Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, v. 6, n. 1, 2018.

Trabalho apresentado no XXXVII CNMAC, S.J. dos Campos - SP, 2017.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Estratégias para Mitigação de Erros Numéricos em Algoritmos de Determinação de Estruturas Protéicas

Michael Souza[1]
Departamento de Estatística e Matemática Aplicada, UFC, Fortaleza, CE
Carlile Lavor[2]
Instituto de Matemática, Estatística e Computação Científica, Unicamp, Campinas, SP
Luiz Mariano Carvalho[3]
Instituto de Matemática e Estatística, UERJ, Rio de Janeiro, RJ

Resumo. Apresentamos duas estratégias de para mitigação de erros numéricos em algoritmos iterativos que usam apenas informações locais para o problema de geometria de distâncias moleculares. Além disso, realizamos experimentos numéricos em instâncias construídas a partir de proteínas reais, envolvendo milhares de átomos, e mostramos que as estratégias propostas conjugadas em um novo algoritmo BuildUpOpt são capazes de resolver instâncias de grande porte.

Palavras-chave. proteínas, bioinformática, geometria de distâncias, algoritmos iterativos.

1 Introdução

As proteínas desempenham inúmeras funções no organismo, da composição de tecidos e membranas até a catalização de reações químicas. O papel específico de cada proteína está intimamente ligado à sua geometria (sítios ativos) e um dos métodos mais utilizados para a obtenção de estruturas protéicas é a ressonância magnética nuclear (RMN). O resultado da RMN é uma matriz de distâncias euclidianas parcialmente preenchida. Devido à precisão dos equipamentos envolvidos, em lugar de escalares, algumas das entradas desta matriz são intervalos.

Neste contexto, o problema de geometria de distâncias moleculares (PGDM) pode ser formulado matematicamente por

Definição (PGDM). *Determine* $x \in \mathbb{R}^{3n}$ *tal que*

$$l_{ij} \leq ||x_i - x_j|| \leq u_{ij}, \forall (i,j) \in E \subset \{1,\ldots,n\} \times \{1,\ldots,n\}. \quad (1)$$

Na definição do PGDM, os dados dos experimentos são representados pelos limites l_{ij} e u_{ij}. Em instâncias reais, algumas distâncias são conhecidas exatamente ($d_{ij} = l_{ij} = u_{ij}$),

[1]michael@ufc.br
[2]clavor@ime.unicamp.br
[3]luizmc@ime.uerj.br

outras de maneira imprecisa ($l_{ij} < u_{ij}$), mas não temos informações sobre a maioria das distâncias, ou seja, o conjunto E de pares (i, j) é esparso [5].

Os algoritmos propostos para o PGDM variam de acordo com as diferentes hipóteses sobre as restrições dadas pela Eq.(1). A hipótese mais restritiva requer que todas as distâncias sejam conhecidas e exatas ($|E| = n(n-1)/2$ e $l_{ij} = u_{ij}$). Neste caso, podemos reduzir o PGDM ao cálculo dos autovalores uma matriz de posto três [3].

Os algoritmos menos restritivos, que não fazem qualquer hipótese adicional, são os que se baseiam em formulações de otimização do PGDM. Dentre os métodos de otimização, podemos destacar os algoritmos de transformada Gaussiana [7], de suavização e penalização hiperbólicas [8] e de programação semi-definida [2]. A presença de muitos mínimos locais e a não-diferenciabilidade são os maiores desafios no uso dos métodos de otimização.

As outras alternativas são, em geral, iterativas e supõem uma ordem de fixação dos átomos partindo de um conjunto de átomos previamente fixados. Por exemplo, no algoritmo *BuildUp* [4], cada átomo é fixado utilizando as distâncias exatas a quatro átomos previamente fixados (interseção de quatro esferas). À medida que mais átomos são fixados, o algoritmo avança fixando novos átomos. No algoritmo *Branch-and-Prune* (BP) [6], supõe-se que haja uma ordenação, onde são conhecidas as distâncias exatas de um átomo a três átomos previamente fixados. Neste caso, existem duas soluções (interseção de três esferas) e uma árvore binária pode ser utilizada para mapear as configurações válidas. O BP aplica uma estratégia de poda (*pruning*) para explorar eficientemente a árvore binária de configurações.

Em geral, os algoritmos iterativos para o PGDM fazem uso apenas de informações locais para fixar cada um dos pontos da estrutura. No entanto, à medida que aumentamos o tamanho dos problemas de interesse, começamos a enfrentar mais sensivelmente os limites impostos pela representação numérica em ponto flutuante, ou seja, os erros de truncamento e os associados ao cancelamento de dígitos significativos. Em problemas cada vez maiores, as ordens de grandeza dos erros acumulados podem inviabilizar completamente os resultados obtidos. Sendo assim, com o aumento do tamanho dos problemas mesmo algoritmos bem estabelecidos, como o *BuildUp* e o *BP*, podem deixar de apresentar resultados satisfatórios se utilizados sem medidas de contenção de erros.

No presente artigo, discutimos as limitações dos algoritmos de fixação iterativos baseados em informações locais em instâncias de grande porte do PGDM. Dadas as particularidades de cada algoritmo, para facilitar a exposição, focaremos a discussão no algoritmo *BuildUp*, mas as conclusões obtidas são gerais e aplicam-se aos demais algoritmos de fixação que façam uso apenas de informações locais a cada iteração. Além disso, como alternativa, apresentamos uma nova versão do algoritmo *BuildUp*, chamada de *BuildUpOpt*, que utiliza otimização local para controlar os erros numéricos e uma ordenação dinâmica para *speedup*.

Na seção 2, discutimos as principais fontes de erro nos algoritmos de fixação iterativos. O algoritmo BuildUpOpt é apresentado na Seção 3. Na Seção 4, discutimos as estruturas de dados utilizadas na implementação. Os experimentos numéricos são apresentados na Seção 5. Finalmente, a Seção 6 é dedicada às conclusões e trabalhos futuros.

DOI: 10.5540/03.2018.006.01.0360

2 As origens dos erros

Os algoritmos iterativos de fixação para o PGDM constroem uma solução calculando as novas coordenadas x_i em função das coordenadas previamente calculadas. Mais especificamente, a cada iteração, uma coordenada x_i de um átomo i é calculada utilizando as distâncias exatas a um conjunto B_i (base) de átomos cujas coordenadas tenham sido previamente calculadas. No algoritmo *BuildUp*, cada base $B_i = \{a, b, c, d\}$ é formada por quatro átomos. A coordenada $x_i \in \mathbb{R}^3$ é obtida a partir do sistema de equações quadráticas:

$$\begin{aligned} ||x_i - x_a||^2 &= ||x_a||^2 - 2x_a^t x_i + ||x_i||^2 = r_a^2, \\ ||x_i - x_b||^2 &= ||x_b||^2 - 2x_b^t x_i + ||x_i||^2 = r_b^2, \\ ||x_i - x_c||^2 &= ||x_c||^2 - 2x_c^t x_i + ||x_i||^2 = r_c^2, \\ ||x_i - x_d||^2 &= ||x_d||^2 - 2x_d^t x_i + ||x_i||^2 = r_d^2. \end{aligned} \tag{2}$$

Ou seja, x_i é a interseção das esferas com centros x_a, x_b, x_c e x_d anteriormente calculados e raios dados pelas distâncias do átomo i aos átomos em B_i.

Os autores do algoritmo *BuildUp* sugerem a linearização deste sistema subtraindo-se a Eq.(2) das demais equações. Com isto, obtemos o sistema linear:

$$\begin{aligned} (x_b - x_a)^t x_i &= (d_{ia}^2 2 - d_{ib}^2 2 + ||x_b||^2 - ||x_a||^2)/2, \\ (x_c - x_a)^t x_i &= (d_{ia}^2 2 - d_{ic}^2 2 + ||x_c||^2 - ||x_a||^2)/2, \\ (x_d - x_a)^t x_i &= (d_{ia}^2 2 - d_{id}^2 2 + ||x_d||^2 - ||x_a||^2)/2. \end{aligned}$$

A primeira observação é que o problema linearizado e o problema quadrático não são equivalentes. De fato, o problema linear pode ter solução (real), mesmo quando o problema quadrático não possui solução real. Por exemplo, se as distâncias relativas aos átomos x_a, x_b, x_c e x_d e as distâncias d_{ia}, d_{ib}, d_{ic} e d_{ia} violarem alguma desigualdade triangular, o problema quadrático não terá solução real. No entanto, a não-singularidade da matriz do sistema linear e, portanto, a existência de solução depende apenas da independência linear dos vetores $u = x_b - x_a$, $v = x_c - x_a$ e $w = x_d - x_a$ não levando em consideração os raios r_a, r_b, r_c e r_d.

É claro que podemos verificar diretamente se a solução do sistema linear é também solução do sistema quadrático. Contudo, caso não seja, não há uma forma direta de identificar qual das coordenadas x_a, x_b, x_c ou x_d deve ser modificada. Mais ainda, a modificação de qualquer coordenada previamente calculada, digamos x_a, implica em potencialmente violar todas as restrições que envolvam o átomo a e outros átomos já fixados. Em outras palavras, não é possível resolver localmente o problema de inviabilidade de uma solução parcial. Além disso, o erro em uma das coordenadas x_a, x_b, x_c ou x_d só é detectado quando não conseguimos fixar o átomo i. Ou seja, o erro só é percebido várias iterações após os cálculos das coordenadas x_a, x_b, x_c e x_d, as quais podem ter sido utilizadas para calcular muitas outras coordenadas espalhando assim o erro por toda a estrutura.

Por outro lado, ainda que os sistemas quadráticos sejam consistentes, pode ser que a distância entre as coordenadas de dois átomos, digamos x_i e x_j, calculadas independentemente usando bases B_i e B_j diferentes não atenda satisfatoriamente à restrição

$l_{ij} \leq ||x_i - x_j|| \leq u_{ij}$, devido ao cancelamento de dígitos significativos e aos erros de truncamento acumulados durante a execução do algoritmo de fixação. Neste caso, sabemos apenas que as coordenadas x_i e x_j são incompatíveis, mas não há informação local que identifique quais coordenadas em B_i e B_j devem ser corrigidas.

Uma pergunta natural é se os erros locais em instâncias reais do PGDM de fato ocorrem e, mais ainda, se eles impedem a construção de soluções. Aqui, o ponto central é o tamanho das instâncias. Quando resolvemos instâncias que envolvem um número reduzido de átomos, os erros tendem a manter-se em nível aceitável. Contudo, o fato inexorável da computação científica em ponto flutuante é que o erro aumenta com o número de operações aritméticas realizadas. Sendo assim, quanto maior o número de átomos, maior será o erro acumulado já que os novos átomos serão utilizados no cálculo dos demais. Concluímos, portanto, que a intensidade do erro deve ser maior nas coordenadas calculadas tardiamente e mais acentuados em instâncias que envolvam um grande número de átomos.

Sendo assim, os erros nas soluções dos algoritmos iterativos de fixação possuem uma fonte local associada ao método de fixação, e outra global cumulativa pelo uso das coordenadas anteriormente calculadas no cômputo das novas coordenadas. No caso do *BuildUp*, o primeiro erro está ligado ao sistema linear e, o segundo, à escolha da base B_i.

3 O algoritmo *BuildUpOpt*

Propomos duas medidas complementares para mitigar as limitações dos algoritmos de fixação com informação local. A primeira delas é refinar soluções parciais utilizando uma formulação de otimização como, por exemplo,

$$(P) \min_{x} \sum_{(i,j)\in\tilde{E}} \max(l_{ij} - ||x_i - x_j||, 0) + \max(||x_i - x_j|| - u_{ij}), \tag{3}$$

onde $\tilde{E}$ representa o conjunto de arestas incidentes em átomos já fixados. O refinamento deve ocorrer sempre que, ao fixarmos um átomo k, verificarmos que uma restrição não tenha sido satisfeita com precisão $\epsilon > 0$ adequada, ou seja, $l_{ik} - ||x_i - x_k|| > \epsilon$ ou $||x_i - x_k|| - u_{ik} > \epsilon$ para algum átomo i já fixado. A formulação de otimização utilizada no refinamento deve levar em consideração apenas o conjunto $\tilde{E} \subset E$ formado pelas arestas envolvendo os átomos fixados até o momento. A ideia é considerar uma tolerância baixa, digamos $\epsilon = 10^{-3}$. Neste caso, a solução parcial estará próxima de um mínimo global da formulação de otimização, sendo possível obtê-lo em poucas iterações mesmo utilizando rotinas de minimização local tradicionais.

Cada chamada da rotina de minimização terá custo pelo menos de $O(|\tilde{E}|)$, onde $|\tilde{E}|$ representa o número de arestas do subproblema (P), enquanto que a solução do sistema quadrático linearizado tem custo $O(1)$. Ou seja, o processo de refino da solução será a etapa mais cara do algoritmo modificado. Para reduzirmos o número de chamadas da rotina de minimização, precisaríamos reduzir a propagação dos erros. Uma forma de fazê-lo é dar preferência ao uso das coordenadas que apresentem os menores erros na definição das bases utilizadas no cálculo das coordenadas. Não é possível calcular o erro cometido por uma coordenada x_k, antes que todos os átomos envolvendo arestas incidentes em k

DOI: 10.5540/03.2018.006.01.0360

tenham sido fixados, mas podemos estimar o erro cometido até a iteração corrente medindo a violação das restrições envolvendo os átomos já fixados.

Para facilitar a compreensão, fixemos os conceitos de *resíduo relativo de uma aresta* (R_{ij}) e *resíduo relativo total de um átomo* (R_k). Para cada aresta $(i, j) \in E$ cujas coordenadas x_i e x_j tenham sido calculadas, definimos seu resíduo relativo como sendo

$$R_{ij} = \frac{2\max(0, l_{ij} - ||x_i - x_j||, ||x_i - x_j|| - u_{ij})}{l_{ij} + u_{ij}}. \tag{4}$$

Ou seja, R_{ij} indica o resíduo relativo associado à restrição $l_{ij} \leq ||x_i - x_j|| \leq u_{ij}$.

O resíduo relativo total de um átomo k é definido como a soma dos resíduos relativos das arestas incidentes sobre ele, $R_k = \sum_{i \in V_k} R_{ik}$, onde V_k representa o conjunto de vizinhos já fixados do átomo k.

Para reduzir a propagação dos erros no cálculo de um átomo k, propomos que sejam utilizados como base os elementos de V_k que tenham os menores resíduos totais. Desta forma, ainda que as informações sejam locais, estaremos utilizando os dados mais precisos disponíveis, reduzindo potencialmente o número de chamadas das rotinas de refinamento.

As ideias apresentadas foram incorporadas no algoritmo *BuildUpOpt* apresentado no Quadro 1. Na linha 1, o conjunto `C`, formado pelos átomos a serem fixados, é inicializado com uma clique cujas coordenadas dos átomos é fixada usando o algoritmo para instâncias completas. Na linha 3, o vetor `p` que armazena o número de vizinhos fixados de cada átomo é inicializado. Na linha 7, a base selecionada leva em consideração os resíduos locais de cada um dos vizinhos já fixados do átomo i. Já a linha 9 aplica a estratégia de refino da solução parcial. São as linhas 7 e 9 que diferenciam o algoritmo *BuildUpOpt* da versão original.

```
BuildUpOpt
 1  Encontre uma clique C com 4 vértices;
 2  Determine as coordenadas dos vértices em C;
 3  Para cada vértice i em C, faça p(i) = 5;
 4  Enquanto |C| > 0,
 5     Para cada i em C,
 6        Se p(i) == 4,
 7           Determine a base B_i de menor resíduo relativo total para i em C
 8           Determine as coordenadas de k usando B (Eqs.6-8);
 9           Se R_i > eps, refine a solução resolvendo o problema (P);
10        Para cada vizinho k de i em G,
11           Faça p(k) = p(k) + 1;
12           Se p(k) == 4, insira k em C;
13        Remova i de C;
```

Quadro 1: Algoritmo *BuildUpOpt*.

DOI: 10.5540/03.2018.006.01.0360

4 Experimentos numéricos

O algoritmo *BuildUpOpt* foi implementado em linguagem C e compilado com GCC versão 4.8.4. A rotina de otimização utilizada foi o gradiente conjugado do pacote GNU Scientific Library. Os sistemas lineares foram resolvidos usando método de Cramer. Os testes foram realizados em um notebook Dell XPS com 8 GB de memória RAM, sistema operacional Ubuntu e processador Intel Core i5-3337U de 1,8 GHz.

Comparamos a versão original do *BuildUp* com uma versão que incorpora apenas a estratégia de ordenação (*BuildUp+Ord*), com uma outra com apenas a estratégia de refino (*BuildUp+Ref*) e, finalmente, com o BuildUpOpt que aplica as duas estratégias.

As instâncias foram geradas com proteínas reais obtidas no banco de dados *Protein Data Bank* [1]. Cada instância foi gerada considerando-se as distâncias menores que 5 Å.

Na Tabela 1, podemos ver os dados sobre as instâncias consideradas e os resultados experimentais. Nas quatro primeiras colunas, encontramos o código de identificação das proteínas (*PDB ID*), o número de átomos (*Nodes*), o número de distâncias consideradas (*Edges*) e o grau máximo dos átomos (*MaxDeg*).

Os resultados dos experimentos numéricos estão na Tabela 1, onde *MaxRes* representa o resíduo máximo por átomo (em notação científica E), *Time* representa o tempo de execução em segundos. O erro aumenta consideravelmente com o tamanho das instâncias, chegando a exceder a capacidade de representação em ponto flutuante (*overflow*). Podemos ver que a estratégia de refinamento da solução (minimização) controla efetivamente os resíduos. Finalmente, fica também evidente que a estratégia de seleção da base de fixação reduz significativamente o tempo de execução, ao diminuir o número de chamadas da rotina de minimização, mas que isoladamente não é capaz de controlar a magnitude dos resíduos ainda que mitigue seus efeitos. O experimento com a proteína 2FUI evidencia a necessidade de conjugar as duas estratégias, pois a ordenação isoladamente apresenta alto erro relativo, já o refinamento consegue controlar o erro, mas o tempo de execução é consideravelmente maior. Quando unimos as duas estratégias, conseguimos reduzir o erro e também o tempo de execução.

Tabela 1: A coluna *PDB ID* contém o código da proteína, *Nodes* indica o número de átomos, *Edges* representa o número de arestas, *MaxDeg* indica o grau máximo dos átomos, a coluna *MaxRes* apresentam os resíduos máximos (em notação científica E), e *Time* contém o tempo em segundos.

				BuildUp		***BuildUp+Ord***		***BuildUp+Ref***		***BuildUpOpt***	
PDB ID	**Nodes**	**Edges**	**MaxDeg**	**MaxRes**	**Time**	**MaxRes**	**Time**	**MaxRes**	**Time**	**MaxRes**	**Time**
4NIN	50	326	19	2,54E-03	0,00	8,55E-08	0,00	5,11E-04	0,20	8,55E-08	0,00
1PLW	75	1021	47	1,38E+01	0,02	3,07E-06	0,02	9,99E-04	0,06	3,07E-06	0,02
5CYT	800	9144	39	1,02E+22	0,07	8,52E-05	0,06	3,36E-05	14,77	8,52E-05	0,06
2MYQ	691	11916	62	2,71E+20	0,13	6,22E+03	0,14	1,17E-03	60,94	1,57E-01	26,29
2FUI	938	19715	73	5,49E+36	0,32	7,81E+01	0,30	4,29E-04	77,98	8,27E-04	3,60
1WF2	1445	29669	71	*overflow*	0,46	2,62E+02	0,53	4,55E+00	1.818,75	8,65E-03	132,88
5J1H	2657	30418	40	*overflow*	0,20	2,73E+12	0,22	9,34E-04	207,11	2,87E-02	66,41
5FY4	3717	44590	47	5,04E+36	0,28	6,00E+04	0,33	3,21E-06	213,32	1,73E-05	48,08
3KGF	7359	94337	61	*overflow*	0,69	1,52E+03	0,91	7,58E-04	820,70	1,25E-04	139,67
5HFA	8421	102724	44	*overflow*	0,91	*overflow*	0,86	1,81E-04	1.542,84	1,73E-03	230,04
3KGV	20320	195453	34	3,80E+01	0,01	7,96E-08	0,01	1,89E-05	0,76	7,96E-08	0,02

DOI: 10.5540/03.2018.006.01.0360

5 Conclusão e trabalhos futuros

Apresentamos duas estratégias para a mitigação dos erros em algoritmos baseados em fixações iterativas para a resolução do problema de geometria de distâncias moleculares. Uma delas é o ranqueamento dinâmico dos átomos para mitigar o espalhamento dos erros, e a outra é a minimização local para garantir a precisão da solução. Estas estratégias foram implementadas na versão modificada do algoritmo BuildUp, chamada BuildUpOpt. Nos experimentos numéricos realizados, a versão modificada obteve soluções com erros consideravelmente menores que a versão original.

Em trabalhos futuros, desejamos explorar o paralelismo e aplicar variações destas estratégias em outros algoritmos de fixação iterativos.

As instâncias e implementações dos algoritmos utilizados neste trabalho podem ser obtidas no repositório GitHub https://goo.gl/pDyL4c.

Os autores agradecem o suporte do CNPq, CAPES, FAPESP e da Universidade Federal do Ceará, da Universidade Estadual de Campinas e da Universidade do Estado do Rio de Janeiro.

Referências

[1] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I.N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, vol. 28, 235–242, (2000), DOI:10.1038/80734.

[2] P. Biswas, K. C. Toh, and Y. Ye. A distributed sdp approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *SIAM Journal on Scientific Computing*, vol. 30, 1251–1277, (2008), DOI:10.1137/05062754X.

[3] G. M. Crippen, T. F. Havel, et al. Distance geometry and molecular conformation, *Research Studies Press Taunton*, (1988).

[4] Q. Dong and Z. Wu. A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization*, vol. 22, 365–375, (2002), DOI:10.1023/A:1013857218127.

[5] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, vol. 56, 3–69, (2014), DOI:10.1137/120875909.

[6] L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, vol. 18, 33–51, (2011), DOI:10.1111/j.1475-3995.2009.00757.x.

[7] J. J. Moré and Z. Wu. Distance geometry optimization for protein structures. *Journal of Global Optimization*, vol. 15, 219–234, (1999), DOI:10.1023/A:1008380219900.

[8] M. Souza, C. Lavor, A. Muritiba, and N. Maculan. Solving the molecular distance geometry problem with inaccurate distance data. *BMC Bioinformatics*, vol. 14, S7, (2013), DOI:10.1186/1471-2105-14-S9-S7.

