma_2^{*-})$ são os seis primeiros termos da re-ordem de Lavor et al. (2019).

Desta forma, há pelo menos uma outra maneira de extrair uma re-ordem de um passeio em G_3.

Os dois últimos teoremas podem ser reunidos no seguinte corolário:

Corolário 1 (Existência de re-ordem)**.** *Seja $G = (V, A)$ um grafo simples e não direcionado. Existe uma* ^3PGDMD re-ordem *em* G *se, e somente se, existe um passeio γ em $G_3 = (V_3, A_3)$ tal que $\bigcup_i \gamma_i = V$.*

Demonstração. Segue imediatamente dos Teorema 1 e Teorema 2.

□

Nota-se que o Corolário 1 não exige que o passeio em G_3 passe por todos os vértices de V_3. Isto possibilita a criação do próximo corolário, que relaciona a existência de re-ordem com a conectividade de G_3:

Corolário 2 (Existência de re-ordem em componente conexa)**.** *Seja $G = (V, A)$ um grafo simples e não direcionado. Existe uma* ^3PGDMD re-ordem *em* G *se, e somente se, existe um passeio γ em uma* ***componente conexa*** *de $G_3 = (V_3, A_3)$, tal que $\bigcup_i \gamma_i = V$.*

Demonstração. Segue imediatamente do Corolário 1.

□

Todos os resultados demonstrados, que estão sintetizados no Corolário 2, permitem que uma re-ordem de ^3PDGDM seja obtida em tempo polinomial. Lavor et al. (2019) provam esta afirmação para todo ^KPDGDM, com $K \in \mathbb{Z}_+^*$. Na próxima seção, determinamos um algoritmo capaz de realizar este feito para um ^3PDGDM, além de calcular sua complexidade.

1.1 Algoritmo

Os dois teoremas e os dois corolários apresentados abrem um novo leque de possibilidades para encontrar re-ordens de um ${}^3PDGDM$. Em particular, os próprios mecanismos utilizados para provar o Teorema 2 trazem instruções para gerar uma re-ordem de passeios em $G_3 = (V_3, A_3)$: como a sequência r obtida das Equações (1.6) satisfaz os itens 1, 2 e 3 da Definição 7, pode-se determinar se r é uma re-ordem verificando se todos os vértices de $G = (V, A)$ estão presentes em r.

Desta forma, para garantir que alguma re-ordem seja encontrada, é suficiente achar um passeio $\gamma = (\gamma_i)_{i=1}^{p}$ em $G_3 = (V_3, A_3)$ que atinja todos os vértices de $G = (V, A)$, ou seja, $\bigcup_i \gamma_i = V$. Se não existem passeios em G_3 com esta característica, pode-se concluir que não existe re-ordem para o ${}^3PDGDM$ em questão.

O Algoritmo 1 representa um esboço para um método que extraia uma re-ordem de G de seu grafo G_3.

Algoritmo 1 – Esboço para gerar re-ordem

Entrada: $G = (V, A)$: Grafo simples não direcionado
Saída: reordens: Lista de vetores de vértices de G

1 **início**
  `/* Etapa 1 */`
2 Construir $G_3 = (V_3, A_3)$;
  `/* Etapa 2 */`
3 Para cada componente conexa C_i de G_3, descobrir árvore geradora T_i;
4 **para cada** T_i **faça**
5 Realizar percurso completo em T_i: guardar a sucessão de vértices em lista P_i;
  `/* Etapa 3 */`
6 **para cada** P_i **faça**
7 Organizar cada $\tau_j \in P_i$ em $\overline{\tau}_j$ de tal maneira que $seq[i] = (\overline{\tau}_{1,1}, \overline{\tau}_{1,2}, \overline{\tau}_{1,3}, \ldots, \overline{\tau}_{j,1}, \overline{\tau}_{j,2}, \overline{\tau}_{j,3}, \ldots, \overline{\tau}_{|P_i|,1}, \overline{\tau}_{|P_i|,2}, \overline{\tau}_{|P_i|,3})$ é um vetor de elementos de V que satisfaz os itens 1, 2 e 3 da Definição 7.
  `/* Etapa 4 */`
8 **para cada** $seq[i]$ **faça**
9 **se** $seq[i]$ *atinge todos os elementos de* V **então**
10 $reordens$.InserirNoFim($seq[i]$);
11 **retorna** $reordens$;

Antes de entrarmos na determinação dos pseudocódigos que compõem o algoritmo, faz-se necessário estabelecermos algumas estruturas de dados para a representação de vértices e arestas de $G = (V, A)$ e de $G_3 = (V_3, A_3)$.

Começamos por criar uma estrutura de dados para armazenar um elemento de V_3. Similarmente ao que foi observado nas figuras anteriores, os vértices de G são denotados como *inteiros positivos* que variam no intervalo $[1, |V|]$. Consequentemente, os vértices de G_3 são tomados por trincas ordenadas destes inteiros. Formalmente:

Definição 9 (Estrutura de dados *vertice3*)**.** *Representamos por*

$$vertice3 \tag{1.13}$$

a estrutura de dados que armazena um vértice de G_3. Esta estrutura guarda três inteiros, ordenadas de maneira crescente:

$$\begin{cases} v1\text{:} \text{ Inteiro;} \\ v2\text{:} \text{ Inteiro;} \\ v3\text{:} \text{ Inteiro,} \end{cases} \tag{1.14}$$

tais que

$$v1 < v2 < v3. \tag{1.15}$$

Como os vértices de $G_3 = (V_3, A_3)$ simbolizam conjuntos, são representados de maneira única por objetos do tipo *vertice3*. Por exemplo, $\{2, 1, 3\}$ e $\{3, 1, 2\}$ geram o mesmo elemento *vertice3*, aquele cujos $v1 = 1$, $v2 = 2$ e $v3 = 3$.

Por sua vez, tanto as arestas de $G = (V, A)$ quanto as arestas de $G_3 = (V_3, A_3)$ são denotadas como pares crescentemente ordenados de inteiros e de *vertice3*, respectivamente. Referimo-nos às estruturas de dados que as representam como *aresta* e *aresta3*.

Para armazenar grandes conjuntos de maneira ordenada, consideramos *Árvores Binárias de Busca Rubro-Negras*. Esta estrutura de dados é utilizada como molde para representar conjuntos e mapas ordenados de grande porte na linguagem de programação $C++$ (CONTAINERS. . . , 2020). Os números de operações básicas gastos nos piores casos dos métodos de *Inserção*, *Remoção* e *Busca* desta árvore são limitados superiormente por $ln^{|V|}$. Mais detalhes em Cormen et al. (2009).

Desta forma, ao representarmos um conjunto de objetos do tipo *obj* por *Conjunto(obj)*, podemos definir grafos simples e não direcionados da seguinte maneira:

Definição 10 (Estrutura de dados *GrafoG*)**.** *Representamos por*

$$GrafoG \tag{1.16}$$

a estrutura de dados que armazena um grafo simples e não direcionado. Os seus atributos são:

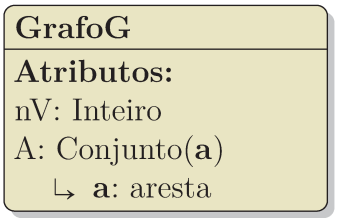
$$\begin{cases} nV\text{: Inteiro;} \\ A\text{: Conjunto(aresta);} \end{cases} \tag{1.17}$$

Guardamos em nV *o número de vértices. Cada vértice do grafo é representado por um rótulo inteiro positivo entre* 1 *e* nV*. As arestas são armazenadas na árvore binária de busca rubro-negra* A*.*

Referimo-nos ao objeto aresta de um nó de A por ***a****.*

A Figura 10 esboça os nomes e tipos dos atributos de um objeto *GrafoG*.

Figura 10 – Atributos de *GrafoG*.



Há diversas maneiras de se retratar computacionalmente um grafo simples e não direcionado. Aquela apresentada na Definição 10 é uma delas. Visto que o principal uso do grafo $G_3 = (V_3, A_3)$ na busca por uma re-ordem acontece na Etapa 2 do Algoritmo 1, interessa-nos que a representação escolhida para G_3 possibilite o menor custo possível de operações básicas para a execução desta etapa. Por razões que serão discutidas adiante, optamos pela Busca em Profundidade (*Depth First Scan*) como método para encontrar uma floresta geradora de $G_3 = (V_3, A_3)$.

Portanto, definimos $G_3 = (V_3, A_3)$ de forma que cada vértice de G_3 tenha acesso direto aos vértices que lhe são adjacentes.

Se representarmos um mapa que associa chaves do tipo *chave* a valores do tipo *val* por *Mapa(chave,val)*, $G_3 = (V_3, A_3)$ pode, por fim, ser formalmente apresentado:

Definição 11 (Estrutura de dados *GrafoG3.*)**.** *Representamos por*

$$GrafoG3 \tag{1.18}$$

a estrutura de dados que armazena um grafo G_3*, que possui o seguinte atributo:*

$$VizinhancaG3\text{: Mapa(vertice3, Conjunto(vertice3));} \tag{1.19}$$

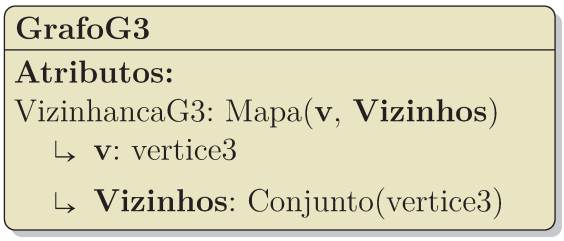
Referimo-nos à chave por ***v*** *e ao valor por* ***Vizinhos****. Então, para cada nó de VizinhancaG3, guarda-se:*

- *Em* $\boldsymbol{v}$*, um vértice de* G_3*;*
- *Em* ***Vizinhos****, todos os vértices que são adjacentes a* $\boldsymbol{v}$*, ou seja,*

  todo $\tau \in V_3$ *tal que* $\{\boldsymbol{v}, \tau\} \in A_3$. (1.20)

A Figura 11 esboça os nomes e tipos dos atributos de um objeto *GrafoG3*.

Figura 11 – Atributos de *GrafoG3*.



1.1.1 Construção de G_3

Os vértices e arestas do grafo $G_3 = (V_3, A_3)$ associado a um grafo $G = (V, A)$ podem ser descobertos por meio do exame de todos os conjuntos de 4 elementos de V, devido a própria Definição 8 de G_3. Naturalmente, consegue-se identificar com isto todas as cliques de tamanhos 3 e 4, e portanto, determinando V_3 e A_3. Esta ideia é apresentada por Lavor et al. (2019) (*Lemma 1*).

Assim, apresentamos o Algoritmo 2, que constrói um *GrafoG3* que representa ou o próprio $G_3 = (V_3, A_3)$ ou um subgrafo dele. De fato, vértices de G_3 que são isolados não precisam ser encontrados, uma vez que este tipo de elemento constitui uma componente conexa C_i de um único vértice, e a menos que G seja exatamente uma 3-clique, C_i não contém re-ordens de G.

O Algoritmo 2 faz uso de um método auxiliar chamado *Verifica4CliqueG*. Tal método é definido no Algoritmo 3 e tem como função verificar se um quarteto de vértices de $G = (V, A)$ corresponde a uma 4-clique. Isto é realizado por meio da busca em A de todos os pares de vértices do quarteto. Se todos os pares são encontrados, então todas as arestas entre aqueles vértices existem em A, implicando que aqueles vértices formam uma 4-clique em G.

Algoritmo 2 – ConstroiG3

Entrada: G: GrafoG

Saída: G_3: GrafoG3 originado de G

```
1  início
2      GrafoG3 G_3;
3      para i = 1 até i ⩽ G.nV − 3 faça
4          para j = i + 1 até j ⩽ G.nV − 2 faça
5              para k = j + 1 até k ⩽ G.nV − 1 faça
6                  para l = k + 1 até l ⩽ G.nV faça
7                      eh4clique ← Verifica4CliqueG(G, i, j, k, l);
8                      se eh4clique então
9                          G_3.Vizinhos[{i,j,k}].Inserir({i,j,l});
10                         G_3.Vizinhos[{i,j,k}].Inserir({i,k,l});
11                         G_3.Vizinhos[{i,j,k}].Inserir({j,k,l});
12                         G_3.Vizinhos[{i,j,l}].Inserir({i,j,k});
13                         G_3.Vizinhos[{i,j,l}].Inserir({i,k,l});
14                         G_3.Vizinhos[{i,j,l}].Inserir({j,k,l});
15                         G_3.Vizinhos[{i,k,l}].Inserir({i,j,k});
16                         G_3.Vizinhos[{i,k,l}].Inserir({i,j,l});
17                         G_3.Vizinhos[{i,k,l}].Inserir({j,k,l});
18                         G_3.Vizinhos[{j,k,l}].Inserir({i,j,k});
19                         G_3.Vizinhos[{j,k,l}].Inserir({i,j,l});
20                         G_3.Vizinhos[{j,k,l}].Inserir({i,k,l});
21     retorna G_3;
```

Algoritmo 3 – Verifica4CliqueG

Entrada: G: GrafoG
$v1$, $v2$, $v3$, $v4$: vértices de G
Saída: *eh4clique*: Booleano

```
1  início
2      arestas4clique ← [ ];
3      arestas4clique.Inserir({v1, v2});
4      arestas4clique.Inserir({v1, v3});
5      arestas4clique.Inserir({v1, v4});
6      arestas4clique.Inserir({v2, v3});
7      arestas4clique.Inserir({v2, v4});
8      arestas4clique.Inserir({v3, v4});
9      p ← 1;
10     eh4clique ← Verdadeiro;
11     enquanto (p ⩽ 6) E (eh4clique) faça
12         nohA ← G.BuscarAresta(arestas4clique[p]);
13         se nohA.a ≠ arestas4clique[p] então
14             eh4clique ← Falso;
15         p++;
16     retorna eh4clique;
```

As duas funções exibidas abarcam toda a Etapa 1 do esboço do Algoritmo 1. Para dar sequência as outras etapas, resta calcular a ordem de complexidade do número de operações realizadas no Algoritmo 2.

Teorema 3 (Complexidade de $ConstroiG3$)**.** *O custo das operações básicas de* ConstroiG3 *é igual a* $O(|V|^4 \log_2^{|V|})$.

Demonstração. Os índices i, j, k e l das estruturas de repetição *para* do Algoritmo 2 elencam, no intervalo de linhas $3-20$, todas as combinações possíveis de 4 inteiros entre 1 e $|V|$. Logo, o bloco de código formado pelas linhas $7-20$ é executado $\binom{|V|}{4}$ vezes.

A linha 7 do Algoritmo 2 possui custo igual ao custo do Algoritmo 3. Este algoritmo, por sua vez, custa $O(\log_2^{|V|})$, valor referente às buscas em A de $G = (V, A)$.

As linhas $8-20$ também custam $O(\log_2^{|V|})$: realizam-se 12 inserções em **Vizinhos**, que é árvore rubro-negra com $|V_3|$ nós ao todo e cada inserção custa $2 * O(\log_2^{|V_3|})$, sendo que $O(\log_2^{|V_3|}) = O(\log_2^{|V|^3}) = O(\log_2^{|V|})$.

Já que as linhas de $7-20$ são repetidas não mais que $|V|^4$ vezes, como descrito no início desta demonstração, concluímos que o custo total do Algoritmo 2 é $O(|V|^4 \log_2^{|V|})$.

□

1.1.2 Construção do passeio em G_3

Uma vez construído $G_3 = (V_3, A_3)$, a Etapa 2 do Algoritmo 1 consiste em encontrar um passeio γ de G_3 tal que $\bigcup_i \gamma_i = V$. Devido à própria Definição 8 de G_3, é natural que passeios que atingem todos os vértices de G_3 também atinjam todos os vértices de G, a menos que algum elemento de V não esteja representado em qualquer elemento de V_3. Vamos chamar tais passeios de δ.

Evidentemente, nem sempre é possível obter passeios que alcancem todos os vértices de $G_3 = (V_3, A_3)$, já que G_3 pode ser desconexo. Nestes casos, deve-se buscar um passeio δ^j para cada componente conexa que chegue a todos os vértices daquela componente. Se nenhum dos passeios δ^j atinge o conjunto V por completo, ou seja, $\bigcup_i \delta_i^j \neq V$, para cada componente C_j, então G_3 certamente não terá passeio γ tal que $\bigcup_i \gamma_i = V$.

Passeios δ como aqueles descritos nos parágrafos anteriores podem ser obtidos de árvores e florestas geradoras por meio de percursos em árvores (passeios que passam por todos os vértices de uma árvore). Isto é justificado pelas próprias definições de árvore e floresta geradora e de percurso em árvore. A Busca em Profundidade (em inglês, *Depth First Search* (DFS)) é um método que realiza um passeio δ^j em cada componente conexa C_j de $G_3 = (V_3, A_3)$, passeio este oriundo de um percurso em alguma árvore geradora T_j da componente C_j.

A Busca em Profundidade parte do pressuposto que se desconhece todos os vértices de $G_3 = (V_3, A_3)$ e tem por objetivo descobrir cada um deles por meio da busca de um vizinho do vértice recém descoberto que ainda é desconhecido pelo algoritmo. Mais precisamente: toma-se $v \in V_3$ qualquer para ser o primeiro vértice a ser avaliado. Daí, se $u \in V_3$ é o elemento avaliado na etapa atual, o algoritmo escolhe um vizinho de u que ainda não foi descoberto como o próximo elemento a ser visitado. Se a busca atingir um vértice que não possui vizinhos não-descobertos, retorna-se pelos vértices do percurso até achar algum que ainda possui vizinhos não-descobertos. O algoritmo funciona desta maneira até que todos os vértices atingidos não tenham mais vizinhos desconhecidos.

Se o conjunto dos vértices atingidos pela busca é igual a V_3, então há apenas uma componente conexa C_1 e obtém-se um passeio δ^1 desta componente C_1. O algoritmo é então encerrado.

Senão, significa que nem todos os vértices foram atingidos, e portanto, existem outras componentes C_j, com $j \geqslant 2$. Neste caso, reinicia-se a Busca em Profundidade nos vértices restantes. Este procedimento é repetido até que todos os vértice de $G_3 = (V_3, A_3)$ tenham sido descobertos. No fim, todas as componentes conexas C_j de G_3 são determinadas,

assim como um passeio δ^j que passa por todos os vértices de C_j, para cada C_j. Mais detalhes acerca do *DFS* podem ser encontrados em Bondy e Murty (2008) e Cormen et al. (2009).

A Figura 12 ilustra a execução da Busca em Profundidade no grafo $G_3 = (V_3, A_3)$ da Figura 5, que aqui está representado horizontalmente.

O Algoritmo 4 implementa a Busca em Profundidade para o grafo G_3.

Algoritmo 4 – DFSgrafoG3

Entrada: G_3: GrafoG3
Saída: P_3s: Lista de listas de $vertice3$

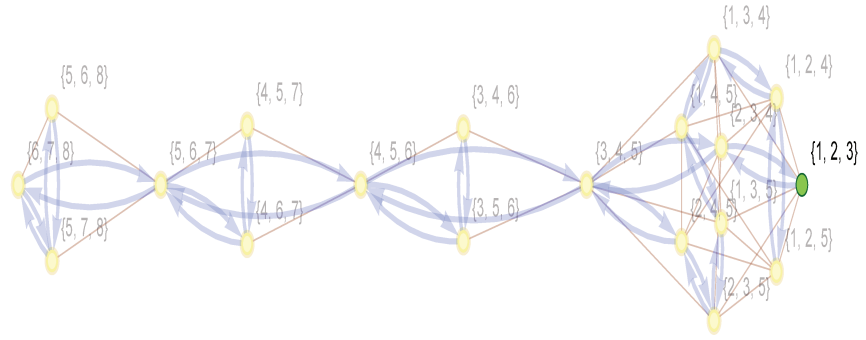
```
1  início
2      Conjunto(vertice3) NaoVisitados ← ∅;
3      P3s ← { };
4      NaoVisitados ← G3.V3();
5      enquanto NaoVisitados ≠ ∅ faça
6          pilha ← ∅;
7          pilha.Inserir(NaoVisitados.Menor());
8          NaoVisitados.Remover(NaoVisitados.Menor());
9          passeio ← { };
10         enquanto pilha ≠ ∅ faça
11             vertice3 vAtual ← pilha.Topo();
12             passeio.InserirNoFim(vAtual);
13             vertice3 viz;
14             viz ←EncontraVizinho(vAtual, NaoVisitados, G3);
15             se viz ≠ Nulo então
16                 pilha.Inserir(viz);
17                 NaoVisitados.Remover(viz);
18             senão
19                 pilha.Remover();
20         P3s.InserirNoFim(passeio);
21     retorna P3s;
```

Neste algoritmo, a função *EncontraVizinho* é responsável por retornar para a Busca em Profundidade um vizinho ainda não-descoberto do vértice u que está sendo avaliado naquele momento. *EncontraVizinho* é descrito em detalhes no Algoritmo 5.

Para achar um vizinho ainda não-visitado de algum $u \in V_3$, *EncontraVizinho* precisa utilizar algum critério de escolha deste vizinho. Visto que um conjunto ordenado

Figura 12 – Busca em Profundidade em $G_3 = (V_3, A_3)$.

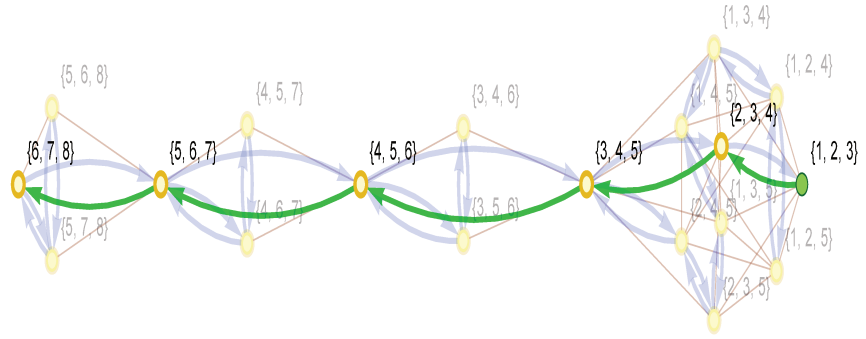
(a) Etapa 1



δ^1:

{1, 2, 3}	{2, 3, 4}	{3, 4, 5}	{4, 5, 6}	{5, 6, 7}	{6, 7, 8}	{5, 7, 8}	{5, 6, 8}	{5, 7, 8}	{6, 7, 8}	{5, 6, 7}
{4, 6, 7}	{4, 5, 7}	{4, 6, 7}	{5, 6, 7}	{4, 5, 6}	{3, 5, 6}	{3, 4, 6}	{3, 5, 6}	{4, 5, 6}	{3, 4, 5}	{2, 4, 5}
{2, 3, 5}	{1, 3, 5}	{1, 4, 5}	{1, 3, 4}	{1, 2, 4}	{1, 2, 5}	{1, 2, 4}	{1, 3, 4}	{1, 4, 5}	{1, 3, 5}	{2, 3, 5}
{2, 4, 5}	{3, 4, 5}	{2, 3, 4}	{1, 2, 3}							

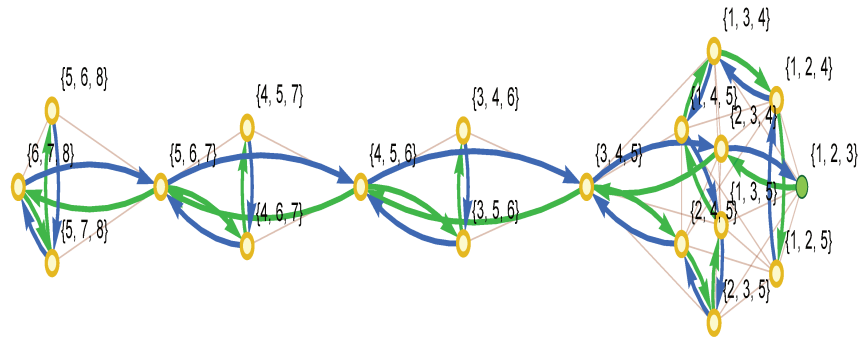
(b) Etapa 6



δ^1:

{1, 2, 3}	{2, 3, 4}	{3, 4, 5}	{4, 5, 6}	{5, 6, 7}	{6, 7, 8}	{5, 7, 8}	{5, 6, 8}	{5, 7, 8}	{6, 7, 8}	{5, 6, 7}
{4, 6, 7}	{4, 5, 7}	{4, 6, 7}	{5, 6, 7}	{4, 5, 6}	{3, 5, 6}	{3, 4, 6}	{3, 5, 6}	{4, 5, 6}	{3, 4, 5}	{2, 4, 5}
{2, 3, 5}	{1, 3, 5}	{1, 4, 5}	{1, 3, 4}	{1, 2, 4}	{1, 2, 5}	{1, 2, 4}	{1, 3, 4}	{1, 4, 5}	{1, 3, 5}	{2, 3, 5}
{2, 4, 5}	{3, 4, 5}	{2, 3, 4}	{1, 2, 3}							

(c) Etapa 37 (última etapa)



δ^1:

{1, 2, 3}	{2, 3, 4}	{3, 4, 5}	{4, 5, 6}	{5, 6, 7}	{6, 7, 8}	{5, 7, 8}	{5, 6, 8}	{5, 7, 8}	{6, 7, 8}	{5, 6, 7}
{4, 6, 7}	{4, 5, 7}	{4, 6, 7}	{5, 6, 7}	{4, 5, 6}	{3, 5, 6}	{3, 4, 6}	{3, 5, 6}	{4, 5, 6}	{3, 4, 5}	{2, 4, 5}
{2, 3, 5}	{1, 3, 5}	{1, 4, 5}	{1, 3, 4}	{1, 2, 4}	{1, 2, 5}	{1, 2, 4}	{1, 3, 4}	{1, 4, 5}	{1, 3, 5}	{2, 3, 5}
{2, 4, 5}	{3, 4, 5}	{2, 3, 4}	{1, 2, 3}							

Nota – O vértice de partida da Busca em Profundidade (elemento $\{1, 2, 3\}$) está colorido em *verde*. Os arcos em *verde* representam a descoberta de vértices, enquanto os arcos em *azul* ilustram o retorno ao elemento avaliado na etapa anterior da Busca em Profundidade. Um retorno é realizado quando o vértice avaliado não possui vizinhos ainda desconhecidos. O passeio δ^1 é obtido por meio do registro em sequência dos vértices avaliados pela Busca em Profundidade.

pode ser percorrido do menor elemento ao maior, um critério simples consiste em, ao percorrer os vizinhos de u, escolher o primeiro vértice não-descoberto encontrado. Isto exige que o objeto *GrafoG3* retenha em cada nó não somente um vertice3 e o seu conjunto de vizinhos, mas também um ponteiro para aquele vizinho a partir do qual a Busca em Profundidade deve iniciar a procura por um vértice desconhecido. A discussão acima sugere a seguinte definição:

Definição 12 (Estrutura de dados *GrafoG3* com atributo *criterio*)**.** *Representamos por*

$$GrafoG3 \tag{1.21}$$

a estrutura de dados que armazena um grafo G_3, que possui os seguintes atributos:

$$\begin{aligned} &\textit{VizinhancaG3: Mapa(} \\ &\qquad \textit{vertice3,} \\ &\qquad \textit{Conjunto(vertice3),} \\ &\qquad \textit{Ponteiro para nó de Conjunto(vertice3)} \\ &\qquad \textit{);} \end{aligned} \tag{1.22}$$

Referimo-nos à chave de VizinhancaG3 por $\boldsymbol{v}$*, ao valor Conjunto(vertice3) por* ***Vizinhos*** *e ao valor que é um ponteiro para nó de Conjunto(vertice3) por* ***criterio****. Então, para cada nó de VizinhancaG3, guarda-se:*

- *Em* $\boldsymbol{v}$*, um vértice de G_3;*
- *Em* ***Vizinhos****, todos os vértices que são adjacentes a* $\boldsymbol{v}$*, ou seja,*

  $$\textit{todo } \tau \in V_3 \textit{ tal que } \{\boldsymbol{v}, \tau\} \in A_3; \tag{1.23}$$

- *Em* ***criterio****, um ponteiro para um vizinho de* $\boldsymbol{v}$ *armazenado em* ***Vizinhos****, indicando o próximo vizinho a ser visitado durante o processo de construção de uma Floresta Geradora pela Busca em Profundidade.*

Inicialmente, o atributo ***criterio*** *aponta para o menor vértice de* ***Vizinhos****.*

A Figura 13 mostra os atributos de um objeto *GrafoG3* acrescido de **criterio**.

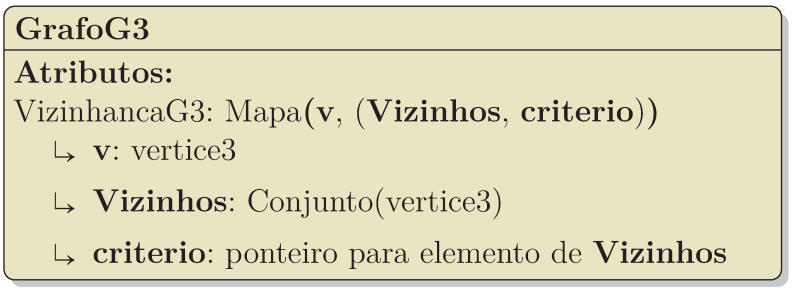
Ainda a respeito do Algoritmo 4, o bloco de linhas $6-20$ está encarregado de desvendar uma componente conexa C_j de $G_3 = (V_3, A_3)$. O algoritmo faz uso de uma pilha para guiar a avaliação dos vértices: cada iteração do *enquanto* da linha 10 equivale a uma etapa da Busca em Profundidade, e o vértice a ser avaliado naquela etapa corresponde aquele que está no topo da pilha, chamado de *vAtual*. Assim, entra na pilha um vértice

que é vizinho do *vAtual* daquela etapa. Uma vez que este vizinho está sendo descoberto, o mesmo deve ser removido do conjunto de vértices não-descobertos. Sai da pilha o *vAtual*, caso ele não possua vizinhos desconhecidos.

A sequência de vértices avaliados é armazenada na lista *passeio*. Quando a pilha torna-se enfim vazia, significa que aquela sequência representa um percurso completo de uma árvore geradora de alguma componente C_j, fazendo com que os vértices de *passeio* integrem o conjunto de vértices de C_j. Portanto, *passeio* corresponde ao passeio δ^j.

O *enquanto* da linha 5, por sua vez, garante que todas as componentes C_j e todos os respectivos passeios δ^j sejam obtidos ao fim do algoritmo. Todos os passeios são retornados como uma lista de passeios.

Figura 13 – Adição de atributo **criterio** a *GrafoG3*.



Algoritmo 5 – EncontraVizinho

Entrada: v: vertice3
$NaoVisitados$: Conjunto(vertice3)
G_3: GrafoG3
Saída: viz: vertice3

```
1  início
2      achou ← Falso;
3      Nó de VizinhancaG3 noh ← G3.Buscar(v);
4      Nó de Vizinhos nohViz ← noh.criterio;
5      enquanto (nohViz ≠ Nulo) E (¬achou) faça
6          viz ← NaoVisitados.Buscar(nohViz.v);
7          nohViz ← Itera+(nohViz);
8          se viz ≠ Nulo então
9              noh.criterio← nohViz;
10             achou ← Verdadeiro;
11     retorna viz
```

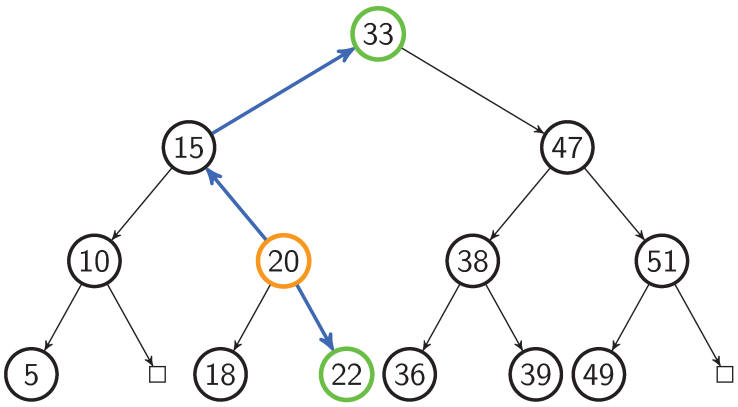
O algoritmo *EncontraVizinho* recorre, na linha 7, a um método chamado *Itera+*. Esta função é responsável por identificar o *vertice3* u imediatamente maior que o *vertice3* v passado como parâmetro. Se conjuntos de *vertice3* fossem armazenados em vetores ordenados e v ocupasse a posição i no vetor, o método equivalente ao *Itera+* para vetores retornaria ou a posição $i + 1$ ou o elemento desta posição.

Contudo, em árvores binárias de busca, o procedimento de *iterar para frente* demanda mais operações. Há dois possíveis candidatos ao vértice procurado por *Itera+*: o primeiro deles é o filho de v que é maior que v; o segundo é o primeiro elemento maior que v que é encontrado ao se partir de v em direção a raiz da árvore. Se o primeiro candidato existe, então é ele o vértice procurado. Senão, retorna-se um ponteiro para o segundo candidato.

No caso do segundo candidato também não existir, significa que v é o maior vértice da árvore, retornando um ponteiro *Nulo*.

A Figura 14 exemplifica o método *Itera+* numa árvore binária de busca que guarda inteiros em seus nós.

Figura 14 – Candidatos de iteração para frente em árvore rubro-negra.



Nota – O vértice em *laranja* (de valor 20) é aquele a partir do qual se pretende iterar para frente. Os vértices em *verde* são os candidatos a elemento imediatamente maior que o vértice em *laranja*: o filho de valor maior que 20 (de valor 22); e o primeiro antecessor no caminho do vértice em *laranja* à raiz que tem valor maior que 20 (neste caso, a própria raiz, de valor 33). Os arcos em *azul* ilustram os caminhos do vértice em *laranja* a cada um dos candidatos.

Itera+ avalia, no máximo, os vértices no caminho de uma folha à raiz. Como a altura de uma árvore rubro-negra é da ordem de $\log_2^n$, cujo n é o número de nós da árvore, o custo de operações básicas deste método é $O(\log_2^{|V|^3}) = O(3\log_2^{|V|}) = O(\log_2^{|V|})$ (CORMEN et al., 2009).

Por fim, calculamos a complexidade da Busca em Profundidade:

Teorema 4 (Complexidade de $DFSgrafoG3$)**.** *O custo das operações básicas de* DFSgrafoG3 *é igual a* $O(|V|^4 \log_2^{|V|})$.

Demonstração. O custo do Algoritmo 4 equivale ao custo do bloco de linhas $5-20$, que por sua vez é equivalente à soma dos custos de cada uma das operações efetuadas no bloco: inserções em *pilha*, *NaoVisitados* e *passeio*, remoções de *pilha* e *NaoVisitados* e chamadas à função *EncontraVizinho*.

Uma vez que cada vértice de $G_3 = (V_3, A_3)$ é descoberto e finalizado exatamente uma vez, há no máximo $|V|^3$ inserções e $|V|^3$ remoções na pilha, $|V|^3$ inserções em *passeio*, e $|V|^3$ inserções e $|V|^3$ remoções em $NaoVisitados$.

Como o custo de inserção e remoção em pilhas é $O(1)$, o custo total de suas inserções e remoções é $O(|V|^3)$. Similarmente, já que inserção no fim de listas custa $O(1)$, o custo de todas as inserções em *passeio* é $O(|V|^3)$.

Por outro lado, o custo de uma inserção ou uma remoção na árvore rubro-negra de $NaoVisitados$ é não mais que $O(\log_2^{|V|^3}) = O(\log_2^{|V|})$. Consequentemente, o custo de todas as inserções e remoções em $NaoVisitados$ é igual a $O(|V|^3 \log_2^{|V|})$.

Seja m_u o número de vizinhos do vértice u de G_3. Para cada $u \in V_3$, ao fim de $DFSgrafoG3$, o algoritmo $EncontraVizinho$ avalia exatamente uma vez cada elemento do conjunto **Vizinhos** associado a u. Portanto, o número de avaliações de todos os vizinhos é $\sum_{u \in V_3} m_u$. Cada aresta $\{u, v\}$ de A_3 está representada exatamente duas vezes em $GrafoG3$: uma com v em **Vizinhos** do nó de u e outra com u em **Vizinhos** do nó de v. Então, $\sum_{u \in V_3} m_u = 2|A_3|$. Como a busca pelo nó de $VizinhancaG3$ de um vértice u custa $O(\log_2^{|V|^3}) = O(\log_2^{|V|})$, concluímos que o custo total do uso de $EncontraVizinho$ em $DFSgrafoG3$ é igual a $O(|V|^4 \log_2^{|V|})$, pois $2|A_3|O(\log_2^{|V|}) \leqslant 2|V|^4 O(\log_2^{|V|}) = O(|V|^4 \log_2^{|V|})$.

Sumarizando, o custo de *DFSgrafoG3* é igual a:

$$
\begin{aligned}
&O(|V|^4 \log_2^{|V|}) \\
&\qquad = \\
&\qquad + 2O(|V|^3), \text{ das inserções e remoções na } pilha \\
&\qquad + O(|V|^3), \text{ das inserções em } passeio \\
&\qquad + 2O(|V|^3 \log_2^{|V|}), \text{ das inserções e remoções em } NaoVisitados \\
&\qquad + O(|V|^4 \log_2^{|V|}), \text{ do uso de } EncontraVizinho.
\end{aligned}
\tag{1.24}
$$

□

1.1.3 Construção de candidata a re-ordem

O método *ConstroiCandidataAReordem* é responsável por construir a sequência de vértices $seq_r = (r_1, r_2, \ldots, r_{3p})$ apresentada nas Equações (1.6) a partir de um passeio γ de G_3. Portanto, é aquele que executa a Etapa 3 do esboço do Algoritmo 1. *ConstroiCandidataAReordem* é descrito no Algoritmo 6.

Algoritmo 6 – ConstroiCandidataAReordem

Entrada: $passeioT_3$: Lista de vertice3
Saída: seq_r: Vetor de vértices de GrafoG

```
1  início
2      seq_r ←[ ];
3      nPasseioT_3 ← passeioT_3.Tamanho();
4      se nPasseioT_3 > 1 então
5          [γ_1^{*+}, γ_1^{1+}, γ_1^{2+}] ←DefineRotulos(passeioT_3[1], Verdadeiro);
6          seq_r.InserirNoFim(γ_1^{1+});
7          seq_r.InserirNoFim(γ_1^{2+});
8          seq_r.InserirNoFIm(γ_1^{*+});
9      senão
10         se nPasseioT_3 = 1 então
11             seq_r.InserirNofim(passeioT_3[1].v1);
12             seq_r.InserirNoFim(passeioT_3[1].v2);
13             seq_r.InserirNoFim(passeioT_3[1].v3);
14     γ_i ← passeioT_3[1].posterior();
15     para i = 2 até i ⩽ nPasseioT_3 faça
16         [γ_i^{*−}, γ_i^{1−}, γ_i^{2−}] ←DefineRotulos(γ_i, Falso);
17         seq_r.InserirNoFim(γ_i^{*−});
18         para j = 3(i − 1) − 2 até j ⩽ 3(i − 1) faça
19             se seq_r[j] = γ_i^{1−} então
20                 seq_r.InserirNoFim(γ_i^{1−});
21                 seq_r.InserirNoFim(γ_i^{2−});
22             senão
23                 se seq_r[j] = γ_i^{2−} então
24                     seq_r.InserirNoFim(γ_i^{2−});
25                     seq_r.InserirNoFim(γ_i^{1−});
26         γ_i ← γ_i.posterior();
27     retorna seq_r;
```

A sequência seq_r de vértices de V pode não ser uma re-ordem, já que há a chance de nem todos os vértices de G serem alcançados. Por outro lado, se o conjunto V é atingido por completo, seq_r é certamente uma re-ordem, pelo Teorema 2.

A função *DefineRotulos* exibida no Algoritmo 7 tem como papel atribuir aos elementos do passeio as notações positivas e negativas retratadas nas Equações (1.1), (1.2), (1.3), (1.4) e (1.5).

Algoritmo 7 – DefineRotulos

Entrada: $nohP$: Nó de lista de vertice3
$ehRotuloPositivo$: Booleano
Saída: $rotulos$: Vetor de vértices de G

```
1  início
2      singular ← 0;
3      rotulos ← [ ];
4      intersecao ← [ ];
5      vizinho ← [ ];
6      se ehRotuloPositivo então
7          vizinho ← nohP.posterior();
8      senão
9          vizinho ← nohP.anterior();
10     γ_k ←[nohP.v1, nohP.v2, nohP.v3];
11     γ_viz ←[vizinho.v1, vizinho.v2, vizinho.v3];
12     para i = 1 até i ⩽ 3 faça
13         ehSingular ← Verdadeiro;
14         para j = 1 até j ⩽ 3 faça
15             se γ_k[i] = γ_viz[j] então
16                 intersecao.InserirNoFim(γ_k[i]);
17                 ehSingular ← Falso;
18         se ehSingular então
19             singular ← γ_k[i];
20     rotulos.InserirNoFim(singular);
21     se intersecao[1] < intersecao[2] então
22         rotulos.InserirNoFim(intersecao[1]);
23         rotulos.InserirNoFim(intersecao[2]);
24     senão
25         rotulos.InserirNoFim(intersecao[2]);
26         rotulos.InserirNoFim(intersecao[1]);
27     retorna rotulos;
```

A Figura 8 mostra a aplicação dos Algoritmo 7 e Algoritmo 6 num passeio de $G_3 = (V_3, A_3)$.

Como conclusão desta subseção, calculamos a complexidade do método *ConstroiCandidataAReordem*.

Teorema 5 (Complexidade de *ConstroiCandidataAReordem*)**.** *O custo das operações básicas de* ConstroiCandidataAReordem *é igual a* $O(|PasseioT_3|)$.

Demonstração. O método *DefineRotulos* possui custo computacional de operações básicas $O(1)$, pois a inserção no fim de vetores custa $O(1)$. O acesso aos elementos anterior e posterior a um elemento u de uma lista encadeada também custa $O(1)$, e os dois *paras* encadeados geram nove repetições das linhas $15 - 17$.

O custo do Algoritmo 6 é proporcional ao custo de operações básicas do *para* da linha 15, pois todas as funções utilizadas no algoritmo possuem custo $O(1)$. Como o *para* da linha 18 realiza três iterações, o interior do bloco de linhas $16 - 26$ também custa $O(1)$. Então, o Algoritmo 6 tem custo $O(|passeioT_3|)$.

□

1.1.4 Gerador de re-ordem

O Algoritmo 8 tem por função realizar a Etapa 4 do esboço apresentado no Algoritmo 1, ou seja, checar se uma sequência de vértices de $G = (V, A)$ atinge todos os elementos de V.

Algoritmo 8 – ChecaSobrejetividadeReordem

Entrada: *seq*: Lista de vértices de GrafoG
nV: Número de vértices de GrafoG
Saída: ehSobrejetiva: Booleano

```
1  início
2      i ← 1;
3      nSeq ← seq.Tamanho();
4      Conjunto(Inteiro) verticesAtingidos;
5      enquanto (verticesAtingidos.Tamanho() < nV) E (i ⩽ nSeq) faça
6          verticesAtingidos.Inserir(seq[i]);
7          i + +;
8      se verticesAtingidos.Tamanho() = nV então
9          ehSobrejetiva ← Verdadeiro;
10     senão
11         ehSobrejetiva ← Falso;
12     retorna ehSobrejetiva;
```

ChecaSobrejetividadeReordem assume que os inteiros da sequência *seq* passada como entrada são vértices de G, e que esta sequência satisfaz os três itens da Definição 7.

Então, o algoritmo conta quantos inteiros diferentes *seq* possui. Se esta quantidade for igual a $|V|$, *seq* é de fato uma re-ordem de G, e o algoritmo retorna *Verdadeiro*. No caso contrário, retorna-se *Falso*.

Finalmente, ao se juntar num único método todos os algoritmos apresentados, temos o Algoritmo 9, que gera uma re-ordem de um grafo $G = (V, A)$ simples e não direcionado.

Algoritmo 9 – GeraReordem

Entrada: G: GrafoG
Saída: reordem: Vetor de vértices de GrafoG

```
1  início
2      ehReordem ← Falso;
3      reordem ← ∅;
4      GrafoG3 G3 ← ConstroiG3(G);
5      P3s ← DFSgrafoG3(G3);
6      para cada passeio ∈ P3s faça
7          seq ← ConstroiCandidataAReordem(passeio);
8          ehReordem ← ChecaSobrejetividadeReordem(seq, G.nV);
9          se ehReordem então
10             reordem ← seq;
11             Interromper para;
12     retorna reordem;
```

Teorema 6 (Complexidade de *GeraReordem*)**.** *O custo das operações básicas de* GeraReordem *é igual a* $O(|V|^4 \log_2^{|V|})$.

Demonstração. Seja *seq* a sequência de vértices de V passada como entrada para o Algoritmo 8. O custo das operações básicas de *ChecaSobrejetividadeReordem* é $|seq|O(\log_2^{|seq|})$, pois o *enquanto* da linha 5 é executado $|seq|$ vezes, e a inserção da linha 6 na árvore rubro-negra *verticesAtingidos* custa $O(\log_2^{|seq|})$.

Agora, calculamos o custo de operações básicas de *GeraReordem*. As linhas 4 e 5 têm custos iguais a $O(|V|^4 \log_2^{|V|})$, os custos de $ConstroiG3$ e $DFSgrafoG3$, respectivamente (Teorema 3 e Teorema 4).

O custo total de todas as execuções possíveis da linha 7 na estrutura de repetição *para* é igual a soma dos custos de *ConstroiCandidataAReordem* para cada passeio de P_3s. Então, pelo Teorema 5, este custo total é $\sum_{passeio \in P_3s} O(|passeio|) = O(2|V_3| + k) = O(|V^3|)$, cujo k é o número de componentes conexas de G_3.

Similarmente, o custo total de todas as aplicações do Algoritmo 8 é igual a soma dos custos de *ChecaSobrejetividadeReordem* de todas as sequências *seq* obtidas pelo Algoritmo 6, ou seja, $\sum_{seq} |seq| O(\log_2^{|seq|})$. Então, como $\sum_{seq} |seq| = 3(2|V_3| + k)$, o custo total do Algoritmo 8 é da ordem de $3(2|V_3|+k)O(\log_2^{3(2|V_3|+k)}) = O(|V|^3 \log_2^{|V|^3}) = O(|V|^3 \log_2^{|V|})$.

Em conclusão, o custo de *GeraReordem* é $O(|V|^4 \log_2^{|V|}) + O(|V|^4 \log_2^{|V|}) + O(|V|^3) + O(|V|^3 \log_2^{|V|}) = O(|V|^4 \log_2^{|V|})$.

□

1.2 Razão de Uso da Busca em Profundidade

A designação da Busca em Profundidade para determinar um passeio γ de $G_3 = (V_3, A_3)$, tal que $\bigcup_{i=1}^{p} \gamma_i = V$, pretende obter uma re-ordem da qual se possa extrair uma ${}^3PDGDM$ ordem.

As ordens estão relacionadas com uma propriedade peculiar de um ${}^3PDGDM$: a quantidade de soluções para o problema é sempre uma potência de *dois*. Liberti et al. (2014b) mostram esta propriedade ao descobrir a existência de uma forma de simetria entre as soluções de um ${}^3PDGDM$. Estas simetrias podem ser usadas para obter, em tempo linear, qualquer outra solução X_j de um ${}^3PDGDM$, a partir de uma solução X_i já descoberta e da própria ${}^3PDGDM$ *ordem* (LIBERTI et al., 2014b).

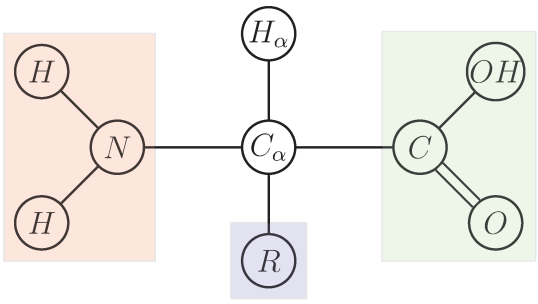
Para fundamentar a pretensão de encontrar também uma ${}^3PDGDM$ ordem, faz-se necessário apresentar algumas propriedades atreladas às moléculas de proteínas.

Uma proteína é um polímero, ou seja, uma molécula formada pela ligação em sequência de moléculas menores ditas monômeros. No caso das proteínas, estes monômeros são os aminoácidos. A grande maioria das proteínas são sintetizadas a partir de conjunto de 20 aminoácidos comuns (NELSON; COX, 2014, p. 75).

Parte da estrutura química dos aminoácidos se repete: um grupamento amino (NH_2) está ligado a um carbono central, dito *carbono alfa* (C_α), que, por sua vez, está ligado a um grupamento carboxila ($COOH$). Por fim, há um hidrogênio ligado ao C_α, chamado de *hidrogênio alfa* (H_α) (NELSON; COX, 2014, p. 76). A distinção entre eles se dá por um conjunto de átomos que se conecta ao C_α, denominado de *grupo R* ou *cadeia lateral* (NELSON; COX, 2014, p. 76).

A Figura 15 ilustra as partes mencionadas de um aminoácido.

Figura 15 – Estrutura química geral de um aminoácido.



Nota – Em rosa está destacado o grupamento amino, em verde o grupamento carboxila e em azul a cadeia lateral.

Um aminoácido liga-se a outro por meio de uma ligação covalente particular.

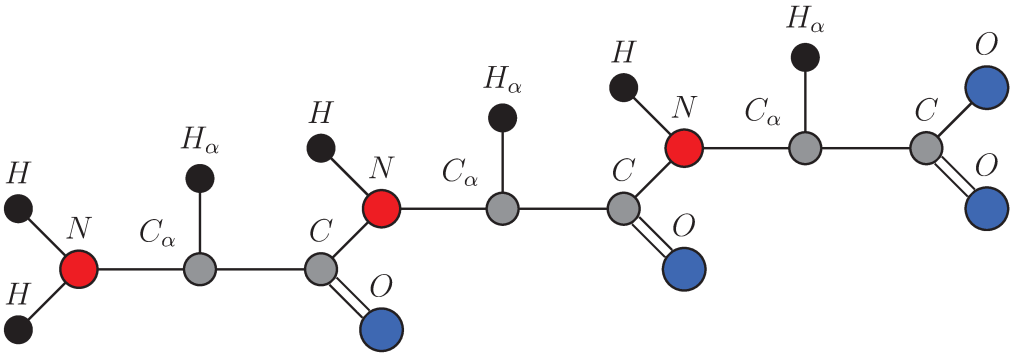
A chamada *ligação peptídica* conecta o carbono da carboxila do primeiro aminoácido e o nitrogênio do grupo amino do segundo aminoácido. A ligação ocorre quando a hidroxila (OH) do primeiro aminoácido junta-se a um dos hidrogênios do segundo aminoácido para a liberação de uma molécula de água (NELSON; COX, 2014, p. 85-86).

Então, quando conectadas em sequência, as moléculas de aminoácidos diferem de suas moléculas originais, pois faltarão os dois átomos de hidrogênio e o átomo de oxigênio associados àquela molécula de água que é liberada para a criação da ligação peptídica. Esta molécula reduzida de aminoácido é denominada de *resíduo* (NELSON; COX, 2014, p. 86).

Esta natureza sequencial das proteínas possibilita a definição do início e do fim da molécula. Por convenção, o resíduo que possui o grupo amino completo (*resíduo aminoterminal*) é considerado o primeiro resíduo, ou seja, é aquele representado na extremidade esquerda da proteína, enquanto o resíduo que possui o grupo carboxila completo (*resíduo carboxiterminal*) é considerado o último resíduo, disposto na extremidade direita (NELSON; COX, 2014, p. 86).

Desta forma, define-se como *Estrutura Primária da Proteína* a descrição de todas as ligações covalentes que unem os resíduos da proteína. O principal elemento da estrutura primária é a sua sequência de aminoácidos (NELSON; COX, 2014, p. 96-97). Por fim, chama-se de *cadeia principal* ou *esqueleto* da proteína a parte da molécula constituída pelas sequências de $N - C_\alpha - C$ (NELSON; COX, 2014, p. 118). A Figura 16 mostra o esqueleto de uma molécula de 3 resíduos de aminoácidos em conjunto com os átomos de oxigênio e hidrogênio ligados a este esqueleto.

Figura 16 – Representação de cadeia de três aminoácidos, omitindo-se os grupos R.



Fonte: Adaptado de Lavor (2019)

Ao se considerar proteínas como estruturas *rígidas* (LAVOR, 2019), conhece-se, *a priori*, as distâncias exatas entre átomos conectados por ligação covalente e os ângulos entre átomos **a**, **b** e **c**, tais que **a** e **b** e **b** e **c** estão conectados covalentemente (GIBSON;

SCHERAGA, 1997; JACKSON; JORDÁN, 2008; LAVOR et al., 2019; LAVOR, 2019; LAVOR et al., 2017; LAVOR; LIBERTI, 2014). Consequentemente, a distância entre **a** e **c** também é conhecida (JACKSON; JORDÁN, 2008; LAVOR et al., 2019; LAVOR et al., 2017).

Em adição, outras distâncias são provindas da seguinte característica das ligações peptídicas: se C^i e N^{i+1} são os átomos que compõem a ligação peptídica entre os *i-ésimo* e *i+1-ésimo* resíduos de uma proteína, então os átomos C^i_α e O^i ligados a C^i, os átomos C^{i+1}_α e H^{i+1} ligados a N^{i+1}, e os próprios C^i e N^{i+1} estão presentes num mesmo plano. Tal plano é chamado de *plano peptídico* (NELSON; COX, 2014, p. 117-118). Portanto, são conhecidas todas as distâncias exatas entre os *seis* átomos contidos no plano (LAVOR et al., 2019).

Por outro lado, pode-se obter limites inferiores e superiores para as distâncias entre alguns pares de átomos dos quais não se conhecem as distâncias exatas. O procedimento experimental de *Ressonância Magnética Nuclear* determina intervalos para as distâncias de pares de átomos de hidrogênio que estão a, no máximo, 5 Angstroms (Å) de distância (HAVEL; WÜTHRICH, 1984).

Da natureza sequencial das proteínas e das distâncias conhecidas entre seus átomos, derivam-se duas propriedades particulares dos grafos de ${}^3PDGDM$ associados a cadeias principais de proteínas, incluindo os hidrogênios e oxigênios ligados ao esqueleto: **(a)** a matriz de adjacência de grafos deste tipo é esparsa; **(b)** como a rotulação dos vértices segue a ordem dos resíduos de aminoácidos da proteína, tal matriz possui largura de banda pequena.

Desta forma, quando nos referirmos a um ${}^3PDGDM$ de proteína, estaremos considerando o ${}^3PDGDM$ associado à cadeia principal, como em Lavor et al. (2019).

A Figura 17 ilustra a matriz de adjacência do grafo do ${}^3PDGDM$ associado à molécula Alfa-T-Alfa (PDB ID: 1abz).

Empiricamente, os grafos $G_3 = (V_3, A_3)$ de ${}^3PDGDM$ de proteínas parecem apresentar as mesmas propriedades mencionadas. A Figura 18 ilustra a matriz de adjacência do grafo G_3 da molécula Alfa-T-Alfa (PDB ID: 1abz).

Assim como ilustrado na Figura 12b, a Busca em Profundidade descobre um novo vértice que é vizinho de um vizinho do vértice atual u antes de explorar todos os vizinhos de u. Ao ser utilizada na *Etapa 2* do Algoritmo 1, a cada iteração, pode-se verificar se o passeio obtido até o momento já atingiu todo o conjunto V de $G = (V, A)$. Isto possibilita encontrar passeios de tamanhos menores do que o esperado.

Portanto, quando este método é aplicado em grafos de largura de banda pequena, aumentam-se as chances de obtenção de passeios de menor tamanho possível que atinjam todos os vértices de $G = (V, A)$. Finalmente, afirmando-se que alguns destes passeios de

Figura 17 – Matriz de Adjacência de $G = (V, A)$ associada à instância *1abz* do *PDB*.

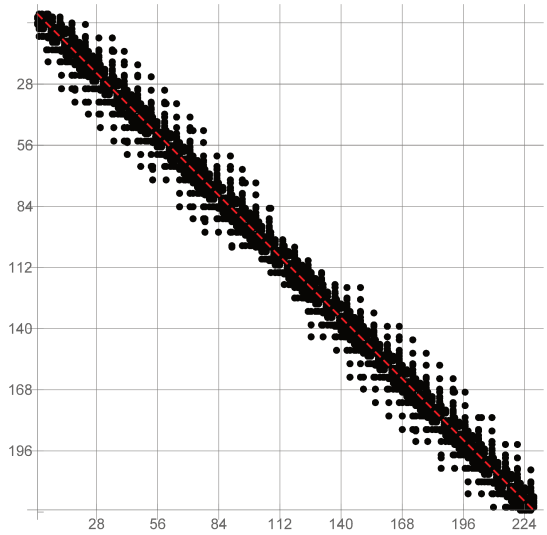
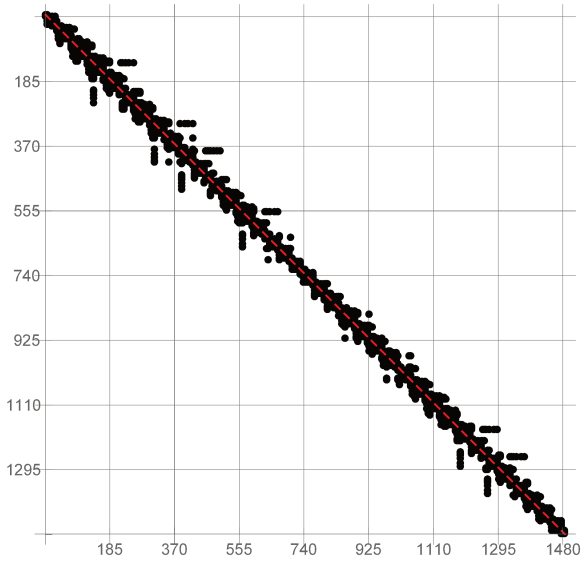


Figura 18 – Matriz de Adjacência de $G_3 = (V_3, A_3)$ associada à instância *1abz* do *PDB*.



$G_3 = (V_3, A_3)$ podem gerar ${}^3PDGDM$ ordens, estaria justificada a escolha da Busca em Profundidade.

Corolário 3. *Seja $G = (V, A)$ um grafo simples e não direcionado. Se existe uma ${}^3PDGDM$ ordem em G, então existe um caminho γ em $G_3 = (V_3, A_3)$ tal que* $\bigcup_{i=1}^{|V|-2} \gamma_i = V$.

Demonstração. [⇒] Suponhamos que exista uma ${}^3PDGDM$ ordem z. Pelas definições de ${}^3PDGDM$ ordem e re-ordem, z é também uma re-ordem. Daí, pelo Teorema 1, os

seguintes trios formam um passeio γ em $G_3 = (V_3, A_3)$, tal que $\bigcup_{i=1}^{|V|-2} \gamma_i = V$:

$$\begin{aligned} \gamma_1 &= \{z_1, z_2, z_3\}; \\ \gamma_2 &= \{z_2, z_3, z_4\}; \\ &\vdots \\ \gamma_i &= \{z_i, z_{i+1}, z_{i+2}\}; \\ &\vdots \\ \gamma_{|z|-2} &= \{z_{|z|-2}, z_{|z|-1}, z_{|z|}\}. \end{aligned} \tag{1.25}$$

Como z é uma sequência constituída por elementos distintos, por construção, todos os elementos de γ são também distintos. Portanto, γ é um caminho em G_3.

□

O Corolário 3 mostra como é possível construir um caminho de $|V|-2$ elementos de $G_3 = (V_3, A_3)$ que atinge todos os vértices de $G = (V, A)$, a partir de uma ^3PDGDM ordem. De maneira inversa, com a construção de uma sequência s como aquela ilustrada na Figura 7, pode-se reaver a ordem z removendo-se de s, a partir do quarto elemento, os vértices que são repetidos. Chamemos esta sequência final de $\overline{s}$.

A possibilidade de obtenção de uma ^3PDGDM ordem de um caminho $\gamma = (\gamma_i)_{i=0}^{|V|-2}$, tal que $\bigcup_{i=1}^{|V|-2} \gamma_i = V$, leva-nos ao desenvolvimento do seguinte teorema:

Teorema 7. *Seja $G = (V, A)$ um grafo simples e não direcionado. Se γ é um passeio em $G_3 = (V_3, A_3)$ de $|V| - 2$ elementos, tal que $\bigcup_{i=1}^{|V|-2} \gamma_i = V$, então γ é um caminho em G_3.*

Demonstração. Suponhamos, por contradição, que γ não seja um caminho. Então, algum vértice $\tau \in V_3$ aparece pelo menos duas vezes em γ.

Ao percorrer o passeio γ, registramos os vértices de G que são descobertos por cada um dos elementos γ_i, à medida que são avaliados. Então, na avaliação do primeiro elemento do passeio, γ_1, descobrem-se três vértices: $\gamma_{1,1}$, $\gamma_{1,2}$ e $\gamma_{1,3}$. De γ_2 em diante, a avaliação de cada γ_i descobre no máximo um novo vértice de G, que seria $\gamma_i \backslash \gamma_{i-1}$. Como τ se repete em algum γ_j de γ, *nenhum* novo vértice de G é descoberto na *j-ésima* avaliação. Então, o número de vértices de G representados em γ é, *no máximo*, $|V| - 1$, pois:

$$|\bigcup_{i=1}^{|V|-2} \gamma_i| = 3 + |\bigcup_{i=2}^{|V|-2} \gamma_i| \leqslant 3 + (|\gamma| - 2) = 3 + (|V| - 2 - 2) = |V| - 1. \tag{1.26}$$

Isto é uma contradição, já que $\bigcup_{i=1}^{|V|-2} \gamma_i = V$. Portanto, γ é caminho em G_3.

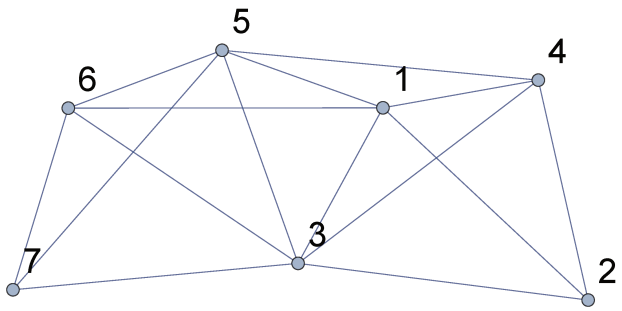
□

Pelo Teorema 7, pode-se afirmar que passeios de $G_3 = (V_3, A_3)$ de $|V| - 2$ elementos, que atingem todos os vértice de $G = (V, A)$, são certamente caminhos em G_3. Como consequência imediata, os menores passeios de G_3 que alcançam V por completo possuem $|V| - 2$ elementos.

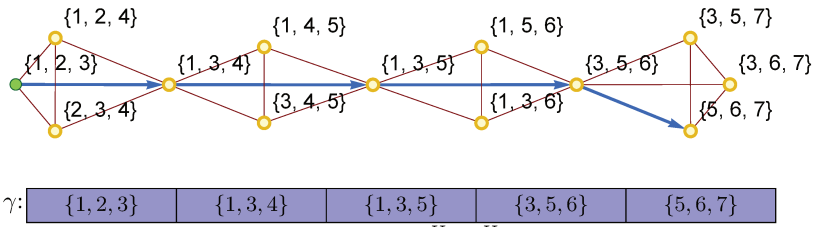
Deve-se salientar, no entanto, que nem sempre é possível extrair uma ${}^3PDGDM$ ordem por meio da sequência s descrita anteriormente. A Figura 19 mostra um caminho de $H_3 = (V_3^H, A_3^H)$ deste tipo. As sequências do tipo $\overline{s}$ possíveis para este exemplo são: (a) $(1, 2, 3, 4, 5, 6, 7)$; (b) $(1, 3, 2, 4, 5, 6, 7)$; (c) $(2, 1, 3, 4, 5, 6, 7)$; (d) $(2, 3, 1, 4, 5, 6, 7)$; (e) $(3, 1, 2, 4, 5, 6, 7)$ e (f) $(3, 2, 1, 4, 5, 6, 7)$. Para qualquer uma das combinações dos inteiros do primeiro vértice do caminho, a sequência $\overline{s}$ obtida desrespeita a definição de ${}^3PDGDM$ ordem: o *sexto* elemento, que é sempre o inteiro 6, não está conectado por uma aresta ao *quarto* elemento, que é sempre o inteiro 4.

Figura 19 – Caminho no grafo $H_3 = (V_3^H, A_3^H)$ associado ao grafo simples e não direcionado $H = (V^H, A^H)$

(a) Grafo $H = (V^H, A^H)$



(b) Caminho em $H_3 = (V_3^H, A_3^H)$



γ:	$\{1,2,3\}$	$\{1,3,4\}$	$\{1,3,5\}$	$\{3,5,6\}$	$\{5,6,7\}$

Nota – O primeiro vértice do caminho γ em $H_3 = (V_3^H, A_3^H)$ está colorido em *verde* (elemento $\{1, 2, 3\}$). Os arcos em *azul* representam a sucessão entre os vértice de γ.

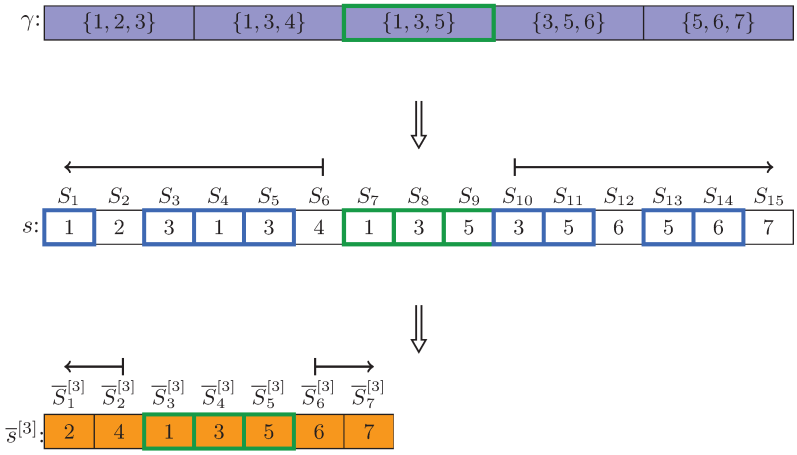
Além das sequências do tipo $\overline{s}$, há outras formas de se obter sequências que podem vir a ser ${}^3PDGDM$ ordens. Dado um passeio $\gamma = (\gamma_i)_{i=1}^{|V|-2}$ de $G_3 = (V_3, A_3)$, tal

que $\bigcup_{i=1}^{|V|-2} \gamma_i = V$, escolhendo-se um elemento γ_j qualquer, obtém-se um caminho $\overline{s}^{[j]}$ de $G = (V, A)$ da seguinte maneira:

1. Constrói-se a sequência s atrelada a γ, como na Figura 7;
2. Escolhe-se um dos arranjos possíveis para a trinca de vértices de G associados a γ_j;
3. Partindo-se dos elementos associados de γ_{j+1} até $\gamma_{|V|-2}$, ou seja, para $k \in \{j+1, j+2, \ldots, |V|-2\}$, eliminam-se aqueles elementos associados a γ_k que também pertencem a γ_{k-1};
4. Partindo-se dos elementos associados de γ_{j-1} até γ_1, ou seja, para $k \in \{1, 2, \ldots, j-1\}$, eliminam-se aqueles elementos associados a γ_k que também pertencem a γ_{k+1}.

A Figura 20 exemplifica este procedimento ao aplicá-lo no caminho da Figura 19b. Além disto, tal procedimento está precisamente descrito no Algoritmo 10.

Figura 20 – Sequência $\overline{s}^{[3]}$.



Nota – No topo, o caminho γ no grafo $H_3 = (V_3^H, A_3^H)$ da Figura 19b. O terceiro elemento de γ é escolhido (em *verde*). No meio, a sequência s de vértices de $H = (V^H, A^H)$ da Figura 19a construída pela justaposição dos inteiros de γ. Escolhe-se um arranjo possível para 3-clique escolhida (neste caso, o arranjo crescentemente ordenado: 1, 3, 5). A partir da trinca em *verde*, eliminam-se os elementos repetidos tanto à esquerda quanto à direita (em *azul*). Então, na base, obtém-se uma ${}^3PDGDM$ ordem $\overline{s}^{[3]}$ de H.

Verifica-se facilmente que a sequência $\overline{s}^{[3]}$ da Figura 20 é uma re-ordem do grafo $H = (V^H, A^H)$ da Figura 19a.

A próxima subseção propõe alterações no método para encontrar re-ordens da seção 1.1 que visam melhorar o seu desempenho. Estas alterações não aumentam o limite superior para o custo de operações básicas.

Algoritmo 10 – GeraSbarraI

Entrada: γ: Caminho de $G_3 = (V_3, A_3)$;
i: Inteiro positivo menor ou igual a $|\gamma|$;
$arranjoI$: Trio de Inteiros de γ_i arranjados numa dada ordem.

Saída: $\overline{s}^{[i]}$: Vetor de Inteiros.

início
- $\overline{s}^{[i]} \leftarrow$ Vetor de Inteiros de tamanho $|\gamma| + 2$;
- **para** $j = 1$ **até** $j \leqslant |\gamma|$ **faça**
  - $[\,\overline{s}^{[i]}_{j},\ \overline{s}^{[i]}_{j+1},\ \overline{s}^{[i]}_{j+2}\,] \leftarrow [\,\gamma_{j,1},\ \gamma_{j,2},\ \gamma_{j,3}\,]$;
- $[\,\overline{s}^{[i]}_{3i-2},\ \overline{s}^{[i]}_{3i-1},\ \overline{s}^{[i]}_{3i},\,] \leftarrow arranjoI$;
- **para** $j = i - 1$ **até** $j = 1$ **faça**
  - **se** $\overline{s}^{[i]}_{3j-2} \in \{\,\overline{s}^{[i]}_{3(j+1)-2},\ \overline{s}^{[i]}_{3(j+1)-1},\ \overline{s}^{[i]}_{3(j+1)}\,\}$ **então**
    - Remover $\overline{s}^{[i]}_{3j-2}$;
  - **se** $\overline{s}^{[i]}_{3j-1} \in \{\,\overline{s}^{[i]}_{3(j+1)-2},\ \overline{s}^{[i]}_{3(j+1)-1},\ \overline{s}^{[i]}_{3(j+1)}\,\}$ **então**
    - Remover $\overline{s}^{[i]}_{3j-1}$;
  - **se** $\overline{s}^{[i]}_{3j} \in \{\,\overline{s}^{[i]}_{3(j+1)-2},\ \overline{s}^{[i]}_{3(j+1)-1},\ \overline{s}^{[i]}_{3(j+1)}\,\}$ **então**
    - Remover $\overline{s}^{[i]}_{3j}$;
- **para** $j = i + 1$ **até** $j = |\gamma|$ **faça**
  - **se** $\overline{s}^{[i]}_{3j-2} \in \{\,\overline{s}^{[i]}_{3(j-1)-2},\ \overline{s}^{[i]}_{3(j-1)-1},\ \overline{s}^{[i]}_{3(j-1)}\,\}$ **então**
    - Remover $\overline{s}^{[i]}_{3j-2}$;
  - **se** $\overline{s}^{[i]}_{3j-1} \in \{\,\overline{s}^{[i]}_{3(j-1)-2},\ \overline{s}^{[i]}_{3(j-1)-1},\ \overline{s}^{[i]}_{3(j-1)}\,\}$ **então**
    - Remover $\overline{s}^{[i]}_{3j-1}$;
  - **se** $\overline{s}^{[i]}_{3j} \in \{\,\overline{s}^{[i]}_{3(j-1)-2},\ \overline{s}^{[i]}_{3(j-1)-1},\ \overline{s}^{[i]}_{3(j-1)}\,\}$ **então**
    - Remover $\overline{s}^{[i]}_{3j}$;
- **retorna** $\overline{s}^{[i]}$;

1.2.1 Algoritmo Modificado

Em razão da esparsidade e da baixa largura de banda dos grafos $G_3 = (V_3, A_3)$ de ${}^3PDGDM$ de proteínas, não é necessário checar todos os subconjuntos de *quatro* vértices de $G = (V, A)$ para determinar G_3.

É possível encontrar todas as 4-cliques de arestas associadas a distâncias exatas apenas com a verificação dos quartetos de vértices cujos rótulos distam não mais que 13 unidades, uma vez que: **(a)** as distâncias exatas do problema ocorrem entre pares de átomos que estão ou no mesmo resíduo ou em resíduos sucessivos; **(b)** cada resíduo tem, no máximo, *sete átomos*: N, C_α, C, O, H, H_α e H' *ou* O'. Os dois últimos átomos

correspondem, respectivamente, ao segundo hidrogênio do grupo amino do primeiro resíduo e ao oxigênio da hidroxila do último resíduo.

Quanto às distâncias intervalares, sabemos que ocorrem somente entre hidrogênios que estão a 5 Å de proximidade, no máximo. Então, guardando-se o conjunto de vizinhos de cada átomo de hidrogênio, é possível determinar aquelas 4-cliques que têm alguma aresta de distância intervalar por meio da inspeção dos quartetos de vértices formados por um átomo de hidrogênio e três de seus vizinhos.

Concretamente, tais ideias exigem modificações na estrutura de dados que retém um grafo simples e não direcionado e no algoritmo que constrói o grafo $G_3 = (V_3, A_3)$, o que motiva a próxima definição.

Definição 13 (Estrutura de dados *GrafoG* com *VizinhancaH*)**.** *Representamos por*

$$GrafoG \tag{1.27}$$

a estrutura de dados que armazena um grafo simples e não direcionado de um ^3PDGDM *de proteína. Os seus atributos são:*

$$\begin{cases} nV\text{: Inteiro;} \\ A\text{: Conjunto(aresta);} \\ VizinhancaH\text{: Mapa}\big(\text{Inteiro, Conjunto(Inteiro)}\big). \end{cases} \tag{1.28}$$

Referimo-nos ao objeto aresta de um nó de A por $\boldsymbol{a}$*, à chave de VizinhancaH por* $\boldsymbol{v}$ *e ao valor Conjunto(vertice3) por* ***Vizinhos****.*

Guarda-se em nV *o número de vértices. Cada vértice do grafo é representado por um rótulo inteiro positivo entre* 1 *e* nV*. As arestas são armazenadas na árvore binária de busca rubro-negra* A*. Por fim, em cada nó de* ***VizinhancaH****, guarda-se:*

- *Em* $\boldsymbol{v}$*, um vértice de* G *que representa um hidrogênio;*
- *Em* ***Vizinhos****, todos os vértices que são adjacentes a* $\boldsymbol{v}$*, ou seja,*

  $$\text{todo } u \in V \text{ tal que } \{\boldsymbol{v}, u\} \in A. \tag{1.29}$$

A Figura 21 mostra um esquema dos atributos de um objeto *GrafoG* que retém os vizinhos dos hidrogênios.

O bloco de linhas $7 - 20$ do Algoritmo 2 foi transformado numa função de nome *AtribuiVizinhos*, descrita no Algoritmo 11. O novo método de construção de $G_3 = (V_3, A_3)$ é descrito no Algoritmo 12.

Algoritmo 11 – AtribuiVizinhos

Entrada: G: GrafoG
G_3: GrafoG3
quarteto: Vetor de 4 inteiros
Saída: *eh4clique*: Booleano

```
1  início
2      [i, j, k, l] ← quarteto;
3      eh4clique ← Verifica4CliqueG(G, i, j, k, l);
4      se eh4clique então
5          G3.InserirVizinho({i, j, k}, {i, j, l});
6          G3.InserirVizinho({i, j, k}, {i, k, l});
7          G3.InserirVizinho({i, j, k}, {j, k, l});
8          G3.InserirVizinho({i, j, l}, {i, j, k});
9          G3.InserirVizinho({i, j, l}, {i, k, l});
10         G3.InserirVizinho({i, j, l}, {j, k, l});
11         G3.InserirVizinho({i, k, l}, {i, j, k});
12         G3.InserirVizinho({i, k, l}, {i, j, l});
13         G3.InserirVizinho({i, k, l}, {j, k, l});
14         G3.InserirVizinho({j, k, l}, {i, j, k});
15         G3.InserirVizinho({j, k, l}, {i, j, l});
16         G3.InserirVizinho({j, k, l}, {i, k, l});
17     retorna eh4clique;
```

Figura 21 – Atributos de *GrafoG* com *VizinhancaH*.

GrafoG

Atributos:
nV: Inteiro
A: Conjunto(**a**)
↳ **a**: aresta
VizinhancaH: Mapa(**v**,**Vizinhos**)
↳ **v**: Inteiro
↳ **Vizinhos**: Conjunto(Inteiro)

O método *InserirVizinho(**a**,**b**)* insere o vertice3 **b** no conjunto de vizinhos do nó que tem o vertice3 **a** como chave.

Algoritmo 12 – ConstroiG3

Entrada: G: GrafoG
Saída: G_3: GrafoG3 originado de G

```
1  início
2      GrafoG3 G3;
3      alcance ← 13;
4      para i = 1 até i ⩽ G.nV − 3 faça
5          fj ←Min(i + alcance − 2, G.nV − 2);
6          para j = i + 1 até j ⩽ fj faça
7              fk ←Min(i + alcance − 1, G.nV − 1);
8              para k = j + 1 até k ⩽ fk faça
9                  fl ←Min(i + alcance, G.nV);
10                 para l = k + 1 até l ⩽ fl faça
11                     AtribuiVizinhos(G, G3, [i, j, k, l]);
12     para cada h ∈ G.VizinhancaH faça
13         maiorViz ← h.Vizinhos.MaiorVizinho();
14         fj ←Itera-(maiorViz, 2);
15         para v1 ← h.Vizinhos.MenorVizinho() até v1 = fj faça
16             fk ←Itera-(maiorViz, 1);
17             para v2 ←Itera+(v1) até v2 = fk faça
18                 fl ← maiorViz;
19                 para v3 ←Itera+(v2) até v3 = fl faça
20                     quarteto ←Ordena([h.v, v1.v, v2.v, v3.v]);
21                     AtribuiVizinhos(G, G3, quarteto);
22     retorna G3;
```

O método *Itera-* é similar ao *Itera+* apresentado na subseção 1.1.2 e ilustrado na Figura 14. Porém, ao invés de retornar o elemento imediatamente maior que o elemento u passado como parâmetro, retorna o elemento imediatamente menor. Um número inteiro n maior ou igual a 1 pode ser passado como segundo parâmetro e faz com que *Itera-* retorne o *n-ésimo* elemento imediatamente menor que u. Por exemplo, *Itera-(u,2)* retorna o elemento imediatamente menor que o elemento imediatamente menor que u. *Itera-(u,1)* é equivalente a *Itera-(u)*.

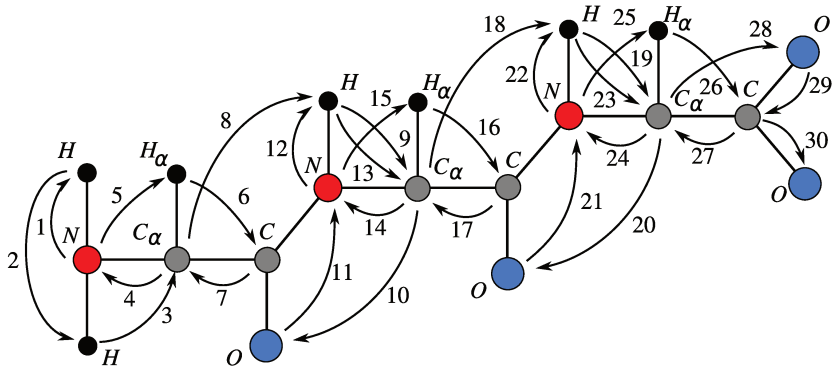
Por fim, uma vez que o próximo capítulo trata de resultados da execução do algoritmo polinomial para encontrar re-ordens, deve-se assinalar dois conjuntos de informações.

O primeiro deles refere-se à existência de re-ordens. Lavor et al. (2019) mostram que, para qualquer instância do ^3PDGDM associado a uma proteína, existe sempre uma re-ordem particular, a qual dar-se o nome de *re-ordem* ***hc*** (do inglês, *hand-crafted*). Segue abaixo a definição desta re-ordem atrelada a um ^3PDGDM de uma proteína de p resíduos de aminoácidos (ver Figura 22):

$$\begin{aligned}\mathbf{hc} = \{&N^1, H^1, H^{1'}, C^1_\alpha, N^i, H^1_\alpha, C^1, C^1_\alpha, \dots \\ &H^i, C^i_\alpha, O^{i-1}, N^i, H^i, C^i_\alpha, N^i, H^i_\alpha, C^i, C^i_\alpha \dots \\ &H^p, C^p_\alpha, O^{p-1}, N^p, H^p, C^p_\alpha, N^p, H^p_\alpha, C^p, C^p_\alpha, O^p, C^p, O^{p'}\}\end{aligned} \tag{1.30}$$

.

Figura 22 – Re-ordem **hc** de cadeia de três resíduos de aminoácidos, omitindo-se os grupos R.



Fonte – Lavor et al. (2019)

Nota – Os arcos em *preto* representam a sucessão entre vértices consecutivos na re-ordem **hc**. Cada arco está numerado com a sua posição na re-ordem. Por exemplo: o arco 8 (do C_α do aminoácido 1 ao H do grupo amino do aminoácido 2) ilustra a passagem do *oitavo* elemento da re-ordem ao nono.

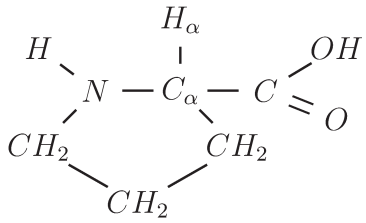
Os átomos $H^{1'}$ e $O^{p'}$ correspondem, respectivamente, ao segundo hidrogênio do grupo amino do primeiro resíduo de aminoácido e ao segundo oxigênio do grupo carboxila do último resíduo.

O outro conjunto de informações refere-se ao aminoácido prolina, que é um dos 20 aminoácidos comuns. Em estudo de 18.666 proteínas humanas, Morgan e Rubenstein (2013) mostraram que 99.8% destas proteínas contém prolina em sua composição. Então, é razoável pensar que esse aminoácido ocorre com elevada frequência em proteínas.

A prolina detém uma característica única: a sua cadeia lateral conecta-se ao esqueleto do aminoácido por ligação covalente em dois pontos diferentes. O seu grupo R é formado por três grupos CH_2 ligados em sequência. Porém, além da ligação com o C_α, o grupo R está ligado ciclicamente ao nitrogênio do grupo amino. A estrutura química da prolina pode ser observada na Figura 23.

Prolinas podem assumir dois tipos de configurações tridimensionais, uma delas chamada de *trans*, a outra chamada de *cis*. Como consequência, tal aminoácido tende a dobrar uma proteína nas posições em que ocorrem, induzindo o enrolamento da mesma (GIBSON; SCHERAGA, 1997; MORGAN; RUBENSTEIN, 2013).

Figura 23 – Estrutura química do aminoácido Prolina.



2 Resultados

O algoritmo de obtenção de uma ^3PDGDM re-ordem introduzido na seção 1.1 e sua versão modificada apresentada na subseção 1.2.1 foram implementados na linguagem de programação *C++*, e testados em instâncias do repositório de estruturas tridimensionais de proteínas *RCSB Protein Data Bank* (RCSB PDB) (RCSB. . . , 2020). Estas instâncias são as mesmas utilizadas em Mucherino, Lavor e Liberti (2012).

Os arquivos das instâncias foram baixados do servidor do RCSB PDB, por meio da biblioteca para análise de estruturas de proteínas *Protein Dynamics and Sequence Analysis* (ProDy), para linguagem de programação *Python* (PROTEIN. . . , 2019). Tais arquivos possuem simbologia e organização padronizados, que são detalhados em Fitzgerald et al. (2006), Protein. . . (2008), Introduction. . . (2020).

ProDy é utilizada ainda para processar os arquivos do PDB, a fim de montar arquivos menores que possuem apenas as informações necessárias para construção do grafo $G = (V, A)$ de um ^3PDGDM. Então, para cada distância conhecida de um ^3PDGDM (seção 1.2) entre átomos que ***não*** estão em ***cadeias laterais***, capturam-se: os rótulos inteiros e símbolos dos átomos, os rótulos inteiros e símbolos dos resíduos aos quais os átomos pertencem e os limites inferior e superior para a distância. Quando a distância em questão é exata, aqueles limites são iguais.

A Tabela 1 mostra o número de resíduos e o número de átomos (descontados os átomos das cadeias laterais) de cada uma das instâncias.

Tabela 1 – Número de Resíduos e Número de Átomos das Instâncias.

Instância	nResíduos	nÁtomos	Referências
1a23	189	1137	Schirra et al. (1998)
1aqr	40	240	Peterson, Narula e Armitage (1996)
1b4c	92	555	Drohat et al. (2008)
1bqx	77	464	Aono et al. (1998)
1brz	54	324	Caldwell et al. (1998)
1ccq	60	357	Dementieva, Bocharov e Arseniev (1999)
1d8v	263	1585	Wang et al. (1999)
1erp	38	224	Brown et al. (1993)
1k1v	41	247	Kusunoki et al. (2002)
1la3	201	1127	Ames, Hamasaki e Molchanova (2002)

Nota – A coluna *Instância* identifica cada uma das moléculas por seu código de identificação do PDB. A coluna *nResíduos* mostra o número de resíduos de aminoácidos da molécula. A coluna *nÁtomos* mostra o número de átomos da molécula ***sem*** os átomos das cadeias laterais.

A Tabela 2 mostra os tempos de leitura do grafo $G = (V, A)$ e de execução de cada etapa do algoritmo para obtenção de re-ordem da seção 1.1. Nota-se que todas as

células de tempo das instâncias *1a23*, *1d08* e *1la3* associadas à execução do algoritmo propriamente dito foram preenchidas com o símbolo -. Isto significa que o algoritmo não conseguiu terminar a etapa de construção de $G_3 = (V_3, A_3)$, após *duas horas* de execução. Consequentemente, nenhuma das etapas posteriores são realizadas.

Tabela 2 – Execução do Algoritmo Gerador de Re-ordem - Tempo.

Instância	G $\lfloor s \rfloor$	G_3 $\lfloor s \rfloor$	DFS $\lfloor s \rfloor$	Candid. $\lfloor s \rfloor$	Sobrejet. $\lfloor s \rfloor$	Total $\lfloor s \rfloor$
1a23	0	-	-	-	-	$\geqslant 7200$
1aqr	0	56	0	0	0	56
1b4c	0	1530	1	0	0	1531
1bqx	0	760	0	0	0	760
1brz	0	181	0	0	0	181
1ccq	0	266	0	0	0	266
1d8v	0	-	-	-	-	$\geqslant 7200$
1erp	0	40	0	0	0	40
1k1v	0	0	0	0	0	0
1la3	0	-	-	-	-	$\geqslant 7200$

Nota – A coluna *Instância* indica cada uma das moléculas por seu código de identificação do PDB. O símbolo $\lfloor s \rfloor$ indica o tempo em *segundos* cujo valor é o maior inteiro menor que o tempo gasto. A coluna G $\lfloor s \rfloor$ mostra o tempo de leitura de $G = (V, A)$; G_3 $\lfloor s \rfloor$, o tempo de construção de $G_3 = (V_3, A_3)$; *DFS* $\lfloor s \rfloor$, o tempo de aplicação da Busca em Profundidade; *Candid.* $\lfloor s \rfloor$, o tempo de construção das sequências candidatas a re-ordem; *Sobrejet.* $\lfloor s \rfloor$, o tempo de checagem da sobrejetividade de cada uma das candidatas; e em *Total* $\lfloor s \rfloor$, a soma de todos os tempos. O sinal $\geqslant n$ indica que a instância foi executada por cerca de n segundos e não foi resolvida.

A Tabela 3 é similar à Tabela 2, porém, é relativa à execução do algoritmo modificado da subseção 1.2.1. Observa-se a efetividade das alterações feitas no método: as instâncias *1a23*, *1d08* e *1la3*, que não haviam sido resolvidas mesmo com *duas horas* de execução, foram finalizadas em não mais que *onze segundos*. De fato, em cada um dos dez exemplos, a etapa de construção de $G_3 = (V_3, A_3)$ foi efetuada em menos de *um segundo*.

Isto se repete para a leitura de $G = (V, A)$ e para as etapas de construção de uma sequência candidata a re-ordem e checagem da sobrejetividade de tal sequência.

Portanto, o gasto substancial de tempo ocorre na etapa de aplicação da Busca em Profundidade. Ainda assim, para todas as instâncias, ***exceto*** as *quatro* maiores, a Busca em Profundidade consome menos de *um segundo*, e para as demais instâncias, este consumo não passa de onze segundos. Uma vez que a maior parte do custo da Busca em Profundida deriva da avaliação de todos os vizinhos de todos os vértices, a performance deste método sugere que cada vértice de $G_3 = (V_3, A_3)$ compartilha poucas arestas.

Em suma, os desempenhos de tempo das duas versões do algoritmo reforçam a seguinte tese: grafos $G = (V, A)$ de ${}^3PDGDM$ de proteínas têm matriz de adjacência esparsa e de pequena largura de banda, implicando que os grafos $G_3 = (V_3, A_3)$ associados também possuem matriz de adjacência esparsa e de pequena largura de banda (seção 1.2).

Estes resultados experimentais estimulam a busca de inequações para verificar se a largura de banda de G_3 estaria limitada superiormente por algum valor proporcional à largura de banda de G.

Tabela 3 – Execução do Algoritmo Gerador de Re-ordem Modificado - Tempo.

Instância	G $\lfloor s \rfloor$	G_3 $\lfloor s \rfloor$	DFS $\lfloor s \rfloor$	Candid. $\lfloor s \rfloor$	Sobrejet. $\lfloor s \rfloor$	Total $\lfloor s \rfloor$
1a23	0	0	5	0	0	5
1aqr	0	0	0	0	0	0
1b4c	0	0	1	0	0	1
1bqx	0	0	0	0	0	0
1brz	0	0	0	0	0	0
1ccq	0	0	0	0	0	0
1d8v	0	0	10	0	0	10
1erp	0	0	0	0	0	0
1k1v	0	0	0	0	0	0
1la3	0	0	5	0	0	5

Nota – A coluna *Instância* indica cada uma das moléculas por seu código de identificação do PDB. O símbolo $\lfloor s \rfloor$ indica o tempo em *segundos* cujo valor é o maior inteiro menor que o tempo gasto. A coluna G $\lfloor s \rfloor$ mostra o tempo de leitura de $G = (V, A)$; G_3 $\lfloor s \rfloor$, o tempo de construção de $G_3 = (V_3, A_3)$; *DFS* $\lfloor s \rfloor$, o tempo de aplicação da Busca em Profundidade; *Candid.* $\lfloor s \rfloor$, o tempo de construção das sequências candidatas a re-ordem; *Sobrejet.* $\lfloor s \rfloor$, o tempo de checagem da sobrejetividade de cada uma das candidatas; e em *Total* $\lfloor s \rfloor$, a soma de todos os tempos.

A Tabela 4 exibe o resultado derradeiro do método: em um dado ${}^3PDGDM$, há ou não re-ordens. Nas instâncias testadas, o algoritmo encontrou re-ordem em apenas ***duas*** delas, *1b4c* e *1k1v*. Consequentemente, conclui-se que não existem re-ordens nas demais.

Este resultado é inesperado, pois, para toda instância de ${}^3PDGDM$ de proteínas, deveria existir pelo menos uma re-ordem: a re-ordem ***hc*** (Equação 1.30).

Ao averiguar-se a distribuição dos tipos de aminoácidos presentes em cada uma das instâncias, percebe-se um fato curioso: nos casos em que o resultado do algoritmo é negativo para existência de re-ordens, há sempre a presença de *prolinas* na composição das moléculas; já nas outras duas instâncias, cujos resultados são positivos, ***não*** há *prolinas*.

Uma vez que o grupo R da prolina forma um ciclo com os átomos N e C_α do próprio aminoácido (Figura 23), a ligação peptídica entre a carboxila de um aminoácido qualquer e o grupo amino de uma prolina consome o único hidrogênio ligado ao nitrogênio da prolina. Consequentemente, o resíduo de prolina gerado não possui hidrogênio no grupo amino. A Figura 24 ilustra uma molécula de três resíduos cujo segundo é uma prolina.

A *ausência* de tal hidrogênio implica diretamente na *inexistência* da re-ordem **hc**, já que o átomo H^i não existe quando o *i-ésimo* resíduo de aminoácido é uma prolina.

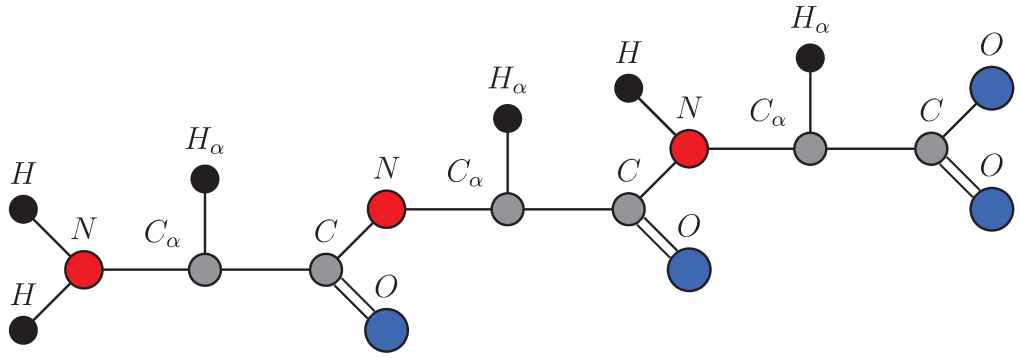
Definamos a sequência hc_{mod} como aquela equivalente à sequência **hc**, mas sem

Tabela 4 – Execução do Algoritmo Gerador de Re-ordem Modificado - Existência.

Instância	Existência	nComponentes
1a23	F	15
1aqr	F	4
1b4c	V	1
1bqx	F	9
1brz	F	2
1ccq	F	6
1d8v	F	16
1erp	F	9
1k1v	V	1
1la3	F	12

Nota – A coluna *Instância* indica cada uma das moléculas por seu código de identificação do PDB. A coluna *Existência* indica se o algoritmo gerador de re-ordens modificado obteve ou não uma re-ordem: *V* quando obteve; *F* quando não obteve. A coluna *nComponentes* indica quantas componentes conexas o grafo $G_3 = (V_3, A_3)$ possui.

Figura 24 – Cadeia de três resíduos de aminoácidos cujo o segundo é prolina, omitindo-se os grupos R.



Nota – Ao contrário dos aminoácidos 1 e 3, a prolina (aminoácido 2) não possui hidrogênios (H) ligados ao nitrogênio (N) do grupo amino.

os hidrogênios do grupo amino das prolinas. Nota-se que hc_{mod} também não é re-ordem. De fato: toma-se uma instância de ${}^3PDGDM$ de uma proteína cujo *i-ésimo* resíduo é de prolina. A primeira ocorrência de C_α^i em hc_{mod} não respeita a definição de re-ordem: os três átomos imediatamente anteriores a C_α^i são C_α^{i-1}, C^{i-1} e H_α^{i-1}; ainda que se conheçam as distâncias de C_α^i a C_α^{i-1} e C^{i-1}, provindas do plano peptídico, desconhece-se a distância de C_α^i a H_α^{i-1}. Portanto, hc_{mod} não é re-ordem.

Deve-se salientar que a inexistência não somente da re-ordem **hc**, mas de qualquer outra re-ordem, em todas as instâncias que possuem prolina, pode não ser uma

mera coincidência. As moléculas *1b4c* e *1k1v* possuem re-ordem (Tabela 4), têm 92 e 41 resíduos de aminoácidos, respectivamente (Tabela 1), e nenhum deles é prolina. Os grafos $G_3 = (V_3, A_3)$ de ambas as instâncias possuem somente uma componente conexa (Tabela 4). Por outro lado, a molécula *1brz* tem 54 resíduos (Tabela 1), *exatamente um* deles é prolina, e seu grafo G_3 possui duas componentes conexas, nenhuma delas carregando todos os átomos da molécula representados em seus vértices (Tabela 4). Seria interessante demonstrar se é verdadeira ou falsa a seguinte afirmação: inexistem re-ordens para qualquer ${}^3PDGDM$ de proteína que tem prolina, com distâncias conhecidas como aquelas apresentadas na seção 1.2.

Por fim, é preciso buscar, novamente, uma re-ordem que exista em qualquer proteína. Como observado na instância *1brz*, a ausência de um único átomo ligado ao esqueleto é suficiente para que não existam re-ordens em um ${}^3PDGDM$ de proteína. Utilizar os átomos da cadeia lateral das prolinas é uma medida que poderia garantir a existência de re-ordens. Fatos relevantes apoiam tal medida: a prolina é o único aminoácido comum cujo resíduo pode ter menos átomos do que o esperado; a cadeia lateral deste aminoácido é formada somente por carbonos e hidrogênios, tipos de átomos já presentes nos atuais ${}^3PDGDM$ de proteínas; o ciclo da cadeia lateral traria novas propriedades geométricas (por exemplo, conformações *trans* e *cis* do resíduo).

3 Conclusão

O presente trabalho reapresenta alguns teoremas de existência de ${}^3PDGDM$ re-ordens e propõe um algoritmo com custo de $O(|V|^4 \log_2^{|V|})$ para encontrar essas re-ordens. Tal custo se mostra menor que o $O(|V|^{2K})$ reportado na literatura. Além disto, é inovadora a maneira pela qual o método obtém uma re-ordem de um passeio γ em G_3 tal que $\bigcup_i \gamma_i = V$.

O algoritmo foi testado em instâncias de proteínas retiradas do PDB, com moléculas que chegam a ter mais de 1500 átomos. Foram analisados o tempo de execução e a obtenção ou não de re-ordem. Neste contexto, uma versão modificada do algoritmo é apresentada, visando diminuir o tempo de construção do grafo $G_3 = (V_3, A_3)$. A nova versão é efetiva, já que o método resolve o problema em não mais que *onze* segundos, mesmo nas maiores moléculas. Estes resultados viabilizam o uso de re-ordens como pré-requisito para aplicação do BP. Quanto à existência, encontraram-se re-ordens apenas naquelas instâncias que não têm o aminoácido prolina em sua composição. Já nos casos que possuem prolina, constatou-se a *inexistência* de re-ordens, inclusive da *re-ordem* ***hc***. Isto é uma evidência de que a ausência do átomo de hidrogênio de uma prolina pode estar causando a inexistência de re-ordens.

Finalmente, são elencadas algumas ideias para as próximas investigações:

1. Mostrar o quão esparsas são as matrizes de adjacência de grafos $G = (V, A)$ de instâncias de proteínas;
2. Buscar inequações que relacionem as larguras de banda das matrizes de adjacência de $G = (V, A)$ e $G_3 = (V_3, A_3)$, com o objetivo de mostrar se as matrizes de adjacência de grafos G_3 de ${}^3PDGDM$ de proteínas são sempre esparsas;
3. Aplicar o algoritmo em mais instâncias de proteínas que *não contenham* prolina;
4. Provar se de fato *inexistem* re-ordens em todo ${}^3PDGDM$ de proteínas que possuem prolina;
5. Encontrar uma nova re-ordem que exista para todo ${}^3PDGDM$ de proteína. Uma possível maneira para realizar esta tarefa consistiria em utilizar os átomos da *cadeia lateral* das prolinas.

Uma vez que o método apresentado utiliza a Busca em Profundidade para gerar não somente ${}^3PDGDM$ re-ordens, mas também ${}^3PDGDM$ *ordens*, será interessante induzir a escolha de vizinhos não visitados da Busca em Profundidade para que o passeio

γ encontrado tenha exatamente $|V| - 2$ elementos. Desta forma, o algoritmo gerador de re-ordens funcionaria como uma heurística para determinação de ${}^3PDGDM$ ordens.

Referências

AMES, J. B.; HAMASAKI, N.; MOLCHANOVA, T. Structure and Calcium-Binding Studies of a Recoverin Mutant (E85Q) in an Allosteric Intermediate State †. *Biochemistry*, v. 41, n. 18, p. 5776–5787, may 2002. ISSN 0006-2960. Disponível em: <https://pubs.acs.org/doi/10.1021/bi012153k>. Citado na página 65.

ANDERSON, B. D. O.; BELHUMEUR, P. N.; EREN, T.; GOLDENBERG, D. K.; MORSE, A. S.; WHITELEY, W.; YANG, Y. R. Graphical properties of easily localizable sensor networks. *Wireless Networks*, v. 15, n. 2, p. 177–191, feb 2009. ISSN 1022-0038. Disponível em: <http://link.springer.com/10.1007/s11276-007-0034-9>. Citado na página 14.

AONO, S.; BENTROP, D.; BERTINI, I.; COSENZA, G.; LUCHINAT, C. Solution structure of an artificial Fe8S8 ferredoxin : the D13C variant of Bacillus schlegelii Fe7S8 ferredoxin. *European Journal of Biochemistry*, v. 258, n. 2, p. 502–514, dec 1998. ISSN 0014-2956. Disponível em: <http://doi.wiley.com/10.1046/j.1432-1327.1998.2580502.x>. Citado na página 65.

BAZARAA, M. S.; JARVIS, J. J.; SHERALI, H. D. *Linear Programming and Network Flows*. 4. ed. New Jersey: John Wiley & Sons. Inc., 2010. 1–748 p. ISBN 978-0-470-46272-0. Citado na página 13.

BLUMENTHAL, L. M. *Theory and Applications of Distance Geometry*. 2. ed. New York, NY: Chelsea Publishing Company, 1970. 1–357 p. ISBN 0-8284-0242-6. Citado na página 13.

BONDY, J. A.; MURTY, U. S. R. *Graph Theory*. Springer, 2008. 1–654 p. (Graduate Texts in Mathematics). ISBN 978-1-84628-969-9. Disponível em: <http://link.springer.com/10.1007/978-1-84628-970-5>. Citado 2 vezes nas páginas 15 e 40.

BROWN, L.; MRONGA, S.; BRADSHAW, R.; ORTENZI, C.; LUPORINI, P.; WÜTHRICH, K. Nuclear Magnetic Resonance Solution Structure of the Pheromone Er-10 from the Ciliated Protozoan Euplotes raikovi. *Journal of Molecular Biology*, v. 231, n. 3, p. 800–816, jun 1993. ISSN 00222836. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0022283683713276>. Citado na página 65.

CALDWELL, J. E.; ABILDGAARD, F.; DŽAKULA, Ž.; MING, D.; HELLEKANT, G.; MARKLEY, J. L. Solution structure of the thermostable sweet-tasting protein brazzein. *Nature Structural Biology*, v. 5, n. 6, p. 427–431, jun 1998. ISSN 1072-8368. Disponível em: <http://www.nature.com/doifinder/10.1038/nsb0698-427>. Citado na página 65.

CASSIOLI, A.; GÜNLÜK, O.; LAVOR, C.; LIBERTI, L. Discretization vertex orders in distance geometry. *Discrete Applied Mathematics*, Elsevier B.V., v. 197, p. 27–41, dec 2015. Disponível em: <http://dx.doi.org/10.1016/j.dam.2014.08.035https://linkinghub.elsevier.com/retrieve/pii/S0166218X14003874>. Citado 2 vezes nas páginas 19 e 20.

CONTAINERS em C++. 2020. Estruturas de dados genéricas da linguagem C++. Disponível em: <http://www.cplusplus.com/reference/stl/>. Acesso em: 22 maio 2020. Citado na página 34.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms*. 3. ed. [S.l.]: Massachusetts Institute of Technology, 2009. 1–1313 p. ISBN 978-0-262-03384-8. Citado 3 vezes nas páginas 34, 40 e 44.

DEMENTIEVA, D. V.; BOCHAROV, E. V.; ARSENIEV, A. S. Two forms of cytotoxin II (cardiotoxin) from Naja naja oxiana in aqueous solution . Spatial structures with tightly bound water molecules. *European Journal of Biochemistry*, v. 263, n. 1, p. 152–162, jul 1999. ISSN 0014-2956. Disponível em: <http://doi.wiley.com/10.1046/j.1432-1327.1999.00478.x>. Citado na página 65.

DONALD, B. R. Algorithms in Structural Molecular Biology. In: . [S.l.: s.n.], 2011. Citado na página 14.

DROHAT, A. C.; TJANDRA, N.; BALDISSERI, D. M.; WEBER, D. J. The use of dipolar couplings for determining the solution structure of rat apo-S100B($\beta\beta$). *Protein Science*, v. 8, n. 4, p. 800–809, dec 2008. ISSN 09618368. Disponível em: <http://doi.wiley.com/10.1110/ps.8.4.800>. Citado na página 65.

FITZGERALD, P. M. D.; WESTBROOK, J. D.; BOURNE, P. E.; MCMAHON, B.; WATENPAUGH, K. D.; BERMAN, H. M. Macromolecular dictionary (mmCIF). In: HALL, S. R.; MCMAHON, B. (Ed.). *International Tables For Crystallography Volume G: Definition and Exchange of Crystallographic Data*. International Union Of Crystallography, 2006. cap. 4.5, p. 295–443. ISBN 978-1-4020-3138-0. Disponível em: <it.iucr.org/g>. Citado na página 65.

GIBSON, K. D.; SCHERAGA, H. A. Energy minimization of rigid-geometry polypeptides with exactly closed disulfide loops. *Journal of Computational Chemistry*, v. 18, n. 3, p. 403–415, feb 1997. ISSN 0192-8651. Disponível em: <https://doi.org/10.1002/(SICI)1096-987X(199702)18:3<403::AID-JCC10>3.0.CO;2-J>. Citado 2 vezes nas páginas 54 e 64.

HAVEL, T.; WÜTHRICH, K. A distance geometry program for determining the structures of small proteins and other macromolecules from nuclear magnetic resonance measurements of intramolecular1H-1H proximities in solution. *Bulletin of Mathematical Biology*, v. 46, n. 4, p. 673–698, 1984. ISSN 15229602. Citado na página 54.

INTRODUCTION to Protein Data Bank Format. 2020. Convenções dos símbolos de arquivos de formato do PDB. Disponível em: <https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html>. Acesso em: 22 maio 2020. Citado na página 65.

JACKSON, B.; JORDÁN, T. On the rigidity of molecular graphs. *Combinatorica*, v. 28, n. 6, p. 645–658, 2008. ISSN 02099683. Citado na página 54.

KOSTROWICKI, J.; PIELA, L. Diffusion equation method of global minimization: Performance for standard test functions. *Journal of Optimization Theory and Applications*, v. 69, n. 2, p. 269–284, may 1991. ISSN 0022-3239. Disponível em: <http://link.springer.com/10.1007/BF00940643>. Citado na página 16.

KUCHERENKO, S.; SYTSKO, Y. Application of Deterministic Low-Discrepancy Sequences in Global Optimization. *Computational Optimization and Applications*, v. 30, n. 3, p. 297–318, mar 2005. ISSN 0926-6003. Disponível em: <http://link.springer.com/10.1007/s10589-005-4615-1>. Citado na página 16.

KUSUNOKI, H.; MOTOHASHI, H.; KATSUOKA, F.; MOROHASHI, A.; YAMAMOTO, M.; TANAKA, T. Solution structure of the DNA-binding domain of MafG. *Nature Structural Biology*, v. 9, n. 4, p. 252–256, apr 2002. ISSN 10728368. Disponível em: <http://www.nature.com/doifinder/10.1038/nsb771>. Citado na página 65.

LAVOR, C. *Rigidez em Grafos de Proteínas*. Rio de Janeiro: 32o Colóquio Brasileiro de Matemática, IMPA, 2019. 1–66 p. ISBN 978-85-244-0428-3. Citado 3 vezes nas páginas 17, 53 e 54.

LAVOR, C.; LIBERTI, L. Um convite à geometria de distâncias. In: Rodrigo Rafaeli, F.; Avansini Botta Pirani, V.; Loureiro Madureira, A.; Luiz Cataldo Ferreira, E.; Manuel Vieira Capela, J.; Augusta Santos, S. (Ed.). *Notas em Matemática Aplicada*. São Carlos: Sociedade Brasileira de Matemática Aplicada e Computacional, 2014. v. 71, p. 1–57. ISBN 978-85-8215-050-4. Disponível em: <http://www.sbmac.org.br/p{_}notas.> Citado 5 vezes nas páginas 15, 16, 17, 18 e 54.

LAVOR, C.; LIBERTI, L.; DONALD, B.; WORLEY, B.; BARDIAUX, B.; MALLIAVIN, T. E.; NILGES, M. Minimal NMR distance information for rigidity of protein graphs. *Discrete Applied Mathematics*, Elsevier B.V., v. 256, p. 91–104, mar 2019. ISSN 0166218X. Disponível em: <https://doi.org/10.1016/j.dam.2018.03.071https://linkinghub.elsevier.com/retrieve/pii/S0166218X18301793>. Citado 2 vezes nas páginas 54 e 63.

LAVOR, C.; LIBERTI, L.; MACULAN, N. Computational Experience with the Molecular Distance Geometry Problem. In: PINTÉR, J. D. (Ed.). *Global Optimization. Nonconvex Optimization and Its Applications*. Boston, MA: Springer US, 2006. cap. 9, p. 213–225. ISBN 978-0-387-30927-9. Citado na página 16.

LAVOR, C.; LIBERTI, L.; MACULAN, N.; MUCHERINO, A. The discretizable molecular distance geometry problem. *Computational Optimization and Applications*, v. 52, n. 1, p. 115–146, may 2012. ISSN 0926-6003. Disponível em: <http://link.springer.com/10.1007/s10589-011-9402-6>. Citado 2 vezes nas páginas 19 e 20.

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*, v. 56, n. 3, p. 855–871, 2013. ISSN 09255001. Citado na página 20.

LAVOR, C.; MACULAN, N.; SOUZA, M.; ALVES, R. *Álgebra e Geometria no Cálculo de Estrutura Molecular*. Rio de Janeiro: 31o Colóquio Brasileiro de Matemática, IMPA, 2017. 1–117 p. ISBN 978-85-244-0432-0. Citado na página 54.

LAVOR, C.; SOUZA, M.; CARVALHO, L. M.; LIBERTI, L. On the polynomiality of finding KDMDGP re-orders. *Discrete Applied Mathematics*, Elsevier B.V., v. 267, p. 190–194, aug 2019. ISSN 0166218X. Disponível em: <https://doi.org/10.1016/j.dam.2019.07.021https://linkinghub.elsevier.com/retrieve/pii/S0166218X19303373>. Citado 8 vezes nas páginas 20, 21, 22, 23, 25, 31, 32 e 36.

LIBERTI, L.; DRAŽIC, M. Variable Neighbourhood Search for the Global Optimization of Constrained NLPs. *Proceedings of GO Workshop*, p. 1–5, 2005. Citado na página 16.

LIBERTI, L.; LAVOR, C. Six mathematical gems from the history of distance geometry. *International Transactions in Operational Research*, v. 23, n. 5, p. 897–920, sep 2016. ISSN 09696016. Disponível em: <http://doi.wiley.com/10.1111/itor.12170>. Citado na página 13.

LIBERTI, L.; LAVOR, C.; MACULAN, N. A Branch-and-Prune algorithm for the Molecular Distance Geometry Problem. *International Transactions in Operational Research*, v. 15, p. 1–17, jan 2008. Disponível em: <http://doi.wiley.com/10.1111/j.1475-3995.2011.00807.xhttp://doi.wiley.com/10.1111/j.1475-3995.2007.00622.x>. Citado 3 vezes nas páginas 16, 18 e 20.

LIBERTI, L.; LAVOR, C.; MACULAN, N.; MUCHERINO, A. Euclidean distance geometry and applications. *SIAM Review*, v. 56, n. 1, p. 3–69, 2014. ISSN 00361445. Citado 2 vezes nas páginas 14 e 16.

LIBERTI, L.; MASSON, B.; LEE, J.; LAVOR, C.; MUCHERINO, A. On the number of realizations of certain Henneberg graphs arising in protein conformation. *Discrete Applied Mathematics*, Elsevier B.V., v. 165, p. 213–232, mar 2014. ISSN 0166218X. Disponível em: <http://dx.doi.org/10.1016/j.dam.2013.01.020https://linkinghub.elsevier.com/retrieve/pii/S0166218X13000449>. Citado 2 vezes nas páginas 19 e 52.

LINDEGREN, L.; LAMMERS, U.; HOBBS, D.; O'MULLANE, W.; BASTIAN, U.; HERNÁNDEZ, J. The astrometric core solution for the Gaia mission. *Astronomy & Astrophysics*, v. 538, p. A78, feb 2012. ISSN 0004-6361. Disponível em: <http://www.aanda.org/10.1051/0004-6361/201117905>. Citado na página 14.

MORÉ, J. J.; WU, Z. Global Continuation for Distance Geometry Problems. *SIAM Journal on Optimization*, v. 7, n. 3, p. 814–836, aug 1997. ISSN 1052-6234. Disponível em: <http://epubs.siam.org/doi/10.1137/S1052623495283024>. Citado na página 16.

MORGAN, A. A.; RUBENSTEIN, E. Proline: The Distribution, Frequency, Positioning, and Common Functional Roles of Proline and Polyproline Sequences in the Human Proteome. *PLoS ONE*, v. 8, n. 1, p. e53785, jan 2013. ISSN 1932-6203. Disponível em: <https://dx.plos.org/10.1371/journal.pone.0053785>. Citado 2 vezes nas páginas 63 e 64.

MUCHERINO, A.; LAVOR, C.; LIBERTI, L. The discretizable distance geometry problem. *Optimization Letters*, v. 6, n. 8, p. 1671–1686, dec 2012. Disponível em: <http://link.springer.com/10.1007/s11590-011-0358-3>. Citado na página 65.

NELSON, D. L.; COX, M. M. *Princípios de Bioquímica de Lehninger*. 6. ed. [S.l.]: ARTMED EDITORA LTDA, 2014. 1–1250 p. ISBN 978-85-8271-073-9. Citado 3 vezes nas páginas 52, 53 e 54.

NIELSEN, J.; ROTH, B. On the Kinematic Analysis of Robotic Mechanisms. *The International Journal of Robotics Research*, v. 18, n. 12, p. 1147–1160, dec 1999. ISSN 0278-3649. Disponível em: <http://journals.sagepub.com/doi/10.1177/02783649922067771>. Citado na página 14.

PETERSON, C. W.; NARULA, S. S.; ARMITAGE, I. M. 3D solution structure of copper and silver-substituted yeast metallothioneins. *FEBS Letters*, v. 379, n. 1, p. 85–93, jan 1996. ISSN 00145793. Disponível em: <https://www.sciencedirect.com/science/article/pii/0014579395014926>. Citado na página 65.

PROTEIN Data Bank Contents Guide: Atomic Coordinate Entry Format Description. wwPDB, 2008. 1–194 p. Disponível em: <http://mmcif.pdb.org/dictionaries/mmcif{_}pdbx.dic/Index/index.h>. Citado na página 65.

PROTEIN Dynamics and Sequence Analysis (ProDy). 2019. Biblioteca para Python de manipulação de arquivos do PDB. Disponível em: <http://prody.csb.pitt.edu/>. Acesso em: 07 julho 2020. Citado na página 65.

RCSB Protein Data Bank. 2020. Página inicial do PDB. Disponível em: <https://www.rcsb.org>. Acesso em: 06 julho 2020. Citado na página 65.

SANTIAGO, C. P.; LAVOR, C.; MONTEIRO, S. A.; KRONER-MARTINS, A. A new algorithm for the small-field astrometric point-pattern matching problem. *Journal of Global Optimization*, Springer US, v. 72, n. 1, p. 55–70, sep 2018. ISSN 0925-5001. Disponível em: <https://doi.org/10.1007/s10898-018-0653-yhttp://link.springer.com/10.1007/s10898-018-0653-y>. Citado na página 14.

SAXE, J. B. Saxe-Embeddability.Pdf. In: *17th Annual Allerton Conference on Communication, Control and Computing*. Monticello, Illinois: [s.n.], 1979. p. 480–489. Citado na página 15.

SCHIRRA, H. J.; RENNER, C.; CZISCH, M.; HUBER-WUNDERLICH, M.; HOLAK, T. A.; GLOCKSHUBER, R. Structure of Reduced DsbA from Escherichia coli in Solution. *Biochemistry*, v. 37, n. 18, p. 6263–6276, may 1998. ISSN 0006-2960. Disponível em: <https://pubs.acs.org/doi/10.1021/bi980136y>. Citado na página 65.

WANG, Y.-X.; NEAMATI, N.; JACOB, J.; PALMER, I.; STAHL, S. J.; KAUFMAN, J. D.; HUANG, P. L.; HUANG, P. L.; WINSLOW, H. E.; POMMIER, Y.; WINGFIELD, P. T.; LEE-HUANG, S.; BAX, A.; TORCHIA, D. A. Solution Structure of Anti-HIV-1 and Anti-Tumor Protein MAP30. *Cell*, v. 99, n. 4, p. 433–442, nov 1999. ISSN 00928674. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0092867400815299>. Citado na página 65.