Universidade Federal Fluminense

WARLEY GRAMACHO DA SILVA

Algoritmos para o Cálculo de Estruturas de Proteínas

NITERÓI

2008

Universidade Federal Fluminense

WARLEY GRAMACHO DA SILVA

Algoritmos para o Cálculo de Estruturas de Proteínas

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. Luiz Satoru Ochi, D.Sc.

Prof. Carlile Campos Lavor, D.Sc.

NITERÓI

2008

Algoritmos para o Cálculo de Estruturas de Proteínas

Warley Gramacho da Silva

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:

Prof. Luiz Satoru Ochi, D.Sc. / IC-UFF (Presidente)
(Orientador)

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

Prof. Fábio Protti, D.Sc. / DCC-IM-UFRJ

Profa. Loana Tito Nogueira, D.Sc. / IC-UFF

Profa. Simone de Lima Martins, D.Sc. / IC-UFF

Niterói, 04 de Abril de 2008.

"Aquele que recebe de mim uma idéia tem aumentada a sua instrução sem que eu tenha diminuído a minha. Como aquele que acende sua vela na minha recebe luz sem apagar a minha vela. Que as idéias passem livremente de uns aos outros no planeta, para a instrução moral e mútua dos homens e a melhoria de sua condição, parece ter sido algo peculiar e benevolentemente desenhado pela natureza ao criá-las, como o fogo, expansível no espaço, sem diminuir sua densidade em nenhum ponto."

Thomas Jefferson

Aos meus pais, Zilda e Joaquim, que dedicaram suas vidas
para me dar oportunidades as quais nunca tiveram.

Agradecimentos

Primeiramente, agradeço a Deus, por tudo que me é proporcionado todos os dias e por ter permitido a conclusão deste trabalho.

Agradeço aos meus pais Zilda e Joaquim por todo o amor e carinho, pelo apoio incondicional em todas as fases da minha vida. Aos meus irmãos, Glenda, Wandrey e Wesley que muitas vezes, mesmo não percebendo, tiveram influência no incentivo à busca do conhecimento e na minha formação como pessoa.

Um agradecimento muito especial aos professores, orientadores e amigos: Satoru e Carlile. Pelo constante incentivo, sempre indicando a direção a ser tomada nos momentos de maior dificuldade, depositando em mim e em meu trabalho confiança e credibilidade, no auxílio na revisão da dissertação, contribuindo significativamente para a qualidade desta, mas principalmente pela quantidade e qualidade de conhecimento transmitido.

Agradeço ao Silas e ao Thársis Tuani(UNICAMP) pela ajuda com as instâncias reais, contribuindo para melhora desse trabalho.

Agradeço à minha doce e pequena Glêndara, pelo seu apoio absoluto, amor e paciência em todos os momentos durante a realização desta dissertação. Nada disso teria graça alguma se não existisse você. Agradeço também a toda a sua família.

Aos amigos de república: Rodrigo, Rafael (Guto), Rafael, Emmanuel, Daniel (Chileno), Diego e Diney que, na maioria das vezes, fizeram os momentos em Niterói bem mais agradáveis e divertidos.

Um agradecimento a todos os novos amigos que fiz no IC. São tantas pessoas que até corro o risco de esquecer um eventual nome. Aletéia, Aline e Alexandre, Alisson, Anand, André Renato, Ary, Augusto, Carlão, Copetti, Cris, Dani e Rodrigo, Diego, Edgar, Haroldo, Helder, Higor, Idalmes, Ivan, Jackson, Jacques, Jairo, Janine, Johnny, Kennedy, Luciana Brugiolo, Luciano Bertini, Marcelo, Mário Mestria, Renathinha, Sanderson, Sillas, Stênio, Synara, Thiago (Facada), Vinicio, Vivi e Jonivan, Yuri.

Agradeço à Maria e à Ângela por sempre se prontificarem a me ajudar.

Agradeço ao Instituto de Computação da Universidade Federal Fluminense e a todos seus professores.

Agradeço aos membros da banca examinadora pelas sugestões e comentários: Fábio, Protti, Loana e Simone.

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

Peço desculpas às pessoas cujos nomes deveriam estar aqui, mas acabaram ficando de fora.

Resumo

Um dos problemas mais importantes em biologia computacional é a determinação da estrutura tridimensional de uma proteína. Esta estrutura pode ser determinada experimentalmente através de técnicas de RMN. Geralmente, dados de RMN provêem apenas um conjunto esparso de distâncias entre os átomos de uma molécula. Neste caso, o problema é determinar a estrutura tridimensional de uma molécula usando um conjunto de distâncias entre alguns pares de átomos da molécula. 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 para tratá-lo como um problema de otimização combinatória, através de uma formulação discreta. Dois algoritmos e dois *softwares* de visualização são propostos nesta dissertação. Para testar os algoritmos, foram utilizadas instâncias artificiais e reais.

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

Abstract

One of the most important problems in computational biology is the determination of the three-dimensional structure of a protein. This structure can be determined experimentally using NMR techniques. Generally, the NMR data provide 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 certain pairs of atoms of the molecule, and 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 were presented recently in the literature by using a discrete formulation. Two algorithms and two softwares for visualizing are proposed in this work. In order to test the algorithms we have used real and artificial instances.

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

Siglas e Abreviações

PGDM	:	Problema de Geometria das Distâncias em Moléculas
PDGDM	:	Problema Discreto de Geometria das Distâncias em Moléculas
RMN	:	Ressonância Magnética Nuclear
BP	:	*Branch and Prune*
LDE	:	*Largest Distance Error*
PDB	:	*Protein Data Bank*
RMSD	:	*Root-Mean-Square Deviation*

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

As proteínas são moléculas fundamentais dos sistemas biológicos, sendo constituídas por uma cadeia linear de aminoácidos [4]. A função protéica é determinada pela sua estrutura tridimensional, que pode ser obtida de duas formas: experimentalmente, via Ressonância Magnética Nuclear (RMN) ou Cristalografia de Raio-X [2], e teoricamente, através da minimização da energia potencial ou por simulação de dinâmica molecular [7]. Neste trabalho, será considerado o problema da determinação da estrutura através da RMN. Mais especificamente, iremos tratar o problema de determinar a estrutura de uma proteína, usando um conjunto de distâncias entre pares de átomos que podem ser obtidas através do conhecimento dos comprimentos de ligações e ângulos entre ligações covalentes, e através de experimentos de RMN. 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 [1], reconhecimento de imagens [13], etc.

No PGDM, caso as distâncias entre todos os pares de átomos sejam previamente conhecidas, uma única estrutura tridimensional pode ser determinada, em tempo polinomial [6]. Caso contrário, o problema passa a ser NP-difícil [27]. Recentemente, Lavor, Liberti e Maculan [16] 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 foi denotada por Problema Discreto de Geometria das Distância em Molécula (PDGDM).

Existem várias abordagens para o PGDM, por exemplo, o algoritmo *EMDED* de Crippen e Havel [5, 10], a estratégia de redução de grafo de Hendrickson [11, 12], o algoritmo DGSOL de Moré e Wu [21, 22, 23, 25], o método de perturbação estocástica

de Zou, Bird e Schnabel [31], o método de escala multidimensional de Trosset [29], o algoritmo VNS de Lavor, Liberti e Maculan [18, 17], o *geometric build-up* algoritmo de Dong, Wu e Wu [30], o algoritmo BP de Lavor, Liberti e Maculan [16, 19], etc. *Surveys* sobre o problema são encontrados em [2, 5, 9, 15, 20].

Nesta dissertação, propomos dois novos algoritmos para o PDGDM, além de dois *softwares* de visualização. Apesar da simplicidade dos algoritmos, eles gerenciam de forma eficiente o custo de memória e tempo, comparados com uma versão do algoritmo existente na literatura. Para testar os algoritmos, foram utilizadas instâncias artificiais e reais da literatura.

O trabalho está organizado em cinco capítulos. O contéudo de cada capítulo é apresentado a seguir.

- Capítulo 2: introduz o PGDM, bem como sua formulação discreta: o PDGDM.
- Capítulo 3: apresenta as principais contribuições deste trabalho: dois novos algoritmos para o PDGDM e dois softwares de visualização.
- Capítulo 4: faz uma análise experimental, comparando os algoritmos propostos com um algoritmo 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 de uma molécula. Este problema pode ser formulado da seguinte forma: encontre as posições $x_1, \ldots, x_n \in \mathbb{R}^3$ dos átomos da molécula, 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 $d_{i,j}$ são conhecidas a priori e $||.||$ é a norma Euclidiana.

A formulação 2.1 corresponde ao PGDM exato. Devido ao 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, encontrando 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 das distâncias, respectivamente.

O PGDM pode ser formulado como um problema de otimização contínua, onde a

função objetivo é dada por

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

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 glocal [12].

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 [16], introduzindo assim uma subclasse de problemas do PGDM, chamada de Problema Discreto de Geometria das Distâncias em Moléculas (PDGDM). Também em [16], foi demostrado que o PDGDM é NP-difícil.

2.2.1 Formulação discreta

Considere uma molécula como sendo uma seqüência de n átomos, onde os comprimentos de ligações covalentes, que corresponde à distância média entre os núcleos de dois átomos ligados na posição de maior estabilidade (menor energia), são denotadas por $d_{i-1,i}$, para $i = 2, \ldots, n$, os ângulos de ligações covalentes 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: os comprimentos de ligações e os ângulos de ligações, bem como as distâncias entre átomos separados por 3 ligações consecutivas são conhecidos.
- Hipótese 2: Os ângulos de ligações não podem ser múltiplos de π.

A Hipótese 1 é aplicável à maioria das proteínas, pois os comprimentos de ligações e os ângulos de ligações são conhecidos a priori. 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, com valores menores do que 5Å, que é a "precisão" da RMN [4, 28]. A Hipótese 2 é igualmente aplicável às proteínas, dado que não se conhece proteína com ângulos de ligações covalentes com valor exato de π.

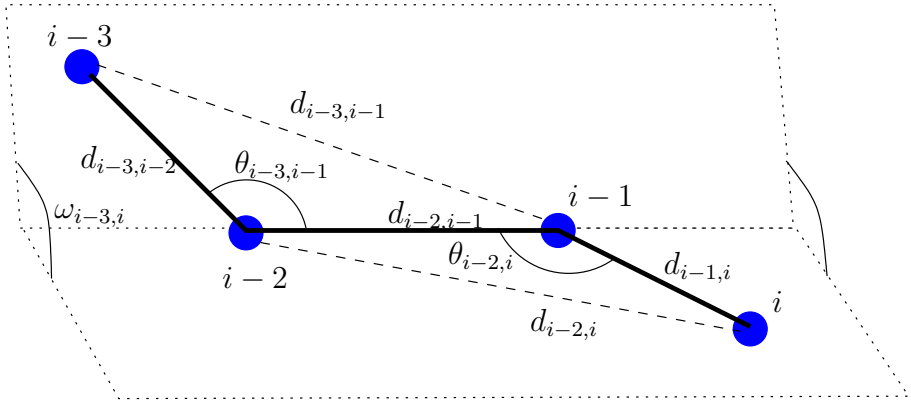


Figura 2.1: Definições de *comprimento de ligações*, *ângulos de ligações* e ângulos de torção.

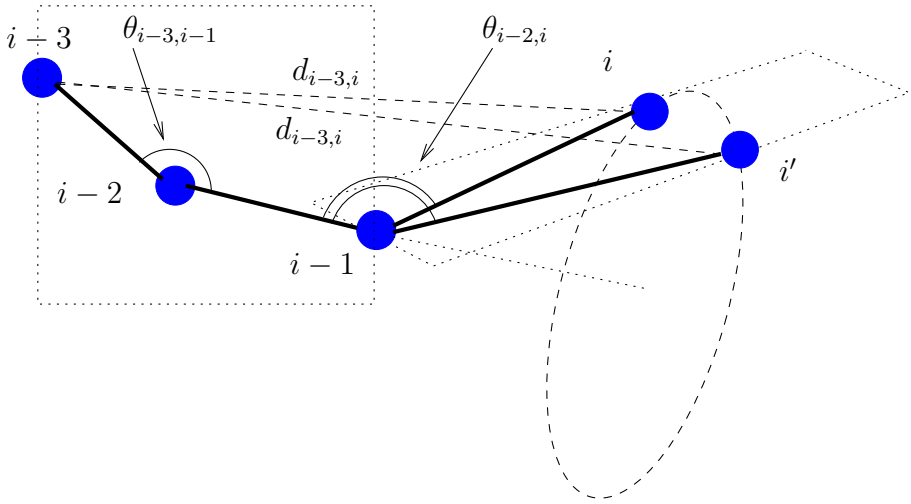


Figura 2.2: No PDGDM, o átomo i pode estar somente em duas posições (i e i') para ser "viável" com a distância $d_{i-3,i}$.

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, dando-nos 2^{n-3} possíveis moléculas.

Dados todos os comprimentos de ligações $d_{1,2}, \ldots, d_{n-1,n}$, ângulos de ligações

$\theta_{13}, \ldots, \theta_{n-2,n}$, e ângulos de torção $\omega_{1,4}, \ldots, \omega_{n-3,n}$ de uma molécula com n átomos, as coordenadas cartesianas $x_i = (x_{i_1}, x_{i_2}, x_{i_3})$, para cada átomo i na molécula, podem ser obtidas utilizando a seguinte fórmula [26]:

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

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

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

que é apenas um rearranjo da lei dos cossenos para os ângulos de torção [16].

Usando os comprimentos de ligações $d_{1,2}, d_{2,3}$ e o ângulo de ligação $\theta_{1,3}$, podemos

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, pela Hipótese 1.

Uma vez que a distância $d_{1,4}$ é conhecida, novamente 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), obtemos apenas duas posições possíveis (x_4, 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, obtemos 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, podemos observar que para o i-ésimo átomo, existem 2^{n-3} posições possíveis. Desta forma, para representarmos uma molécula como uma seqüência linear de n átomos, temos 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 $x = (x_1, \ldots, x_n) \in \mathbb{R}^3$.

Capítulo 3

Algoritmos

Este capítulo apresenta os dois algoritmos propostos e implementados para o PDGDM, assim como as ferramentas de visualização, que são as contribuições desta dissertação.

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

Seja $F = \{(j,i) \mid i - j \geq 4\}$ o conjunto de pares de átomos no qual a distância entre (j,i) é menor ou igual a 5Å. Então, 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.

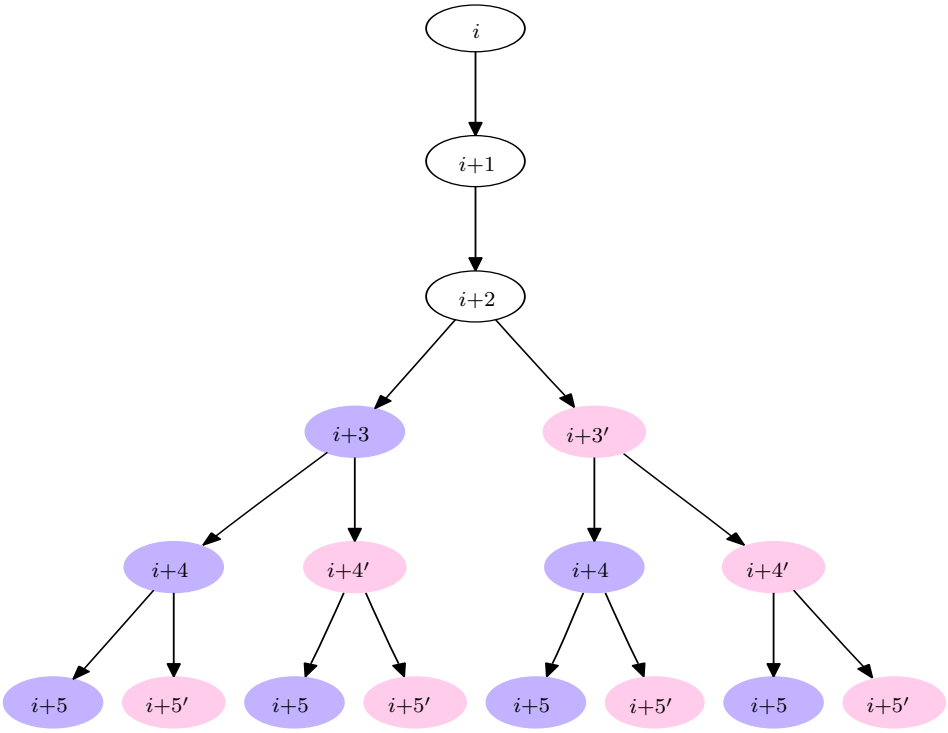


Figura 3.1: *Representação em árvore binária*

3.1 Algoritmos Propostos para o PDGDM

3.1.1 Algoritmo I

O 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 serão 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 serão soluções para o problema. Note que, se a quantidade de nós remanescentes em um dado nível é muito grande, o algoritmo poderá ter problemas com o tamanho da memória ocupada. Observe ainda que, ou o algoritmo encontra todas as soluções possíveis para o problema, ou não encontra nenhuma solução, caso tenha tido problema com o tamanho da memória.

No pseudo-código apresentado para o algoritmo I, T é uma representação de árvore. T é inicializado 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. O quarto átomo pode ser fixado em x_4 ou x_4', devido existir uma solução simétrica em torno do plano definido pelo

três primeiros átomos. Mais especificamente, qualquer solução formada em um lado deste plano causa também uma outra solução simétrica do outro lado do plano, permitindo assim reduzir o custo computacional pela metade, [16] .

Para cada nó da árvore no nível i, as seguintes informações são atribuídas:

- Posição $x_i \in \mathbb{R}^3$ para o i-ésimo átomo;
- Matriz de torção B_i, definida em 2.5, e o produto acumulativo das matrizes de torção, denotado por $Q_i = \prod_{j=1}^{i} B_j$.

Algoritmo 1 *Algoritmo I*

1: $T = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$;
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.

Algoritmo 2 *ExpandeArvore(* T, f)

1: **para** *cada nó folha* **faça**
2: *Expande árvore binária até o nível de altura* $j \in f$*, onde em cada nível* i*, calcula matrizes de torção* B_i' *para os nós da esquerda e* B_i *para os nós da direita, via Eq. 2.5*;
3: *busca matriz de torção acumulada* Q_{i-1} *do nível anterior*;
4: *calcula* $Q_i = Q_{i-1}B_i$, $Q_i' = Q_{i-1}B_i'$ *e* x_i, x_i' *de* $Q_i y, Q_i' y$;
5: *cria* v' *que representa um nó da árvore, guarde* Q_i' *e* x_i';
6: *cria* v *que representa um nó da árvore, guarde* Q_i *e* x_i;
7: $T \leftarrow$ *adicionando* v' *aos nós da esquerda e* v *aos nós da direita*;
8: **fim para**
9: **retorna** T;

Algoritmo 3 *PodaArvore(* T, f)

1: **para** *cada nó folha* **faça**
2: **se** *Desigualdade 3.1 é satisfeita* **então**
3: *Continue*;
4: **senão**
5: *Elimina respectivo nó folha de* T ;
6: **fim se**
7: **fim para**
8: **retorna** T;

Exemplo: Veja um exemplo simples da aplicação do Algoritmo 1, considerando uma instância gerada artificialmente, como explicado em [14]. A instância em questão (chamada `lavor6`) tem 6 átomos e todos os ângulos de ligações têm 1.91 radianos. Ou seja,

$$\begin{aligned} \delta(1) \cup F &= \{5\}, d_1^F = (2.618574532) \\ \delta(2) \cup F &= \{6\}, d_2^F = (2.573247413), \end{aligned}$$

onde $\delta(i) = \{(j,i) \in E\}$ é o conjunto dos átomos adjacentes ao átomo i e E é o conjunto de pares de átomos (i,j) cujas as distâncias d_{ij} são conhecidas a priori. Denotando por H o conjunto de todas as distâncias entre átomos separados por 3 ligações consecutivas, obtemos

$$\begin{aligned} H &= \{(i,i+1) | 1 \leq i \leq n-1\} \cup \\ &\quad \{(i,i+2) | 1 \leq i \leq n-2\} \cup \\ &\quad \{(i,i+3) | 1 \leq i \leq n-3\}. \end{aligned}$$

O vetor das distâncias em H é dado por:

$$\begin{aligned} d^H = (&1.526, 2.491389536, 3.030585264, \\ &1.526, 2.491389535, 2.909957921, \\ &1.526, 2.491389535, 2.997115238, \\ &1.526, 2.491389536, \\ &1.526). \end{aligned}$$

Através da Figura 3.2, pode-se verificar o comportamento do algoritmo, onde a árvore é inicializada com os três primeiros átomos fixos (Figura 3.2(a)). Em seguida, a árvore é expandida até o nível 5 (Figura 3.2(b)), pois $(j,k) \in F$, com $k = 5$. No próximo passo, verifica-se quais nós folhas estão corretos, para que seja feita a poda, Figura 3.2(c). Novamente, o algoritmo faz a expansão na árvore até o nível indicado por $(j,k) \in F$, agora com $k = 6$ (Figura 3.2(d)). Por fim, uma nova poda é feita e os nós folhas remanescente são as soluções do problema. Note que, para esta instância, são encontradas duas soluções.

3.1.2 Algoritmo II

O Algoritmo II explora a árvore ilustrada na Figura 3.1 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. No algoritmo proposto, a implementação é não-recursiva, fazendo-se uso de uma estrutura de pilha para auxiliar na implementação. Neste sentido, todos os nós expandidos recentemente são adicionados a uma pilha, para realizar a exploração. Este processo se repete até que se chegue ao último nível da árvore. Neste nível, encontra-se uma solução para o problema, o que implica dizer que a pilha está cheia. Caso se deseje que o algoritmo continue percorrendo o espaço de busca para encontrar outras possíveis soluções, então a solução encontrada é armazenada e repete-se o processo explorando outro ramo da árvore.

O algoritmo 4 apresenta o pseudo-código dessa estratégia. Cada nó da árvore, no nível i, contém as seguintes informações:

- Posição $x_i \in \mathbb{R}^3$ para o i-ésimo átomo;
- Matriz de torção B_i e o produto cumulativo das matrizes de torção $Q_i = \prod_{j=1}^{i} B_j$;
- Variável (subarvore) que auxilia a pilha no controle de descer e retrocede na árvore.

Exemplo: Considere novamente a mesma instância usada na subseção 3.1.1, p. 11. A Figura 3.3 mostra como o algoritmo simula a exploração da árvore através de uma pilha.

3.2 *Softwares* de Visualização

Dois *softwares de visualização* foram desenvolvidos nesta dissertação. O primeiro trata de gerar a representação visual 3D das proteínas associadas às soluções encontradas por um dos algoritmos. O segundo é responsável por desenhar o percurso feito por algum dos algoritmos até encontrar soluções para o problema, onde uma representação de estrutura de arvore binária é usada para este fim. Em seguida descrevemos mais detalhes de cada *software*:

Algoritmo 4 *Algoritmo II*

1: P *é a reprentação da pilha, onde cada elemento de* P *representa um nó da arvore(que será simulada na pilha)*;
2: $P = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$ *(representa os quatros primeiros nós que podem ser fixados na árvore)*;
3: $k \leftarrow 0$;
4: $Flag \leftarrow true$;
5: **enquanto** $Flag$ & $Card(P) > 3$ **faça**
6: **enquanto** $Card(P) < F[k].j$ **faça**
7: $P \leftarrow$ *Empilha(P)*;
8: **fim enquanto**
9: $t \leftarrow$ *P[topo da pilha]*;
10: **se** *em t a desigualdade 3.1 não é satisfeita* **então**
11: $P \leftarrow$ *Desempilha(P, k)*;
12: **senão**
13: $k \leftarrow k + 1$;
14: **se** $k = Card(F)$ **então**
15: *Encontrou solução*;
16: **se** *algoritmo executado para encontrar apenas uma solução* **então**
17: $Flag \leftarrow false$;
18: **senão**
19: *guarda solução*;
20: $P \leftarrow$ *Desempilha(P, k)*;
21: **fim se**
22: **fim se**
23: **fim se**
24: **fim enquanto**

Algoritmo 5 *Empilha*(P)

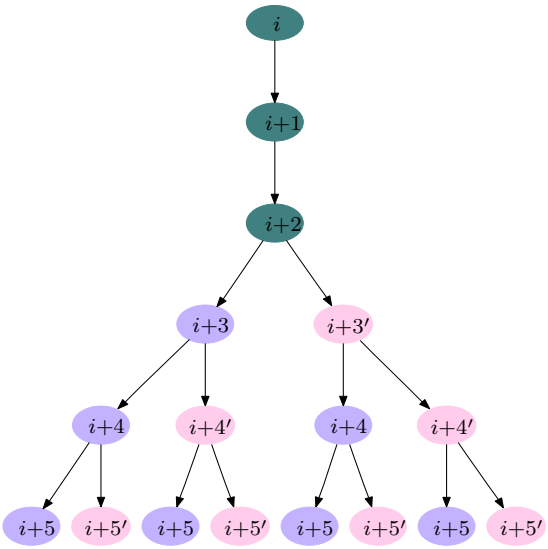
1: Calcula possíveis posições para i-ésimo átomo:
2: *calcula matrizes de torção* B_i, B'_i *via Eq. (2.5)*;
3: *restaura matriz de torção acumulada* Q_{i-1} *do topo da pilha*;
4: *calcula* $Q_i = Q_{i-1}B_i$, $Q'_i = Q_{i-1}B'_i$ *e* x_i, x'_i *de* $Q_i y, Q'_i y$;
5: $t \leftarrow$ *P[topo da pilha]*;
6: **se** $t.subarvore = ESQUERDA$ **então**
7: *cria v'(resentação do nó esquerdo da árvore), guarde* Q'_i *e* x'_i;
8: $v'.subarvore \leftarrow ESQUERDA$;
9: *P[topo da pilha].subarvore* $\leftarrow DIREITA$;
10: $P \leftarrow P \cup \{v'\}$;
11: **senão**
12: *cria v(resentação do nó direito da árvore), guarde* Q_i *e* x_i;
13: $v.subarvore \leftarrow ESQUERDA$;
14: *P[topo da pilha].subarvore* $\leftarrow PODA$;
15: $P \leftarrow P \cup \{v\}$;
16: **fim se**
17: **retorna** P;

Algoritmo 6 *Desempilha*(P, k)

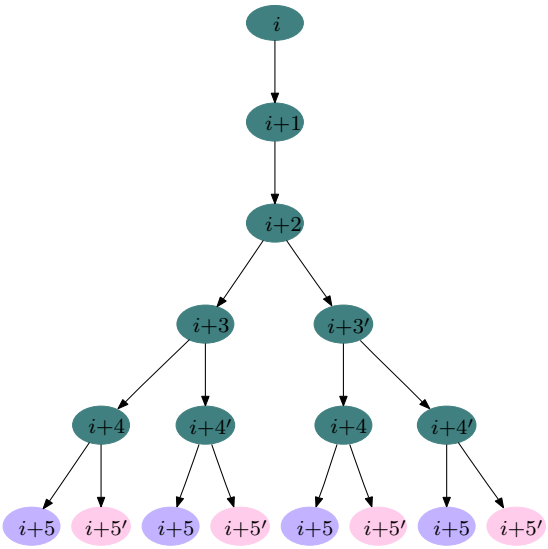
1: **repita**
2: $P \leftarrow P-$ {*P[topo da pilha]*};
3: $t \leftarrow$ *P[topo da pilha]*;
4: **enquanto** $P < F[k-1]$ **faça**
5: $k \leftarrow k-1$;
6: **fim enquanto**
7: **até** $t.subarvore = PODA$
8: **retorna** P;

- *Visualizador 3D:* Dada uma solução encontrada por um dos algoritmos, o visualizador desenha a proteína no espaço dimensional, permitindo observar o comportamento da mesma. As operações de rotação, translação e *zoom* são também características do visualizador. Veja Figura 3.3, para exemplos de telas do *software*.

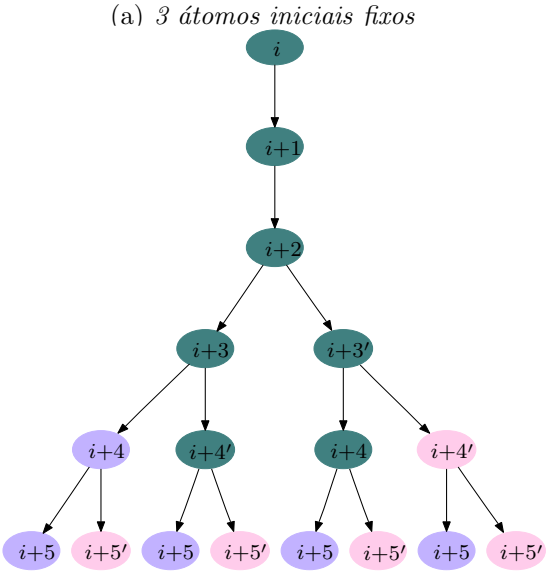
- *Visualizador de Percurso Algorítmico:* Este *software* permite desenhar uma árvore que contém o percurso seguido por um dos algoritmos até encontrar as soluções. A motivação do desenvolvimento deste *software* baseia-se na possibilidade de estudar as soluções encontradas, a fim de descobrir informações de padrões de solução, bem como outras informações que venham a ser relevantes para implementações futuras. A Figura 3.4 apresenta a saída do Visualizador de Percurso Algorítmico desenvolvido, quando este encontra soluções para uma dada instância.



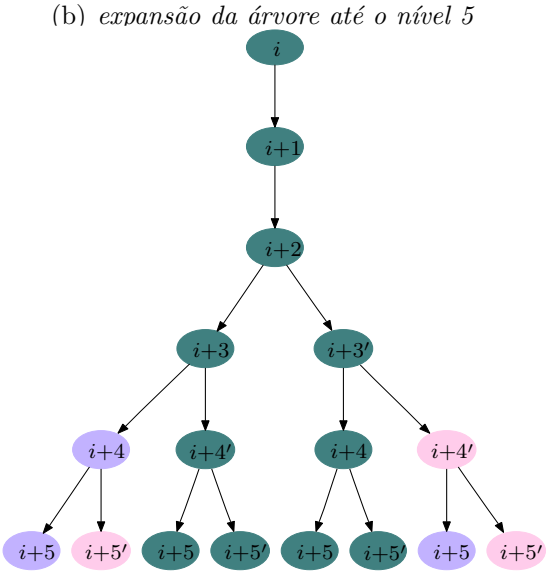
(a) *3 átomos iniciais fixos*



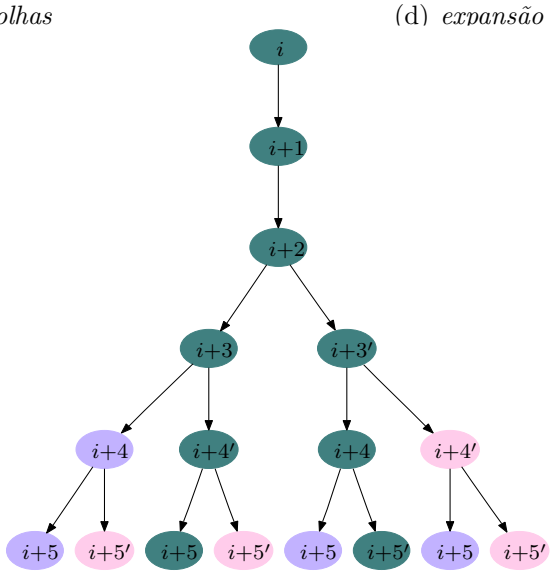
(b) *expansão da árvore até o nível 5*



(c) *Poda dos nós folhas*

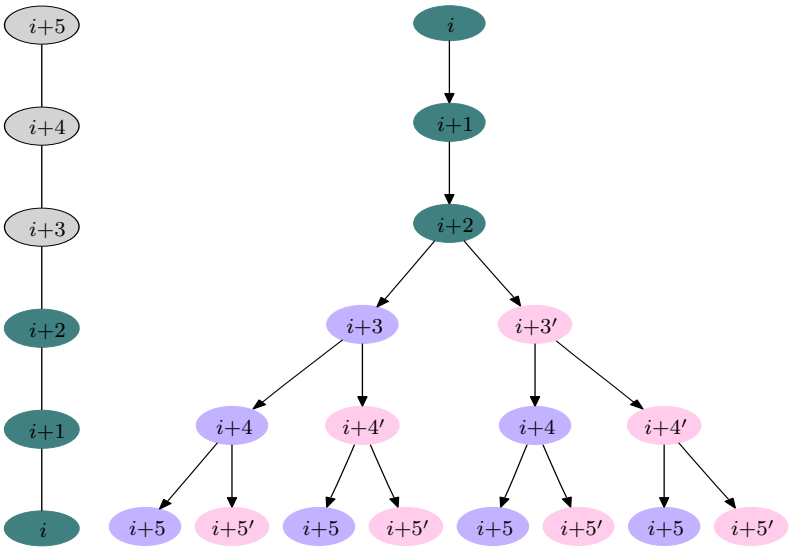


(d) *expansão da árvore até o nível 6*

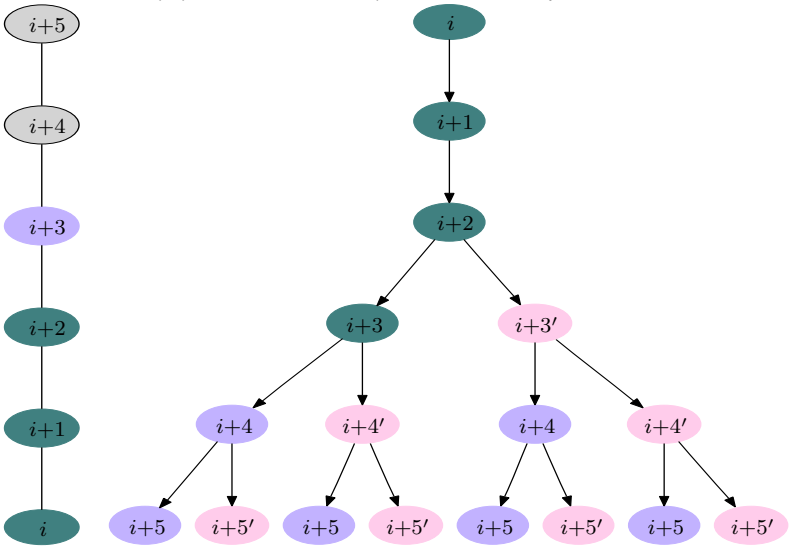


(e) *Poda dos nós folhas*

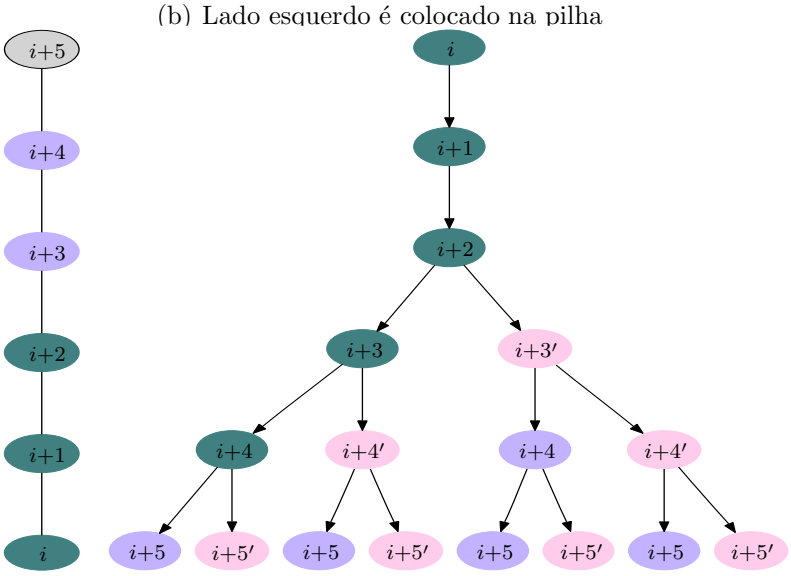
Figura 3.2: *Exemplo do Algoritmo I*



(a) 3 átomos fixos / 3 átomos na pilha

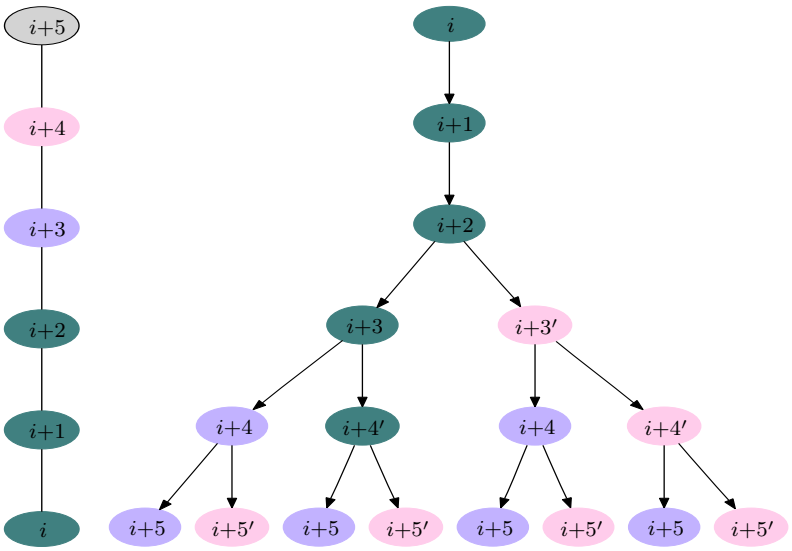


(b) Lado esquerdo é colocado na pilha

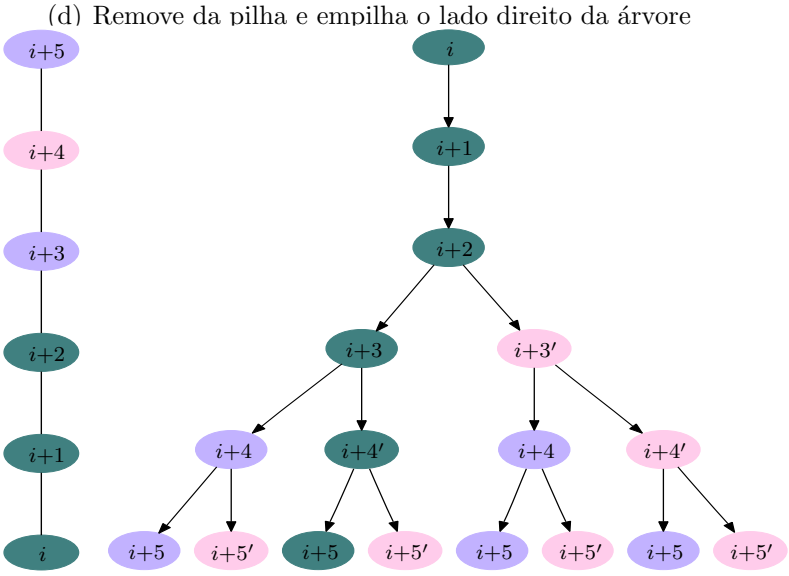


(c) Lado esquerdo é colocado na pilha

Figura 3.3: *Exemplo do Algoritmo II*



(d) Remove da pilha e empilha o lado direito da árvore



(e) Lado esquerdo é colocado na pilha

Figura 3.3: *Exemplo do Algoritmo II (continuação)*

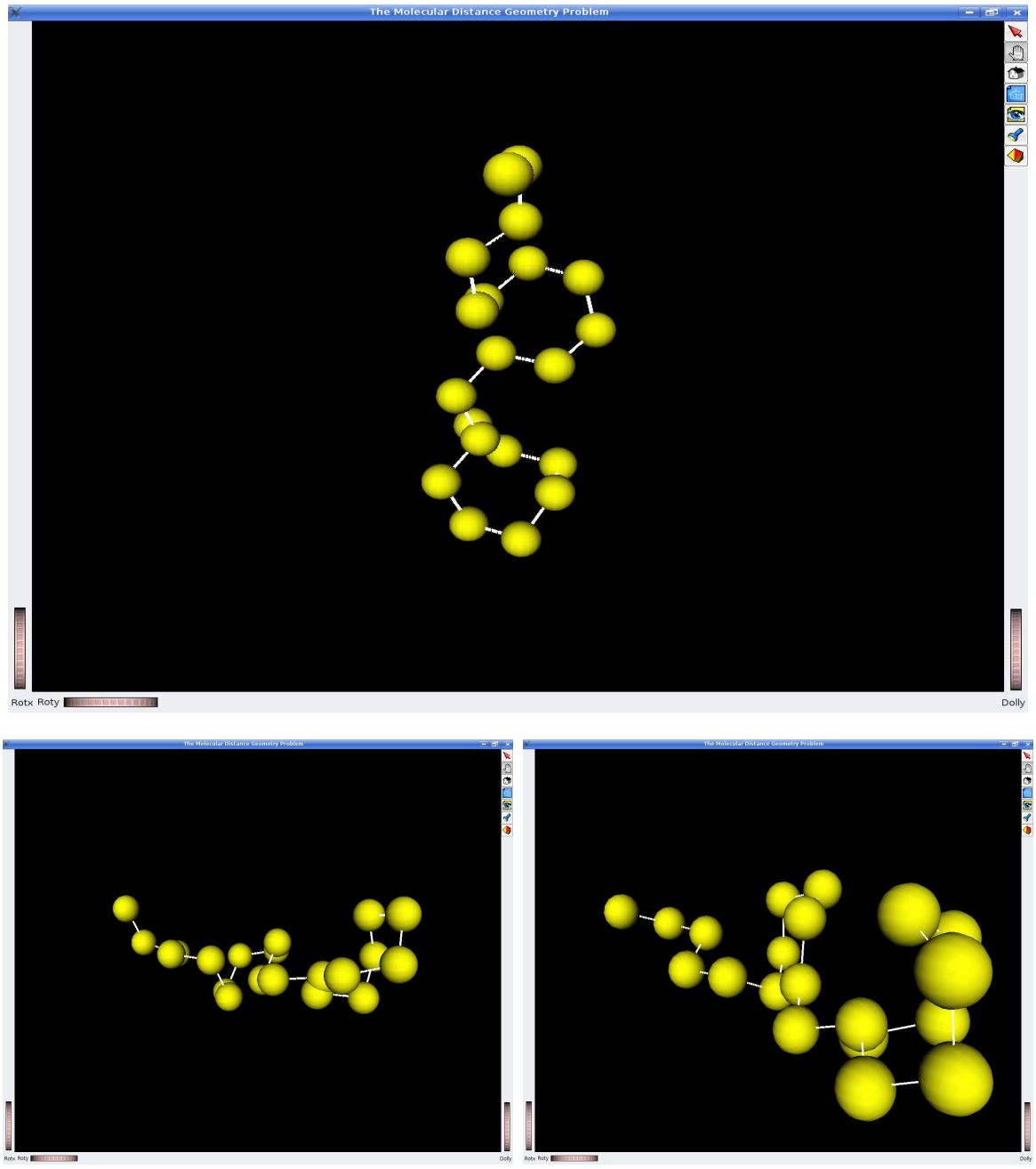


Figura 3.3: *Visualizador 3D*

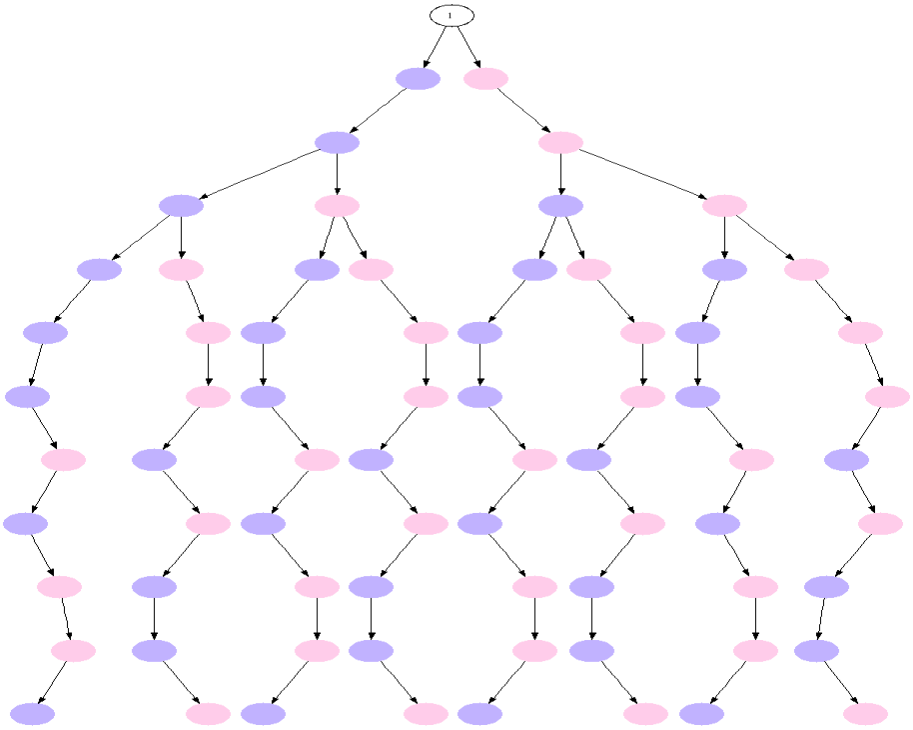


Figura 3.4: *Visualizador de Percurso Algorítmico*

Capítulo 4

Resultados Experimentais

Neste capítulo, serão apresentados os experimentos computacionais para analisar os algoritmos propostos. Primeiramente, os experimentos foram realizados com instâncias artificiais, comparando os algoritmos propostos com outro algoritmo já existente, denominado `Branch and Prune - (BP)` [16, 19], que é também um algoritmo para o PDGDM. Em seguida, apresentamos os resultados com instâncias reais, sem comparar com o BP, pois não existem testes com instâncias reais para este algoritmo.

Os algoritmos propostos foram implementadas em C++, utilizando-se a *Standard Template Library* e compilados com g++ 4.0.2, utilizando os comandos `g++ -O2`, que otimiza o código binário gerado. Utilizamos um computador Dell Optiplex, com processador Pentium D de 3.0 GHz e 2 Gbytes de memória RAM, usando o sistema operacional Linux, versão 2.6.12. Não foram permitidos acessos à memória secundária *"swap"*, por meio do comando `swapoff` existente no sistema operacional.

4.1 Experimentos com Instâncias Artificiais

Esta seção apresenta os experimentos com instâncias artificiais, chamadas *instância Lavor* [14], que são baseadas em um modelo proposto por [26], onde a molécula é representada como uma cadeia linear de átomos. Os comprimentos de ligações e ângulos de ligações 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 envolvido e m é o ID da instância.

Foram utilizadas 10 diferentes *instâncias Lavor*, para cada tamanho $n = 10, \ldots, 70$ ("instâncias pequenas"), e 4 diferentes *instâncias Lavor*, para cada tamanho de $n = 100i$, $i = 1, \ldots, 10$ ("instâncias grandes"). 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 e a terceira coluna indica o número n de átomos na molécula. As colunas seguintes contêm os conjuntos $|H|$, $|F|$ e $|E|$, respectivamente.

Instâncias Lavor					
Instância	**Nome**	n	$\|H\|$	$\|F\|$	$\|E\|$
1	`lavor10_0`	10	24	9	33
2	`lavor15_0`	15	39	18	57
3	`lavor20_0`	20	54	51	105
4	`lavor25_0`	25	69	62	131
5	`lavor30_0`	30	84	85	169
6	`lavor35_0`	35	99	72	171
7	`lavor40_0`	40	114	181	295
8	`lavor45_0`	45	129	110	239
9	`lavor50_0`	50	144	127	271
10	`lavor55_0`	55	159	392	551
11	`lavor60_0`	60	174	203	377
12	`lavor65_0`	65	189	78	267
13	`lavor70_0`	70	204	227	431
14	`lavor100_2`	100	294	311	605
15	`lavor200_2`	200	594	1250	1844
16	`lavor300_2`	300	894	1611	2505
17	`lavor400_2`	400	1194	1406	2600
18	`lavor500_2`	500	1494	3083	4577
19	`lavor600_2`	600	1794	3679	5473
20	`lavor700_2`	700	2094	2094	4188
21	`lavor800_2`	800	2394	4456	6850
22	`lavor900_2`	900	2694	5271	7965
23	`lavor1000_2`	1000	2994	5235	8229

Tabela 4.1: Instâncias Lavor

4.1.1 Experimentos Comparativos

O algoritmo `BP` explora a estrutura do problema de forma similar aos algoritmos propostos nessa dissertação. Porém, diferentemente do modo como fizemos, a estratégia do `BP` foi implementada de forma recursiva. O `BP` explora a árvore em profundidade, da esquerda para direita. 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, as informações armazenadas na pilha de recursividade se tornam muito grandes, podendo causar *overflow* de memória. Dessa forma, o algoritmo se torna muito rápido para as instâncias menores e impraticável para

as instâncias maiores. O algoritmo I, como mencionado na seção 3.1.1, pode, igualmente, causar *overflow* de memória, devido à sua estratégia de explorar a árvore em nível e tendo em vista que, em algum nível, a quantidade de nós pode ser muito grande (lembramos que este algoritmo ou encontra todas soluções ou é encerrado, caso haja problema com a memória). O algoritmo II resolve o problema de *overflow* 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 de todos os testes, o valor do parâmetro ε, que representa a tolerância de erro, foi 1×10^{-3}, o mesmo valor utilizado para o `BP` em [19]. A Tabela 4.2 apresenta os resultados referentes a cinco métodos: `BP` executado até encontrar a primeira solução (`BP-One`), `BP` executado até encontrar todas as soluções (`BP-All`), algoritmo II executado até encontrar a primeira solução (`AII-One`), algoritmo II executado até encontrar todas soluções (`AII-All`) e algoritmo I que pode ser executado somente para encontrar todas soluções (`AI-All`). Para o `BP-One` e o AII-One, apresentamos o tempo de CPU (em segundos), assim como o LDE - *Largest Distance Error*, definido como

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

empregado como uma medida de precisão da solução (quanto menor, melhor). Para os algoritmos `BP-All`, `AI-All` e `AII-All`, apresentamos o tempo de CPU (também em segundos), e o número de soluções encontradas (# Sol).

Para os algoritmos que foram executados até encontrar uma solução (`BP-One` e AII-One), as Figuras 4.1 e 4.2 exibem gráficos para o LDE e o tempo de CPU, respectivamente. Observamos uma pequena vantagem do algoritmo AII-One, no que se refere ao LDE (veja Figuras 4.1(a) e 4.1(b)). A diferença de tempo entre os algoritmos pode ser vista nas Figuras 4.2(a) e 4.2(b) que mostram que o algoritmo `BP-One` apresenta menor tempo de CPU. Isso já era esperado, tendo em vista que o `BP-One` é implementado recursivamente e o AII-One usa estruturas auxiliares para gerenciar a limitação de memória.

Para os algoritmos que foram executados até encontrar todas as soluções (`BP-All`, AI-All e AII-All), observamos que apenas o algoritmo AII-All conseguiu ser executado para todas as instâncias. Os algoritmos `BP-All` e AI-All só foram executados para encontrar todas as soluções com 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. No entanto, para o algoritmo AII-All, que foi

I	BP-One		AII-One		BP-All		AII-All		AI-All	
-	**CPU**	**LDE**	**CPU**	**LDE**	**CPU**	**#Sol**	**CPU**	**#Sol**	**CPU**	**#Sol**
1	0.00	5.36E-10	0.00	5.36E-10	0.00	4	0.00	4	0.00	4
2	0.00	2.84E-09	0.00	2.84E-09	0.00	16	0.02	16	0.05	8
3	0.00	6.13E-09	0.00	6.12E-09	0.00	8	0.02	8	0.06	4
4	0.00	1.38E-09	0.00	1.38E-08	0.00	8	0.02	8	0.03	4
5	0.00	1.23E-09	0.00	1.23E-09	0.00	2	0.00	2	0.01	2
6	0.00	1.52E-09	0.00	1.51E-09	0.01	256	0.12	256	0.51	256
7	0.00	2.87E-09	0.00	2.87E-09	0.01	128	0.16	128	0.72	128
8	0.00	6.92E-09	0.00	6.92E-09	0.00	64	0.11	64	0.49	64
9	0.00	3.96E-08	0.00	3.96E-08	0.46	256	0.10	256	0.44	256
10	0.00	2.66E-09	0.01	2.66E-09	0.00	8	0.03	8	0.12	8
11	0.00	3.51E-09	0.00	3.51E-09	0.03	1024	0.09	1024	0.43	1024
12	0.00	7.76E-10	0.00	7.76E-10	–	–	27.51	**32768**	–	–
13	0.02	1.64E-09	0.31	4.49E-08	–	–	45.49	**32768**	–	–
14	2.26	4.01E-09	32.21	4.01E-09	–	–	89.05	**2**	–	–
15	0.00	5.66E-08	0.03	5.66E-08	–	–	0.93	**32**	–	–
16	0.03	1.56E-08	0.40	1.56E-08	–	–	0.83	**4**	–	–
17	0.01	3.35E-09	0.12	3.35E-09	–	–	7200	**9.82E+06**	–	–
18	0.27	4.69E-07	3.38	8.05E-07	–	–	7200	**484736**	–	–
19	0.01	4.94E-08	0.09	4.94E-08	–	–	7200	**1.15E+08**	–	–
20	0.16	1.83E-06	1.98	4.49E-08	–	–	7200	**927461**	–	–
21	3.34	3.37E-06	39.01	1.37E-08	–	–	7200	**156479**	–	–
22	3.08	5.62E-06	53.41	4.59E-08	–	–	7200	**4.71E+07**	–	–
23	0.81	2.04E-06	13.11	3.24E-08	–	–	7200	**263**	–	–

Tabela 4.2: Experimentos comparativos usando as *instâncias Lavor*

executado para todas as instâncias, foi estipulado um tempo de CPU limite de duas horas. Neste caso, o algoritmo é finalizado, ao encontrar todas as soluções ou até que tenha se passado duas horas de tempo de CPU. Para as instâncias $I = \{17, 18, 19, 20, 21, 23\}$, o tempo limite foi atingido, ou seja, na Tabela 4.2, a quantidade de soluções encontradas, refere-se apenas às soluções encontradas durante esse limite de tempo. Note que para as instâncias em que todos algoritmos foram executados, o número de soluções encontradas é o mesmo. O número de soluções encontradas pelos os algoritmos é a metade dos valores expressos na Tabela 4.2, colunas # Sol, devido a simetria em torno dos átomos, descrita em (sec. 3.1.1, p. 10).

4.1.2 Experimentos Adicionais

Os experimentos apresentados nesta subseção foram feitos usando 10 instâncias de mesmo tamanho, com 65 átomos e variando o tamanho do conjunto $|E|$, que são todas as distâncias conhecidas a priori. Para estes experimentos, foi utilizado o algoritmo II, executado para encontrar todas as soluções possíveis. A Tabela 4.3 apresenta os resultados obtidos nestes experimentos.

Instâncias lavor					AII-All	
Nome	n	$\|H\|$	$\|F\|$	$\|E\|$	**CPU**	**#Sol**
lavor65_0	65	189	78	267	27.51	32768
lavor65_1	65	189	266	455	0.27	16
lavor65_2	65	189	164	353	0.08	16
lavor65_3	65	189	141	330	0.15	32
lavor65_4	65	189	265	454	0.01	2
lavor65_5	65	189	194	383	0.03	8
lavor65_6	65	189	194	383	0.10	2
lavor65_7	65	189	122	311	0.09	512
lavor65_8	65	189	466	655	0.01	128
lavor65_9	65	189	156	345	0.18	128

Tabela 4.3: Experimentos com 10 diferentes instâncias com 65 átomos.

Os gráficos 4.3(a) e 4.3(b) mostram a relação do tempo de CPU e a quantidade de soluções, com o tamanho do conjunto E, respectivamente.

4.2 Experimentos com Instâncias Reais

Os experimentos desta seção foram realizados com instâncias reais. As proteínas utilizadas para gerar essas instâncias foram obtidas do *Protein Data Bank* (PDB), que podem ser acessadas em

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

Em cada uma das estruturas de proteínas, são geradas apenas as distâncias com valores de até 5Å. A escolha por esse corte nas distâncias foi feita para simular os dados obtidos a partir de experimentos de RMN.

Para comparar as estruturas encontras pelo algoritmo com as estruturas obtidas no PDB, foi utilizado o calculo de *Root-Mean-Square Deviation* (RMSD), que de maneira geral, mede o grau de semelhança entre duas estruturas [8]. O RMSD de duas estruturas X e Y com mesmo centro geométrico pode ser definido da seguinte forma:

$$RMSD(X,Y) \quad = \quad \min_{Q} ||X - YQ||/\sqrt{n}, \tag{4.1}$$

onde Q é a matriz de rotação e $QQ^T = I$. Seja $C = Y^T X$, seja $C = USV^T$ a decomposição por valores singulares da matriz C. Como descrito em [8], $Q = UV^T$ resolve o problema de minimização em 4.1. Portanto, computacionalmente, podemos calcular os centros

geométricos a partir de duas estruturas:

$$xc = \frac{1}{n}\sum_{i=1}^{n} X(i,:), \qquad yc = \frac{1}{n}\sum_{i=1}^{n} X(i,:).$$

E então atualizamos Y:

$$\begin{aligned} Y(:,1) &= Y(:,1) - [yc(1) - xc(1)], \\ Y(:,2) &= Y(:,2) - [yc(2) - xc(2)], \\ Y(:,3) &= Y(:,3) - [yc(3) - xc(3)]. \end{aligned}$$

As duas estruturas possuem agora o mesmo centro geométrico. Então computa-se a matriz $C = Y^T X$ e sua decomposição por valor singular $C = USV^T$. Com $Q = UV^T$, o RMSD das duas estruturas é calculado da seguinte forma:

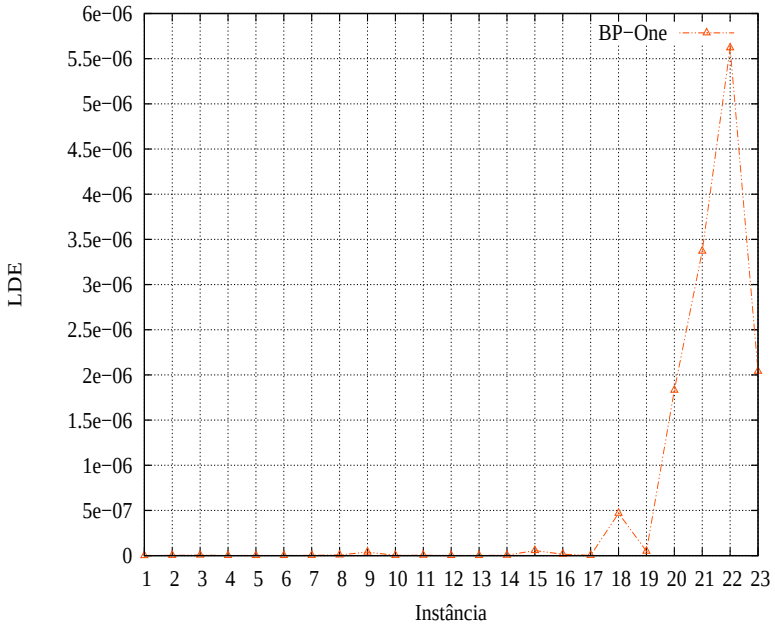
$$RMSD(X,Y) = ||X - YQ||/\sqrt{n}.$$

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

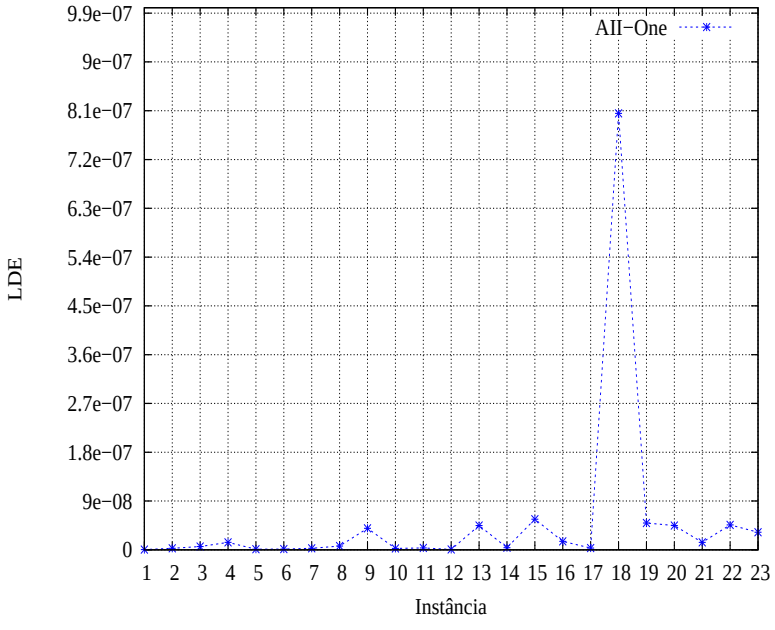
Na Tabela 4.4, são apresentados os resultados dos testes com instâncias reais usando o algoritmo II, na coluna `PDB`, mostramos qual das soluções encontras para cada instância tem maior grau de semelhança com a proteína obtida no PDB, com seu respectivo RMSD na coluna seguinte. Os demais algoritmos `BP` e o algoritmo I não foram submetidos a testes com as instâncias reais, devido ao tamanho dessas instâncias (esses algoritmos teriam problemas de memória). Para os testes com instâncias reais, também foi usado o valor de $\epsilon = 1 \times 10^{-3}$ para todas as instâncias. Para as instâncias reais, o algoritmo II foi capaz de encontrar todas as soluções possíveis, exceto para a instância `1BQV`, na qual o algoritmo não encontrou todas soluções no período limite de duas horas de tempo de CPU.

Instâncias reais						AII-One		AII-All			
I	**Proteína**	n	$\|H\|$	$\|F\|$	$\|E\|$	**CPU**	**LDE**	**CPU**	**#Sol**	PDB	RMSD
1	1BRV	57	165	157	322	0.00	1.19E-14	0.00	2	1	5.67E-17
2	1BRZ	159	471	451	922	0.65	8.30E-07	28.83	64	27	1.95E-15
3	1BRO	831	2487	2919	5406	0.09	2.76E-13	0.12	2	1	2.33E-17
4	1BSA	321	957	892	1849	0.04	7.08E-06	0.28	4	1	5.41E-14
5	1PTQ	150	444	385	829	0.05	1.09E-13	0.05	2	1	2.40E-14
6	1HOE	222	660	599	1259	0.01	1.44E-13	0.01	2	1	3.53E-14
7	1LFB	232	690	802	1492	0.01	9.74E-15	0.02	4	1	2.50E-14
8	1F39	303	903	797	1700	2.06	4.81E-07	5.92	32	9	3.65E-16
9	1PHT	249	741	707	1448	0.08	1.14E-12	0.11	8	4	8.92E-15
10	1POA	354	1056	1145	2201	0.03	5.42E-14	0.03	2	1	5.82E-14
11	1RGS	792	2370	2566	4936	1.28	5.73E-07	110.42	128	29	1.23E-14
12	1BPM	1443	4323	4980	9303	0.10	2.67E-13	0.22	2	1	5.08E-15
13	1BQV	330	984	1040	2024	0.16	4.18E-06	7200.00	1.32E+06	361611	1.86E-10
14	1BQX	231	687	638	1325	0.01	3.09E-07	0.02	8	1	3.98E-17
15	1AQR	120	354	268	622	0.08	2.87E-06	1.03	32	14	7.10E-17
16	1ACZ	324	966	1051	2017	10.60	7.27E-06	2484.24	2048	687	1.68E-14
17	1AHL	147	435	344	779	1.98	2.14E-07	57.62	64	21	1.45E-14
18	1B4C	276	822	992	1814	0.07	1.63E-06	4.33	2048	409	8.42E-16
19	1Q8O	522	1560	1669	3229	0.04	1.77E-06	0.11	32	1	6.36E-16

Tabela 4.4: Experimentos com instâncias reais

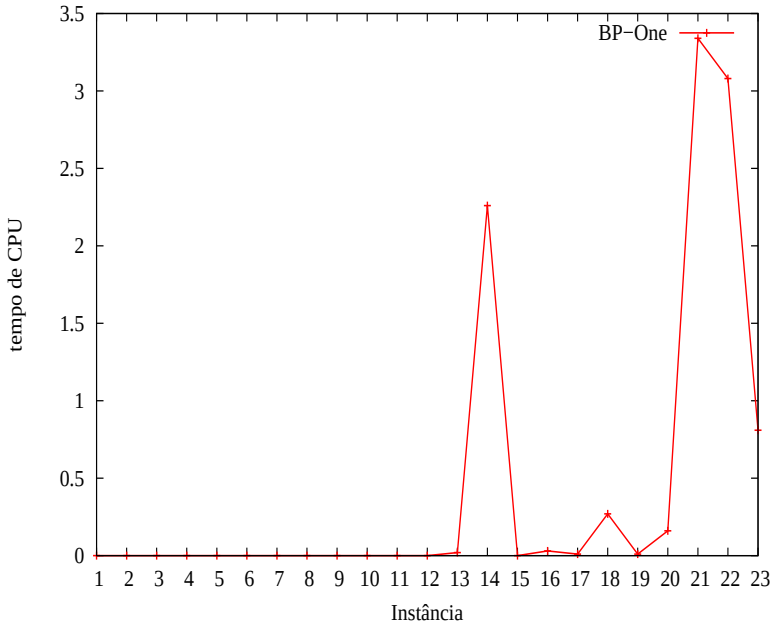


(a) `BP-One`

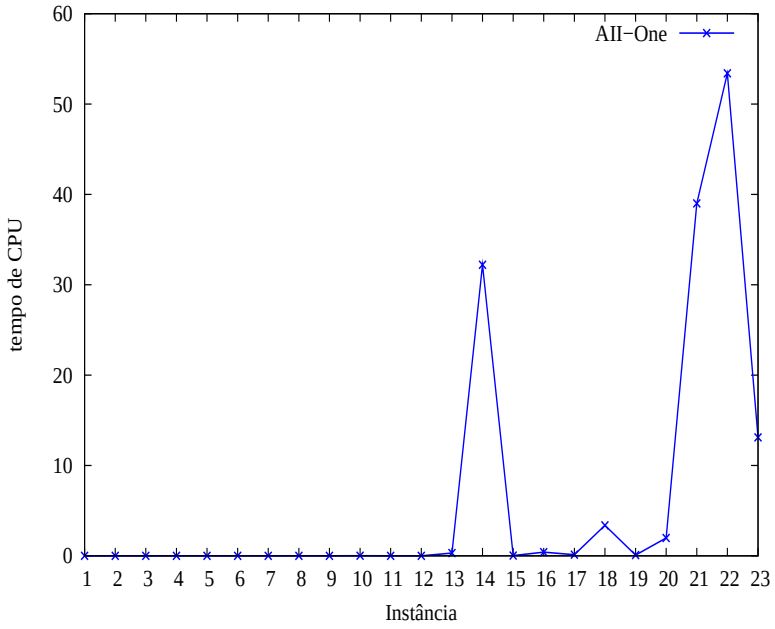


(b) *AII-One*

Figura 4.1: *Comparação do LDE entre* `BP-One` *e AII-One*

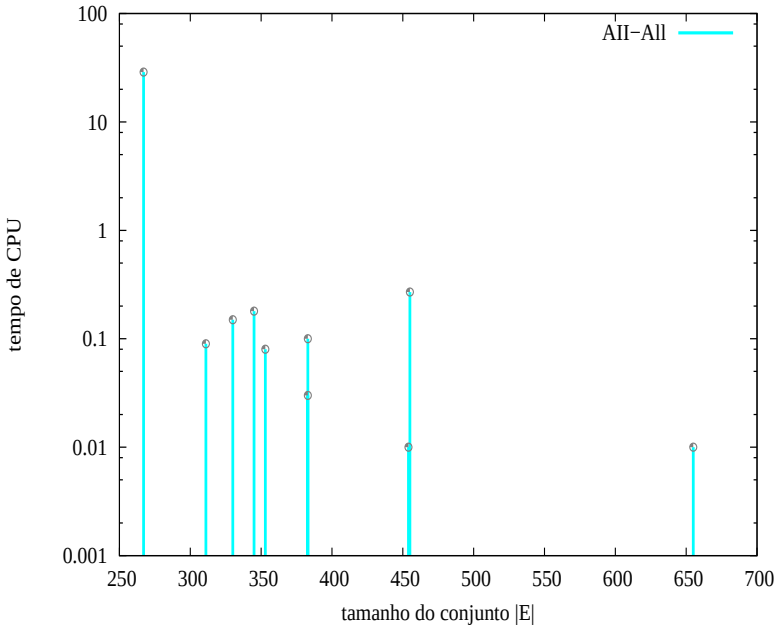


(a) `BP-One`

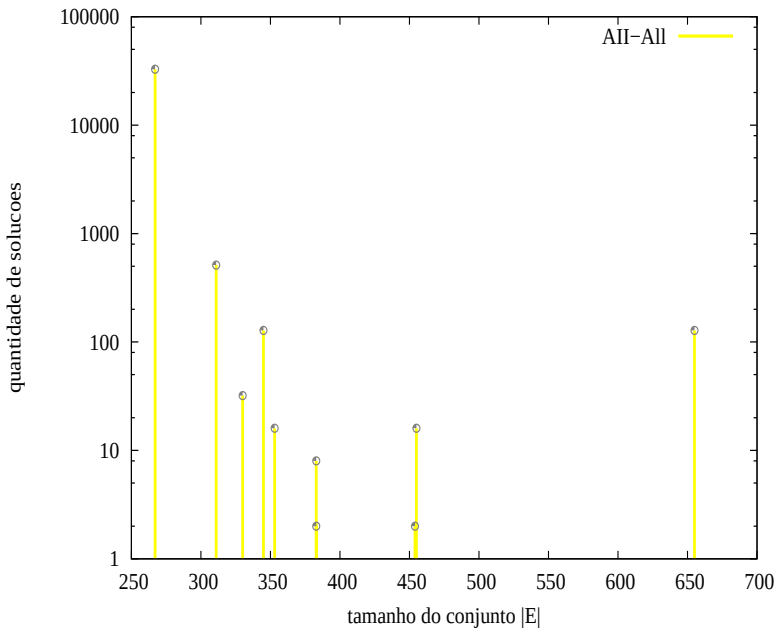


(b) *AII-One*

Figura 4.2: *Comparação do tempo de CPU entre* `BP-One` *e AII-One*



(a) *Tempo de CPU*



(b) *Quantidade de Soluções*

Figura 4.3: *Gráficos de experimentos com* 10 *diferentes instâncias de mesmo tamanho.*

Capítulo 5

Conclusões e Trabalhos Futuros

Esta dissertação abordou a formulação discreta do Problema de Geometria das Distâncias em Moléculas, considerando conjuntos esparsos de distâncias com valores exatos.

Foram propostos algoritmos e softwares de visualização. Dentre os dois algoritmos propostos, o algoritmo II apresentou melhor resultado quando analisado a eficiência em gerenciar memória, comparado com outro algoritmo da literatura. O algoritmo I teve problemas em gerenciar a memória, assim como o algoritmo da literatura. Contudo, o algoritmo II teve um desempenho inferior, quando comparamos o tempo de CPU com o algoritmo da literatura. Com o gerenciamento eficiente de memória do algoritmo II, foi possível a realização de experimentos com instâncias reais para este algoritmo.

Como possibilidades de trabalhos futuros, podemos considerar o problema com todos os átomos de uma proteína, e não apenas a cadeia principal. Além disso, considerar ainda as distâncias inexatas, tendo em vista que, na prática, os experimentos NMR determinam apenas limites inferiores e superiores para as distâncias entre átomos. Trabalhos preliminares nos permitiram conjecturar que a paralelização do algoritmo I pode vir a ter resultados melhores que o algoritmo II e o algoritmo da literatura. Portanto, trabalhos futuros neste sentido apontam para um caminho promissor.

Referências

[1] 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.

[2] 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.

[3] Carvalho, R. S. Extensão do algoritmo de construção geométrica para a determinação de estruturas moleculares. Dissertação de Mestrado, UFRJ/IM/NCE, 2005.

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

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

[6] 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.

[7] Floudas, C. A., Pardalos, P. M., Eds. *Optimization in computational chemistry and molecular biology.* Kluwer Academic Publishers, Dordrecht, 2000.

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

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

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

[27] 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.

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

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

[30] 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.

[31] 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.