Documentos

Dezembro, 2004 46

ISSN 1677-9274

Um Modelo Orientado a Objetos para o Cálculo da Estimativa da Área Acessível a Solvente

Embrapa

ISSN 1677-9274
Dezembro, 2004

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 46

Um Modelo Orientado a Objetos para o Cálculo da Estimativa da Área Acessível a Solvente

Maria Fernanda Moura
Roberto Hiroshi Higa
Michel Eduardo Beleza Yamagishi

Campinas, SP
2004

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. André Tosello, 209
Cidade Universitária "Zeferino Vaz" Barão Geraldo
Caixa Postal 6041
13083-970 - Campinas, SP
Telefone (19) 3789-5743 - Fax (19) 3289-9594
URL: http://www.cnptia.embrapa.br
e-mail: sac@cnptia.embrapa.br

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel Fugisawa Souza*
Editoração eletrônica: *Área de Comunicação e Negócios (ACN)*

1ª. Edição on-line - 2004

Moura, Maria Fernanda.
Um modelo orientado a objetos para o cálculo da estimativa da área acessível a solvente / Maria Fernanda Moura, Roberto Hiroshi Higa, Michel Eduardo Beleza Yamagishi. – Campinas : Embrapa Informática Agropecuária, 2004.
25 p. : il. (Documentos / Embrapa Informática Agropecuária ; 46).

ISSN 1677-9274

1. Bioinformática. 2. Área acessível a solvente. 3. Modelo orientado a objeto. 4. Análise da estrutura de proteínas. 5. Grid cúbico. I. Higa, Roberto Hiroshi. II.Yamagishi, Michel Eduardo Beleza. III. Título. IV. Série.

CDD – 570-285 (21st ed.)

Autores

Maria Fernanda Moura
M.Sc. em Engenharia Elétrica, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970- Campinas, SP
e-mail: fernanda@cnptia.embrapa.br

Roberto Hiroshi Higa
M.Sc. em Engenharia Elétrica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970- Campinas, SP
e-mail: roberto@cnptia.embrapa.br

Michel Eduardo Beleza Yamagishi
Ph.D. em Matemática Aplicada, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970- Campinas, SP
e-mail: michel@cnptia.embrapa.br

Apresentação

Atualmente, a Embrapa Informática Agropecuária adota uma política de produção de código fonte próprio para o Laboratório de Bioinformática Estrutural, onde alguns dos módulos do Sting Millenium Switch - SMS, quer cedidos ou comprados de terceiros, devem ser reimplementados a partir de uma solução própria, de modo a diminuir-lhe as dependências externas e poder utilizar os seus resultados para fins comerciais.

Os programas desenvolvidos sob essa política nem sempre apresentam algoritmos ou métodos inéditos ou melhorados. Porém consistem em soluções criativas para métodos já bastante explorados e divulgados, de modo que os resultados anteriormente obtidos permaneçam inalterados, mantendo ou reduzindo o tempo de processamento necessário.

O módulo *Protein Dossier* do SMS, apresenta um conjunto de propriedades físico-químicas das proteínas, entre as quais a área acessível a solvente (ASA - Accessible Surface Area). O método utilizado para o cálculo da ASA disponível no software é o Double Cubic Lattice Method - DCLM, que foi proposto por Eisenhaber, Lijnzaad, Argos, Sander e Scharf em 1995. Porém, a implementação utilizada é um código de acesso restrito ao meio acadêmico.

Neste trabalho é apresentada uma descrição do DCLM e uma possível solução de projeto orientada a objetos. Esta solução será codificada em C + +, provendo uma nova implementação para o método de propriedade da Embrapa.

José Gilberto Jardine
Chefe-Geral

Sumário

Um Modelo Orientado a Objetos para o Cálculo da Estimativa da Área Acessível a Solvente

Maria Fernanda Moura
Roberto Hiroshi Higa
Michel Eduardo Beleza Yamagishi

Introdução

O Sting Millennium Suite - SMS compreende um conjunto de programas para análise da estrutura de proteínas, que pode ser acessado pela *web*. Dentre suas funcionalidades, permite a visualização simultânea da estrutura e da seqüência da proteína e, por meio do módulo Protein Dossier (Protein..., 2004), provê um conjunto de propriedades físico-químicas mapeadas para cada resíduo de aminoácido (Higa et al., 2002b). Dentre essas propriedades, um subconjunto refere-se especificamente às propriedades da superfície da molécula protéica, entre as quais, a área acessível a solvente (ASA - Accessible Surface Area).

As interações entre moléculas são fortemente dependentes de suas superfícies; em especial, a superfície acessível a solvente permite inferir sobre interações entre moléculas e superfícies, auxiliando na localização de sítios escondidos *(binding sites*) e outros (Nicolau et al., 2003). A idéia básica para o cálculo da área acessível a solvente, descrita por Conolly (1983), consiste em fazer as seguintes considerações:

- a proteína como um conjunto de átomos $[a_1,a_2,\ldots,a_n]$;
- cada átomo (a_i) como uma aproximação esférica, dada pelo seu centro (x,y,z) e seu raio (r_i). Logo, considera-se a molécula uma nuvem de esferas; e
- existe uma esfera imaginária, denominada *probe*, com raio próximo ao de uma molécula de solvente. No caso, se o solvente for a água o raio será de 1.4 Å.

Para estimar a área acessível a solvente - ASA, gira-se o *probe* sobre a superfície da estrutura tridimensional da molécula da proteína; tomando-se como área de interesse aquela delimitada pelo raio do *probe*. Essa área é formada por um conjunto de envelopes: um que recobre toda a estrutura tridimensional da molécula e outros que recobrem áreas internas à molécula, desde que com alguma abertura externa.

A estimativa mais utilizada, para aproximar o cálculo dessa área, considera o raio de cada átomo e o raio de cada molécula de *probe* somados como uma esfera única, para depois somar os pedaços de áreas de cada esfera que se encontram na superfície externa da proteína. Os pedaços das esferas são estimados pelo número de pontos que se encontram na superfície; considerados como as contribuições de cada esfera à área total. Logo, os problemas resumem-se a conhecer os raios de cada átomo e do *probe*, utilizar um método que permita gerar os pontos de cada esfera de uma maneira uniforme sobre suas superfícies, e, então calcular a área e o volume dados pelas contribuições dos pontos na superfície externa à proteína.

Embora pareça um problema simples, deve-se notar que muitas proteínas são compostas por milhares de átomos, o que significa que são necessárias milhares de comparações entre pontos, para saber quais se encontram na superfície externa da proteína, além de gerar milhões de pontos para cobrirem as superfícies esféricas de cada átomo. Para diminuir o número de comparações entre átomos, existem várias soluções na literatura (Nicolau et al., 2003) que passam pela divisão da molécula em setores – modelos geométricos de *lattice*, e várias soluções com algoritmos paralelizados – por exemplo, o proposto por Futamura et al. (2002). Outro problema não-trivial é gerar pontos uniformemente distribuídos em uma superfície esférica. Se a distribuição dos pontos não for muito homogênea, a estimativa da contribuição de cada átomo para a ASA estará negativamente comprometida.

A estimativa da ASA no SMS é calculada pelo *Double Cubic Lattice Method* (Eisenhaber et al., 1995) que resolve de um modo bastante satisfatório os problemas acima relacionados. O SMS utiliza o código fonte cedido pelos autores do método – com a restrição de uso em ambiente acadêmico ou para pesquisa sem fins comerciais e, de acordo com a política de produção de código fonte próprio para Laboratório de Bioinformática Estrutural, esse método deve ser implementado para o SMS a partir de uma solução própria. Assim, o objetivo deste trabalho é descrever a solução de projeto orientado a objetos, a ser implementada em C + +, do *Double Cubic Lattice Method*, proposto por Eisenhaber et al. (1995), e que está em desenvolvimento no Laboratório de Bioinformática Estrutural da Embrapa Informática Agropecuária.

Material e Métodos

Neste item faz-se uma descrição mais detalhada da obtenção da estimativa da área acessível a solvente (ASA), do método utilizado para calculá-la e a descrição dos dados que devem ser utilizados para o teste funcional do programa.

Estimativa da ASA

A primeira questão é a definição da área acessível a solvente de uma molécula. A Fig. 1 ilustra a superfície acessível a solvente e a superfície molecular (ProShape Software, 2004). Para melhor visualizar os conceitos na figura, primeiro deve-se considerar que cada átomo (a_i), em cada molécula, corresponde a uma aproximação esférica, a partir de seu ponto central (x,y,z) e seu raio de Van der Waals (r_i). A superfície molecular, também conhecida por superfície de Van der Waals, é a definida pela união das superfícies das esferas que representam todos os seus átomos, que na figura é a mostrada em rosa (*Molecular surface*). A superfície acessível a solvente é aquela gerada pelo raio de uma molécula de solvente (em geral água), rolando sobre a superfície de Van der Waals da molécula (em verde na figura, *Accessible surface*). Geralmente considera-se o raio da molécula solvente como 1.4 Angstrons e denomina-se essa molécula de *probe*.

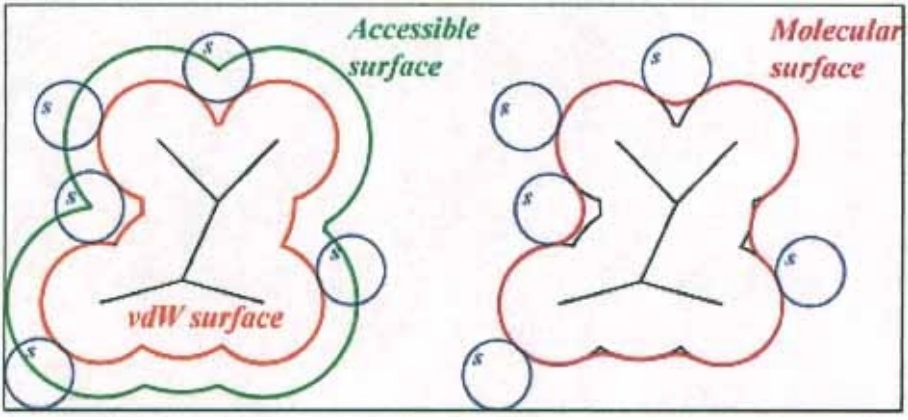

FIG. 1. Superfície acessível a solvente e superfície molecular.
Fonte: ProShape Software (2004).

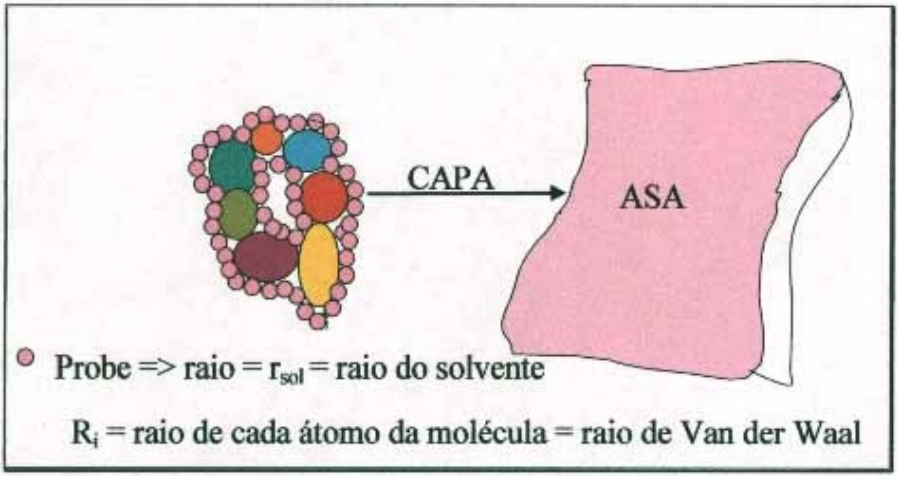

FIG. 2. Aproximação para a superfície acessível a solvente.

Deve-se considerar também as áreas internas à molécula no cálculo de sua área acessível a solvente, desde que essas áreas tenham comunicação com a parte externa, como na Fig. 2. De uma forma mais grosseira, a partir do desenho de uma molécula completamente fictícia formada pelas esferas coloridas (aproximações de cada átomo). Os dois colares de esferas rosas correspondem ao giro do *probe* (esfera rosa) em torno da molécula. A estimativa da área acessível a solvente (ASA) é a capa (envelope) em rosa representado do lado direito da figura. Cada raio de cada átomo corresponde ao raio de *Van der Waals* do átomo, considerando n átomos por molécula, cada um desses raios será representado por r_i, $i = 1,\ldots,n$ e a cada qual deve-se somar o raio do *probe* (r_{sol} = 1.4 Angstrons).

Para simplificar a forma de cálculo da superfície assume-se que cada átomo é uma esfera, assim como o *probe*. Gira-se o *probe* em torno de cada átomo formando uma grande esfera de raio $r_i + r_{sol}$. Despreza-se todos os pontos da superfície dessa esfera que caem em regiões de intersecção dessa esfera com as esferas vizinhas (também formadas por $r_j + r_{sol}$, r_j diferente de r_i) – ver ilustração na Fig. 3. Logo, para cada esfera de raio $r_i + r_{sol}$, temos um certo número de pontos que pertence à superfície acessível a solvente, que será denominada $m_{acc(i)}$. A estimativa total da ASA corresponde à soma das contribuições de cada esfera (de raio $r_i + r_{sol}$,) à área total, da seguinte forma:

$$\hat{A}SA = 4 \sum_{i}^{n} (r_i^2 (m_{acci}/m)),$$ *onde,* ***i varia de 1 a n***

n *é o número total de átomos da molécula*

m *é o número de pontos gerados por esfera.*

Equação 1. Estimativa da ASA.

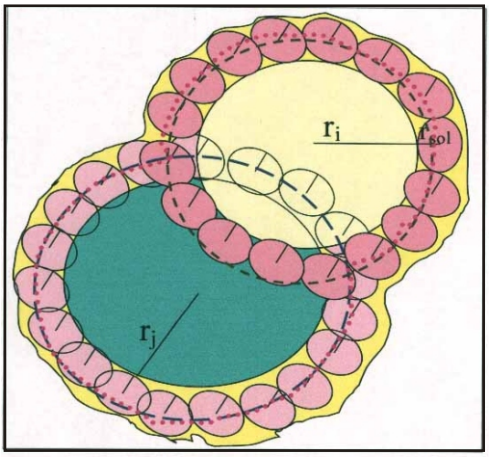

FIG. 3. Dois átomos vizinhos de raios r_i e r_j mais o *probe* e a aproximação esférica.

Deve-se lembrar que:

- o perímetro da esfera é $2\pi r$;
- a área da esfera é $4\pi r;^2$
- o volume da esfera é $8\pi r3$;
- o número de pontos gerados para cada esfera é fixo e denominado m^1;
- o número de pontos acessíveis, não oclusos, para cada esfera i é $m_{acc(i)}$; e,
- a contribuição de cada esfera, em proporção de pontos para a superfície acessível a solvente ou para o volume acessível a solvente é $m_{acc(i)}/m$.

A variável aleatória $X_i = m_{acc(i)}$ tem a seguinte esperança[2]:

$$\hat{A}_i = (X_i/m)*S_i \Rightarrow S_i = 4\pi r_i^2 \Rightarrow X_i = \sum_k O_k$$

$$P[O_k] = \begin{cases} 1, \text{ ponto acessível} \\ 0, \text{ ponto não acessível} \end{cases}$$

$$E(O_k) = 1*(A_i/S_i) + 0*(1-A_i/S_i) = (A_i/S_i)$$

$$\text{Logo, } E(X_i) = = \sum_k E(O_k) = m*(A_i/S_i)$$

$$E(ASA) = E(\Sigma A_i) = \Sigma (1/m)*S_i*m*(A_i/S_i) = \Sigma A_i$$

Equação 2. Dedução da esperança da área acessível a solvente.

Assim, precisa-se de um método capaz de gerar os pontos nas superfícies de cada esfera de forma homogênea e de um método para determinar sistematicamente quais pontos de uma esfera são oclusos pelas demais esferas, isto é, não contribuem para a estimativa da ASA.

Descrição do método utilizado

O *Double Cubic Lattice Method,* proposto por Eisenhaber et al. (1995), é um método que utiliza a idéia de dividir a molécula em cubos (*lattices)*, de modo a comparar átomos apenas em suas vizinhanças, diminuindo o número de comparações entre os pontos gerados para cada esfera. O primeiro cubo determina a vizinhança entre os átomos e o segundo determina os quadrantes de um átomo que interessam efetivamente às comparações – isto é, os intervalos de pontos em um átomo que são candidatos à oclusão pela projeção de um segundo átomo sobre ele.

Cubo 1

A primeira divisão da molécula em cubos visa reduzir o número de comparações entre os átomos. Para isso o método:

- considera para cada átomo o seu raio $\mathbf{r}_i$ e seu ponto central, a sua coordenada **(x,y,z)** na molécula;
- calcula o maior r_i => $\mathbf{r_{máx} = \max_i (r_i + r_{sol})}$

[1] M é um parâmetro com o qual pode-se trabalhar a fim de comparar tempos de processamento.

[2] A esperança, valor médio esperado, de uma variável aleatória discreta corresponde a somatória de seus valores multiplicados pelas suas probabilidades de ocorrência (Mood et al., 1974).

- calcula um *grid* cúbico cujos lados são **2*$r_{máx}$**.
- dispõe-se os átomos que formam a molécula nesse *grid*, de acordo com a transformação de suas coordenadas cartesianas nas novas coordenadas dadas pela origem do *grid*, da seguinte forma:
  - calcula o ponto de origem no *grid*, que agora consiste em um novo sistema de ordenadas:
    - **origem = (x_0 = mín x, y_0 = mín y, z_0 = mín z)**
  - para cada átomo calcula em que cubo ele cai, ou seja, suas novas coordenadas no *grid*:
    - $I_x = ((x_i - x_o)/2 * r_{max})$
    - $I_y = ((y_i - y_o)/2 * r_{max})$
    - $I_z = ((z_i - z_o)/2 * r_{max})$
  - ordena os átomos dentro de cada cubo
- cada átomo corresponde a uma esfera ampliada de raio $r_i + r_{sol}$. Para que uma esfera invada a área de uma outra esfera ampliada, de raio $r_j + r_{sol}$, com j#i, deve estar a um raio menor que $2*r_{max}$ da outra. Com isso o número de comparações entre átomos é reduzido aos outros átomos que caem no mesmo cubo que ele e nos outros cubos vizinhos a ele, que são no máximo 26 outros cubos. A Fig. 4 ilustra um corte transversal do *grid*, onde o átomo em verde possui raio máximo (r_{max}). Deve-se notar que todos os átomos diferentes do verde terão raio menor que o dele e serão seus vizinhos se e somente se estiverem no mesmo cubo que ele, nos cubos laterais a ele (+ 8 cubos), nos cubos abaixo dele (+ 9 cubos) ou nos cubos acima dele (+ 9 cubos).

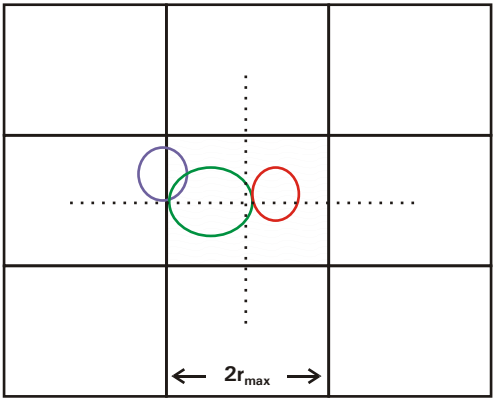

FIG. 4. Ilustração do *grid*, corte frontal com três átomos apenas.

Alguns dados são importantes para detalhes da implementação do método, que podem ser calculados com os dados conhecidos neste ponto:

- ponto limite no *grid*: **limite = (x_f = máx x, y_f = máx y, z_f = máx z)**
- número de células na ordenada x do *grid*: $b_x = int((x_f - x_o)/2 * r_{max})$
- número de células na ordenada y do *grid*: $b_y = int((y_f - y_o)/2 * r_{max})$
- número de células na ordenada z do *grid*: $b_z = int((z_f - z_o)/2 * r_{max})$
- e, se o cubo for implementado como um vetor unidimensional, funcionando como uma *hash table*[3], a *hashing function*[4] para acertar o número do cubo para cada átomo deve ser: $l_x + (l_y * b_x) + (l_z * b_x * b_y)$.

Cubo 2

O segundo cubo é uma nova subdivisão de cada um dos cubos anteriores. O objetivo é minimizar o número de cálculos para verificar que pontos pertencem a alguma intersecção entre as esferas. Para isso considera-se cada esfera como um novo cubo dividido em M^3 pequenos cubos. Existem várias recomendações em relação ao valor de M. Eisenhaber et al. (1995) recomendam utilizar M = 4, pois esse valor demonstrou-se geralmente aceitável para qualquer densidade de pontos e estrutura de proteína estudada.

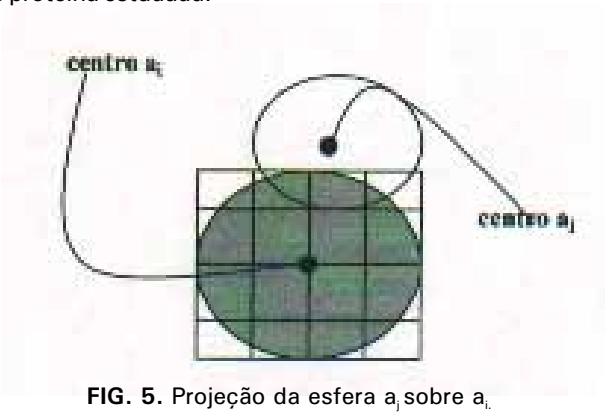

FIG. 5. Projeção da esfera a_j sobre a_i.

[3] Para referências sobre *hash tables* e suas formas de implementação, bem como *hashing function*, consulte (Wirth, 1976).

[4] Uma *hashing function* é a fórmula utilizada para armazenar e recuperar dados de forma eficiente em uma *hash table*.

A idéia é fixar um átomo a_i e então para cada átomo a_j, com j#i, pertencente à vizinhança de a_i (os 27 cubos em torno de a_i, resultantes da primeira divisão da molécula), calcular os intervalos que formam projeções da esfera a_j nos eixos da *grid* local, onde provavelmente estarão os únicos pontos da esfera a_i que podem ser oclusos por a_j; como ilustrado na Fig. 5.

Para calcular os intervalos utiliza-se:

$$X'_t = \text{máx}((x_j - x_i + (r_i - r_j))/2r_i, 0) \quad e \quad X'_u = \text{mín}((x_j - x_i + (r_i + r_j))/2r_i, 1\text{-}1/M) => [X'_t, X'_u]$$

$$Y'_t = \text{máx}((y_j - y_i + (r_i - r_j))/2r_i, 0) \quad e \quad Y'_u = \text{mín}((y_j - y_i + (r_i + r_j))/2r_i, 1\text{-}1/M) => [Y'_t, Y'_u]$$

$$Z'_t = \text{máx}((z_j - z_i + (r_i - r_j))/2r_i, 0) \quad e \quad Z'_u = \text{mín}((z_j - z_i + (r_i + r_j))/2r_i, 1\text{-}1/M) => [Z'_t, Z'_u]$$

Equação 3. Cálculo dos intervalos para o cubo 2.

Assim, serão testados apenas os pontos gerados para cada esfera a_i que caiam nesses intervalos, a partir do seguinte resultado apresentado por Eisenhaber et al. (1995):

- um ponto p_k gerado para a_i é ocluso pela esfera a_j se e somente se:

$$< (a_j \; a_i), p_k > \; > \; ((d^2_{ij} + r^2_i - r^2_j)/ 2r_i)$$

Equação 4. Inequação para decidir sobre oclusão de um ponto.

Isto é, a diferença vetorial formada pelos centros de a_i e a_j produto escalar com p_k for maior que o quadrado das distâncias de a_i e a_j, somados ao quadrado do raio de a_i, subtraídos do quadrado do raio de a_j e divididos pelo dobro de r_j.

Deve-se notar também que o lado direito da inequação acima precisa ser calculado uma única vez para cada átomo j.

Algoritmo para encontrar o número de pontos não oclusos

A solução algorítmica adaptada do método de Eisenhaber et al. (1995), a ser implementada para o Laboratório de Bioinformática Estrutural da Embrapa Informática Agropecuária é:

```
Após dividir a proteína segundo o Cubo 1
    Percorrer os cubos do grid
        Fixar um átomo a_i
        Gerar conjunto de pontos para a_ai => T_i(p_1, p_2, p3, ..., p_n)
        Gerar vetor de probabilidade de não oclusão O = {1, 1, ..., 1} => O_k = 1, qualquer k
        Para cada aj no mesmo cubo e em sua vizinhança
            Calcular intervalo, segundo a equação 3
            e parte direita da expressão da equação 4
            Para cada pk, com O_k # 0:
                p_k é ocluso, se e somente se, a equação4 é satisfeita,
                    Nesse caso => O_k recebe 0
        m_acc(i) = soma( O_k )
```

Algoritmo 1. Cálculo de $m_{acc(i)}$ para cada a_i.

Gerando pontos nas esferas

O método utilizado para gerar os pontos em cada esfera não é o mais importante para *o Double Cubic Lattice*, desde que a distribuição dos pontos sobre a esfera seja razoavelmente homogênea e que sejam gerados em torno de *600 pontos por esfera*, o que foi empiricamente demonstrado como um bom número (Eisenhaber et al., 1995).

Desta forma, escolheu-se um método já implementado em C, sem restrições de distribuição e de uso, a fim de evitar-se mais um estudo e teste de métodos. Foi escolhido o *algoritmo Sphere tesselation* desenvolvido por Leech & Buddenhagen (2004). No método utilizado pelos autores, gera-se uma malha de triângulos aproximando-se da esfera por divisões recursivas, sendo que a primeira divisão cai em um sólido platônico[5] e então cada nova divisão aumenta o número de triângulos em um fator quádruplo.

O problema da escolha dessa implementação é que ela foi projetada para um modelo procedimental e terá que ser transformada em um ou mais métodos de uma classe, respeitando um modelo de projeto orientado a objetos – tal como proposto neste trabalho.

Dados a serem utilizados

Os dados utilizados serão lidos de vários arquivos textos denominados PDBs - Protein Data Bank (Bourne & Weissig, 2003; Westbrook & Fitzgerald, 2003). Cada PDB corresponde a uma proteína, ou um complexo de proteínas ou a uma proteína em complexo com RNA/ DNA.

[5] Os sólidos platônicos, também chamados regulares ou poliedros regulares, são poliedros convexos com faces equivalentes compostas de polígonos regulares congruentes e convergentes (Weisstein, 2004).

Para ler os dados de interesse em um PDB será utilizada uma biblioteca disponível no controlador de versões do Laboratório de Bioinformática, denominada PDBReader (Higa et al., 2002a). A leitura será restrita a uma cadeia, que corresponde a uma proteína ou a um DNA. Para cada cadeia lida será calculada a estimativa da ASA e do volume acessível a solvente.

Do PDB virão os dados relativos à posição do átomo na molécula, ou seja, as coordenadas (x_i, y_i, z_i) de cada átomo. O raio de cada átomo deverá ser obtido de um arquivo, de domínio público e disponível no controlador de versões do Laboratório, que contém uma tabela de raios, denominado Radii.

Modelo Orientado a Objetos Proposto

A Fig. 6 ilustra o modelo orientado a objetos proposto para solucionar o problema apresentado, procurando-se ser fiel à notação de classes da OMT (Rumbaugh et al., 1999). O modelo apresenta apenas a agregação de classes necessárias para a implementação do Double Cubic Lattice Method – DCLM. É necessário completar a fonte de dados, isto é, a colocar a classe PDBReader e inserir a leitura dos dados do arquivo Radii.

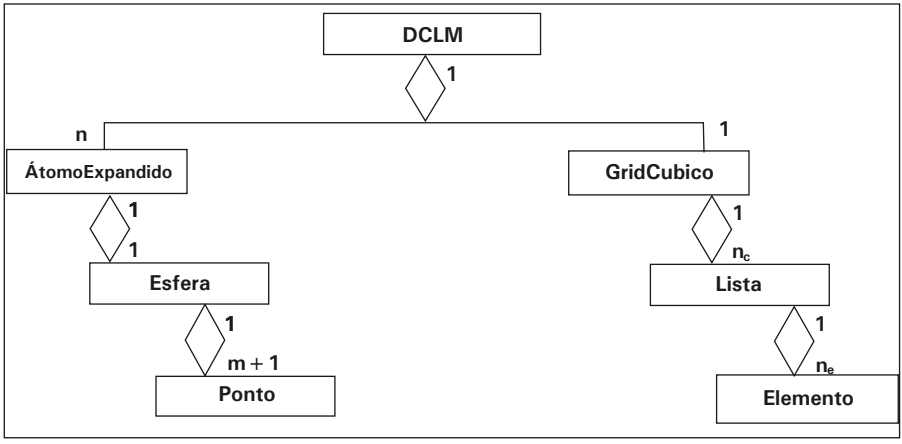

FIG. 6. Modelo de agregação de classes para o DCLM.

A intenção de construir esse modelo desta forma foi separar classes que pudessem ser reusadas em outras aplicações de bioinformática, dado que a maior parte delas provavelmente não servirá a outros domínios do conhecimento. Por exemplo, para as classes geométricas Ponto e Esfera serão implementados apenas os métodos de interesse na solução deste problema, mas as duas classes podem ser expandidas para comportarem novos métodos necessários a outras aplicações. A opção por desenvolver as classes Ponto e Esfera procurou evitar a compra de novos pacotes ou a utilização de pacotes de geometria computacional com restrição de uso, assim como a classe Lista.

Nos próximos subitens são descritas essas classes e enumeradas algumas das soluções específicas mais importantes.

Classe Ponto

Nesta classe há apenas três atributos, que correspondem às coordenadas do ponto em relação à origem (0,0,0). Uma instância desta classe pode ser considerada também como um vetor cuja origem é (0,0,0) e o ponto final a sua coordenada (x,y,z); dessa forma os métodos dessa classe incluem, além das atribuições e das obtenções de valores das ordenadas:

- distância: calcula a distância euclidiana entre a instância da classe e um dado ponto;
- produtoCartesiano: calcula o produto cartesiano entre o vetor dado pela instância da classe e o vetor dado pelo ponto que é passado como parâmetro;
- norma: calcula o módulo (comprimento) do vetor dado pela instância da classe;
- diferençaVetorial: calcula a diferença vetorial entre a instância da classe e um dado ponto, devolvendo o ponto limite do vetor resultante;
- origem: calcula o menor x, y e z entre a instância da classe e um dado ponto;
- limite: calcula o maior x, y e z entre a instância da classe e um dado ponto;
- pertenceIntervalo: se a instância da classe pertencer ao intervalo formado pelos dois pontos recebidos como parâmetro será devolvido o valor verdadeiro, caso contrário o valor falso.

A classe pode ser ampliada à medida que outros métodos forem sendo necessários.

Classe Esfera

O objetivo desta classe é concentrar os métodos responsáveis por cálculos em esferas tais como: superfície, perímetro, volume, etc. Esta classe possui quatro atributos que são:

- as coordenadas do centro da esfera, da classe Ponto;
- o valor do raio, que é um número real de dupla precisão;
- um vetor contendo valores reais de dupla precisão, que formam a nuvem de pontos gerados para a esfera. Esse vetor é alocado na criação da nuvem de pontos, pois o número de pontos gerados é um parâmetro conhecido em tempo de execução; e
- o número de pontos gerados para cada esfera.

O método mais importante para a aplicação é responsável por gerar a nuvem de pontos, que deve ter uma distribuição homogênea sobre a superfície da esfera. Inicialmente será implementado apenas um método para gerar a nuvem de pontos, porém, a idéia é implementar diferentes métodos com diferentes algoritmos, de forma que se possa, futuramente experimentar várias técnicas a fim de observar quais melhoram ou não a estimativa da ASA e em quais situações.

Classe ÁtomoExpandido

O átomo neste modelo corresponde a uma Esfera, que é a aproximação do átomo neste problema, e agrega também um número real de dupla precisão que corresponderá ao número de pontos não oclusos para cada instância da classe.

Os métodos, a princípio, serão apenas os de atribuição e obtenção dos valores dos atributos.

Classe Elemento

O objetivo desta classe é generalizar os elementos que podem compor uma lista ou alguma outra estrutura de dados que venha a ser utilizada. Para isso ele tem apenas um atributo que é um apontador para um caracter, que pode ser transformado a cada atribuição a partir de uma operação de "casting type" (mudança de tipo em C++). Esse truque está sendo utilizado porque o C++ não faz a introspecção dos tipos dos objetos - como na linguagem JAVA.

Os métodos da classe Elemento serão os de atribuir valores, a partir de um *casting type* predefinido em um arquivo *header* em comando a ser preprocessado.

O método mais importante desta classe é o de comparação entre Elementos, que será utilizado na ordenação dos elementos em cada cubo do *grid* cúbico. A implementação dessa comparação vai depender de cada tipo de Elemento, e será realizada, neste problema, para comparar pontos segundo uma dada origem.

Classe Lista

Cada cubo do *grid* cúbico conterá uma lista de pontos, que correspondem aos pontos que caem neste cubo. Após a montagem dessa lista, os pontos precisam ser ordenados.

Logo, esta classe deve conter métodos para incluir e retirar pontos da lista, para percorrer a lista e para ordenar seus elementos. Para percorrer a lista, no mínimo, serão implementados: primeiroElemento, temProximoElemento e proximoElemento. Para a ordenação de elementos será utilizado o método de comparação da classe Elemento.

Classe GridCubico

O gridCubico é uma nova abstração do conjunto de átomos expandidos que formam a molécula protéica, considerando-se uma nova origem para os pontos e dividindo-se o espaço em cubos. Dessa forma os atributos devem ser: nova origem; limite superior para os pontos; número de cubos no novo eixo x (bx); número de cubos no novo eixo y (by); número de cubos no novo eixo z (bz); e, um vetor de listas de dimensão bx*by*bz que, de fato, é a tabela de *hashing* onde os pontos serão colocados.

Os métodos da classe devem ser:

- Cria: criar uma instância desta classe iniciando seus atributos;
- Aloca: aloca o vetor de listas de dimensão bx*by*bz;
- PosicionaElemento: calcula a posição do elemento no vetor, utilizando a *hashing function* do item Cubo 1. e insere o elemento na lista dessa posição do vetor;
- OrdenaElementos: ordena os elementos de cada cubo, utilizando a ordenação da lista;
- Percorrer os cubos: deve-se lembrar que é necessário percorrer os cubos, e cada átomo dentro de cada cubo:
  - PrimeiroCubo: posiciona no cubo de origem;
  - TemProximoCubo: verifica se existe um próximo cubo;
  - ProximoCubo: posiciona no próximo cubo.
- Percorrer a vizinhança do cubo: a partir do momento que um cubo estiver fixado, é necessário percorrer toda a sua vizinhança. Essa vizinhança pode atingir até 26[6] cubos além dela:
  - PrimeiroVizinho: busca o primeiro cubo vizinho ao cubo fixado;
  - TemProximoVizinho: verifica se existe outro cubo vizinho;
  - ProximoVizinho: fixa o próximo cubo vizinho.

Classe DCLM

O objetivo desta classe é efetivamente implementar o método *Double Cubic Lattice*. Logo, é nesta classe que todo o processo de cálculo é controlado a partir da leitura dos dados.

[6] Veja subsubitem Cubo 1.

Seus atributos são: o conjunto de átomos que forma a molécula protéica; os valores limites dos pontos, Origem = ($x_{mín}$,$y_{mín}$,$z_{mín}$) e Limite = ($x_{máx}$,$y_{máx}$,$z_{máx}$); os raios mínimo e máximo; o número total de átomos; o valor da ASA; e, o valor do volume acessível a solvente. Desta forma, os valores dos atributos que correspondem aos máximos e mínimos são recalculados a cada novo átomo que é inserido na coleção.

Quando a coleção de átomos está completa então pode-se calcular os cubos 1 e 2 e finalmente a ASA e o volume, utilizando-se os seguintes métodos:

- Cubo1: cria uma instância do *gridCubico*; aloca o vetor do *grid*; executa um laço onde os átomos são colocados em suas posições no *grid*; e, ordena os elementos de cada cubo do *grid;*
- Cubo 2: implementa o Algoritmo 1. Cálculo de macc(i) para cada ai;
- Calcula a ASA: segundo a Equação 1. Estimativa da ASA.

Trabalhos Futuros

O modelo de objetos proposto precisa ser implementado e integrado a um programa que execute as leituras dos átomos nos PDBs e dos raios dos átomos no Radii. Os resultados obtidos por essa implementação deverão ser comparados aos resultados obtidos atualmente pelo Java Protein Dossier - JPD. Algum erro é permitido, desde que não seja estatisticamente significante; assim, deverá ser estabelecido qual o valor do erro aceitável e que testes utilizar para verificá-lo. Também deverá ser considerado o tempo de processamento gasto pelo JPD e pelo programa implementado a partir deste trabalho, permitindo concluir se houve melhora ou piora dessa implementação em relação à anterior.

Realizada essa implementação, deve ser feito um estudo de maneira de melhorar a estimativa da ASA a partir apenas da troca de algoritmos para geração dos pontos nas esferas. Sabe-se de antemão, por outros trabalhos, que a distribuição dos pontos precisa ser homogênea e utilizar-se perto de 600 pontos por esfera. Entretanto, é possível que métodos diferentes consigam diminuir significativamente o erro da estimativa se permitirem melhorar a precisão do cálculo dos pontos.

O arquivo Radii possui os valores dos raios dos átomos para cada resíduo. Seria interessante manter esses valores em uma estrutura ordenada da qual fosse possível obtê-los rápida e eficientemente. Assim, um outro trabalho futuro, é desenvolver uma biblioteca que permita colocar os dados do Radii em uma árvore binária em memória, a fim de facilitar e padronizar o acesso aos mesmos em aplicações do Laboratório de Bioinformática Estrutural.

Referências Bibliográficas

BOURNE, P.; WEISSIG, H. The Protein Data Bank. In: BOURNE, P.; WEISSIG, H. (Ed.). **Structural bioinformatics**. Hoboken, N. J.: Wiley-Liss, 2003. p. 181-198.

CONOLLY, M. J. Analytical molecular surface calculation. **J. Appl. Crystallog**., v., 16, n. 548-558, 1983.

EISENHABER, F.; LIJNZAAD, P.; ARGOS, P.; SANDER, C.; SCHARF, M. The double cubic lattice method: efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies. **Journal of Computational Chemistry**, v. 16, n. 3, p. 273-284, 1995.

FUTAMURA, N.; ALURU, S.; RANJAN, D.; HARIHARAN, B. Efficient parallel algorithmics for solvent accessible surface area of proteins. **IEEE Transactions on Parallel and Distributed Systems**, v. 13, n. 6, p. 544-555, June 2002.

HIGA, R. H.; MANCINI, A. L.; FALCÃO, P. K.; NESHICH, G. **Cálculo de área acessível por solvente utilizando SURFV – definição de interface intramolecular pelo SMS**. Campinas: Embrapa Informática Agropecuária, 2002. 4 p. (Embrapa Informática Agropecuária. Comunicado Técnico, 36). Disponível em: <http://www.cnptia.embrapa.br/modules/tinycontent3/content/2002/comuntec36.pdf>. Acesso em: 12 nov. 2004.

HIGA, R. H.; BAUDET, C.; MANCINI, A. L. MATTIUZ, A. R.; FREITAS, E. M. de; FALCÃO, P. K.; NESHICH, G. **SMSLib – Biblioteca C++ do Sting Millenium Suite**. Campinas: Embrapa Informática Agropecuária, 2002. 10 p. (Embrapa Informática Agropecuária. Comunicado Técnico, 39). Disponível em: <http://www.cnptia.embrapa.br/modules/tinycontent3/content/2002/comuntec39.pdf>. Acesso em: 12 nov. 2004b.

LEECH, J.; BUDDENHAGEN, J. **Sphere tesselation code**. Disponível em: <http://www.neubert.net/Htmapp/SPHEmesh.htm>. Acesso em: 12 nov. 2004.

MOOD, A. M.; GRAYBILL, F. A.; BOES, D. C. **Introduction to the theory of statistics**. 3rd ed. New York: MacGraw-Hill Book, 1974. 564 p.

NICOLAU JUNIOR, D. V.; FULGA, F.; NICOLAU, D. V. A new program to compute the surface properties of biomolecules. In: CHEN, Y.-P. (Ed.). **First Asia-Pacific Bionformatics Conference (APBC 2003)**. Adelaide: Australia Computer Society, 2003. (CRPIT, v. 19). Disponível em: <http://crpit.com/confpapers/CRPITV19Nicolau.pdf>. Acesso em: 12 nov. 2004.

PROSHAPE Software: understanding the shape of proteins structure - volume, surface and pockets of proteins. Disponível em: <http://biogeometry.cs.duke.edu/software/proshape/>. Acesso em: 9 nov. 2003.

PROTEIN Dossier. In: STRUCTURAL BIOINFORMATICS GROUP. **Sting Millennium Suite 3 - Structural Bioinformatics Group - EMBRAPA/CNPTIA, Brazil**. Disponível em: <http://sms.cbi.cnptia.embrapa.br/SMS/index_m.html>. Acesso em: 12 nov. 2004.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G**. The unified modeling language reference manual**. Reading, Mass. : Addison-Wesley, 1999. 550 p.

WEISSTEIN, E. W. **Platonic Solid -- from MathWorld**. Disponível em: <http://mathworld.wolfram.com/PlatonicSolid.html>. Acesso em: 20 dez. 2004.

WESTBROOK, J. D.; FITZGERALD, P. M. D. The PDB format, mmCIF formats, and other data formats. In: BOURNE, P.; WEISSIG, H. (Ed.). **Structural bioinformatics**. Hoboken, N. J.: Wiley-Liss, 2003. p. 161-179.

WIRTH, N. **Algorithms + data structures = programs**. Englewood Cliffs, N. J.: Prentice-Hall, 1976. 366 p. 1976.

Ministério da Agricultura,
Pecuária e Abastecimento

BRASIL
UM PAÍS DE TODOS
GOVERNO FEDERAL