CÁLCULO DE ESTRUTURAS DE PROTEÍNAS

Warley Gramacho da Silva
Curso de Ciência da Computação - Universidade Federal do Tocantins
Av. NS 15, S/N – ALCNO 14 - Bloco 02 – Sala 21, CEP: 77020-210, Palmas - TO
wgramacho@uft.edu.br

Carlile Lavor
Instituto de Matemática, Estatística e Computação Científica - Universidade Estadual de Campinas
Rua Sergio Buarque de Holanda, 651, Caixa Postal: 6065, CEP: 13083-859, Campinas - SP
clavor@ime.unicamp.br

Luiz Satoru Ochi
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 3º andar, São Domingos - Niterói – RJ – CEP: 24210-240
satoru@ic.uff.br

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 Ressonância Magnética Nuclear (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. O trabalho tem como objetivo apresentar testes computacionais de um novo algoritmo que resolve o Problema Discreto de Geometria das Distâncias em Moléculas. O algoritmo proposto usa a técnica de busca em árvore binária em profundidade, sem procedimentos de recursividade. Os testes foram realizados com instâncias artificiais e reais associadas às distâncias de átomos de cadeias principais de proteínas. O algoritmo proposto é uma adaptação do algoritmo existente *Branch-and-Prune*, permitindo resolver instâncias maiores do problema, com complexidade de tempo maior e complexidade de memória menor.

PALAVRAS CHAVE: Problema de Geometria das Distâncias em Moléculas, Estrutura tridimensional de proteínas, Biologia computacional, Otimização combinatória.

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 Nuclear Magnetic Resonance (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, 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. The work aims to tests a new algorithm that solves the Discretizable Molecular Distance Geometry Problem. The proposed algorithm uses the technique of finding a binary tree in depth, without procedures of recursion. The tests were with real and artificial instances associated with distances of atoms of a main backbone (a chain of protein). The proposed algorithm is an adaptation of existing Branch-and-Prune algorithm, allowing more instances solve the problem, with more complexity of time and less complexity of memory.

KEYWORDS: Discretizable Molecular Distance Geometry Problem, Three-dimensional structure of proteins, Computational biology, Combinatorial optimization.

1. Introdução

As proteínas são moléculas fundamentais dos sistemas biológicos, sendo constituídas por uma cadeia linear de aminoácidos (CREIGHTON, 1993). 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 (BRÜNGER e NILGES, 1993), e teoricamente, através da minimização da energia potencial ou por simulação de dinâmica molecular (FLOUDAS e PARDALOS, 2000). Entretanto, nem todas as proteínas podem ter suas conformações tridimensionais obtidas experimentalmente, i. e., por cristalografia de raios X ou RMN. O problema tratado no trabalho é o de determinar a estrutura tridimensional da cadeia principal de uma proteína 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 (BISWAS *et al.*, 2006), reconhecimento de imagens (KLOCK e BUHMANN, 1997), 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 (DONG e WU, 2002). Caso contrário, o problema passa a ser NP-difícil (SAXE, 1979). Recentemente, Lavor, Liberti e Maculan (LAVOR e*t al.*, 2006) 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 Distâncias em Molécula (PDGDM).

Existem várias abordagens para o PGDM, por exemplo, o algoritmo *EMDED* de Crippen e Havel (CRIPPEN e HAVEL, 1988**)**, (HAVEL, 1996), a estratégia de redução de grafo de Hendrickson (HENDRICKSON**,** 1992), (HENDRICKSON**,** 1995), o algoritmo DGSOL de Moré e Wu (MORÉ e WU, 1996), (MORÉ e WU, 1997), (MORÉ e WU, 1999), o método de perturbação estocástica de Zou, Bird e Schnabel (ZOU *et al.***,** 1997), o método de escala multidimensional de Trosset (TROSSET, 1998), o algoritmo VNS de Liberti, Lavor, e Maculan (LIBERTI *et al.*, 2005),(LAVOR *et al*., 2006), o *geometric build-up* algoritmo de Wu e Wu (Wu e Wu, 2007), o algoritmo *Branch-and-Prune* (BP) de Lavor, Liberti e Maculan (LAVOR e*t al.*, 2006), (LIBERTI *et al*., 2008), etc. *Surveys* sobre o problema são encontrados em (BRÜNGER e NILGES, 1993), (CRIPPEN e HAVEL, 1988), (HAVEL, 1991), (LAVOR *et al.*, 2007).

O artigo está organizado da seguinte forma: a próxima seção traz uma descrição do problema; na seção 3 é descrita a versão do algoritmo proposto; na seção 4 encontram-se os resultados computacionais e, por fim, na seção 5, são apresentadas as conclusões e algumas sugestões de trabalhos futuros.

2. O Problema de Geometria das Distâncias em Moléculas – PGDM

O Problema de Geometria das Distâncias em Moléculas (PGDM) esta associado à determinação da estrutura tridimensional de uma molécula. Este problema pode ser formulado da seguinte forma: encontre as posições, $x_1, \dots, x_n$ dos átomos na molécula, tal que

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

onde S é um subconjunto dos pares de átomos cujas distâncias $d_{i,j}$ são conhecidas a priori e ||.|| é a norma Euclidiana.

O PGDM pode ser formulado como um problema de otimização contínua, onde a função objetivo é dada por

$$f(x_1, \dots, x_n) = \sum_{(i,j) \in \mathrm{S}} (|| x_i - x_j ||^2 - d^2{}_{i,j})^2. \tag{2.2}$$

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 (HENDRICKSON**,** 1995).

2.1. 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 formulação discreta foi proposta em (LAVOR e*t al.*, 2006), introduzindo assim, uma subclasse de problemas do PGDM, chamada de Problema Discreto de Geometria das Distâncias em Moléculas (PDGDM). Também em (LAVOR e*t al.*, 2006), foi demonstrado que o PDGDM é NP-difícil.

Considere uma molécula como sendo uma seqüência de n átomos, onde os comprimentos de ligações covalentes 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 são 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).

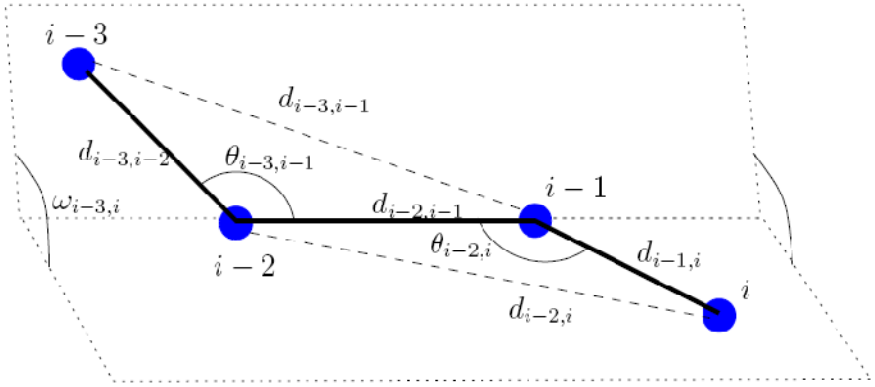

Figura 2.1: Definições de comprimento de ligações, ângulos de ligações e ângulos de torção.

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 (CREIGHTON, 1993), (SCHLICK, 2002). A Hipótese 2 é igualmente aplicável às proteínas, dado que não se conhece proteína com ângulos de ligações covalentes com valor exato de π.

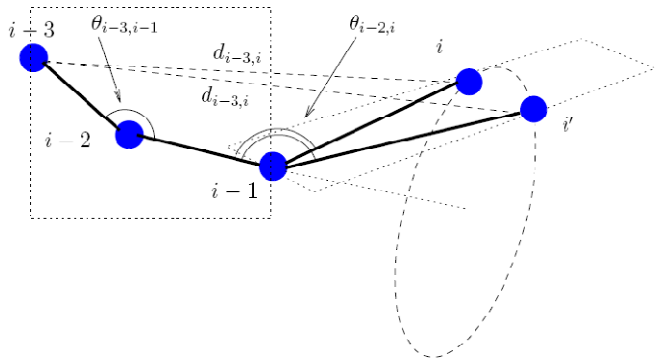

Figura 2.2: No PDGDM, o átomo *i* pode estar somente em duas posições (*i* e *i'*) para ser "viável" com a distância $d_{i-3,i}$.

A intuição da formulação discreta é que o i-ésimo átomo reside na intersecção de três esferas centradas nos átomos $i-3$, $i-2$ e $i-1$, de raios $d_{i-3,i}$, $d_{i-2,i}$ e $d_{i-1,i}$, 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 $d_{1,2}$,..., $d_{n-1,n}$, ângulos de ligações $\theta_{1,3}$,..., $\theta_{n-2,n}$, e ângulos de torção $\omega_{1,4}$,..., $\omega_{n-3,n}$ 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 (PHILLIPS *et al.*, 1996):

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

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 $i = 4, \ldots, n$.

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 $i = 4, \ldots, n$, 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 (LAVOR e*t al.,* 2006).

Usando os comprimentos de ligações $d_{1,2}$, $d_{2,3}$ e o ângulo de ligação$\theta_{1,3}$, podemos calcular as matrizes B_1, 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: $\sin = \pm\sqrt{1 - cos^2\omega_{1,4}}$. 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 $\omega_{1,4}$,... , $\omega_{n-3,n}$, 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 $x = (x_i, \ldots,\ x_n) \in \mathbb{R}^3$.

3. Algoritmo Proposto para o PDGDM

O algoritmo aqui proposto é um algoritmo exato, cujas soluções são mínimos globais. A sua estratégia se baseia 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.

Seja $F = \{(j, i) \mid i - j > 4\}$ o conjunto de pares de átomos no qual a distância entre (i, j) 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 $(i, j) \in F$. Para isso, verifica-se

$$(||\, x_j - x_i \,||^2 - d^2{}_{j,i}\,)^2 < \epsilon, \tag{3.1}$$

onde $\epsilon > 0$ 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.

O algoritmo proposto neste trabalho, chamado de Algoritmo I é uma adaptação do algoritmo BP que resolve PDGD, no sentido de substituir a implementação recursiva pela não

recursiva, usando estrutura de dados do tipo pilha, esse algoritmo explora a estrutura de árvore binária 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.

No pseudo-código apresentado para o algoritmo proposto, P é uma representação de pilha (que simula a exploração da árvore). P é inicializado com P={1, 2, 3, 4 }, que representa os quatro primeiros nós da árvore, em que os três primeiros átomos podem ser fixados nas posições x_1, x_2 e x_3. 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, (LAVOR *et al.*, 2006).

O algoritmo I, dado a seguir, 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 retroceder na árvore.

Algoritmo I

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

```
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 ← 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 ← ESQUERDA;
 9:    P[ topo da pilha ].subarvore ← DIREITA;
10:    P ← P ∪ {v'};
11: senão
12:    cria v(resentação do nó direito da árvore), guarde Q_i e x_i;
13:    v.subarvore ← ESQUERDA;
14:    P[ topo da pilha ].subarvore ‹  PODA;
15:    P ‹  P ∪ {v};
16: fim se
17: retorna P;
```

```
Desempilha ( P, k )
1: repita
2:    P ← P− {P[ topo da pilha ]};
3:    t ← P[ topo da pilha ];
4:    enquanto P < F[k − 1] faça
5:       k ← k − 1;
6:    fim enquanto
7: até t.subarvore = PODA
8: retorna P;
```

4. Resultados Computacionais

Nessa seção, serão apresentados os experimentos computacionais para analisar o algoritmo proposto. Primeiramente, os experimentos foram realizados com instâncias artificiais, comparando o algoritmo proposto com outro algoritmo já existente, denominado *Branch and Prune* (BP) (LAVOR e*t al.*, 2006),(LIBERTI *et al*., 2008), 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.

4.1. Experimentos com Instâncias Artificiais

Esta subseção apresenta os experimentos com instâncias artificiais, chamadas instância Lavor (*lavor-instance*), que são baseadas em um modelo proposto por (PHILLIPS *et al.*, 1996), 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 identificador da instância.

Foram utilizadas 23 diferentes instâncias Lavor, 13 variando de tamanho n=10,...,70 (“instâncias pequenas”) e 10 de tamanho n=100*i*, i=1,...,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 a cardinalidade dos conjuntos |E|, |H| e |F|, respectivamente, onde E é o conjunto de todas as distâncias conhecidas a priori, que é particionado nos H e F, sendo H o conjunto de pares de átomos separados por até 3 ligações consecutivas, H = $\{(i , i + 1) | 1 \leq i \leq n - 1\} \cup \{(i , i + 2) | 1 \leq i \leq n - 2\} \cup \{(i , i + 3) | 1 \leq i \leq n - 3\}$ e 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.

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

Tabela 4.1: Instâncias Lavor

O algoritmo BP explora a estrutura do problema de forma similar aos algoritmos propostos neste artigo. 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 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 $1x10^{-3}$, o mesmo valor utilizado para o BP em (LIBERTI *et al*., 2008). A Tabela 4.2 apresenta os resultados referentes a quatro métodos: BP executado até encontrar a primeira solução (BP-One), BP executado até encontrar todas as soluções (BP-All), algoritmo I executado até encontrar a primeira solução (AI-One) e algoritmo I executado até encontrar todas as soluções (AI-All). Para o BP-One e o AI-One, apresentamos o tempo de CPU (em segundos), assim como o *Largest Distance Error* (LDE), definido como

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

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

	BP-One		AI-One		BP-All		AI-All	
I	CPU	LDE	CPU	LDE	CPU	#Sol	CPU	#Sol
1	0.00	5.36E-10	0.00	5.36E-10	0.00	4	0.00	4
2	0.00	2.84E-09	0.00	2.84E-09	0.00	8	0.02	8
3	0.00	6.13E-09	0.00	6.12E-09	0.00	4	0.02	4
4	0.00	1.38E-09	0.00	1.38E-08	0.00	4	0.02	4
5	0.00	1.23E-09	0.00	1.23E-09	0.00	2	0.00	2
6	0.00	1.52E-09	0.00	1.51E-09	0.01	256	0.12	256
7	0.00	2.87E-09	0.00	2.87E-09	0.01	128	0.16	128
8	0.00	6.92E-09	0.00	6.92E-09	0.00	64	0.11	64
9	0.00	3.96E-08	0.00	3.96E-08	0.46	256	0.10	256
10	0.00	2.66E-09	0.01	2.66E-09	0.00	8	0.03	8
11	0.00	3.51E-09	0.00	3.51E-09	0.03	1024	0.09	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	-	-	45.59	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

Para os algoritmos que foram executados até encontrar uma solução (BP-One e AI-One), pela Tabela 4.2, podemos observar uma pequena vantagem do algoritmo AI-One, no que se refere ao LDE. A diferença de tempo entre os algoritmos também pode ser vista na Tabela 4.2, onde mostra 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 AI-One usa estruturas auxiliares para gerenciar a limitação de memória.

Para os algoritmos que foram executados até encontrar todas as soluções (BP-All e AI-All), observamos que apenas o algoritmo AI-All conseguiu ser executado para todas as instâncias. O algoritmo BP-All só foi executado 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 mostra a Tabela 4.2. Para instâncias maiores que estas, o algoritmo BP teve problemas de falta de memória. No entanto, para o algoritmo AI-All, que foi 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 para encontrar todas soluções, 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, mencionado na seção anterior e descrita em (LAVOR *et al.*, 2006).

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 cálculo de *Root-Mean-Square Deviation* (RMSD), que de maneira geral, mede o grau de semelhança entre duas estruturas (GOLUB e VAN LOAN, 1989). O RMSD de duas estruturas X e Y com mesmo centro geométrico pode ser definido da seguinte forma:

$$RMSD(X,Y) = min_Q \left|\left|X - YQ\right|\right| / \sqrt{n}\,, \tag{4.1}$$

onde Q é a matriz de rotação e $QQ^T = \mathrm{I}$. Seja $C = Y^T\mathrm{X}$ e $C = USV^T$ a decomposição por valores singulares da matriz C. Como descrito em (GOLUB e VAN LOAN, 1989), $Q = UV^T$ resolve o problema de minimização em (4.1). Portanto, computacionalmente, podemos calcular os centros geométricos a partir de duas estruturas:

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

que são usadas para atualizar 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 $C = Y^T\mathrm{X}$ e sua decomposição por valor singular $C = USV^T$. Com $Q = UV^T$, o RMSD das duas estruturas é calculado da seguinte forma:

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

Na Tabela 4.3, são apresentados os resultados dos testes com instâncias reais usando o algoritmo I. Na coluna PDB, mostramos qual das soluções encontradas para cada instância tem maior grau de semelhança com a proteína obtida no PDB, com seu respectivo RMSD na coluna seguinte. O algoritmo BP não foi submetido a testes com as instâncias reais, devido ao tamanho dessas instâncias (esse algoritmo teria problemas de gerência de memória). Para os testes com instâncias reais, também foi usado o valor de $\epsilon = 1x10^{-3}$ para todas as instâncias. Para as instâncias reais, o algoritmo I 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						AI-One		AI-All			
I	Proteína	N	|E|	|H|	|F|	CPU	LDE	CPU	#Sol	PDB	RMSD
1	1BRV	57	322	165	157	0.00	1.19E-14	0.00	2	1	5.67E-17
2	1BRZ	159	922	471	451	0.65	8.30E-07	28.83	64	27	1.95E-15
3	1BRO	831	5406	2487	2919	0.09	2.76E-13	0.12	2	1	2.33E-17
4	1BSA	321	1849	957	892	0.04	7.08E-06	0.28	4	1	5.41E-14
5	1PTQ	150	829	444	385	0.05	1.09E-13	0.05	2	1	2.40E-14
6	1HOE	222	1259	660	599	0.01	1.44E-13	0.01	2	1	3.53E-14
7	1LFB	232	1492	690	802	0.01	9.74E-15	0.02	4	1	2.50E-14
8	1F39	303	1700	903	797	2.06	4.81E-07	5.92	32	9	3.65E-16
9	1PHT	249	1448	741	707	0.08	1.14E-12	0.11	8	4	8.92E-15
10	1POA	354	2201	1056	1145	0.03	5.42E-14	0.03	2	1	5.82E-14
11	1RGS	792	4936	2370	2566	1.28	5.73E-07	110.42	128	29	1.23E-14
12	1BPM	1443	9303	4323	4980	0.1	2.67E-13	0.22	2	1	5.08E-15

13	1BQV	330	2024	984	1040	0.16	4.18E-06	7200.0	1E+06	361611	1.86E-10
14	1BQX	231	1325	687	638	0.01	3.09E-07	0.02	8	1	3.98E-17
15	1AQR	120	622	354	268	0.08	2.87E-06	1.03	32	14	7.10E-17
16	1ACZ	324	2017	966	1051	10.6	7.27E-06	2484.2	2048	687	1.68E-14
17	1AHL	147	779	435	344	1.98	2.14E-07	57.62	64	21	1.45E-14
18	1B4C	276	1814	822	922	0.07	1.63E-06	4.33	2048	409	8.42E-16
19	1Q80	522	3229	1560	1669	0.04	1.77E-06	0.11	32	1	6.36E-16

Tabela 4.3: Experimentos com instâncias reais

5. Conclusões e Trabalhos Futuros

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

Foi proposto um novo algoritmo, que apresentou melhor resultado quando analisado a eficiência em gerenciar memória, comparado com outro algoritmo da literatura. Com o gerenciamento eficiente de memória, foi possível a realização de experimentos com instâncias reais para este algoritmo.

A técnica apresentada neste trabalho pode enriquecer algumas estruturas tridimensionais obtidas experimentalmente por RMN, pois existem casos de proteínas que somente puderam ter parte de sua conformação espacial conhecida.

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 de RMN determinam apenas limites inferiores e superiores para as distâncias entre átomos.

Agradecimentos

Além do apoio financeiro da FAPERJ, da FAPESP e do CNPQ, gostaríamos de agradecer os comentários e sugestões dos revisores do artigo.

Referências

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

Brünger, A. T. e Nilges, M. (1993), Computational challenges for macromolecular structure determination by X-ray crystallography and solution NMR-spectroscopy, *Quarterly Reviews of Biophysics,* 26, 1, 49-125.

Creighton, T. E, *Proteins: Structures and Molecular Properties*, 2 ed. W. H. Freeman, New York, 1993.

Crippen, G. e Havel, T. Distance Geometry and Molecular Conformation. Research Studies Press Ltd., New York, 1988.

Dong, Q. e Wu, Z. (2002), A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization*, 22, 1-4, 65–375.

Floudas, C. A. e Pardalos, P. M. Optimization in computational chemistry and molecular biology. *Nonconvex Optimization and its Applications*, Kluwer Academic Publishers, Dordrecht, 2000.

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

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

Havel, T. F. Distance geometry. *Encyclopedia of Nuclear Magnetic Resonance*, J. Wiley & Sons, p. 1701–1710, 1996.

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

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

Klock, H. e Buhmann, J. M. Multidimensional scaling by deterministic annealing. *Atas do First International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 245–260.

Lavor, C. On generating instances for the molecular distance geometry problem, L. Liberti and N. Maculan, *Global Optimization: from Theory to Implementation*, ed. Springer, Berlin, 2006.

Lavor, C., Liberti, L. e Maculan, N. An overview of distinct approaches for the molecular distance geometry problem. *Encyclopedia of Optimization 2 ed.*, P. Pardalos e C. Floudas, New York, 2007.

Lavor, C., Liberti, L. e Maculan, N. The discretizable molecular distance geometry problem, http://arxiv.org/abs/q-bio.BM/0608012, (2006).

Lavor, C., Liberti, L. e Maculan, N. Computational Experience With The Molecular Distance Geometry Problem, J. Pintér, *Global Optimization: Scientific and Engineering Case Studies*, ed. Springer, New York, 2006.

Liberti, L., Lavor, C. e Maculan, N. (2005), Double vns for the molecular distance geometry problem, in Proceedings of Mini Euro Conference Variable Neighborhood Search, p 23-25.

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

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

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

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

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

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

Saxe, J. B. Embeddability of weighted graphs in k-space is strongly NP-hard. *Atas do 17th Allerton Conference in Communications, Control and Computing, 480–489, 1979.*

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

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

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

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