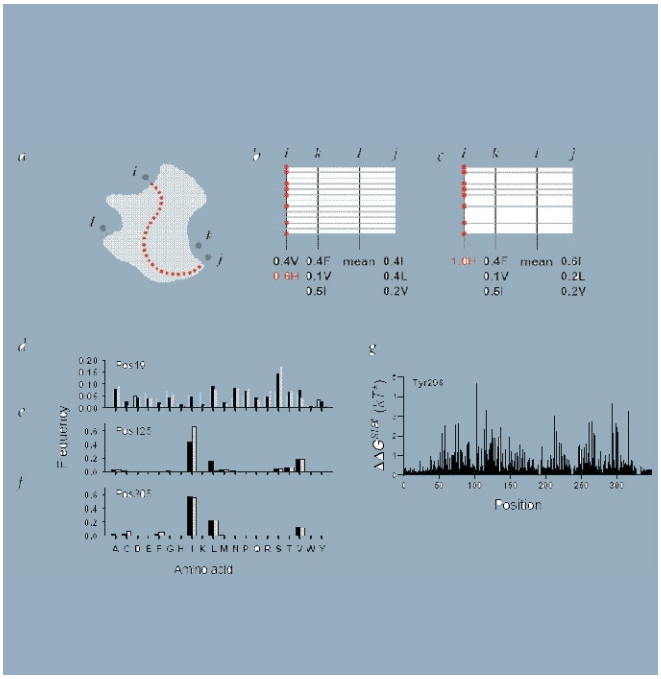
Comunicado Técnico 77

Novembro, 2006
Campinas, SP



ISSN 1677-8464

Aspectos Computacionais da Análise da Co-evolução de Aminoácidos que pertencem a uma Proteína qualquer

Marcelo Gonçalves Narciso[1]
Michel Eduardo Beleza Yamagishi[2]
Thiago Quinaglia[3]
Edgard Henrique dos Santos[4]
Fábio Danilo Vieira[5]
José Gilberto Jardine[6]
Ivan Mazoni[7]
Paula Regina Kuser Falcão[8]
Goran Neshich[9]

Uma área que tem crescido muito nos últimos anos é a Bioinformática. Segundo definição extraída da Wikipedia (Wikimedia Foundation, 2006), a Bioinformática combina conhecimentos de Química, Física, Biologia e Informática para processar dados biológicos. Assim, para tratar os dados existentes, é necessário desenvolver softwares para identificar genes, prever a configuração tridimensional de proteínas, identificar inibidores de enzimas, organizar e relacionar informação biológica ou simular células biológicas. O software Sting (Embrapa Informática Agropecuária, 2006a) é um software voltado para a Bioinformática e, de forma geral, faz análise de estrutura de proteínas e mostra de forma espacializada qualquer proteína cuja estrutura está em arquivos .pdb, que contém dados sobre proteínas, podendo ser acessados em Research Collaboratory for Structural Bionformatics (2006). Estes arquivos, em torno de 37.000, são acessados pelo Sting para diversos fins de visualização, estudo e mineração de dados (Oliveira, 2006). Diversos tipos de análises podem ser feitos com este software, bastando apenas acessá-lo pela internet.

Um dos recursos do Sting é a análise, de forma visual, da co-evolução de um aminoácido (unidade constituinte das proteínas) em uma proteína (que é uma cadeia formada por aminoácidos). Co-evolução de aminoácidos é um fenômeno no qual o deslocamento de um aminácido de uma dada posição na proteína é seguido por uma mudança paralela de outros aminoácidos, ou ainda, um aminoácido tem correlação com outros aminoácidos da cadeia proteica. No software Sting, é possível ver a co-evolução de todos os aminoácidos que compõem uma dada proteína. Esta funcionalidade está disponível em Embrapa Informática Agropecuária (2006b).

[1] Doutor em Computação, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: narciso@cnptia.embrapa.br)
[2] Doutor em Matemática Aplicada, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: michel@cbi.cnptia.embrapa.br)
[3] Bacharel em Ciência da Computação, Bolsita da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: quinaglia@cbi.@cnptia.embrapa.br)
[4] Bacharel em Ciência da Computação, Analista da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: edgard@cbi.@cnptia.embrapa.br)
[5] Bacharel em Tecnologia de Informação, Analista da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: fabio@cbi.@cnptia.embrapa.br)
[6] Ph.D. em Engenharia de Alimentos, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (jardine@cnptia.embrapa.br)
[7] Bacharel em Ciência da Computação, Analista da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: ivan@cbi.cnptia.embrapa.br)
[8] Doutora em Cristalografia de Proteínas, Pesquisadora da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (e-mail: paula@cbi.cnptia.embrapa.br)
[9] Ph.D. em Biofísica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP. (Neshich@cbi.cnptia.embrapa.br)

Embrapa

O algoritmo desenvolvido para o cálculo de co-evolução de aminoácidos é o proposto por Süel et al. (2003) com alguns ajustes para ser implementado no software Sting. Estes ajustes serão descritos mais adiante. Após o cálculo dos valores de co-evolução, uma interface feita em java mostra graficamente o resultado. Um exemplo desta interface está descrito a seguir (Fig. 1).

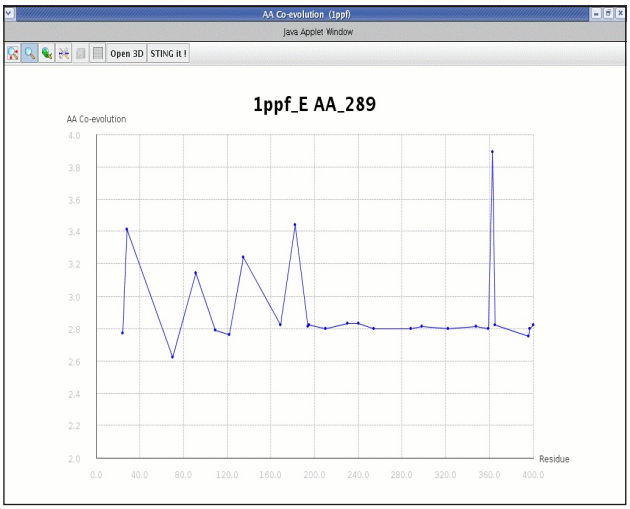


Fig. 1. Co-evolução do aminoácido da posição 289 da proteína 1ppf, cadeia E.

O gráfico acima descreve os valores de co-evolução em relação às demais posições da cadeia, onde se situam outros aminoácidos. Observa-se neste gráfico que o aminoácido da posição 361 tem um valor de 3.9. Isto significa que o aminoácido que está na posição 289, o qual está sendo comparado com os demais aminoácidos da cadeia, tem co-evolução com o aminoácido da posição 361 e alguns outros aminoácidos ao longo da cadeia que forma a proteína 1ppf. Na Fig. 1, tem-se a posição e o valor de co-evolução de todos os aminoácidos da cadeia proteica descrita em 1ppf. Mais detalhes podem ser observados no software Sting, clicando-se no gráfico, na posição desejada.

O objetivo deste artigo é mostrar o algoritmo para se calcular a co-evolução entre aminoácidos e, principalmente, as soluções obtidas para contornar as dificuldades encontradas em cálculo de fatoriais muito grandes e conseqüentes erros de *overflow* e *underflow*, os quais não são cobertos na proposta inicial de Süel et al. (2003)

Material e Métodos

O trabalho de Süel et al. (2003) é a base para se fazer os cálculos de co-evolução entre os aminoácidos e gerar os resultados a serem mostrados pelo software Sting. Este algoritmo foi desenvolvido em C e executado em 37.000 arquivos pdb (Research Collaboratory for Structural Bionformatics, 2006), em conjunto com respectivos arquivos de alinhamento. Os arquivos de alinhamento são gerados a partir de arquivos .pdb. Por exemplo, a proteína que está no arquivo 1cho.pdb tem os seguintes arquivos de alinhamentos gerados: 1choE.aln e 1choI.aln. Não é objetivo deste trabalho entrar em detalhes de como estes arquivos foram gerados, mas sim descrever os algoritmos e como este é apresentado no software Sting.

De forma resumida, o algoritmo de Süel et al. (2003) pode ser colocado da seguinte forma:

1. dado um conjunto de múltiplas seqüências de alinhamentos de aminoácidos (MSA), escolher um aminoácido X e verificar como ele influencia os demais. Calcular os valores;
2. escolher alguns alinhamentos do conjunto de alinhamento anterior e verificar a influência do aminoácido X relativamente aos demais aminoácidos, tal como anteriormente. Recalcular os valores;
3. obter a diferença entre os resultados obtidos nos itens 1 e 2 e verificar o valor. Se o valor for **acima de 2.0**, então considera-se que existe uma co-evolução.

No trabalho de Süel et al. (2003) existe uma figura que ilustra melhor este algoritmo.

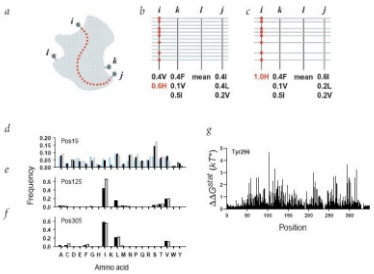


Fig. 2. Exemplo de Co-evolução de um aminoácido da posição 296 da cadeia proteica da Serine Protease.

No exemplo da Fig. 2, a figura (a) ilustra os aminoácidos das posições i, j, k e l. A figura (b) ilustra os alinhamentos iniciais com os valores de energia para cada posição i, j, k e l. A figura (c) ilustra os valores de energia ao se considerar apenas a coluna i com um dado aminoácido A, retirando-se os alinhamentos que não contenham este aminoácido A na coluna i. As figuras (d), (e) e (f) ilustram as distribuições de energia dos aminoácidos em diversas posições antes da perturbação (em preto) e após a perturbação (em cinza) de um dado aminoácido da posição 296 da cadeia proteica. A figura (g) ilustra a distribuição da energia total de todas as posições com a perturbação do aminoácido da posição 296 (Tyr296).

Observa-se que a figura (f) ilustra pouca variação de energia para cada aminoácido e a figura (e) ilustra uma maior variação de energia com os aminoácidos I e L. Assim, o aminoácido da posição 296 tem mais influência (é mais correlacionado) com o aminoácido da posição 125 do que o aminoácido da posição 305.

A forma de calcular a influência de um aminoácido X na cadeia de aminoácidos é através da probabilidade conjunta de um dado aminoácido X aparecer antes e depois de uma perturbação em uma determinada posição na seqüência, cujo cálculo, segundo Süel et al. (2003), é feito através da expressão

$$K.T.\log(Pxj/PxMSA),$$

onde K e T são constantes e Pxj é a probabilidade binomial de se obter o aminoácido X na posição j, dada sua freqüência média no alinhamento. PxMSA é a probabilidade de se observar o aminoácido X em todas as seqüências de alinhamento múltiplos (MSA). Para obter mais detalhes, consultar Süel et al. (2003).

A probabilidade condicional Pxj é dada por:

$Pxj = (N!/(n!*(N-n)!))*p^N*(1-p)^{N-n}$
N! (assim com n! e (N-n)!) significa fatorial de N, isto é
$N! = N*(N-1)*(N-2)*...*1$.

A variável p é a probabilidade de um dado aminoácido X aparecer no alinhamento j. N é o número de aminoácidos do alinhamento e n é o número de aminoácidos X nos alinhamentos.

Um problema para este cálculo é quando N = 500 e n = 250, por exemplo. Não adianta simplificar pois 500!/(250!*250!) = 500*499*498*...*251 é um número muito elevado e pode resultar em erro de *overflow*.

Por outro lado, como p é um número menor ou igual a 1, se for elevado a um número muito grande, pode dar erro de *underflow*, pois um número menor que 1 elevado a um número muito grande pode estourar a capacidade da máquina de armazenar um número muito pequeno.

Assim, este cálculo para calcular a probabilidade condicional Pxj, em princípio, era inviável. Para resolver este problema, após várias tentativas sem sucesso, usou-se o seguinte procedimento. Seja log a representação do logaritmo na base 10. Assim,

$\log\{Pxj\} = \log\{(N!/(n!*(N-n)!))(*p^N*(1-p)^{N-n}\}$
$\log\{Pxj\} = \log N! - \log n! - \log (N-n)! + N \log(p) + (N-n) \log (1\ p)$,
onde p é diferente de 1.

Por outro lado, $\log N! = \log N + \log (N-1) + \log (N-2) + ... + \log 1$

Assim, como o log de um número é finito e não muito grande, a soma dos logaritmos acima também será um valor finito de tal forma que não ocorra erro de *overflow* e *underflow*. Suponha que a soma destes logaritmos seja igual a Z. Então,

$Z = \log\{Pxj\} = \log N! - \log n! - \log (N-n)! + N \log(p) + (N-n) \log (1\ p)$
Se $\log\{Pxj\} = Z$, então $Pxj = 10^Z$ e $0 \le Pxj \le 1$.

Desta forma, a probabilidade foi calculada sem problemas de *overflow* e *underflow*.

Outro problema que apareceu foi o cálculo do fatorial. A somatória dos logaritmos, embora resolvesse o problema de *overflow*, demorava mais para ser calculado. Uma forma alternativa de cálculo de fatorial é a fórmula de Stirling (Matemática, 2006). Esta fórmula é dada por:

$n! = (2*3.1415*n)^{0.5}*[(n/e)^n]*[1 + 1.0/(12*n) + 1.0/(288*n^2) - 139.0/(51840*n^3) - 571.0/(2488320*n^4) + ...]$, (e = 2,718...).

Levando-se em conta a série até $571.0/(2488320*n^4)$, a aproximação fica muito boa a partir de fatorial de 7. Assim, usou-se a forma comum de calcular o fatorial para N variando de 0 até 7 e a fórmula de Stirling para valores de N acima de 7. A precisão é muito boa, neste caso. Com isto, o cálculo do fatorial usando logaritmo ficou muito mais fácil pois o número de operações diminuiu sensivelmente.

Se aplicar o logaritmo na base 10, simbolizado por log, na fórmula de Stirling (sem considerar a série), tem-se a seguinte expressão:

$Log10(n!) = 0.5*\log(2*3.1415) + 0.5*\log(n) + n*(\log(n) - \log(\exp(1))) + \log\{1 + 1.0/(12*n) + 1.0/(288*n^2) - 139.0/(51840*n^3) - 571.0/(2488320*n^4)) + ...\}$.

Observe que a série da fórmula acima é apenas mais um cálculo de um logaritmo.

Suponha que

$T = 0.5*\log(2*3.1415) + 0.5*\log10(n) + n*(\log(n) - \log(2.7182)) + ...$

Assim, $\log(n!) = T$, e portanto, $n! = 10^T$.

Desta forma, o cálculo ficou muito mais rápido e os erros de *overflow* e *underflow* não existem mais. Observe que, sem usar logaritmo, o cálculo de um fatorial através da fórmula de Stirling também pode dar problemas de *overflow* por causa do termo $(n/e)^n$. Se n for maior do que 100, por exemplo, erros de *overflow* podem aparecer, conforme a capacidade do computador. Com valores acima de 200, observou-se que erros de *overflow* ocorriam freqüentemente. Portanto, o uso do logaritmo resolveu o problema.

Este problema enquadra-se em Cálculo Numérico, devido às considerações feitas anteriormente, ou computação de alto desempenho, visto que para rodar os 78.000 casos (arquivos .pdb e .aln) foi necessário um *cluster* Linux Mosix (Barak, 2006), com 10 processos em paralelo. O tempo para processar todos estes arquivos foi de 25 horas aproximadamente. Antes, para processar estes arquivos em uma máquina eram necessários em média 3 dias.

Resultados

Devido ao grande número de arquivos .pdb e .aln para serem executados, usou-se um *cluster* linux Mosix. Com este *cluster*, o qual contém 25 nós (processadores), foram divididas as tarefas de processamento em 10 nós. O *cluster* vai distribuindo o processamento nos 10 processadores livres que encontrar e monitora todo o processamento. Assim, não foi necessário executar este programa em outras máquinas da rede e o processamento foi muito leve, sem *overflow* ou qualquer outro problema parecido.

Um outro problema interessante, digno de nota, é a distribuição dos 78.000 arquivos de alinhamento de proteínas a serem processados pelo algoritmo do co-evolution modificado entre 10 processadores. Foi feito um algoritmo que distribuia os arquivos entre os 10 processadores de forma que todos os processadores tivessem a mesma carga. Os arquivos têm tamanhos diferentes e, assim, procurou-se distribuí-los de forma que a soma total dos arquivos, em termos de tamanho, fosse a mesma para cada processador. Como o número de arquivos é muito grande, a solução deste problema não foi trivial, isto é, foi necessário resolver um problema combinatório. Assim, o tempo de execução total, do início até o término do processamento do último nó do *cluster*, foi menor, visto

que a carga distribuída entre os processadores foi praticamente a mesma.

Após todos os valores de co-evolução serem calculados, estes são acessados pelo software Sting e os resultados podem ser mostrados individualmente ou globalmente, como é o caso da Fig. 3.

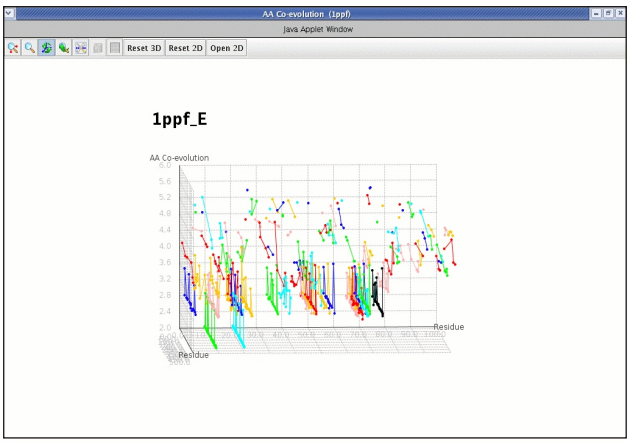


Fig. 3. Co-evolução de todos os aminoácidos no alinhamento 1ppf, Cadeia E.

Conclusões

Embora o algoritmo de Süel et al. (2003) tenha tido problemas para implementação, acabou por ser resolvido com uma transformação logaritmica, que impediu erros de *overflow* e *underflow*. Sem esta transformação não seria possível gerar os resultados em muitos casos. Por outro lado, a fórmula de Stirling ajudou em muito a melhorar o desempenho em termos de tempo de execução do algoritmo. Assim, este problema de cálculo numérico foi resolvido e o uso de *cluster* Linux também foi de grande importância para o processamento mais rápido de uma grande massa de dados, que eram os arquivos pdb e alinhamentos (.aln).

A interface Java do Sting proporciona uma visão mais fácil de um alinhamento de uma proteína e a co-evolução de cada aminoácido desta proteína, conforme o tipo de cadeia, que pode ser escolhida antes do Sting montar a saída graficamente. Porém, de nada valeria esta interface sem o algoritmo para calcular a co-evolução e os artifícios usados para o cálculo dos fatoriais para evitar erros de *overflow* e *underflow*.

Referências Bibliográficas

BARAK, A. *MOSIX*: grid and cluster management. Disponível em: <http://www.mosix.org>. Acesso em: 14 ago. 2006.

EMBRAPA INFORMÁTICA AGROPECUÁRIA. *Blue Star Sting - Structural Bioinformatics Group*. Disponível em: <http://www.cbi.cnptia.embrapa.br/SMS/>. Acesso em: 14 ago. 2006a.

EMBRAPA INFORMÁTICA AGROPECUÁRIA. *Blue Star Sting Suite - AA Co-evolution*. Disponível em: <http://www.cbi.cnptia.embrapa.br/SMS/coevolution/index.html>. Acesso em: 14 ago. 2006b.

MATEMÁTICA fatorial no Brasil. Disponível em: <http://www.gigabusca.com.br/wiki/Factorial.html>. Acesso em: 14 ago. 2006.

OLIVEIRA, S. R. de M. *Heuristics for protecting competitive knowledge in association rule mining*. Campinas: Embrapa Informática Agropecuária, 2006. 51 p. (Embrapa Informática Agropecuária. Boletim de Pesquisa e Desenvolvimento, 13). Disponível em: <http://www.cnptia.embrapa.br/modules/tinycontent3/content/2006/bp13.pdf>. Acesso em: 14 ago. 2006.

RESEARCH COLLABORATORY FOR STRUCTURAL BIOINFORMATICS. Protein Data Bank. *RCSB PDB - latest released structures*. Disponível em: <http://www.pdb.org/>. Acesso em: 14 ago. 2006.

SÜEL, G. M.; LOCKLESS, S. W.; RANGANATHAN, A. Evolutionarily conserved networks of residues mediate allosteric communication in proteins. *Nature Structural Biology*, v. 10, n. 1, p. 59-69, Jan. 2003. Disponível em: <http://www.nature.com/nsmb/journal/v10/n1/pdf/Nsb881.pdf>. Acesso em: 14 ago. 2006.

WIKIMEDIA FOUNDATION. Bioinformática. Disponível em: <http://pt.wikipedia.org/wiki/Bioinformática>. Acesso em: 14 ago. 2006.

Comunicado Técnico, 77

Ministério da Agricultura, Pecuária e Abastecimento

BRASIL UM PAÍS DE TODOS GOVERNO FEDERAL

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-970 - Campinas, SP
Fone: (19) 3789-5743
Fax: (19) 3289-9594
e-mail: sac@cnptia.embrapa.com.br

1ª edição on-line - 2006