Universidade Federal Fluminense

SILAS SALLAUME

Cálculo de estruturas protéicas através da resolução do Problema Discreto de Geometria das Distâncias em Moléculas

NITERÓI

2009

Universidade Federal Fluminense

SILAS SALLAUME

Cálculo de estruturas protéicas através da resolução do Problema Discreto de Geometria das Distâncias em Moléculas

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Orientadores:
Prof. Simone de Lima Martins, D.Sc.

Prof. Carlile Campos Lavor, D.Sc.

NITERÓI

2009

Calculo de estruturas protéicas através da resolução do Problema Discreto de Geometria das Distâncias em Moléculas

Silas Sallaume

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Aprovada por:

Profa. Simone de Lima Martins, D.Sc. / IC-UFF
(Presidente) (Orientador)

Prof. Carlile Campos Lavor, D.Sc. / IMECC-UNICAMP
(Orientador)

Prof. Fábio Protti, D.Sc. / IC-UFF

Prof. Luiz Satoru Ochi, D.Sc. / IC-UFF

Prof. Víctor Manuel Parada Daza, D.Sc. / DII-USACH

Niterói, 13 de Março de 2009.

"Deixem a trilha conhecida de vez em quando e entrem pela floresta, certamente encontrarão alguma coisa que nunca viram."

Alexandre G. Bell

À minha família pelo apoio incondicional.

Agradecimentos

Primeiramente, à Deus, por ter me iluminado em mais uma jornada.

Aos meus pais Jamil e Marilize, e ao meu irmão Thales, bem como, à toda a minha família, por todo amor e carinho, não só durante a realização deste trabalho, como na vida inteira.

Agradeço à Erica o amor dedicado a mim durante o tempo em que estive realizando este trabalho. Obrigado pelos bons momentos e pela paciência nos momentos mais difíceis.

Aos professores e amigos Simone, Carlile e Satoru, pela valiosa orientação, apoio e incentivo, os quais foram indispensáveis para a realização deste trabalho.

Ao Warley, pelas sugestões oferecidas e por disponibilizar todas as informações sobre o seu trabalho, o qual foi a base desta dissertação.

A professora Helena que me convidou para participar do projeto da FAPERJ do qual me tornei bolsista.

Ao professor Marcone da UFOP, que me incentivou e apoiou o meu ingresso no programa de mestrado da IC.

Ao IC da UFF e à seus professores e funcionários que não mediram esforços para me ajudar durante todo o meu mestrado.

Aos companheiros de "república", Marcio e Matheus pela amizade.

Aos amigos André, Rodrigo, Thiago e Xico pelo companheirismo e pela hospitalidade em minha "primeira semana" em Niterói.

Prefiro não citar os nomes, para não me esquecer de ninguém, mas agradeço à todos meus amigos que me encorajaram nos momentos mais difíceis desta trajetória, que Deus possa retribuir-lhes em dobro.

Agradeço também à FAPERJ por financiar uma grande parte dos meus estudos.

E finalmente, à todos que de alguma forma participaram e participam da minha vida.

Muito obrigado.

Resumo

Um problema muito importante em biologia computacional é a determinação da estrutura tridimensional de proteínas. Algumas informações a respeito da proteína, podem ser obtidas através de Ressonância Magnética Nuclear (RMN), porém, esta técnica, provê apenas um conjunto esparso de distâncias entre os átomos de uma molécula. Neste caso, o problema é determinar a estrutura tridimensional de uma molécula, utilizando um conjunto de distâncias entre alguns átomos. Na literatura, este problema é conhecido como Problema de Geometria das Distâncias em Moléculas e é geralmente formulado como um problema de otimização contínua. Entretanto, recentemente, foram apresentadas condições, sob as quais ele pode ser tratado como um problema de otimização combinatória, através de uma formulação discreta. Nesta dissertação, é apresentada uma abordagem que determina as estruturas por completo, incluindo as ramificações da cadeia principal. São propostos três algoritmos, um somente para cadeia principal e outros dois que tratam a proteína inteira. Os algoritmos foram testados com instâncias artificiais e reais e obtiveram bons resultados em relação aos métodos existentes na literatura.

Palavras-chave: Problema Discreto de Geometria das Distâncias em Moléculas, estrutura tridimensional de proteína, biologia computacional.

Abstract

One important problem in computational biology is the determination of the three-dimensional structure of proteins. Some information about the protein can be caught by the Nuclear magnetic resonance (NMR), but this technique provides only a sparse set of distances between atoms in a molecule. In this case, the problem is to determine the three-dimensional structure of a molecule using a set of distances between some atoms. In the literature, this problem is known as the Molecular Distance Geometry Problem which is generally expressed as a problem of continuous optimization. However, conditions for dealing with it as a combinatorial optimization problem using a discrete formulation were presented recently. In this dissertation, an approach for determine the whole protein structure, including the ramifications of the backbone. Three algorithms are proposed, one that deals only with the backbone and other two that consider the whole protein. In order to test the algorithms were used real and artificial instances and were obtained good results in relation to methods presented in the literature.

Keywords: Discretizable Molecular Distance Geometry Problem, three-dimensional protein structure, computational biology.

Siglas e Abreviações

ACPN	:	Algoritmo para Cadeia Principal por Níveis
APC	:	Algoritmo para Proteína Completa
APCS	:	Algoritmo para Proteína Completa com Simetria
BP	:	*Branch and Prune*
LDE	:	*Largest Distance Error*
PDB	:	*Protein Data Bank*
PDGDM	:	Problema Discreto de Geometria das Distâncias em Moléculas
PGDM	:	Problema de Geometria das Distâncias em Moléculas
POAPDGDM	:	Problema de Ordenação de Átomo para o PDGDM
RMN	:	Ressonância Magnética Nuclear
RMSD	:	*Root-Mean-Square Deviation*

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

As proteínas são fundamentais nos sistemas biológicos, onde desempenham diversas funções como: catálise de reações, transporte, suporte e movimento, resposta imunitária, entre outras [18]. Cada proteína é constituída por uma cadeia linear de aminoácidos [6] e sua função é determinada pela sua estrutura tridimensional. Técnicas de Ressonânica Magnética Nuclear (RMN) fornecem distâncias entre os átomos de uma proteína que estejam separados por menos de 6Å. Usualmente, para calcular as estruturas, estas distâncias são combinadas com outras informações tais como sequência primária, geometrias de referêcia para ângulos entre átomos, entre outras. Formula-se uma função de energia e tenta-se minimizá-la de modo a se encontrar coordenadas para o atomos mais compatíveis com os dados experimentais.

Como os experimentos de RMN fornecem apenas um conjunto esparso de distâncias entre átomos da molécula, neste trabalho, é considerado o problema de se determinar a estrutura de uma proteína, utilizando esse conjunto esparso de distâncias. Este problema é chamado de Problema de Geometria das Distâncias em Moléculas (PGDM). Com as devidas adaptações, também existem aplicações em outras áreas, como localização em redes de sensores [3], reconhecimento de imagens [17], entre outros, que podem ser modelados com um PGDM.

No PGDM, caso sejam conhecidas as distâncias entre todos os pares de átomos, o problema pode ser resolvido em tempo polinomial e uma única estrutura tridimensional pode ser determinada [8]. Caso contrário, o problema passa a ser NP-completo [32].

Existem, na literatura, várias abordagens para o PGDM, por exemplo, o algoritmo *EMDED* de Crippen e Havel [7, 14], a estratégia de redução de grafo de Hendrickson [15, 16], o algoritmo DGSOL de Moré e Wu [28, 29, 30, 27], o método de perturbação

estocástica de Zou, Bird e Schnabel [37], o método de escala multidimensional de Trosset [34], o algoritmo VNS de Lavor, Liberti e Maculan [23, 21], o algoritmo *Geometric Build-Up* de Dong, Wu e Wu [35], a extensão do algoritmo Geometric Build-Up feita por Carvalho, Lavor e Protti [5], entre outros. Existem também, alguns levantamentos sobre métodos para a resolução deste problema, que podem ser encontrados em [4, 7, 13, 22, 25].

Em 2006, Lavor, Liberti e Maculan [20] propuseram uma formulação discreta para o PGDM, observando que é possível formular o PGDM, aplicado à cadeia principal de uma proteína, como um problema de busca em um espaço discreto. Essa nova formulação recebeu o nome de Problema Discreto de Geometria das Distâncias em Molécula (PDGDM). Foi apresentado, também, o algoritmo *Branch-and-Prune* [20, 24] que utiliza essa formulação para resolver o problema.

Em [11], foram propostos dois algoritmos para o PDGDM, que apesar de serem bem simples, gerenciam de forma eficiente o uso de memória e o tempo computacional, quando comparados com outras versões existentes na literatura. Contudo, os algoritmos determinam somente a estrutura da seqüência principal de átomos da proteína. No presente trabalho, é apresentada uma abordagem que leva em consideração todos os átomos pertencentes a proteína, com o objetivo de determinar a estrutura global da mesma, incluindo as ramificações da seqüência principal (cadeias laterais).

Esta dissertação está organizada em cinco capítulos. O conteúdo de cada um deles é apresentado a seguir.

- Capítulo 2: introduz o PGDM e a sua formulação discreta, o PDGDM. Além disso, é demonstrado como a formulação discreta pode ser utilizada para determinar somente a cadeia principal, bem como a estrutura de toda a proteína.

- Capítulo 3: apresenta as principais contribuições deste trabalho: três novos algoritmos para o PDGDM, um que resolve o problema somente para cadeia principal e outros dois que consideram todos os átomos da proteína.

- Capítulo 4: faz uma análise experimental, comparando os resultados dos algoritmos propostos com algoritmos da literatura.

- Capítulo 5: apresenta as conclusões e os trabalhos futuros desta dissertação.

Capítulo 2

O Problema de Geometria das Distâncias em Moléculas - PGDM

2.1 Descrição do PGDM

O Problema de Geometria das Distâncias em Moléculas (PGDM) está associado à determinação da estrutura tridimensional da cadeia principal dos átomos de uma proteína. Este problema pode ser formulado da seguinte forma: encontre as posições $x_1, \ldots, x_n \in \mathbb{R}^3$ dos átomos da cadeia, tais que:

$$\| x_i - x_j \| = d_{i,j}, \qquad (i,j) \in S, \tag{2.1}$$

onde S é um subconjunto dos pares de átomos cujas distâncias entre os átomos i e j podem ser obtidas através de experimentos RMN e $||.||$ é a norma Euclidiana.

A formulação 2.1 corresponde ao PGDM exato, mas devido aos erros experimentais na análise de RMN, somente alguns limites inferiores e superiores das distâncias podem ser obtidos. Deste modo, o PGDM pode ser definido, de um modo mais geral, como: encontre as posições $x_1, \ldots, x_n \in \mathbb{R}^3$ tais que:

$$l_{i,j} \leq \| x_i - x_j \| \leq u_{i,j}, \qquad (i,j) \in S, \tag{2.2}$$

onde $l_{i,j}$ e $u_{i,j}$ são os limites inferiores e superiores nas restrições de distância, respectivamente.

Assim, o PGDM pode ser formulado como um problema de otimização contínua, onde

o objetivo é minimizar a função dada por:

$$f(x_1, \ldots, x_n) = \sum_{(i,j)\in S} (||x_i - x_j||^2 - d_{i,j}^2)^2. \tag{2.3}$$

Como mostrado em [16], a grande dificuldade nessa formulação é que a quantidade de mínimos locais cresce exponencialmente com o tamanho da molécula e o que se deseja é encontrar o mínimo global.

2.2 Problema Discreto de Geometria das Distâncias em Moléculas - PDGDM

Como descrito anteriormente, o PGDM pode ser visto como um problema de otimização contínua. Entretanto, usando duas hipóteses adicionais, uma fomulação discreta foi proposta em [20], introduzindo assim uma subclasse de problemas do PGDM, chamada de Problema Discreto de Geometria das Distâncias em Moléculas (PDGDM). Também em [20], foi demonstrado que o PDGDM é NP-completo.

2.2.1 Formulação discreta

Baseado em [20], considere a cadeia principal de uma proteína como sendo uma seqüência de n átomos, onde a distância média entre os núcleos de dois átomos ligados na posição de maior estabilidade (menor energia), são denotados por $d_{i-1,i}$, para $i = 2, \ldots, n$, os ângulos de ligação são denotados por $\theta_{i-2,i}$, para $i = 3, \ldots, n$, e os ângulos de torção denotados por $\omega_{i-3,i}$, para $i = 4, \ldots, n$. Os ângulos de torção são definidos pelos vetores normais dos planos definidos pelos átomos $i-3, i-2, i-1$ e $i-2, i-1, i$, respectivamente (Figura 2.1).

Para a formulação discreta do PGDM, são consideradas as seguintes hipóteses:

- Hipótese 1: Todas as distâncias e ângulos de ligação entre quaisquer quatro átomos consecutivos são conhecidos.
- Hipótese 2: Para quaisquer três átomos consecutivos, denominados A, B e C, o ângulo formado pelos segmentos de reta AB e BC não é múltiplo de π.

De acordo com [20], a Hipótese 1 é aplicável à maioria das proteínas, pois as distâncias entre átomos consecutivos e os ângulos de ligações são conhecidos a priori e com isso as outras distâncias e ângulos necessários podem ser calculados. Além disso, a RMN é

capaz de obter distâncias entre átomos que estão próximos entre si, e grupos de quatro átomos consecutivos da cadeia principal de uma proteína são freqüentemente próximos. A Hipótese 2 é igualmente aplicável às proteínas, dado que não se conhece nenhuma proteína com ângulos de ligação com valor exato de π.

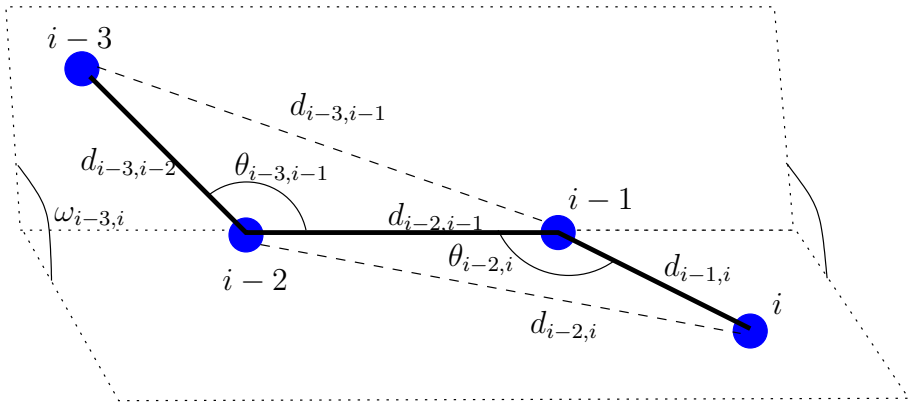


Figura 2.1: Definições de distâncias entre átomos, ângulos de ligação e ângulos de torção.

A intuição da formulação discreta é que o i-ésimo átomo reside na intersecção de três esferas centradas nos átomos $i-3, i-2, i-1$, de raios $d_{i-3,i}, d_{i-2,i}, d_{i-1,i}$, respectivamente. Pela Hipótese 2 e pelo fato de dois átomos não poderem nunca assumir a mesma posição no espaço, a intersecção das três esferas define, no máximo, dois pontos (indexados por i e i' na Figura 2.2). Isto permite expressar a posição do i-ésimo átomo em termos dos últimos três, resultando em 2^{n-3} possíveis estruturas para as seqüências de átomos.

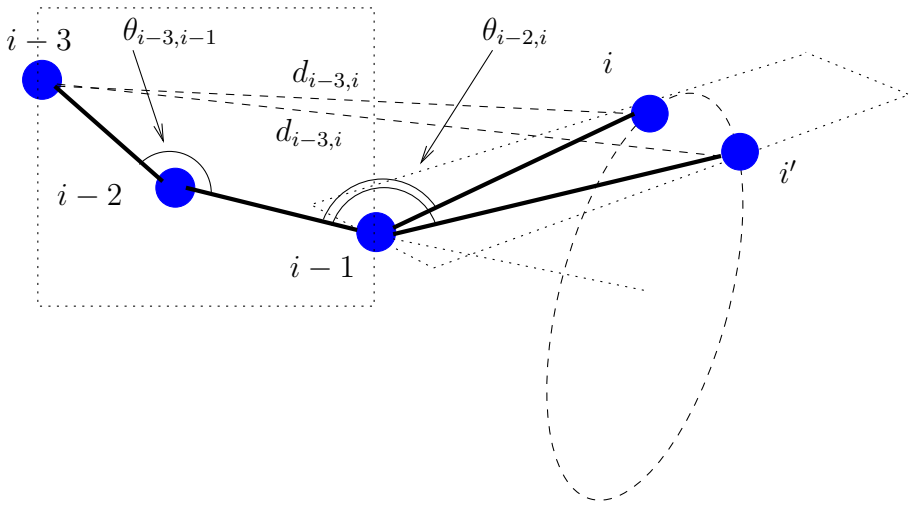


Figura 2.2: No PDGDM, o átomo i pode estar somente em duas posições (i e i') de modo a respeitar as distâncias e os ângulos conhecidos.

Dadas todas as distâncias $d_{1,2}, \ldots, d_{n-1,n}$, os ângulos de ligação $\theta_{1,3}, \ldots, \theta_{n-2,n}$, e ân-

gulos de torção $\omega_{1,4}, \ldots, \omega_{n-3,n}$ de uma cadeia com n átomos, as coordenadas cartesianas $x_i = (x_{i_1}, x_{i_2}, x_{i_3})$, para cada átomo i da seqüência, podem ser obtidas utilizando a seguinte fórmula [31]:

$$\begin{bmatrix} x_{i_1} \\ x_{i_2} \\ x_{i_3} \\ 1 \end{bmatrix} = B_1 B_2 \cdots B_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \ \forall i = 1, \ldots, n,$$

onde

$$B_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -1 & 0 & 0 & -d_{1,2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

$$B_3 = \begin{bmatrix} -\cos\theta_{1,3} & -\sin\theta_{1,3} & 0 & -d_{2,3}\cos\theta_{1,3} \\ \sin\theta_{1,3} & -\cos\theta_{1,3} & 0 & d_{2,3}\sin\theta_{1,3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

e

$$B_i = \begin{bmatrix} -\cos\theta_{i-2,i} & -\sin\theta_{i-2,i} & 0 & -d_{i-1,i}\cos\theta_{i-2,i} \\ \sin\theta_{i-2,i}\cos\omega_{i-3,i} & -\cos\theta_{i-2,i}\cos\omega_{i-3,i} & -\sin\omega_{i-3,i} & d_{i-1,i}\sin\theta_{i-2,i}\cos\omega_{i-3,i} \\ \sin\theta_{i-2,i}\sin\omega_{i-3,i} & -\cos\theta_{i-2,i}\sin\omega_{i-3,i} & \cos\omega_{i-3,i} & d_{i-1,i}\sin\theta_{i-2,i}\sin\omega_{i-3,i} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.5}$$

para $i = 4, ..., n$.

Para cada quatro átomos consecutivos x_{i-3}, x_{i-2}, x_{i-1} e x_i, o cosseno do ângulo de torção $\omega_{i-3,i}$ para $i = 4, ..., n$, pode ser determinado por:

$$\cos\omega_{i-3,i} = \frac{d^2_{i-3,i-2} + d^2_{i-2,i} - 2d_{i-3,i-2}d_{i-2,i}\cos\theta_{i-3,i-1}\cos\theta_{i-2,i} - d^2_{i-3,i}}{2d_{i-3,i-2}d_{i-2,i}\sin\theta_{i-3,i-1}\sin\theta_{i-2,i}}, \tag{2.6}$$

que é apenas um rearranjo da lei dos cossenos para os ângulos de torção [20].

Usando as distâncias $d_{1,2}$ e $d_{2,3}$ e o ângulo de ligação $\theta_{1,3}$, pode-se calcular as matrizes

B_2 e B_3, definidas em (2.4), e obter:

$$x_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$x_2 = \begin{pmatrix} -d_{1,2} \\ 0 \\ 0 \end{pmatrix},$$

$$x_3 = \begin{pmatrix} -d_{1,2} + d_{2,3}\cos\theta_{1,3} \\ d_{2,3}\sin\theta_{1,3} \\ 0 \end{pmatrix},$$

fazendo com que os três primeiros átomos da molécula sejam fixados.

Uma vez que a distância $d_{1,4}$ é conhecida, pela Hipótese 1, o valor de $\cos\omega_{1,4}$ pode ser obtido. Assim, o seno do ângulo de torção $\omega_{1,4}$ pode ter apenas dois valores possíveis: $\sin\omega_{1,4} = \pm\sqrt{1-\cos^2\omega_{1,4}}$. Deste modo, por (2.5), obtém-se apenas duas posições possíveis (x_4 e x'_4) para o quarto átomo da molécula:

$$x_4 = \begin{bmatrix} -d_{1,2} + d_{2,3}\cos\theta_{1,3} - d_{3,4}\cos\theta_{1,3}\cos\theta_{2,4} + d_{3,4}\sin\theta_{1,3}\sin\theta_{2,4}\cos\omega_{1,4} \\ d_{2,3}\sin\theta_{1,3} - d_{3,4}\sin\theta_{1,3}\cos\theta_{2,4} - d_{3,4}\cos\theta_{1,3}\sin\theta_{2,4}\cos\omega_{1,4} \\ d_{3,4}\sin\theta_{2,4}\left(\sqrt{1-\cos^2\omega_{1,4}}\right) \end{bmatrix},$$

$$x'_4 = \begin{bmatrix} -d_{1,2} + d_{2,3}\cos\theta_{1,3} - d_{3,4}\cos\theta_{1,3}\cos\theta_{2,4} + d_{3,4}\sin\theta_{1,3}\sin\theta_{2,4}\cos\omega_{1,4} \\ d_{2,3}\sin\theta_{1,3} - d_{3,4}\sin\theta_{1,3}\cos\theta_{2,4} - d_{3,4}\cos\theta_{1,3}\sin\theta_{2,4}\cos\omega_{1,4} \\ d_{3,4}\sin\theta_{2,4}\left(-\sqrt{1-\cos^2\omega_{1,4}}\right) \end{bmatrix}.$$

Para o quinto átomo, existem quatro possíveis posições, uma para cada combinação de $\pm\sqrt{1-\cos^2\omega_{1,4}}$ e $\pm\sqrt{1-\cos^2\omega_{2,5}}$. Por indução, pode-se observar que para o i-ésimo átomo, existem 2^{n-3} posições possíveis. Desta forma, para se representar uma molécula como uma seqüência linear de n átomos, existem 2^{n-3} possíveis seqüências de ângulos de torção $\omega_{1,4},\ldots,\omega_{n-3,n}$, cada uma definindo uma diferente estrutura tridimensional. Utilizando as matrizes B_i (2.5), essa seqüência de ângulos de torção pode ser convertida em uma outra seqüência de coordenadas cartesianas $x = (x_1,\ldots,x_n) \in \mathbb{R}^3$.

2.2.1.1 Propriedade do posicionamento único

Como ilustrado na Figura 2.2, uma vez que os átomos $i-3$, $i-2$ e $i-1$ estão fixos, existem sempre duas possíveis posições para o átomo i. Contudo, foi observado que existem alguns

casos particulares, onde há somente uma posição possível para se colocar este átomo.

Utilizando como exemplo as duas posições possíveis (x_4 e x'_4) para o quarto átomo, apresentadas anteriormente, pode-se verificar que a única diferença entre elas está na coordenada z. Entretanto, se a igualdade $\sqrt{1-\cos^2\omega_{1,4}} = -\sqrt{1-\cos^2\omega_{1,4}}$ for satisfeita, as posições (x_4 e x'_4) são idênticas, ou seja existe somente uma posição para o quarto átomo. Esta igualdade é verdadeira quando $\cos^2\omega_{1,4} = 1$, e isso ocorre quando $\omega_{1,4}$ é múltiplo de π.

De forma genérica, pode-se definir essa propriedade como: para todo átomo i, se a igualdade $\sqrt{1-\cos^2\omega_{i-3,i}} = -\sqrt{1-\cos^2\omega_{i-3,i}}$ for verdadeira, existe somente uma posição viável para i, uma vez que $x_i = x'_i$.

2.2.1.2 Soluções simétricas

Em [20], onde o PDGDM é definido, foi provado que para qualquer solução S do problema existe uma solução S' simétrica a S. Esta simetria ocorre em relação ao plano definido pelos três primeiros átomos que são fixados, sendo que qualquer solução de um lado deste plano dá origem a uma solução simétrica do outro lado. Em [20], a prova matemática deste teorema é apresentada, porém nesta dissertação optou-se por mostrar somente um exemplo visual (Figura 2.3).

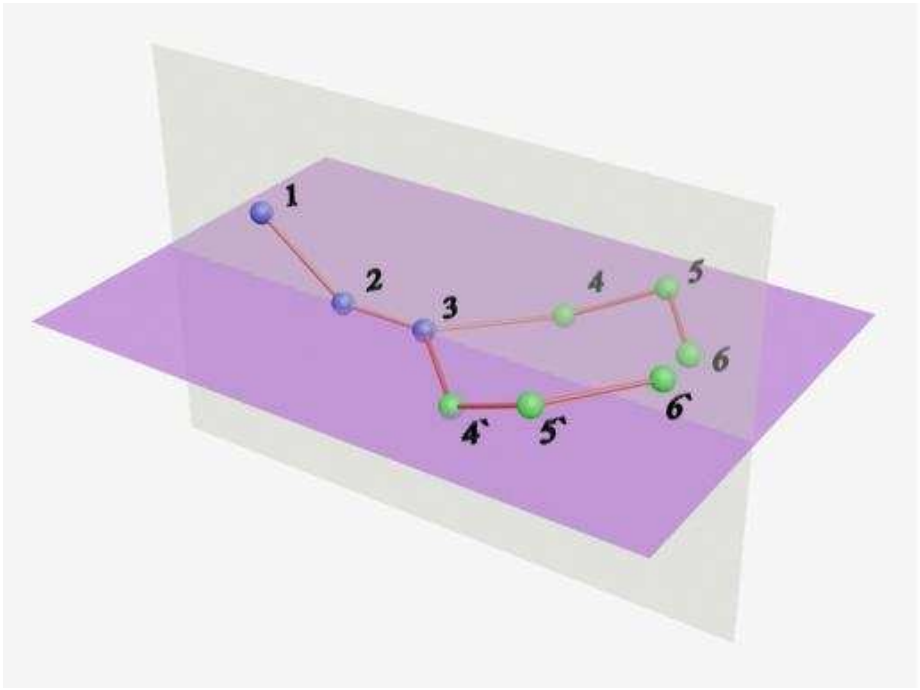


Figura 2.3: Simetria das soluções do PDGDM.

Na Figura 2.3, são mostradas duas soluções simétricas para PDGDM. Nas duas

soluções, os três primeiros átomos estão nas posições 1, 2 e 3, respectivamente, já o quarto, quinto e sexto átomos se encontram em posições distintas em cada uma das soluções. Para uma delas, estes átomos estão nas posições 4, 5 e 6, para a outra, eles estão em $4'$, $5'$ e $6'$ respectivamente. Não é difícil perceber que, em ambas as soluções, a distância entre quaisquer par de átomos é a mesma. Desta forma, se uma delas é válida, a outra também é. Com isso, ao se encontrar uma solução para o problema, pode-se gerar uma solução simétrica a esta.

2.3 Utilização do PDGDM para determinar proteínas completas

Em [11, 20], foram desenvolvidas técnicas que utilizam a definição do PDGDM para determinar a cadeia principal de proteínas. Basicamente, os algoritmos desenvolvidos nestes trabalhos utilizam a formulação discreta apresentada na Seção 2.2.1 para determinar a posição de cada átomo pertencente a cadeia principal. Além disso, quando dispõem de informação adicional, como por exemplo uma distância extra entre algum par de átomos, eles a utilizam para eliminar posições, que a princípio seriam válidas para um determinado átomo, considerando a formulação discreta. Com isso, eles são capazes de reduzir consideravelmente o espaço de busca. Mais detalhes a respeito do funcionamento destes algoritmos serão mostrados no Capítulo 3.

O objetivo principal desta dissertação consiste em estender o Algoritmo II apresentado em [11], de modo que ele seja capaz de determinar estruturas completas de proteínas.

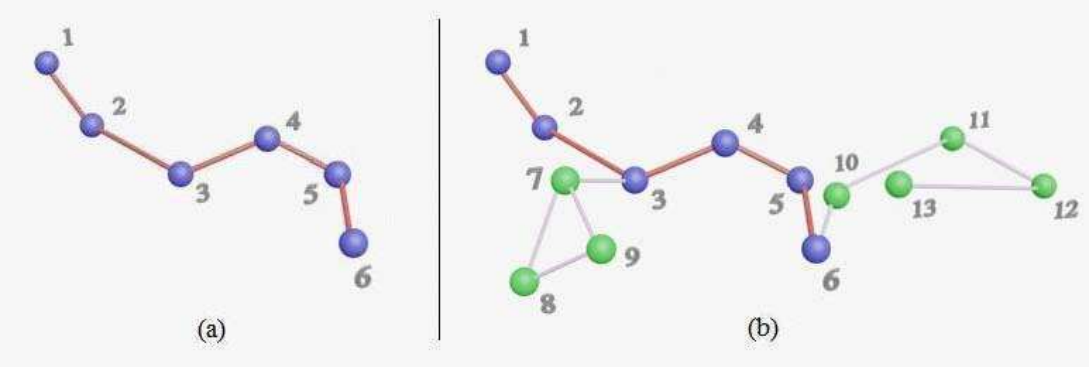


Figura 2.4: Cadeia principal (a) e proteína completa (b).

Na Figura 2.4, é mostrado a cadeia principal de uma proteína (a) e a mesma proteína com todos os seus átomos (b). Nessa figura, em (b), os átomos em azul representam a cadeia principal e os átomos em verde pertencem às cadeias laterais da proteína. Essas

cadeias estão sempre ligadas a cadeia principal e aparecem sistematicamente a cada três átomos da mesma [6], como ilustrado na Figura 2.4 (b).

Verificou-se empiricamente que as cadeias laterais, se tratadas isoladamente, podem ser encaradas como instâncias do PDGDM, uma vez que, com os devidos tratamentos, elas satisfazem as Hipóteses 1 e 2 da formulação apresentada na Seção 2.2.1. Assim, para determinar a estrutura completa de uma proteína, deve ser resolvida não uma, mas várias instâncias do PDGDM. A dificuldade se encontra em como resolver estas instâncias de forma eficiente e integrada.

2.3.1 Simetria de soluções na determinação de proteínas completas

Na Seção 2.2.1.2, foi apresentada a propriedade de simetria das soluções para instâncias do PDGDM. Contudo, quando se deseja determinar a estrutura completa de uma proteína, a cadeia principal e cada cadeia lateral representam instâncias distintas do PDGDM. Assim, a propriedade de simetria de soluções pode ser utilizada isoladamente, para cada uma das cadeias laterias, de modo a reduzir o custo computacional.

Além disso, deseja-se mostrar que a propriedade de simetria pode ser aplicada à soluções de proteínas completas, de forma que para qualquer solução S do problema contendo todos os átomos, existe uma solução S' simétrica a S. Para isso, toma-se como exemplo a Figura 2.5. Neste exemplo, a cadeia principal da proteína é representada pelos átomos 1, 2, 3 e 6, e os átomos 4 e 5 formam uma cadeia lateral. Nesta figura, são mostradas duas soluções simétricas para o problema, sendo que todos os átomos da proteína estão sendo levados em consideração. Em ambas as soluções, os três primeiros átomos da cadeia principal estão nas posições 1, 2 e 3, respectivamente. Para uma das soluções, o quarto átomo da cadeia principal está na posição 6, e para a outra ele está em $6'$. Algo semelhante ocorre com a cadeia lateral dessa proteína: em uma solução os átomos estão nas posições 4 e 5, e na outra, eles estão em $4'$ e $5'$. Note que as posições dos átomos utilizadas na Figura 2.5 são as mesmas da Figura 2.3, a única diferença é a cadeia a qual eles pertecem. Sendo assim, da mesma maneira que a propriedade de simetria funciona para a cadeia principal de uma proteína, ela funciona para a proteína como um todo.

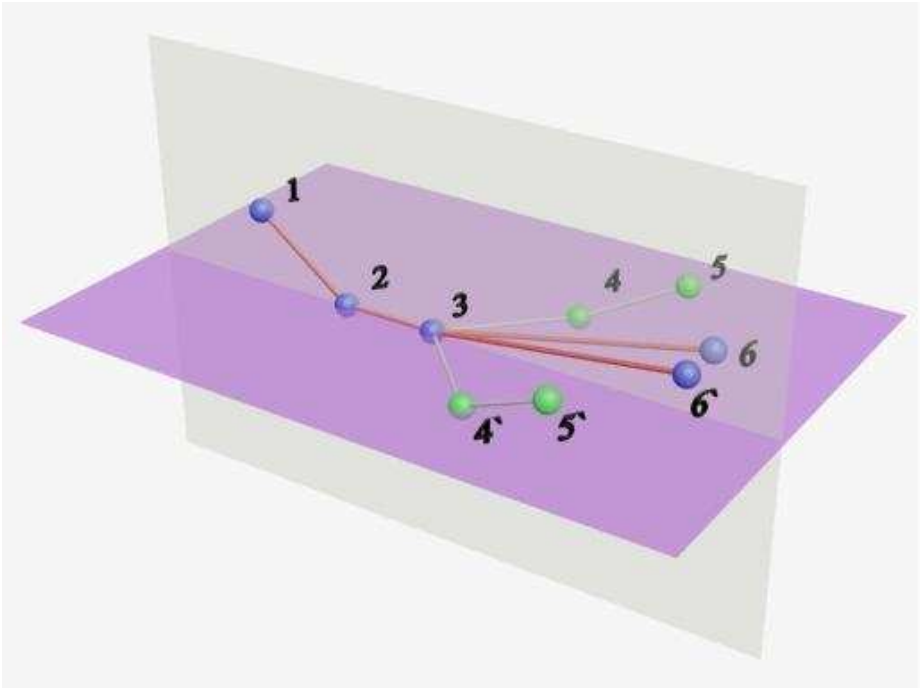


Figura 2.5: Simetria de soluções completas.

2.3.2 Problema de Ordenação de Átomos em Proteínas - POAP

Como descrito na Seção 2.2.1, uma instância do PDGDM deve conter uma sequência de átomos onde duas hipóteses devem ser verificadas. Será mostrado que, dependendo de como esta sequência de átomos está organizada, as hipóteses podem ser satisfeitas ou não. Para que isso seja demonstrado, as duas hipóteses serão tratadas separadamente.

A Hipótese 1 determina que todas as distâncias e ângulos de ligação entre quatro átomos consecutivos devem ser conhecidos. Na realidade, para que esta hipótese seja satisfeita, basta o conhecimento das distâncias, pois com elas é possível calcular os ângulos.

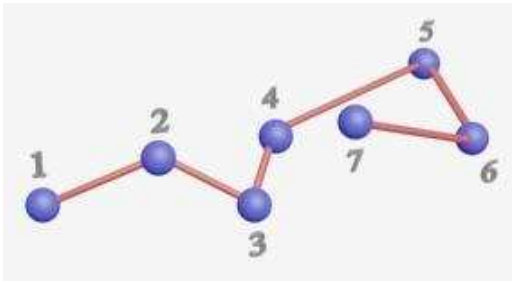


Figura 2.6: Exemplo de instância do POAPDGDM.

Considerando como exemplo a Figura 2.6 e supondo que as distâncias conhecidas entre os átomos dessa figura sejam as seguintes:

$$
\begin{aligned}
d_{1,2} &= 2,0,\\
d_{1,3} &= 3,5,\\
d_{1,4} &= 3,5,\\
d_{2,3} &= 1,5,\\
d_{2,4} &= 2,0,\\
d_{2,7} &= 3,0,\\
d_{3,4} &= 1,5,\\
d_{3,5} &= 4,0,\\
d_{3,6} &= 4,5,\\
d_{3,7} &= 2,5,\\
d_{4,5} &= 3,0,\\
d_{4,6} &= 4,0,\\
d_{4,7} &= 1,5,\\
d_{5,6} &= 2,5,\\
d_{5,7} &= 1,5,\\
d_{6,7} &= 2,5.
\end{aligned}
$$

Suponha que a ordenação da seqüência de entrada dos átomos, considerada pelo método de resolução, seja 1, 2, 3, 4, 5, 6 e 7. Para que a Hipótese I seja satisfeita, seriam necessárias as distâncias: $d_{1,2}$, $d_{1,3}$, $d_{1,4}$, $d_{2,3}$, $d_{2,4}$, $d_{2,5}$, $d_{3,4}$, $d_{3,5}$, $d_{3,6}$, $d_{4,5}$, $d_{4,6}$, $d_{4,7}$, $d_{5,6}$, $d_{5,7}$ e $d_{6,7}$. Como a distância $d_{2,5}$ não é conhecida, esta instância não poderia ser resolvida pelo PDGDM, utilizando esta seqüência de átomos. Contudo, se a ordenação da seqüência fosse 1, 2, 3, 4, 7, 5 e 6, as distâncias necessárias seriam: $d_{1,2}$, $d_{1,3}$, $d_{1,4}$, $d_{2,3}$, $d_{2,4}$, $d_{2,7}$, $d_{3,4}$, $d_{3,7}$, $d_{3,5}$, $d_{4,7}$, $d_{4,5}$, $d_{4,6}$, $d_{7,5}$, $d_{7,6}$ e $d_{5,6}$. Neste caso todas elas são conhecidas, satisfazendo a Hipótese 1.

A Hipótese 2 determina que, para quaisquer três átomos consecutivos, denominados A, B e C, o ângulo formado pelos segmentos de reta AB e BC não pode ser múltiplo de π. Se a seqüência dos átomos fosse 1, 2, 4, 3, 7, 5 e 6 a Hipótese 1 seria verdadeira, porém o ângulo entre segmentos de reta $S1$, formado pelos átomos 3 e 7, e $S2$, formado pelos átomos 7 e 5, é igual a π ($d_{3,7} + d_{7,5} = d_{3,5}$), o que faz com que a Hipótese 2 não seja satisfeita. Entretanto, se fosse utilizada a seqüência 1, 2, 3, 4, 7, 5 e 6, proposta anteriormente, as duas hipóteses seriam válidas.

Logo, para que o método de resolução que será apresentado neste trabalho funcione, é necessário que os átomos sejam organizados de modo a atender as hipóteses necessárias do PDGDM. Não foram encontrados na literatura relatos a respeito deste problema, até

porque os dois únicos trabalhos que utilizaram o PDGDM [20, 11] tratam apenas a cadeia principal. Sendo que esta, geralmente, atende as hipóteses necessárias, se a ordem natural dos átomos for utilizada. Contudo, o mesmo não ocorre com as cadeias laterais. Portanto, o Problema de Ordenação de Átomos em Proteínas (POAP), será definido nesta dissertação.

Problema de Ordenação de Átomos em Proteínas (POAP): dado um conjunto C com n átomos e um conjunto S com k distâncias entre átomos, determine uma seqüência de átomos onde:

- Todas as distâncias entre quaisquer quatro átomos consecutivos sejam conhecidas;
- Para quaisquer três átomos consecutivos, denominados A, B e C, o ângulo formado pelos segmentos de reta AB e BC não seja múltiplo de π.

Este problema pode ser visto como o problema de encontrar um caminho hamiltoniano [2], com duas restrições adicionais, que são justamente as duas condições acima. O Problema do Caminho Hamiltoniano (PCaH) pode ser definido como: dado um grafo não dirigido $G = (V, E)$, verificar se existe um caminho entre dois vértices distintos que passe por cada vértice do grafo exatamente uma vez. Como o PCaH pertence à classe NP-Completo, o POAP também é NP-Completo.

2.3.2.1 Modelo matemático para o POAP

Como necessita-se organizar somente as cadeias laterais e o número de átomos nestas cadeias é pequeno, em média 18 átomos (com base nos 20 principais aminoácidos), optou-se por resolver o problema por meio de um modelo matemático. O modelo desenvolvido é descrito abaixo.

- Dados de entrada:
  - n - número de átomos.
  - $distMax$ - valor da maior distância conhecida entre quaisquer dois átomos da proteína.
  - $distancia_{i,i'}$ - distância entre os átomos i e i' (caso a distância não seja conhecida, seu valor é definido arbitrariamente como 100 x $distMax$).
- Variáveis de decisão:

- $p_{i,j}$ - variável binária que indica se o átomo i está ou não na posição j. A variável tem valor 1, se o átomo i está na posição j e 0 caso contrário.
- $d_{i,i'}$ - variável binária que indica que, obrigatoriamente a distância entre os átomos i e i' tenha que ser conhecida. Esta variavel recebe o valor 1 quando $|$(posição do átomo i) − (posição do átomo i') $| \leq 3$, o que indica que a distância deve ser conhecida.
- $f_{i,i'}$ - variável binária de folga, que indica que a distância entre os átomos i e i' é maior que $distMax$, quando não deveria ser, ou seja quando $d_{i,i'}$ é igual a 1. Esta variável recebe o valor 1 quando a distância entre i e i' é maior que $distMax$ e $d_{i,i'} = 1$, e recebe 0 caso contrário.
- $v_{i,i'}$ - variável binária que indica se a posição do átomo i' é imediatamente posterior a posição do átomo i. Ou seja, esta variável recebe o valor 1 se (posição do átomo i') − (posição do átomo i) = 1, e 0 caso o contrário.

- Função objetivo:

$$min \sum_{i=0}^{i=n-1} \sum_{i'=0}^{i'=n-1} f_{i,i'} \qquad \forall\ i \neq i' \tag{2.7}$$

- Restrições:
  - Em cada posição j deverá haver exatamente um átomo.

$$\sum_{i=0}^{i=n-1} p_{i,j} = 1 \qquad \forall\ j\ tal\ que\ 0 \leq j \leq n-1 \tag{2.8}$$

  - Cada átomo i deverá estar exatamente em uma posição.

$$\sum_{j=0}^{j=n-1} p_{i,j} = 1 \qquad \forall\ i\ tal\ que\ 0 \leq i \leq n-1 \tag{2.9}$$

  - Se a distância entre os átomos i e i' tiver que ser conhecida, ou seja se $|$ (posição do átomo i) − (posição do átomo i') $| \leq 3$, $d_{i,i'}$ deve ser 1.

$$\begin{aligned} p_{i,j} + p_{i',j'} \leq d_{i,i'} + 1 \qquad & \forall\ i, i'\ tal\ que\ 0 \leq i, i' \leq n-1\ e\ i \neq i', \\ & \forall\ j, j'\ tal\ que\ 0 \leq j, j' \leq n-1\ e\ j' - j \leq 3\ e\ j \neq j' \end{aligned} \tag{2.10}$$

– Se a posição do átomo i' é imediatamente posterior a posição do átomo i, $v_{i,i'}$ deve ser 1.

$$p_{i,j} + p_{i',j'} \leq v_{i,i'} + 1 \qquad \forall\ i, i'\ tal\ que\ 0 \leq i, i' \leq n-1\ e\ i \neq i', \\ \forall\ j, j'\ tal\ que\ 0 \leq j, j' \leq n-1\ e\ j' - j = 1 \qquad (2.11)$$

– Todas as distâncias entre quaisquer quatro átomos consecutivos deverão ser menores o iguais a $distMax$ (Hipótese 1).

$$d_{i,i'}.distancia_{i,i'} \leq distMax + (1000.distMax.f_{i,i'}) \\ \forall\ i, i'\ tal\ que\ 0 \leq i, i' \leq n-1 \qquad (2.12)$$

– Para quaisquer três átomos consecutivos, denominados A, B e C, o ângulo formado pelos segmentos de reta AB e BC não pode ser múltiplo de π. Ou seja, $|AB| + |BC| > |AC|$ (Hipótese 2).

$$v_{i,i'}.distancia_{i,i'} + v_{i',i''}.distancia_{i',i''} > distancia_{i,i''} - 10000.(2 - (v_{i,i'} + v_{i',i''})) \\ \forall\ i, i', i''\ tal\ que\ 0 \leq i, i', i'' \leq n-1\ e\ i \neq i'\ e\ i \neq i''\ e\ i' \neq i'' \\ (2.13)$$

Para que uma solução deste modelo, caracterize uma instância do PDGDM, o valor da sua função objetivo deverá ser 0, pois somente assim as duas hipóteses serão satisfeitas.

Capítulo 3

Algoritmos

Este capítulo apresenta os três algoritmos propostos e implementados para o PDGDM, nesta dissertação. O primeiro deles considera apenas a cadeia principal e os outros dois tratam a proteína de forma completa.

As estratégias para os algoritmos aqui propostos se baseiam na estrutura combinatória do PDGDM, onde em cada iteração, o i-ésimo átomo pode ser posicionado em uma das duas possíveis posições: x_i ou x'_i. Mais especificamente, a estrutura do problema pode ser representada por uma árvore binária, como exemplificado na Figura 3.1, que apresenta um cadeia com 6 átomos. Neste exemplo, considera-se que os átomos 1, 2 e 3 sejam fixos e que os átomos 4, 5 e 6 possam ser colocados em duas, quatro e oito possíveis posições, respectivamente.

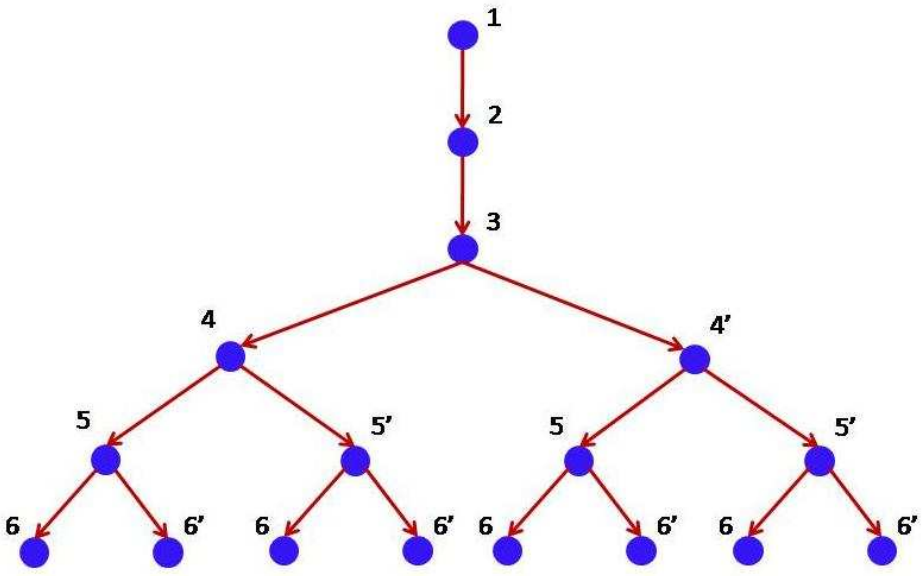


Figura 3.1: *Representação em árvore binária.*

Seja $F = \{(j,i)\ tal\ que\ |Posicao_i - Posicao_j| \geq 4\}$ o conjunto de pares de átomos com distâncias conhecidas, mas que não são essenciais para satisfazer as hipóteses do problema. Quando um átomo é posicionado em x_i ou x'_i, pode ser que esta posição não esteja de acordo com todas as distâncias entre os pares $(j,i) \in F$. Para isso, verifica-se

$$(||x_j - x_i||^2 - d_{j,i}^2)^2 < \varepsilon, \tag{3.1}$$

onde $\varepsilon > 0$ é uma tolerância dada. Diante disso, as seguintes situações podem ocorrer: ambas as posições estão corretas, somente uma das posições está correta, ou nenhuma delas está correta.

Assim, os algoritmos devem percorrer de alguma forma essa estrutura de árvore e realizar podas nos nós onde a posição aferida para o átomo é incorreta. Sendo que, durante o percurso, quando uma folha é alcançada (último átomo da cadeia) e esta tem uma posição que respeita as restrições de distância, uma solução para o problema é encontrada.

3.1 Algoritmo para Cadeia Principal por Níveis - ACPN

Em [11], foram propostos dois algoritmos para o PDGDM, capazes de determinar somente a cadeia principal das proteínas. O primeiro deles, denominado *Algoritmo I*, consiste, basicamente, em explorar a estrutura de árvore da Figura 3.1 por nível. Ou seja, para cada $(i,j) \in F$, a árvore é expandida até o nível j e neste nível, verifica-se, em cada nó folha, se a inequação 3.1 é satisfeita. Caso não seja, o nó folha correspondente é podado. No passo seguinte, somente os nós folhas remanescentes são expandidos até o próximo nível indicado pelo conjunto F. Isso se repete até que todos os elementos de F sejam verificados. Ao final de todas as iterações, os nós folhas remanescentes no nível do último elemento de F representam, juntamente com seus ancestrais, soluções para o problema. Nota-se que, se a quantidade de nós remanescentes em um dado nível for muito grande, o algoritmo poderá ter problemas com o tamanho da memória ocupada. Observa-se ainda que, ou o algoritmo encontra todas as soluções possíveis para o problema, ou não encontra nenhuma solução, caso tenha tido problema de gerenciamento de memória.

O ACPN proposto nesta dissertação foi baseado no *Algoritmo I*, porém optou-se por utilizar uma estrutura de dados diferente. Na implementação realizada em [11] para o *Algoritmo I*, foi utilizado um conjunto de listas e matrizes para representar a estrutura de árvore. Cada lista dessa implementação representa um ramo da árvore que sempre se inicia na raíz e termina em uma folha e cada matriz armazena o produto cumulativo

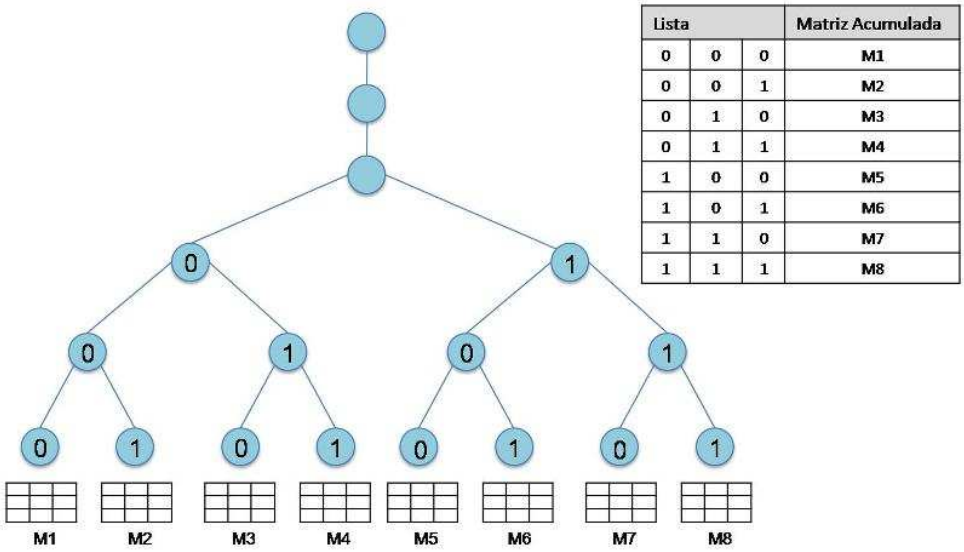


Figura 3.2: Representação da estrutura de dados utilizada pelo *Algoritmo I*.

$(B_1B_2...B_i)$ do ramo ao qual ela está realacionada, como na Figura 3.2.

Nessa figura, os valores 0 e 1 contidos nas listas representam o lado do nó, sendo 0 o filho da esquerda e 1 o filho da direita. O autor queria com isso realizar a implementação da maneira mais simples possível, armazenando em memória apenas valores booleanos e uma única matriz para cada ramo da árvore. No algoritmo proposto pelo presente trabalho, a estrutura de dados utilizada para representar a árvore foi realmente uma Árvore Binária. Como pode ser visto na Figura 3.3, nesta árvore armazena-se para cada nó, um booleano que informa o seu lado (0 ou 1), uma matriz acumulada que define as suas coordenadas e dois ponteiros que apontam para os seus filhos.

Para ambos os algoritmos, utilizando a propriedade de simetria, a complexidade é $O(2^{n-3})$, que representa o número máximo de nós na árvore para uma cadeia com n átomos. Apesar da estrutura de árvore binária ser mais complexa e ter que armazenar mais dados por nó, ela evita o armazenamento redundante da informação de cada nó que é feito no caso da utilização de listas. Como exemplo, a informação "lado" do nó da esquerda do quarto nível da Figura 3.2 é repetida quatro vezes (quatro primeiras listas), se utilizado o Algoritmo I. No caso do ACPN, essa informação não seria repetida. Por outro lado, para esta cadeia da Figura 3.2, que contém 6 átomos, o ACPN necessita armazenar 17 matrizes, enquanto o Algoritmo I somente 8. Contudo, para problemas maiores, a quantidade total de informação armazenada pelo Algoritmo I é maior do que a do ACPN. Por exemplo, para uma proteína com 100 átomos o ACPN armazenaria $2^{97} + 2$ nós, cada um contendo uma matriz, um booleano e dois ponteiros. Por sua vez, o Algoritmo I armazenaria $2^{96} + 2$ listas, cada uma contendo 96 booleanos e $2^{96} + 2$ matrizes.

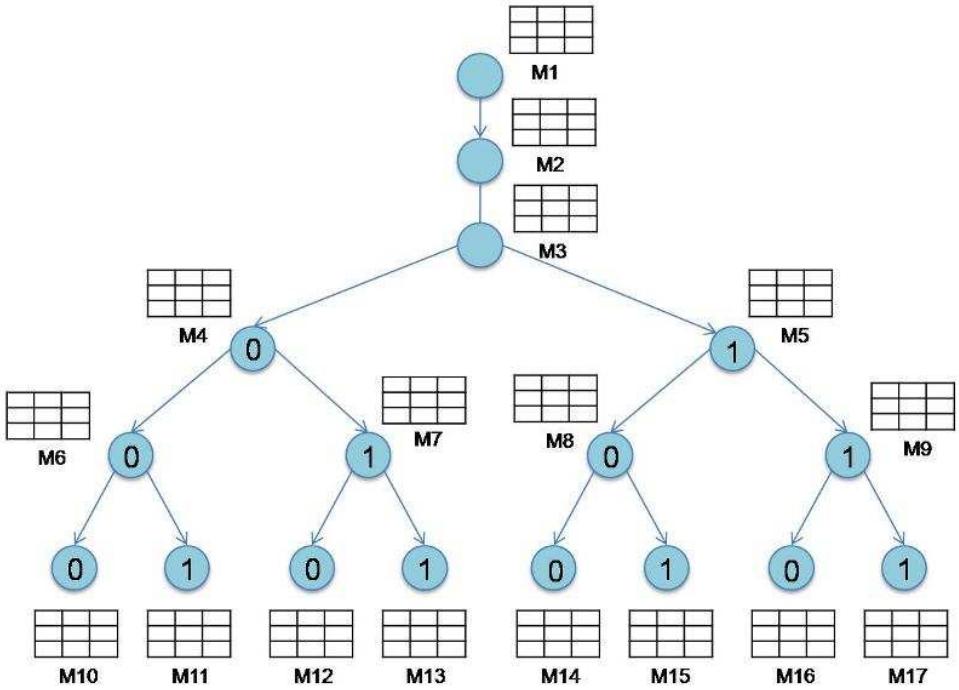


Figura 3.3: Representação da estrutura de dados utilizada pelo ACPN.

No Capítulo 4 são apresentados testes que comparam a eficiência das duas abordagens. Abaixo, é mostrado o pseudo-código do ACPN.

Algoritmo 1 *ACPN*

1: $T = \{x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4\}$;
2: **enquanto** $F \neq \oslash$ **faça**
3: $f \leftarrow$ *primeiro par* $(i, j) \in F$;
4: $T \leftarrow$ *ExpandeArvore(* T, f *)*;
5: $T \leftarrow$ *PodaArvore(* T, f *)*;
6: $F \leftarrow F - \{f\}$;
7: **fim enquanto**
8: Todos os caminhos em T, até os nós folhas, são soluções e através destas são geradas as soluções simétricas.

No pseudo-código apresentado para o ACPN, T é uma representação de árvore. T é inicializada com $T = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$, dado que os três primeiros átomos podem ser fixados nas posições x_1, x_2, x_3, e o quarto átomo em x_4. Na realidade, o quarto átomo pode ser fixado em x_4 ou x'_4, porém devido a existência da simetria explicada na Seção 2.2.1.2, somente uma dessas duas posições é escolhida. Entre as linhas 2 e 7, é executado o *loop* que explora a árvore até que todas as distâncias adicionais (conjunto F) sejam consideradas. Na linha 3, f sempre recebe o par $(i, j) \in F$ o qual contém o menor j do conjunto e através do procedimento *ExpandeArvore*, a árvore é expandida até o nível de altura j. O procedimento *ExpandeArvore* explora a árvore até o nível de

Algoritmo 2 *ExpandeArvore*(T, f)

1: **enquanto** $nivelDaArvore < f.j$ **faça**
2: **para** *cada nó folha* **faça**
3: *busca matriz de torção acumulada* Q_{i-1} *do nível anterior (pai)*;
4: *calcula* $Q_i = Q_{i-1}B_i$, $Q'_i = Q_{i-1}B'_i$;
5: *cria* v *que representa um nó da árvore e armazena neste* Q_i;
6: *cria* v' *que representa um nó da árvore e armazena neste* Q'_i;
7: $T \leftarrow$ *atribui* v *como filho esquerdo e* v' *como filho direito da folha da vez*;
8: **fim para**
9: **retorna** T;
10: **fim enquanto**

Algoritmo 3 *PodaArvore*(T, f)

1: **para** *cada nó folha* **faça**
2: **se** $(||x_{f.j} - x_{f.i}||^2 - d^2_{f.j,f.i})^2 < \varepsilon$ **então**
3: *Continue*;
4: **senão**
5: *Elimina respectivo nó folha de* T ;
6: **fim se**
7: **fim para**
8: **retorna** T;

altura $j \in f$, sendo que em cada nível i são calculadas as matrizes acumuladas Q_i para os nós da esquerda e Q'_i para os nós da direita, via Eq. 2.5. Na linha 5, é chamado o procedimento *PodaArvore* que verifica se algum nó do último nível explorado da árvore não está de acordo com a distância adicional f, sendo que neste caso, o nó é removido.

Exemplo: Veja um exemplo simples da aplicação do ACPN. Será utilizado como instância, somente os 6 primeiros átomos da cadeia principal da proteína 1BRV [1]. Para esta instância, as seguintes distâncias são conhecidas:

- Distâncias essenciais (Hipótese I):
  - $d_{1,2} = 1,49760$
  - $d_{1,3} = 2,61604$
  - $d_{1,4} = 3,20367$
  - $d_{2,3} = 1,57365$
  - $d_{2,4} = 2,59210$
  - $d_{2,5} = 3,99307$
  - $d_{3,4} = 1,36472$
  - $d_{3,5} = 2,52866$

  - $d_{3,6} = 3,14638$
  - $d_{4,5} = 1,49261$
  - $d_{4,6} = 2,60190$
  - $d_{5,6} = 1,55895$
- Distâncias adicionais (conjunto F):
  - $d_{1,5} = 4,64614$
  - $d_{2,6} = 4,67276$

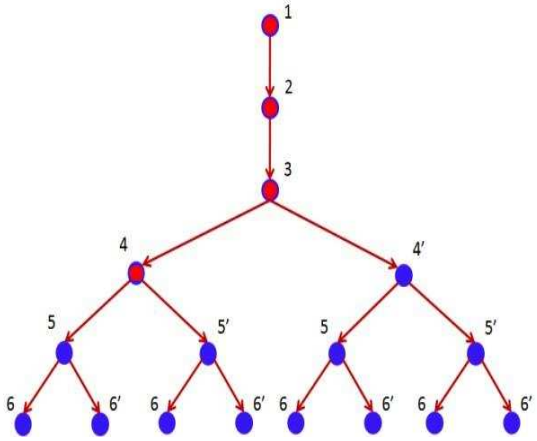
Através da Figura 3.4, pode-se verificar o comportamento do algoritmo, sendo que os nós em vermelho representam os nós que estão em memória no momento. Inicialmente, a árvore é inicializada com os quatro primeiros átomos fixos (Figura 3.4(a)). Em seguida, utilizando a distância adicional $d_{1,5}$, a árvore é expandida até o nível 5 (Figura 3.4(b)), pois $k = 5$. No próximo passo, verificam-se quais nós folhas estão corretos, para que seja feita a poda (Figura 3.4(c)). Novamente, o algoritmo faz a expansão da árvore até o nível indicado pela distancia adicional $d_{2,6}$, uma vez que agora $k = 6$ (Figura 3.4(d)). Em seguida, na Figura 3.4(e), uma nova poda é feita e os nós folhas remanescente são as soluções do problema. Finalmente, as soluções simétricas são geradas (Figura 3.4(f)). Note que, para esta instância, são encontradas duas soluções.

3.2 Algoritmos Propostos para a Proteína Completa

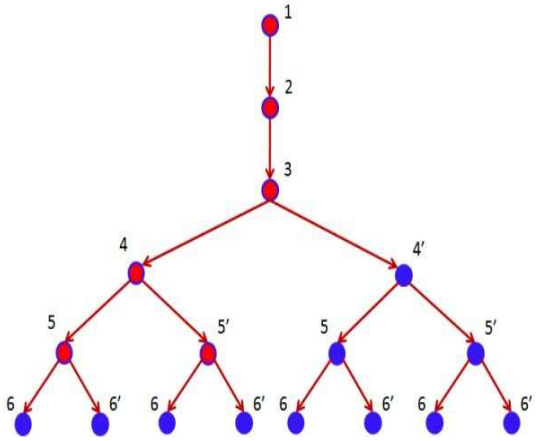
Ao abordar o problema de determinar a estrutura de uma proteína por completo, além da cadeia principal, as cadeias laterais devem ser consideradas, o que produz uma estrutura como a da Figura 3.5, perdendo-se a estrutura de árvore binária apresentada na Figura 3.1. Isso pode não ser interessante, uma vez que esta estrutura tem gerado bons resultados em outros trabalhos [11, 20].

Uma maneira de se manter a estrutura de árvore é através do encapsulamento das cadeias laterais, como é mostrado na Figura 3.6.

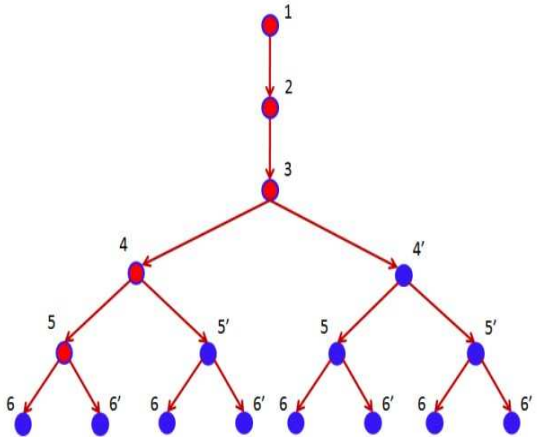
Nesta figura, existem duas cadeias laterais, uma ligada ao terceiro átomo da cadeia principal e outra ao sexto. Para que o problema possa ser resolvido de forma integrada e a estrutura de árvore seja mantida, tratam-se as cadeias laterais como instâncias do PDGDM e estas são resolvidas durante a exploração da árvore (cadeia principal). Para que isso possa ser feito, além de ordenar os átomos das cadeias laterais, como explicado na



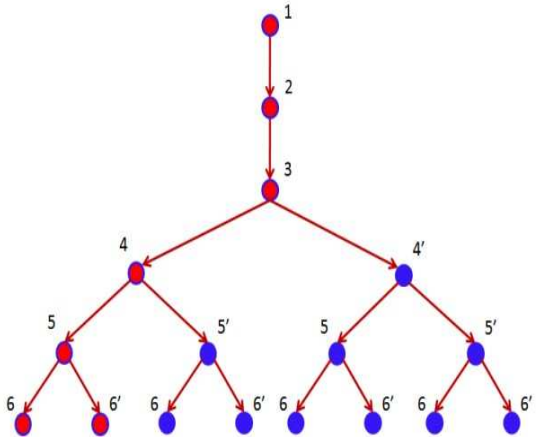
(a) *4 átomos iniciais fixos*



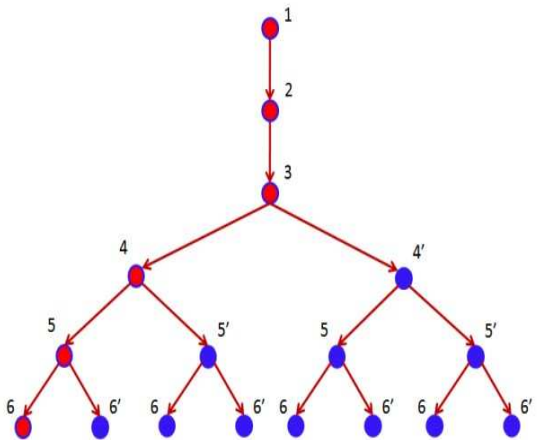
(b) *expansão da árvore até o nível 5*



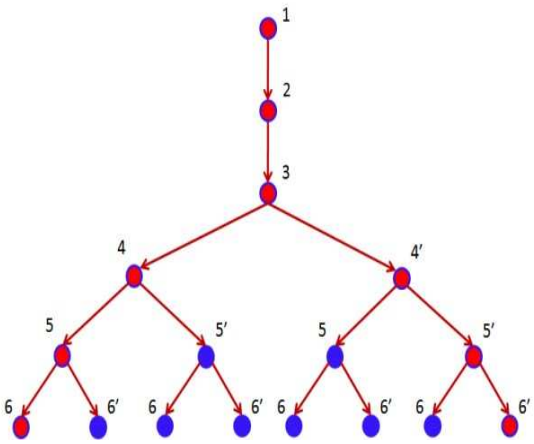
(c) *Poda dos nós folhas incompatíveis (5′)*



(d) *expansão da árvore até o nível 6*



(e) *Poda dos nós folhas incompatíveis (6′)*



(f) *Geração das soluções simétricas*

Figura 3.4: *Exemplo do algoritmo ACP*

Seçao 2.3.2, de modo que as hipóteses sejam satisfeitas, é necessário considerar a cadeia como sendo a cadeia lateral, adicionada do átomo da cadeia principal ao qual ela está ligada, juntamente com os dois átomos da cadeia principal antecedentes a este. Estes três

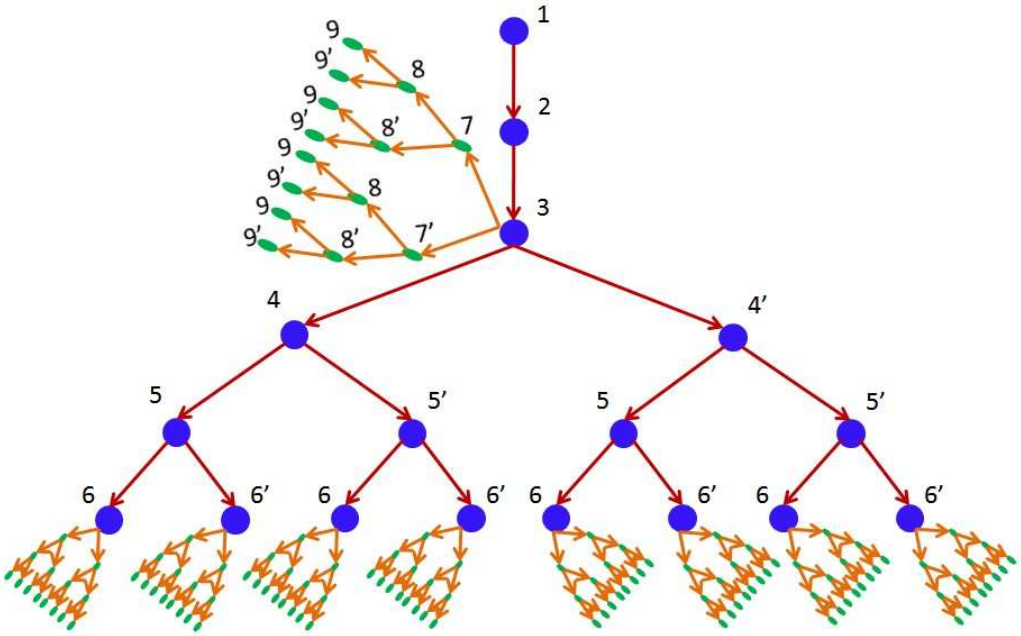


Figura 3.5: *Ilustração da perda de estrutura de árvore, com a inclusão das cadeias laterais.*

átomos da cadeia principal são utilizados como os três átomos fixos do problema a ser resolvido. Como exemplo, para a instância da Figura 3.6, toda vez que o sexto nível da árvore é atingido, a cadeia lateral ligada a este nível deve utilizar a informação da posição determinada para os átomos 4, 5 e 6 da cadeia principal, ficando estes como os três primeiros átomos fixos e os restantes (átomos da cadeia lateral) para serem determinados através da resolução do PDGDM. Com isso, é produzido um conjunto de soluções para cada cadeia lateral e cada conjunto é "anexado" ao seu respectivo nível na árvore.

Com esta nova estrutura, quando um nível da árvore é explorado, além das restrições de distância com os átomos anteriores da cadeia principal, o átomo que está sendo determinado deverá ter uma posição compatível com pelo menos uma das soluções de cada cadeia lateral que já foi explorada. Além disso, ao se explorar uma cadeia lateral, deve ser produzida ao menos uma solução que seja compatível com todos os átomos já fixos na cadeia principal, caso contrário, o ramo em questão deverá ser podado.

3.2.1 Algoritmo para Proteína Completa - APC

Existem alguns modos de se determinar a estrutura tridimensional de toda uma proteína, utilizando o PDGDM. Uma maneira é resolver o PDGDM para a cadeia principal e cada cadeia lateral separadamente. Em seguida, combina-se cada estrutura encontrada para a cadeia principal com as estruturas encontradas para cada uma das cadeias laterais. Para

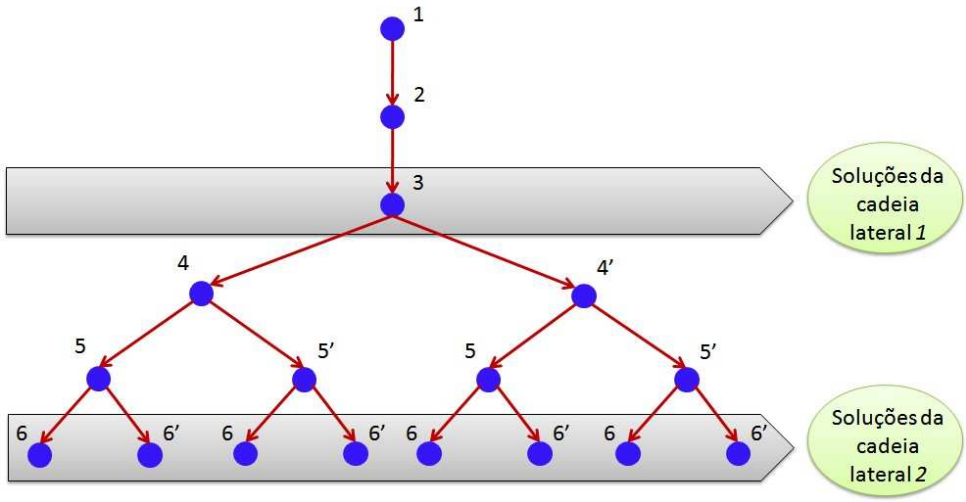


Figura 3.6: *Encapsulamento das cadeias laterais.*

cada combinação realizada, deve se verificar se as estruturas das cadeias laterais utilizadas são compatíveis entre si e também com a estrutura da cadeia principal da combinação da vez. Ser compatível, neste caso, significa que as restrições de distância entre os átomos são satisfeitas.

Utilizando esta abordagem, certamente o custo computacional seria altíssimo e muitos cálculos seriam desperdiçados, pois, provavelmente, diversas combinações seriam inválidas. Acredita-se que uma maneira de se evitar tal esforço computacional seja através de uma resolução integrada da cadeia principal e das cadeias laterais. Baseado nesta idéia, foi criado o APC.

O objetivo do APC é explorar a árvore de modo a respeitar as restrições de distâncias da seguinte maneira: as soluções geradas para as cadeias laterais devem respeitar as restrições de distância em relação aos átomos da cadeia principal que já estão fixos; um átomo da cadeia principal só pode ser posicionado em uma posição compatível com pelo menos uma solução de cada cadeia lateral já explorada. Com essas estratégias, muitos ramos infrutíferos podem ser podados.

No Capítulo 4, onde os resultados computacionais são apresentados, pode-se notar a limitação de gerenciamento de memória dos algoritmos *BP* [20], *Algoritmo I* [11] e ACPN. Portanto, optou-se por desenvolver o APC, baseado no *Algoritmo II* de [11], que por não apresentar problemas de gerenciamento de memória, foi capaz de determinar a cadeia principal de diversas instâncias do PDB.

O *Algoritmo II* [11], ao invés de explorar a árvore ilustrada na Figura 3.1 por nível, a explora em profundidade, ou seja, avança através da expansão do primeiro nó filho da

árvore e se aprofunda, até que chegue no nível j do par $(i, j) \in F$. Neste momento, a restrição de desigualdade 3.1 é verificada. Caso seja satisfeita, o algoritmo continua a busca em profundidade, e caso contrário, ocorre o processo de "poda" e o algoritmo retrocede (*backtracking*) ao nível anterior da árvore e recomeça no próximo nó. A vantagem do caminhamento em profundidade é que um único ramo da árvore precisa ser mantido na memória, diferentemente do caminhamento em nível, onde é necessário manter as informações a respeito de toda a árvore.

O Algoritmo 4 apresenta o pseudo-código do APC. Neste pseudo-código, n é a quantidade de átomos da cadeia principal e P é um vetor com n nós, utilizado para representar o ramo o qual o algoritmo está explorando. Cada nó contém as seguintes informações:

- Produto cumulativo das matrizes de torção $Q_i = \prod_{j=1}^{i} B_j$;
- Variável de controle (subarvore) para auxiliar no caminhamento da árvore.

A variável *nivel* indica o nível em que o algoritmo se encontra na árvore, L é o conjunto de cadeias laterais da proteína, F continua sendo o conjunto de pares de átomos com distâncias adicionais da cadeia principal (tuplas com i, j e *distância*) e os conjuntos B e B' representam as matrizes de torção. O algoritmo começa inicializando P com os três primeiros átomos fixos e atualizando a variável *nivel* para 3. Logo após, na linha 8, é criada uma distância virtual, do último átomo para ele mesmo, sendo esta igual a 0. Essa distância é necessária para forçar o algoritmo a explorar os nós até o ultimo nível da árvore, pois mesmo que não existam restrições de distância até o último nível, podem haver cadeias laterais e estas precisam ser exploradas e tratadas adequadamente. Em seguida, na linha 11, é resolvido o POAP para cada cadeia lateral, organizando-as de modo que possam ser resolvidas pelo PDGDM. Na linha 13, é criada a variável k com o valor 0 (esta variável é utilizada como índice do conjunto F). A partir desse ponto, o algoritmo começa o seu percurso em profundidade na árvore. Sendo que, sempre que uma cadeia lateral é alcançada (níveis que são múltiplo de 3), linha 17, a mesma é resolvida pelo procedimento $resolvePDGDMCadeiaLateral()$, que nada mais é do que uma implementação do *Algoritmo II* de [11].

Resolvida a cadeia lateral, a exploração do ramo somente continua se houver pelo menos uma solução da cadeia lateral que seja compatível com todos os átomos já fixos na cadeia principal. Para fazer tal verificação é chamado o procedimento $filtraSolsCompativeisCadeiaPrincipal()$ que utiliza o conjunto de distâncias entre a cadeia principal e a cadeia lateral. Caso esta verificação falhe, é chamado o procedimento $Poda$, mostrado no

Algoritmo 9. Este procedimento tem a função de retroceder na árvore até um nó onde não houve poda ($subarvore \neq PODA$), ou seja, ainda se pode explorar o seu filho da direita. Além disso nas suas linhas 6, 7 e 8, ele decrementa o valor de k, para garantir que as distâncias adicionais referentes a níveis abaixo do atual, *nivel*, sejam verificadas novamente.

Na linha 22, o algoritmo verifica se o procedimento de poda fez com que toda árvore fosse explorada, subindo até o segundo nível da mesma, sendo que neste caso a execução é finalizada. Na linha 25 do APC, mais um nível da árvore é explorado, através do procedimento $Explora$. Este procedimento analisa qual filho do nó atual ele deve explorar (esquerdo ou direito). Através da variável *subarvore*, cria-se este novo nó e calcula-se a sua matriz Q_i, multiplicando a matriz B_i, no caso do filho da esquerda, ou B'_i, no caso do filho da direita, pela matriz Q_{i-1} que pertence ao seu pai. Para que este nó seja validado, a sua posição (obtida a partir de Q_i) deve ser compatível com pelo menos uma solução de cada cadeia lateral já resolvida. Essa verificação é feita pelo procedimento $CompativelCadeiasLateraisExploradas()$. Se esta verificação não puder ser satisfeita, o procedimento $Poda$ é chamado. Caso contrário, é necessário que o átomo do nível atual seja fixado na sua respectiva cadeia lateral. Essa fixação é feita pelo procedimento $FixaAtomoCadLateral()$, que tem como função designar cada átomo da cadeia principal como um dos três primeiros átomos fixos da cadeia lateral que o utilizará, como explicado na Subseção 3.2.

Na linha 28 do algoritmo APC, é verificado se a exploração da árvore chegou ao final, e neste caso, a execução é finalizada. Caso a exploração ainda não tenha terminado, na linha 29 é verificado se a posição do átomo do nível *nivel* da cadeia principal é compatível com a distância adicional $F[k]$, de modo a satisfazer a Desigualdade 3.1. Caso não seja, o nó é podado pelo procedimento $Poda$. Se o nó pôde ser fixado, na linha 32 o k é incrementado para que a próxima distância adicional seja avaliada. Quando o último nível da árvore é atingido, nas linhas 33 a 40, verifica-se a existência de uma cadeia lateral ligada a ele, e caso exista, ela é resolvida da mesma maneira como foi feito para as demais.

Finalmente, utilizando os conjuntos de distâncias entre a cadeia principal e as cadeias laterais, e os conjuntos de distância entre cadeias laterais, são verificadas quais combinações de cadeias são compatíveis, através do procedimento $combinaEGravaCadeiasLateraisCompativeis$, sendo estas gravadas como soluções do problema. Feita a gravação das soluções encontradas, o algoritmo procede com o caminhamento na árvore em busca de novas soluções.

Este algoritmo não tira proveito da propriedade de simetria de soluções apresentada na Seção 2.3.1. A versão que utiliza este conceito para reduzir o esforço computacional será apresentada na próxima seção.

Exemplo: Considere novamente a mesma instância usada no exemplo 3.1. Porém, agora, como se está tratando a proteína inteira, deve-se considerar, também, as cadeias laterais ligadas a cadeia principal. Nessa instância, como existem seis átomos na cadeia principal, existem duas cadeias laterais, a primeira delas ligada ao terceiro átomo da cadeia principal e a segunda ao sexto. Optou-se por utilizar, como efeito de exemplo, somente os três primeiros átomos da primeira cadeia lateral e os quatro primeiros átomos da segunda. Os átomos da primeira cadeia lateral foram denominados como 7, 8 e 9 e os da segunda como 10, 11, 12 e 13. As distâncias conhecidas são as seguintes:

- Distâncias essenciais da primeira cadeia lateral (Hipótese I):
  - $d_{1,2} = 1,49760$
  - $d_{1,3} = 2,61604$
  - $d_{1,7} = 3,56410$
  - $d_{2,3} = 1,57365$
  - $d_{2,7} = 2,39042$
  - $d_{2,8} = 1,57676$
  - $d_{3,7} = 1,23157$
  - $d_{3,8} = 2,59183$
  - $d_{3,9} = 3,23682$
  - $d_{7,8} = 3,30144$
  - $d_{7,9} = 3,40510$
  - $d_{8,9} = 1,54408$
- Distâncias adicionais da primeira cadeia lateral (conjunto F_1):
  - $d_{1,8} = 2,59560$
  - $d_{1,9} = 3,85746$
  - $d_{2,9} = 2,57075$
- Distâncias essenciais da segunda cadeia lateral (Hipótese I):

Algoritmo 4 *APC*

```
1: P[1].Q = B[1];
2: P[1].subarvore = ESQUERDA;
3: P[2].Q = B[1].B[2];
4: P[2].subarvore = ESQUERDA;
5: P[3].Q = B[1].B[2].B[3];
6: P[3].subarvore = ESQUERDA;
7: nivel = 3;
8: cria uma tupla f, sendo i = j = n e distância = 0;
9: F ← F ∪ {f};
10: para cada cadeia lateral l, pertencente a L faça
11:    resolvePOAPDGDM(l);
12: fim para
13: k ← 0;
14: enquanto nivel > 2 faça
15:    enquanto nivel < F[k].j faça
16:       se nivel mod 3 = 0 então
17:          L[ nivel/3 ].resolvePDGDMCadeiaLateral();
18:          se filtraSolsCompativeisCadeiaPrincipal(L[ nivel/3 ]) = 0 então
19:             P ← Poda(P, k);
20:          fim se
21:       fim se
22:       se nivel <= 2 então
23:          retorna
24:       senão
25:          P ← Explora(P, k);
26:       fim se
27:    fim enquanto
28:    se nivel > 2 então
29:       se em P[nivel] a desigualdade 3.1 não é satisfeita então
30:          P ← Poda(P, k);
31:       senão
32:          k ← k + 1;
33:          se nivel = n então
34:             se nivel mod 3 = 0 então
35:                L[ nivel/3 ].resolvePDGDMCadeiaLateral();
36:                filtraSolsCompativeisCadeiaPrincipal(L[nivel/3])
37:             fim se
38:             combinaEGravaCadeiasLateraisCompativeis();
39:             P ← PodaComSimetria(P, k);
40:          fim se
41:       fim se
42:    fim se
43: fim enquanto
```

- $d_{4,5} = 1,49261$
- $d_{4,6} = 2,60190$

Algoritmo 5 *Explora*(P, k)

```
1: se P[nivel].subarvore = ESQUERDA então
2:    nivel = nivel + 1;
3:    cria v(representação do nó esquerdo da árvore);
4:    v.Q = B[nivel] * P[nivel − 1].Q;
5:    v.subarvore ← ESQUERDA;
6:    P[nivel-1].subarvore ← DIREITA;
7:    se CompativelCadeiasLateraisExploradas( v, nivel ) então
8:       L[ ⌊nivel/3⌋ ].FixaAtomoCadLateral( nivel - ⌊nivel/3⌋ * 3, v.Q );
9:    senão
10:      P ← Poda(P, k);
11:   fim se
12: senão
13:   nivel = nivel + 1;
14:   cria v'(representação do nó direito da árvore);
15:   v'.Q = B'[nivel] * P[nivel − 1].Q;
16:   v'.subarvore ← DIREITA;
17:   P[nivel-1].subarvore ← PODA;
18:   se CompativelCadeiasLateraisExploradas( v', nivel ) então
19:      L[ ⌊nivel/3⌋ ].FixaAtomoCadLateral( nivel - ⌊nivel/3⌋ * 3, v'.Q );
20:   senão
21:      P ← Poda(P, k);
22:   fim se
23: fim se
```

Algoritmo 6 *Poda*(P, k)

```
1: repita
2:    nivel = nivel - 1;
3:    se nivel < 3 ou k <= 0 então
4:       retorna;
5:    fim se
6:    enquanto nivel < F[k − 1].j faça
7:       k ← k − 1;
8:    fim enquanto
9: até P[nivel].subarvore = PODA
```

- $d_{4,10} = 3,00798$
- $d_{5,6} = 1,55895$
- $d_{5,10} = 2,43389$
- $d_{5,11} = 1,54375$
- $d_{6,10} = 1,23443$
- $d_{6,11} = 2,58252$
- $d_{6,12} = 3,80612$
- $d_{10,11} = 3,17769$

  - $d_{10,12} = 4,27117$
  - $d_{10,13} = 3,96014$
  - $d_{11,12} = 1,52948$
  - $d_{11,13} = 2,40137$
  - $d_{12,13} = 1,52614$

- Distâncias adicionais da segunda cadeia lateral (conjunto F_2):
  - $d_{4,11} = 2,35146$
  - $d_{4,12} = 2,34200$
  - $d_{4,13} = 1,45873$
  - $d_{5,12} = 2,44748$
  - $d_{5,13} = 2,47310$
  - $d_{6,13} = 3,71917$

- Distâncias entre átomos da cadeia principal e da primeira cadeia lateral (conjunto PL_1):
  - $d_{1,7} = 3,56410$
  - $d_{1,8} = 2,59560$
  - $d_{1,9} = 3,85746$
  - $d_{2,7} = 2,39042$
  - $d_{2,8} = 1,57676$
  - $d_{2,9} = 2,57075$
  - $d_{3,7} = 1,23157$
  - $d_{3,8} = 2,59183$
  - $d_{3,9} = 3,23682$
  - $d_{4,7} = 2,24888$
  - $d_{4,8} = 3,36545$
  - $d_{4,9} = 4,22646$
  - $d_{5,7} = 2,82839$
  - $d_{5,8} = 4,69508$
  - $d_{6,7} = 2,94006$

- Distâncias entre átomos da cadeia principal e da segunda cadeia lateral (conjunto PL_2):

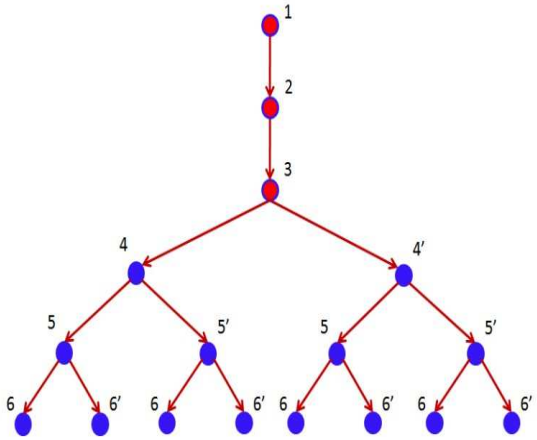
  - $d_{1,15} = 4,57181$
  - $d_{1,16} = 3,05238$
  - $d_{2,13} = 4,70424$
  - $d_{2,14} = 4,90022$
  - $d_{2,15} = 4,45705$
  - $d_{2,16} = 3,00569$
  - $d_{3,13} = 3,36655$
  - $d_{3,14} = 3,67772$
  - $d_{3,15} = 3,64927$
  - $d_{3,16} = 2,46520$
  - $d_{4,13} = 3,00798$
  - $d_{4,14} = 2,35146$
  - $d_{4,15} = 2,34200$
  - $d_{4,16} = 1,45873$
  - $d_{5,13} = 2,43389$
  - $d_{5,14} = 1,54375$
  - $d_{5,15} = 2,44748$
  - $d_{5,16} = 2,47310$
  - $d_{6,13} = 1,23443$
  - $d_{6,14} = 2,58252$
  - $d_{6,15} = 3,80612$
  - $d_{6,16} = 3,71917$

- Distâncias entre átomos da primeira e da segunda cadeia lateral (conjunto $LL_{1,2}$):
  - $d_{7,13} = 3,20765$
  - $d_{7,14} = 4,25583$
  - $d_{7,15} = 4,57284$
  - $d_{7,16} = 3,58559$
  - $d_{8,15} = 4,84776$
  - $d_{8,16} = 3,58213$
  - $d_{9,16} = 4,81402$

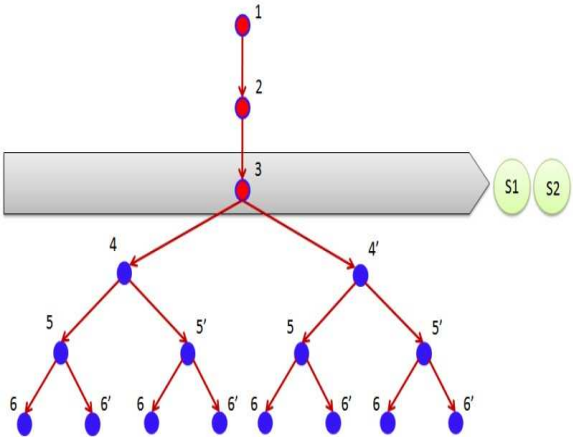
As Figuras 3.7, 3.8 e 3.9 mostram o comportamento do algoritmo, sendo que os nós em vermelho representam os nós que estão em memória no momento. Inicialmente, a árvore é inicializada com os três primeiros átomos fixos (Figura 3.7(a)). Em seguida (Figura 3.7(b)), é resolvido o PDGDM para a primeira cadeia lateral, utilizando como átomos fixos, os átomos 1, 2 e 3 da cadeia principal. Nesse mesmo momento, as soluções encontradas para a primeira cadeia lateral são filtradas, utilizando as informações do conjunto PL_1, ou seja, elimina-se as soluções incompatíveis com os átomos da cadeia principal que já estão fixos. No caso desta cadeia, todas as soluções encontradas são compatíveis, não sendo necessária a remoção de nenhuma delas. Em (Figura 3.7(c)), procede-se com o caminhamento em profundidade, que consegue chegar até o último nível sem que nenhuma poda seja feita. Então, em (Figura 3.7(d)), é resolvido o PDGDM para a segunda cadeia lateral, utilizando os átomos 4, 5 e 6 como átomos fixos. Em seguida (Figura 3.7(e)), as soluções incompatíveis dessa cadeia lateral são removidas, através de informações do conjunto PL_2 (neste caso, é necessária a remoção da solução $S2$). Então, em (Figura 3.7(f)), analisam-se as compatibilidades existentes entre os átomos da cadeia principal e as soluções das cadeias laterais, através dos conjuntos PL_1 e PL_2, e as compatibilidades entre as soluções das cadeias laterais, utilizando o conjunto $LL_{1,2}$. Por fim, as combinações compatíves da cadeia principal com as soluções das cadeias laterais são gravadas. Neste caso, uma única combinação é compatível, ou seja, existe uma única solução para o problema, utilzando a cadeia principal definida pelo ramo que está sendo explorado, sendo esta composta pela cadeia principal, a solução $S2$ da primeira cadeia lateral e a solução $S1$ da segunda cadeia lateral. No próximo passo (Figura 3.8(a)), continua-se com o caminhamento em profundidade que é interrompido (poda) pela restrição de distância, $d_{2,6}$. Logo após (Figura 3.8(b)), o algoritmo continua o seu percurso em outro ramo, e neste também é podado, dessa vez pela distância $d_{1,5}$. A partir daí, o algoritmo segue seu curso normal de execução, apresentando um comportamento simétrico, ao realizado até agora, para o outro "lado" da árvore (Figura 3.8(c) a Figura 3.9(b)).

3.2.2 Algoritmo para Proteína Completa com Simetria - APCS

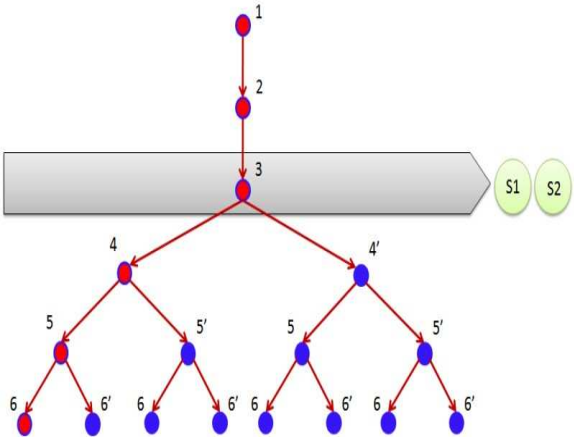
Como explicado anteriormente, existe uma propriedade de simetria de soluções para o PDGDM, onde dada uma solução S, pode-se definir uma solução S'. O APCS é uma extensão do APC, que tira proveito deste conceito.



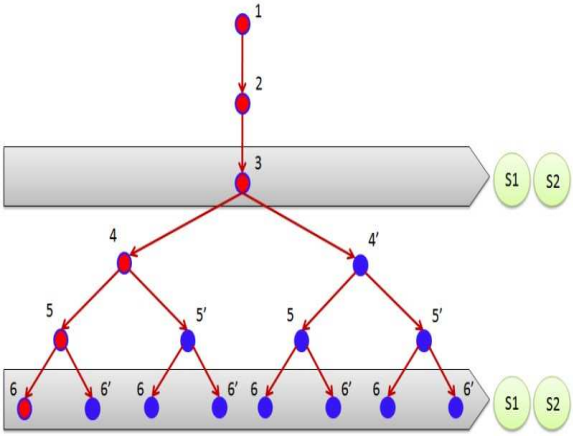
(a) *3 átomos iniciais fixos*



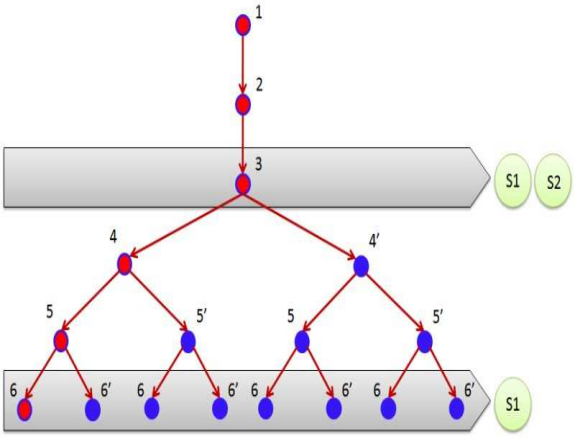
(b) *resolução do PDGDM para a primeira cadeia lateral*



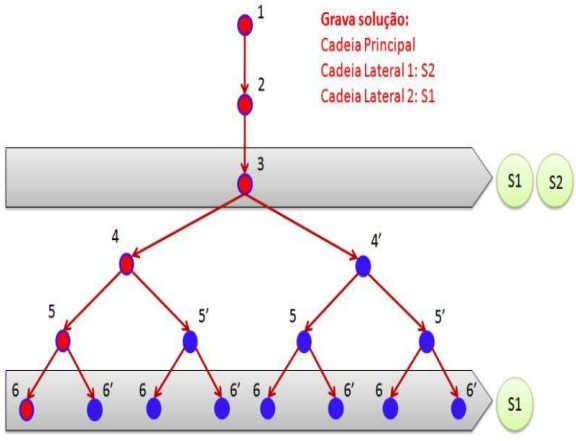
(c) *caminhamento em profundidade*



(d) *resolução do PDGDM para a segunda cadeia lateral*



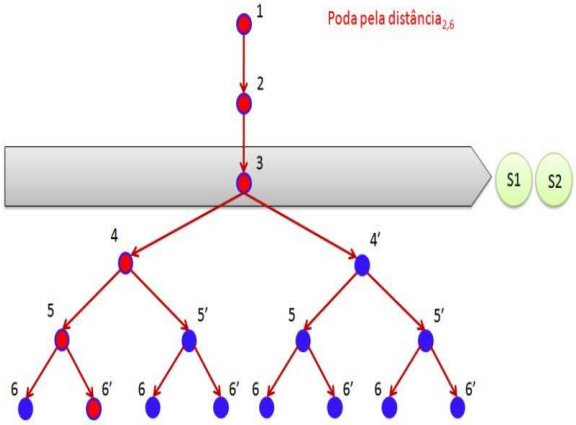
(e) *filtragem de soluções compatíveis da segunda cadeia lateral com a cadeia principal*



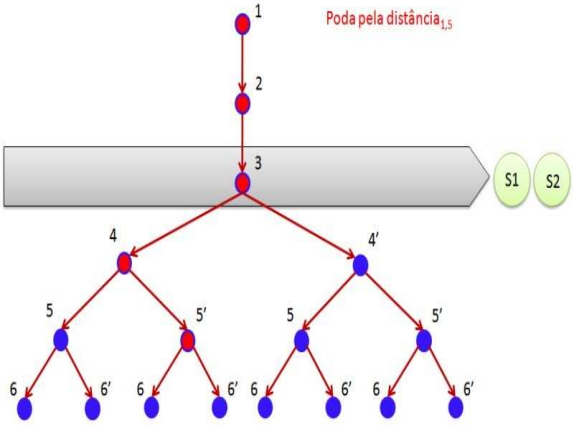
(f) *gravação das combinações da cadeia principal com as cadeias laterais compatíveis*

Figura 3.7: *Exemplo do Algoritmo APC.*

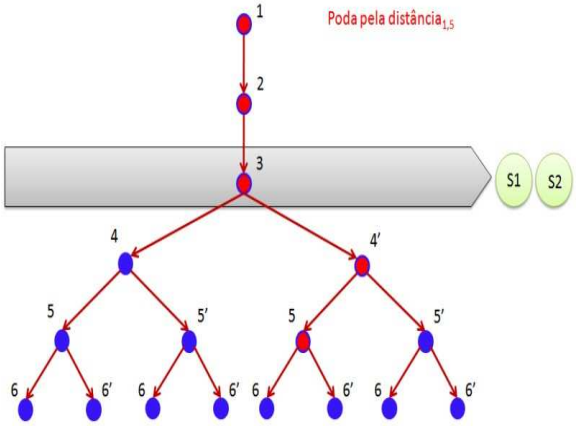
Como no cálculo da estrutura completa de uma proteína o APC precisa resolver várias instâncias do PDGDM, a propriedade de simetria pode ser utilizada em cada uma



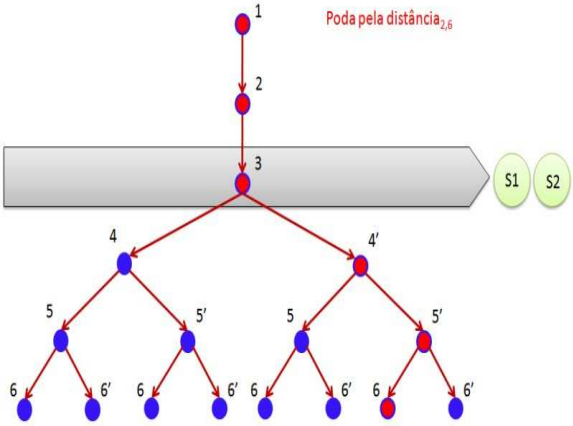
(a) *caminhamento em profundidade e poda por restrição de distância*



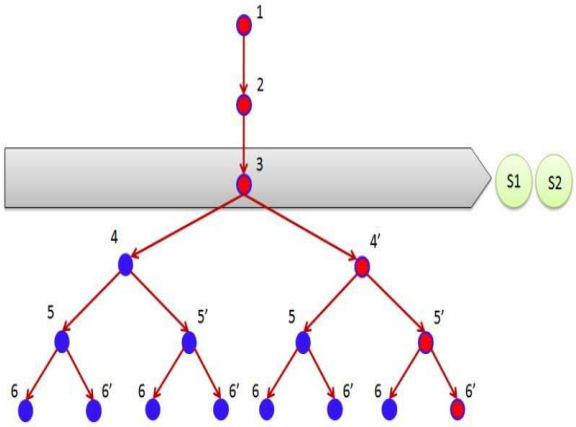
(b) *caminhamento em profundidade e poda por restrição de distância*



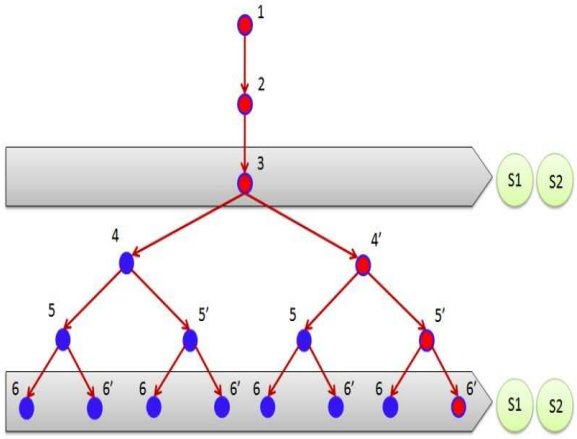
(c) *caminhamento em profundidade e poda por restrição de distância*



(d) *caminhamento em profundidade e poda por restrição de distância*



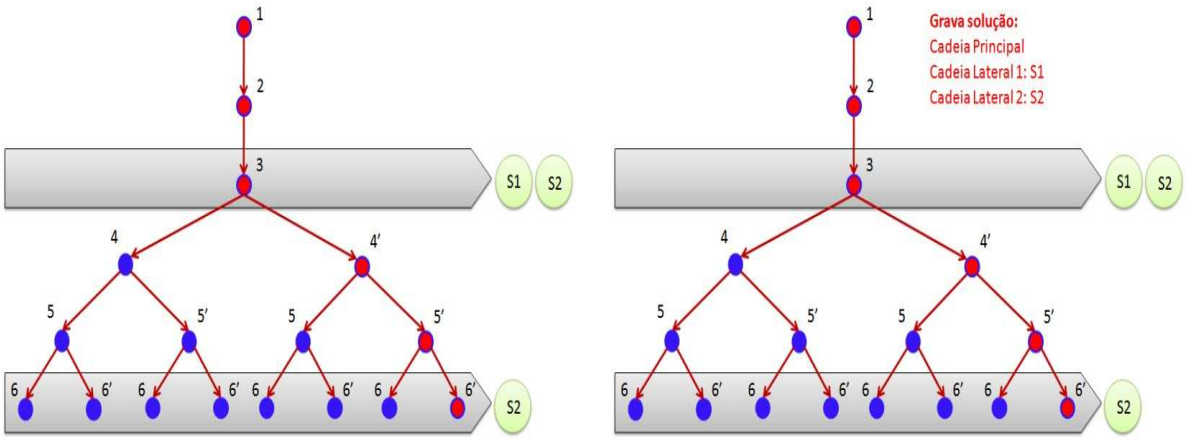
(e) *caminhamento em profundidade*



(f) *resolução do PDGDM para a segunda cadeia lateral*

Figura 3.8: *Exemplo do Algoritmo APC (continuação I).*

delas, para melhorar a eficiência do algoritmo. Além disso, como descrito na Seção 2.3.1, a simetria também existe para soluções de proteínas completas, sendo esta mais uma



(a) *filtragem de soluções compativeis da segunda cadeia lateral com a cadeia principal*

(b) *gravação das combinações da cadeia principal com as cadeias laterais compatíveis*

Figura 3.9: *Exemplo do Algoritmo APC (continuação II).*

propriedade a ser utilizada pelo APCS.

São necessárias pequenas alterações no APC para que a simetria seja utilizada. O Algoritmo 7 apresenta o pseudo-código do APCS, sendo que as partes que se diferenciam do APC estão em negrito.

Como pode ser visto nas linhas 7 a 9, o APCS fixa também o quarto nível da árvore, ficando por conta da simetria determinar as soluções do outro lado (quarto nível como filho direito do terceiro). De maneira geral, as outras alterações servem para que a execução seja finalizada, quando o nó da direita do quarto nível da árvore é explorado, ao invés do terceiro. Além disso, toda vez que é necessário explorar uma cadeia lateral, é utilizado o procedimento $resolvePDGDMCadeiaLateralComSimetria()$, que necessita determinar somente metade das soluções existentes, sendo a outra metade facilmente definida. Isso faz com que o seu esforço computacional seja, aproximandamente, a metade do necessário pelo procedimento $resolvePDGDMCadeiaLateral()$ utilizado no APC.

Como exemplo, se o APCS fosse utilizado para resolver a instância apresentada no exemplo do APC (subseção 3.2.1, p. 27), ele navegaria na árvore da Figura 3.7 da mesma maneira que o APC, porém, somente até o passo da Figura 3.8(b). Logo após, a solução encontrada na Figura 3.9(b) seria calculada pela propriedade de simetria e a execução finalizada.

Como pode-se notar, o APCS é muito mais eficiente que o APC. No Capítulo 4, são mostrados testes que comparam as duas abordagens.

Algoritmo 7 *APCS*

```
1: P[1].Q = B[1];
2: P[1].subarvore = ESQUERDA;
3: P[2].Q = B[1].B[2];
4: P[2].subarvore = ESQUERDA;
5: P[3].Q = B[1].B[2].B[3];
6: P[3].subarvore = ESQUERDA;
7: P[4].Q = B[1].B[2].B[3].B[4];
8: P[4].subarvore = ESQUERDA;
9: nivel = 4;
10: cria uma tupla f, sendo i = j = n e distância = 0;
11: F ← F ∪ {f};
12: para cada cadeia lateral l, pertencente a L faça
13:    resolvePOAPDGDM(l);
14: fim para
15: L[0].resolvePDGDMCadeiaLateralComSimetria();
16: se filtraSolsCompativeisCadeiaPrincipal(L[ nivel/3 ]) = 0 então
17:    retorna;
18: fim se
19: k ← 0;
20: enquanto nivel > 3 faça
21:    enquanto nivel < F[k].j faça
22:       se nivel mod 3 = 0 então
23:          L[ nivel/3 ].resolvePDGDMCadeiaLateralComSimetria();
24:          se filtraSolsCompativeisCadeiaPrincipal(L[ nivel/3 ]) = 0 então
25:             P ← PodaComSimetria(P, k);
26:          fim se
27:       fim se
28:       se nivel <= 3 então
29:          retorna
30:       senão
31:          P ← ExploraComSimetria(P, k);
32:       fim se
33:    fim enquanto
34:    se nivel > 3 então
35:       se em P[nivel] a desigualdade 3.1 não é satisfeita então
36:          P ← PodaComSimetria(P, k);
37:       senão
38:          k ← k + 1;
39:          se nivel = n então
40:             se nivel mod 3 = 0 então
41:                L[ nivel/3 ].resolvePDGDMCadeiaLateralComSimetria();
42:                filtraSolsCompativeisCadeiaPrincipal(L[nivel/3])
43:             fim se
44:             combinaEGravaCadeiasLateraisCompativeis();
45:             P ← PodaComSimetria(P, k);
46:          fim se
47:       fim se
48:    fim se
49: fim enquanto
```

Algoritmo 8 *ExploraComSimetria*(P, k)

```
1: se P[nivel].subarvore = ESQUERDA então
2:    nivel = nivel + 1;
3:    cria v(representação do nó esquerdo da árvore);
4:    v.Q = B[nivel] * P[nivel − 1].Q;
5:    v.subarvore ← ESQUERDA;
6:    P[nivel-1].subarvore ← DIREITA;
7:    se CompativelCadeiasLateraisExploradas( v, nivel ) então
8:       L[ ⌊nivel/3⌋ ].FixaAtomoCadLateral( nivel - ⌊nivel/3⌋ * 3, v.Q );
9:    senão
10:      P ← PodaComSimetria(P, k);
11:   fim se
12: senão
13:   nivel = nivel + 1;
14:   cria v'(representação do nó direito da árvore);
15:   v'.Q = B'[nivel] * P[nivel − 1].Q;
16:   v'.subarvore ← DIREITA;
17:   P[nivel-1].subarvore ← PODA;
18:   se CompativelCadeiasLateraisExploradas( v', nivel ) então
19:      L[ ⌊nivel/3⌋ ].FixaAtomoCadLateral( nivel - ⌊nivel/3⌋ * 3, v'.Q );
20:   senão
21:      P ← PodaComSimetria(P, k);
22:   fim se
23: fim se
```

Algoritmo 9 *PodaComSimetria*(P, k)

```
1: repita
2:    nivel = nivel - 1;
3:    se nivel < 4 ou k <= 0 então
4:       retorna;
5:    fim se
6:    enquanto nivel < F[k − 1].j faça
7:       k ← k − 1;
8:    fim enquanto
9: até P[nivel].subarvore = PODA
```

Capítulo 4

Resultados Computacionais

Este capítulo apresenta os resultados computacionais obtidos pelos algoritmos propostos nesta dissertação. Inicialmente, são apresentadas as instâncias utilizadas para a realização dos testes e as métricas utilizadas para mensurar a qualidade das soluções. Em seguida, compara-se o desempenho do ACPN, com o desempenho dos outros algoritmos que, igualmente ao ACPN, resolvem o problema somente para cadeia principal. Neste primeiro experimento foram utilizadas instâncias artificiais. Posteriormente, são mostrados os resultados obtidos utilizando os algoritmos que resolvem o problema para a cadeia completa. Para a realização destes testes foram utilizadas instâncias reais, extraídas do PDB [1].

Os algoritmos propostos foram implementados em C++, utilizando-se a *Standard Template Library* e compilados com o Visual C++ 2005. Foi utilizado um notebook Toshiba, com processador Intel Core 2 Duo de 1.6 GHz e 2 Gbytes de memória RAM, utilizando o sistema operacional Windows XP, com Service Pack 2.

4.1 Instâncias

Para a realização dos experimentos, foram utilizadas instâncias artificiais e reais. As instâncias artificiais simulam apenas a cadeia principal das proteínas. As instâncias reais que são geradas a partir do *Protein Data Bank*, PDB [1], simulam a proteína inteira.

4.1.1 Instâncias Artificiais

As instâncias artificiais que foram utilizadas são chamadas de *Instâncias Lavor* [19]. Essas instâncias são baseadas em um modelo proposto por [31] e simulam apenas a cadeia principal das proteínas. Neste modelo, as distâncias entre os átomos da cadeia principal e ângulos de ligação são fixos e um conjunto de ângulos de torção é gerado aleatoriamente. Cada *instância Lavor* tem o rótulo `lavor`n-m, onde n é o número de átomos envolvidos e m é o identificador da instância.

Foram utilizadas 13 diferentes *instâncias Lavor* consideradas pequenas, com n variando de 10 até 70, e 10 diferentes *instâncias Lavor* consideradas grandes, com n variando de 100 até 1000. Na Tabela 4.1, as instâncias são descritas em detalhes: a primeira coluna apresenta uma numeração para a instância, a segunda coluna apresenta o nome da instância, a terceira coluna indica o número n de átomos e a quarta coluna indica o tamanho do conjunto F (quantidade de distâncias adicionais).

Instâncias Lavor			
Instância	Nome	n	$\|F\|$
1	`lavor10_0`	10	9
2	`lavor15_0`	15	18
3	`lavor20_0`	20	51
4	`lavor25_0`	25	62
5	`lavor30_0`	30	85
6	`lavor35_0`	35	72
7	`lavor40_0`	40	181
8	`lavor45_0`	45	110
9	`lavor50_0`	50	127
10	`lavor55_0`	55	392
11	`lavor60_0`	60	203
12	`lavor65_0`	65	78
13	`lavor70_0`	70	227
14	`lavor100_2`	100	311
15	`lavor200_2`	200	1250
16	`lavor300_2`	300	1611
17	`lavor400_2`	400	1406
18	`lavor500_2`	500	3083
19	`lavor600_2`	600	3679
20	`lavor700_2`	700	2094
21	`lavor800_2`	800	4456
22	`lavor900_2`	900	5271
23	`lavor1000_2`	1000	5235

Tabela 4.1: Instâncias Lavor

4.1.2 Instâncias Reais

As instâncias reais utilizadas foram geradas através de proteínas obtidas do PDB e podem ser acessadas em:

`http://www.rcsb.org/pdb/`

O PDB é um conhecido banco de dados, onde podem ser encontradas milhares de estruturas protéicas já determinadas. A partir dessas estruturas podem ser obtidas todas as distâncias entre os átomos pertencentes a molécula, uma vez que a posição de cada um deles é disponibilizada. Para que possam ser geradas instâncias a partir dessas estruturas, são extraídas apenas as distâncias com valores menores ou iguais a um parâmetro chamado *corte*. Segundo [33], para simular os dados obtidos a partir de experimentos de RMN, o *corte* deve ser, no máximo, igual a 6Å. Nessas estruturas encontradas no PDB, além da posição de cada átomo, estão disponíveis outras informações como o tipo de cada átomo e a cadeia e o aminoácido ao qual ele pertence. Através dos dados obtidos do PDB, também é possível saber em que átomo da cadeia principal, cada cadeia lateral está ligada.

4.2 Métricas de Qualidade das Soluções

Para que a qualidade das soluções possa ser mensurada, os algoritmos implementados utilizam duas conhecidas métricas da literatura: LDE e RMSD.

4.2.1 *Largest Distance Error* - LDE

O LDE é empregado como uma medida de precisão da solução. Basicamente, ele compara as distâncias entre átomos da estrutura determinada com as distâncias conhecidas previamente. O LDE é definido como:

$$\text{LDE} = \frac{1}{|E|} \sum_{(i,j)\in E} \frac{|\ ||x_i - x_j|| - d_{ij}|}{d_{ij}}, \tag{4.1}$$

onde E é o conjunto de todas as distâncias conhecidas. Quanto menor o LDE, melhor é a qualidade da solução.

4.2.2 *Root-Mean-Square Deviation* - RMSD

Para comparar as estruturas encontras pelos algoritmos com as estruturas existentes no PDB, foi utilizado o cálculo do RMSD, que de maneira geral, mede o grau de semelhança entre duas estruturas [10]. O RMSD de duas estruturas X e Y pode ser definido da seguinte forma:

$$RMSD(X,Y) \quad = \quad \min_{Q} ||X - YQ||/\sqrt{n}, \tag{4.2}$$

onde Q é a matriz utilizada para rotacionar Y de modo que fique o mais semelhante possível de X.

O valor RMSD é expresso em unidades de comprimento. A unidade mais comumente utilizada em biologia estrutural é Ångström (Å).

4.3 Testes com algoritmos para a cadeia principal

Neste experimento, é comparado o desempenho dos algoritmos BP de [24], Algoritmo I e Algoritmo II de [11] e o ACPN proposto nesta dissertação. O algoritmo `BP` foi o primeiro algoritmo criado para o PDGDM e funciona de forma similar a dos algoritmos apresentados nesta dissertação. Ele também utiliza o conceito de árvore e a explora em profundidade, da esquerda para direita, porém, o faz de forma recursiva. Uma vantagem dessa implementação recursiva é o tempo computacional, pois as operações ficam em memória e o reprocessamento dos cálculos não se faz necessário. Contudo, para as instâncias maiores, a quantidade de informações armazenadas na pilha de recursividade se torna muito grande, podendo causar "estouro" de memória. Dessa forma, o algoritmo se torna muito rápido para as instâncias menores e impraticável para as instâncias maiores.

O Algoritmo I e o APCN, como mencionado na seção 3.1, podem, igualmente, causar estouro de memória, devido a estratégia de explorar a árvore em nível e tendo em vista que, em algum nível, a quantidade de nós pode ser muito grande.

Por sua vez, o Algoritmo II resolve o problema de estouro de memória, através de uma estratégia eficiente, usando uma estrutura auxiliar de pilha para explorar a árvore em profundidade, da esquerda para direita. Deste modo, não há problemas com gerenciamento de memória para instâncias maiores.

Para a realização destes testes, o valor do parâmetro ε, que representa a tolerância de erro, foi 1×10^{-3}Å, o mesmo valor utilizado para o `BP` em [24] e para os algoritmos

Algoritmo I e Algoritmo II em [11]. Além disso, o tempo limite de execução dos algortimos foi fixado em duas horas. A Tabela 4.2 apresenta os resultados referentes aos quatro algoritmos: `BP` executado até encontrar todas as soluções (`BP-All`), Algoritmo II executado até encontrar todas soluções (`AII-All`), Algoritmo I (`AI`) e ACPN. Na tabela, são apresentados o tempo de CPU (em segundos), assim como o número de soluções encontradas por cada algoritmo (#Sol). Para as instâncias em que os algoritmos não foram capazes de resolver o problema, os valores de CPU e #Sol não são exibidos.

I	BP-ALL		AII-ALL		Al		ACPN	
-	**CPU**	**#Sol**	**CPU**	**#Sol**	**CPU**	**#Sol**	**CPU**	**#Sol**
1	0.00	4	0.00	4	0.00	4	0.00	4
2	0.00	16	0.02	16	0.05	16	0.02	16
3	0.00	8	0.02	8	0.06	8	0.02	8
4	0.00	8	0.02	8	0.03	8	0.01	8
5	0.00	2	0.00	2	0.01	2	0.00	2
6	0.01	256	0.12	256	0.51	256	0.15	256
7	0.01	128	0.16	128	0.72	128	0.17	128
8	0.00	64	0.11	64	0.49	64	0.16	64
9	0.46	256	0.10	256	0.44	256	0.15	256
10	0.00	8	0.03	8	0.12	8	0.05	8
11	0.03	1024	0.09	1024	0.43	1024	0.28	1024
12	–	–	27.51	**32768**	–	–	–	–
13	–	–	45.49	**32768**	–	–	–	–
14	–	–	89.05	**2**	–	–	–	–
15	–	–	0.93	**32**	–	–	–	–
16	–	–	0.83	**4**	–	–	–	–
17	–	–	7200	**9.82E+06**	–	–	–	–
18	–	–	7200	**484736**	–	–	–	–
19	–	–	7200	**1.15E+08**	–	–	–	–
20	–	–	7200	**927461**	–	–	–	–
21	–	–	7200	**156479**	–	–	–	–
22	–	–	7200	**4.71E+07**	–	–	–	–
23	–	–	7200	**263**	–	–	–	–

Tabela 4.2: Experimentos comparativos dos algoritmos para a cadeia principal

Pode-se observar que apenas o algoritmo AII-All conseguiu resolver todas as instâncias. Os algoritmos `BP-All`, AI e ACPN só foram capazes de resolver as instâncias $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ que têm até 60 átomos, como mostram as Tabelas 4.1 e 4.2. Para instâncias maiores que estas, estes algoritmos tiveram problemas de memória.

Dentre os quatro algoritmos, o que apresentou o melhor resultado foi o Algoritmo II. Contudo, acredita-se que para uma implementação paralela, seja melhor se utilizar de uma abordagem que explore a árvore por nível. Com uma abordagem deste tipo, a divisão de trabalho seria muito mais intuitiva, podendo-se distribuir, por exemplo, ramos da árvore a cada processador. Com este objetivo, devem ser comparados os dois algoritmos que exploram a árvore por nível: Algoritmo 1 e ACPN. Veja que o ACPN gasta, em média,

somente 35% do tempo utilizado pelo Algoritmo I para resolver as instâncias da Tabela 4.2. Assim, caso uma implementação paralela do PDGDM venha a ser feita, uma boa estratégia poderia ser a de estender o ACPN.

4.4 Testes com algoritmos para proteína completa

Existem, na literatura, diversos métodos para determinar estruturas tridimensionais de proteínas. Contudo, os métodos existentes, geralmente, são capazes de produzir uma única solução e não existe informação sobre o tempo computacional necessário para alcançá-la. Outro problema é que alguns deles produzem soluções onde nem todos os átomos da proteína são determinados. Além disso, são utilizadas diferentes métricas para mensurar a qualidade das soluções e são utilizados diversos valores para o *corte* na geração de instâncias a partir do PDB. Todos estes fatores dificultam uma comparação justa entre os métodos. Assim sendo, para os testes desta subseção, procurou-se utilizar as instâncias mais exploradas na literatura, as métricas mais utilizadas (RMSD e LDE) e o *corte* fixo em 6Å, que é o valor máximo permitido para simular dados obtidos via RMN [33]. As soluções produzidas foram comparadas com a melhor solução encontrada na literatura, independentemente dos parâmetros utilizados para alcançá-la. Para a realização deste experimento, o valor de ϵ foi fixado em 1×10^{-5}Å para todas as instâncias.

Na Tabela 4.3, são apresentados os resultados dos testes utilizando o APC, o APCS e os melhores algoritmos presentes na literatura. No iníco da tabela são apresentados os resultados encontrados na literatura, sendo que, nas colunas RMSD e CPU, é exibido o melhor RMSD encontrado para a instância, dentre todos os métodos conhecidos e, quando disponível, o tempo gasto para alcançá-lo, respectivamente. Em seguida, são apresentados os resultados encontrados pelo APC e APCS, #Sol é a quantidade de soluções encontradas pelo método, `PDB` indica qual das soluções encontradas para cada instância tem o maior grau de semelhança com a proteína obtida no PDB, RMSD e LDE mostram o melhor valor encontrado para estas métricas, respectivamente e CPU apresenta o tempo computacional, em segundos, utilizado pelo método para encontrar todas as soluções.

Os resultados da literatura foram obtidos em [9, 35, 12, 5, 36], sendo que estes três últimos se destacaram. Os resultados de [12] eram os melhores da literatura para as instâncias de número 15, 18, 19, 20 e 21, e foram atingidos utilizando um *corte* de 6Å, mesmo valor utilizado pelos algoritmos propostos. Os resultados de [36] eram os melhores para as instâncias 7, 16, 17 e 25, sendo que para a instância 16 foi utilizado um *corte*

Instâncias		Literatura		APC					APCS				
I	**Nome**	**RMSD**	**CPU**	**#Sol**	**PDB**	**RMSD**	**LDE**	**CPU**	**#Sol**	**PDB**	**RMSD**	**LDE**	**CPU**
1	1A29	3,80E-05 [5]	-	4	1	**1,21E-15**	7,74E-17	13,094	4	1	**1,21E-15**	7,74E-17	**7,6870**
2	1ABA			2	1	**2,20E-15**	7,38E-17	7,6560	2	1	**2,20E-15**	7,38E-17	**1,8120**
3	1ACZ			8	2	**9,73E-16**	4,71E-17	46,765	8	2	**9,73E-16**	4,71E-17	**17,078**
4	1AHL			2	1	**3,84E-15**	3,97E-17	7,3900	2	1	**3,84E-15**	3,97E-17	**4,0000**
5	1AQR			2	2	**2,44E-17**	5,25E-17	4,3120	2	2	**2,44E-17**	5,25E-17	**1,9210**
6	1B4C			32	29	**4,70E-16**	7,77E-17	30,937	32	31	**4,70E-16**	7,77E-17	**15,515**
7	1BKR	2,20E-12 [36]	-	4	3	**3,46E-17**	8,00E-17	8,4370	4	3	**3,46E-17**	8,00E-17	**2,2180**
8	1BQV			8	2	**4,40E-18**	4,51E-17	24,828	8	4	**4,40E-18**	4,51E-17	**13,015**
9	1BQX			2	1	**3,51E-19**	3,09E-17	10,312	2	1	**3,51E-19**	3,09E-17	**4,7190**
10	1BRO			4	1	**3,28E-16**	1,07E-16	23,718	4	1	**3,28E-16**	1,07E-16	**7,7650**
11	1BRV			2	2	**2,16E-17**	1,95E-17	1,6400	2	2	**2,16E-17**	1,95E-17	**0,5780**
12	1BRZ			2	1	**5,24E-16**	5,29E-17	10,187	2	1	**5,24E-16**	5,29E-17	**6,1710**
13	1BSA			2	1	**6,03E-16**	8,51E-17	7,7030	2	1	**6,03E-16**	8,51E-17	**2,5000**
14	1FKG	2,10E-02 [5]	-	8	8	**1,77E-16**	9,50E-17	14,734	8	7	**1,77E-16**	9,50E-17	**9,4530**
15	1HOE	2,21E-15 [12]	20	4	1	**1,29E-15**	6,93E-17	5,5620	4	2	**1,29E-15**	6,93E-17	**1,3590**
16	1HYP	2,90E-09 [36]	-	4	2	**1,99E-16**	8,44E-17	6,9530	4	1	**1,99E-16**	8,44E-17	**2,5150**
17	1IOO	6,60E-12 [36]	-	64	13	**3,67E-15**	1,00E-16	33,234	64	24	**3,67E-15**	1,00E-16	**13,421**
18	1LFB	3,03E-15 [12]	99	4	2	**7,18E-16**	5,87E-17	6,0310	4	2	**7,18E-16**	5,87E-17	**1,5310**
19	1PHT	2,86E-14 [12]	111	8	3	**1,03E-16**	6,41E-17	10,484	8	2	**1,03E-16**	6,41E-17	**4,0930**
20	1POA	3,26E-14 [12]	92	32	12	**5,47E-15**	7,50E-17	15,187	32	9	**5,47E-15**	7,50E-17	**5,4210**
21	1PTQ	**2,65E-15** [12]	11	8	3	3,91E-15	6,48E-17	3,9840	8	2	3,91E-15	6,48E-17	**1,1400**
22	1Q80			2	1	**1,65E-15**	7,02E-17	28,390	2	1	**1,65E-15**	7,02E-17	**15,421**
23	1RGS	7,40E-08 [5]	-	4	1	**1,10E-15**	1,58E-16	21,375	4	1	**1,10E-15**	1,58E-16	**5,3590**
24	1RTB	8,80E-06 [5]	-	2	1	**1,50E-15**	1,22E-16	13,671	2	1	**1,50E-15**	1,22E-16	**8,6870**
25	1WRI	5,60E-13 [36]	-	2	1	**9,16E-17**	7,68E-17	6,6560	2	1	**9,16E-17**	7,68E-17	**1,6250**
26	4MBA			4	2	**5,42E-15**	1,63E-16	11,984	4	2	**5,42E-15**	1,63E-16	**2,9210**

Tabela 4.3: Experimentos comparativos dos algoritmos para a proteina completa

de 5Å e para as demais 8,5Å. Por sua vez, os resultados de [5] eram os melhores para as instâncias de número 1, 14, 23 e 24, sendo que o corte utilizado foi de 5Å.

Pode-se observar que APC e APCS tiveram um desempenho muito bom em relação aos outros algoritmos da literatura. Ambos encontraram soluções com erro muito baixo, RMSD na ordem de 1×10^{-15} a 1×10^{-17} e LDE de 1×10^{-16} a 1×10^{-17}, sendo que o tempo computacional também foi bem pequeno, em média de 14,4s, por instância para o APC, e 5,8s para o APCS. A única instância onde o resultado da literatura foi um pouco melhor com relação ao RMSD foi a de número 21. Outro ponto a ser notado é a eficiência do APCS em relação ao APC. Através do uso da propriedade de simetria, ele necessita, em média, de apenas 40% do tempo utilizado pelo APC.

Apesar do bom desempenho dos algoritmos propostos, para as instâncias 1AX8, 1BPM, 1F39 e 1HMV o corte de 6Å não foi suficiente para que estas satisfizessem a Hipótese I do PDGDM. Alguns trabalhos na literatura, como [9, 35, 36], utilizam cortes de 7, 8 e até 16Å, porém estes valores não são adequados para simular o problema real. Uma proposta de como resolver estas instâncias utilizando o corte de 6Å será exposta como um trabalho futuro no Capítulo 5.

Capítulo 5

Conclusões e Trabalhos Futuros

Esta dissertação abordou o problema de derteminação de estruturas protéicas através do PDGDM. Este trabalho pode ser considerado como uma extensão de trabalhos anteriores [11, 20], que utilizam o PDGDM para determinar somente a estrutura da cadeia principal das proteínas.

Foram propostos três algoritmos: o ACPN, para a cadeia principal, e o APC e APCS, para a proteína completa. Foi definido o Problema de Ordenação de Átomos do PDGDM e desenvolvido um modelo matemático para resolvê-lo.

O ACPN, assim como os algoritmos BP de [24] e Algoritmo I de [11], se mostrou incapaz de resolver instâncias grandes por apresentar problemas de gerenciamento de memória. Porém, acredita-se que ele seja o algoritmo mais promissor para uma implementação paralela.

Os algoritmos APC e APCS apresentaram um resultado muito bom em relação aos outros algoritmos da literatura, pois conseguiram produzir soluções melhores para todas as instâncias testadas com exceção de uma. Além da boa qualidade das soluções geradas, o tempo de execução destes algoritmos foi pequeno: em média 14,4s para o APC e 5,8s para o APCS.

Como primeira possibilidade de um trabalho futuro, pode-se estudar métodos de organizar a sequência de átomos das instâncias de modo que elas satisfaçam as hipóteses do PDGDM, com *corte* de no máximo 6Å. Na metodologia desenvolvida neste trabalho, somente os átomos das cadeias laterais passam por este processo que é resolvido pelo POAP. Acredita-se que, se os átomos da cadeia principal forem reorganizados, juntamente com os átomos das cadeias laterais, possivelmente, instâncias como 1AX8, 1BPM, 1F39 e 1HMV possam ser resolvidas utilizando um corte de 6Å. Um outro importante trabalho a ser feito

é o de considerar distâncias inexatas entre os átomos, tendo em vista que, na prática, os experimentos NMR determinam apenas limites inferiores e superiores para as mesmas.

Referências

[1] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., Bourne, P. E. The protein data bank. *Nucleic Acids Research 28* (2000), 235 – 242.

[2] Biggs, N., Lloyd, E. K., Wilson, R. J. *Graph Theory, 1736-1936*. Clarendon Press, New York, 1986.

[3] Biswas, P., Lian, T.-C., Wang, T.-C., Ye, Y. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks 2*, 2 (2006), 188–220.

[4] Brünger, A. T., Nilges, M. Computational challenges for macromolecular structure determination by x-ray crystallography and solution nmr-spectroscopy. *Quarterly Reviews of Biophysics 26*, 1 (1993), 49–125.

[5] Carvalho, R., Lavor, C., Protti, F. Extending the geometric build-up algorithm for the molecular distance geometry problem. *Inf. Process. Lett. 108*, 4 (2008), 234–237.

[6] Creighton, T. *Proteins: Structures and Molecular Properties,* 2nd ed. W.H. Freeman, New York, 1993.

[7] Crippen, G., Havel, T. *Distance Geometry and Molecular Conformation*. Research Studies Press Ltd., New York, 1988.

[8] Dong, Q., Wu, Z. A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization 22*, 1-4 (2002), 365–375.

[9] Dong, Q., Wu, Z. A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. *Journal of Global Optimization 26*, 3 (2003), 321–333.

[10] Golub, G. H., Van Loan, C. F. *Matrix Computations,* 2nd ed. Johns Hopkins University Press, Baltimore, 1989.

[11] Gramacho, W. Algoritmos para o cálculo de estruturas de proteínas. Dissertação de Mestrado, UFF, 2008.

[12] Grosso, A., Locatelli, M., Schoen, F. Solving molecular distance geometry problems by global optimization algorithms. *Computational Optimization and Applications* (2007). A ser publicado.

[13] Havel, T. F. An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. Em *Progress in Biophysics and Molecular Biology*. Permagon Press, Oxford, England, 1991, p. 43–78.

[14] Havel, T. F. Distance geometry. Em *Encyclopedia of Nuclear Magnetic Resonance* (1996), J. Wiley & Sons, p. 1701–1710.

[15] Hendrickson, B. Conditions for unique graph realizations. *SIAM Journal on Computing 21*, 1 (1992), 65–84.

[16] Hendrickson, B. The molecule problem: Exploiting structure in global optimization. *SIAM Journal on Optimization 5*, 4 (1995), 835–857.

[17] Klock, H., Buhmann, J. M. Multidimensional scaling by deterministic annealing. Em *EMMCVPR '97: Proceedings of the First International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition* (London, UK, 1997), Springer-Verlag, p. 245–260.

[18] Krippahl, L. Determinação da estrutura de proteínas através de programação por restrições. Dissertação de Mestrado, Universidade Nova de Lisboa, 1999.

[19] Lavor, C. *Global Optimization: from Theory to Implementation*, in liberti, l. and maculan, n. ed. Springer, Berlin, 2006, ch. On generating instances for the molecular distance geometry problem.

[20] Lavor, C., Liberti, L., Maculan, N. The discretizable molecular distance geometry problem, 2006. arXiv:q-bio/0608012v1.

[21] Lavor, C., Liberti, L., Maculan, N. *Global Optimization: Scientific and Engineering Case Studies*, in pintér, j. ed. Springer, New York, 2006, ch. Computational Experience With The Molecular Distance Geometry Problem.

[22] Lavor, C., Liberti, L., Maculan, N. An overview of distinct approaches for the molecular distance geometry problem. Em *Encyclopedia of Optimization* (Berlin, 2008), P. Pardalos e C. Floudas, Eds. 2nd Edition.

[23] Liberti, L., Lavor, C., Maculan, N. Double vns for the molecular distance geometry problem. Em *Proceedings of Mini Euro Conference on Variable Neighbourhood Search* (Tenerife, Spain, 2005), p. 23–5.

[24] Liberti, L., Lavor, C., Maculan, N. A branch-and-prune algorithm for the molecular distance geometry problem. *International Transactions in Operational Research 15*, 1 (2008), 1–17.

[25] mi Yoon, J., Gad, Y., Wu, Z. Mathematical modeling of protein structure using distance geometry. Relatório Técnico TR0024, Computational and Applied Mathematics Department of Rice University, julho de 2000.

[26] Moré, J. J., Wu, Z. Global smoothing and continuation for large-scale molecular optimization. Relatório Técnico MCS-P–539-1095, Mathematics and Computer Science Division, Argonne National Lab., IL,USA, 1995.

[27] Moré, J. J., Wu, Z. Smoothing techniques for macromolecular global optimization. Em *Nonlinear Optimization and Applications*, P. Press, Ed. Di Pillo, G. and Giannessi, F., New York, 1996, p. 297–312.

[28] Moré, J. J., Wu, Z. ϵ-optimal solutions to distance geometry problems via global continuation. Em *Global minimization of nonconvex energy functions: molecular conformation and protein folding*. American Mathematical Society, Providence, RI, 1996, p. 151–168.

[29] Moré, J. J., Wu, Z. Global continuation for distance geometry problems. *SIAM Journal on Computing 7*, 3 (1997), 814–836.

[30] Moré, J. J., Wu, Z. Distance geometry optimization for protein structures. *Journal of Global Optimization 15*, 3 (1999), 219–234.

[31] Phillips, A., Rosen, J., Walke, V. Molecular structure determination by convex global underestimation of local energy minima, 1996.

[32] Saxe, J. B. Embeddability of weighted graphs in k-space is strongly NP-hard. *Proc. 17th Allerton Conference in Communications, Control and Computing* (1979), 480–489.

[33] Schlick, T. *Molecular Modeling and Simulation: An Interdisciplinary Guide*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[34] Trosset, M. W. Applications of multidimensional scaling to molecular conformation. *Computing Science and Statistics 29*, 1 (1998), 148–152.

[35] Wu, D., Wu, Z. An updated geometric build-up algorithm for solving the molecular distance geometry problems with sparse distance data. *Journal of Global Optimization 37*, 4 (2007), 661–673.

[36] Wu, D., Wu, Z., Yuan, Y. Rigid vs. unique determination of protein structures with geometric buildup. *Optimization Letters 2*, 3 (2008), 319–331.

[37] Zou, Z., Bird, R. H., Schnabel, R. B. A stochastic/perturbation global optimization algorithm for distance geometry problems. *Journal of Global Optimization 11*, 1 (1997), 91–105.