Longest Common Subsequence Aplicada à Comparação de Proteínas

Jessica Costa, Mateus Pereira, Kele Belloze, Eduardo Bezerra

[1] Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Rio de Janeiro, Brasil

{jessica.costa, mateus.pereira}@eic.cefet-rj.br

{kele.belloze, eduardo.silva}@cefet-rj.br

Abstract. *This work compares the performance of two algorithms in the search for the longest common subsequence (LCS) among protein sequences from living organisms. We used sequences of different sizes and carried out tests with the Wagner-Fischer and Needleman-Wunsch algorithms. The results revealed that as the input size increases, both execution time of algorithms increases linearly, providing the best results for the Wagner-Fischer algorithm.*

Resumo. *Este trabalho realiza uma comparação do desempenho de dois algoritmos na busca da maior subsequência comum (LCS - Longest Common Subsequence) entre sequências de proteínas de organismos vivos. Para isso foram utilizadas sequências de diversos tamanhos e realizados testes com os algoritmos de Wagner-Fischer e Needleman-Wunsch. Foi observado que conforme o tamanho da entrada cresce, o tempo de execução aumenta linearmente para ambos os algoritmos, contudo, apresentando melhores resultados para o algoritmo de Wagner-Fischer.*

1. Introdução

A comparação de sequências para identificar similaridade é uma das tarefas importantes da bioinformática na identificação de relações de homologia, ou seja, de mesma descendência [Needleman and Wunsch 1970, Koonin 2005]. Essa comparação pode ser realizada par a par em um aminoácido de cada proteína ou através da maior sequência de aminoácidos de uma proteína que pode coincidir com outra [Needleman and Wunsch 1970]. Essa tarefa em biologia molecular é chamada de alinhamento de sequências e umas das possíveis abordagens computacionais para solução é chamada de *Longest Common Subsequence* (LCS) ou Maior Subsequência Comum.

O problema de LCS tem a finalidade de encontrar a maior subsequência de elementos em comum entre duas ou mais sequências. É importante ressaltar que os elementos não precisam ser contíguos, mas a ordem em que os mesmos aparecem nas sequências deve ser respeitada. Para ilustrar com um exemplo, sejam dois trechos de sequências de proteínas *MDSPMEKFIQ* e *MFSFHERMKK*, a LCS entre os trechos é *MSE*.

A computação do LCS e seus derivados pertence aos problemas NP e especificamente para múltiplas sequências de entrada, ele é classificado como um problema NP-Difícil [Kawade et al. 2018, Garey and Johnson 1979]. Umas das abordagens para a resolução desse tipo de problema é a programação dinâmica que consegue resolver no

tempo proporcional ao produto do tamanho das entradas e usa uma tabela especial de tamanho $m \times n$, onde m e n são os tamanhos das duas sequências [Kawade et al. 2018].

Diante do exposto, este trabalho tem o objetivo de realizar uma análise experimental de dois algoritmos que resolvem o problema de LCS. Os algoritmos escolhidos foram Wagner-Fischer [Wagner and Fischer 1974] e Needleman-Wunsch [Needleman and Wunsch 1970] por trabalharem principalmente com alinhamento global. Para avaliar o desempenho dos algoritmos, sequências de proteínas de diversos tamanhos foram utilizadas. O artigo está organizado em cinco seções, incluindo essa introdução. Na Seção 2 é realizada a revisão bibliográfica, onde são apresentadas a fundamentação teórica e os trabalhos relacionados. Na Seção 3 é descrita a metodologia do trabalho. A Seção 4 discute os resultados encontrados e, na Seção 5, são apresentadas as considerações finais.

2. Revisão Bibliográfica

As subseções a seguir se referem aos conceitos importantes para compreensão do trabalho, principalmente sobre alinhamento de sequências e aos trabalhos relacionados a aplicações do *Longest Common Subsequence* em Bioinformática.

2.1. Fundamentação Teórica

Alinhamento de sequências é um método para organizar sequências de DNA, RNA ou proteínas com a intenção de identificar regiões de similaridade, como subsequências, que podem indicar uma região conservada no ciclo de evolução [Reddy 2020]. Com relação ao alinhamento de pares de sequência, existe o alinhamento local e o alinhamento global que podem ser baseados em programação dinâmica ou métodos heurísticos [Reddy 2020].

O alinhamento local é relativo à comparação de partes das sequências e costuma ser muito utilizado para encontrar regiões de similaridade [Reddy 2020]. O algoritmo Smith-Waterman é uma técnica baseada em programação dinâmica e muito utilizada para o alinhamento local. A complexidade desse algoritmo em termos de tempo ao comparar duas sequências é $O(mn)$, onde m e n denotam os tamanhos das duas sequências da comparação [Shafiq et al. 2016]. Ainda sobre alinhamento local, existem técnicas que utilizam métodos heurísticos para reduzir o espaço de busca, como é o caso da ferramenta *Basic Local Alignment Search Tool* (BLAST) que executa de forma mais rápida quando comparado ao algoritmo Smith-Waterman [Zhang et al. 2017].

O alinhamento global encontra regiões de similaridade entre duas sequências inteiras e é computacionalmente mais custoso [Reddy 2020]. Um algoritmo muito utilizado para alinhamento global é o Needleman-Wunsch, proposto na década de 70 por Saul Needleman e Christian Wunsch e que também utiliza programação dinâmica [Needleman and Wunsch 1970]. O alinhamento global é normalmente referido como o primeiro método na literatura das áreas biológica e biomédica para determinar homologia [Shafiq et al. 2016, Needleman and Wunsch 1970]. O alinhamento global também pode ser realizado através de métodos heurísticos como é o caso do Maximal Unique Match-mer (MUmmer), introduzido nos anos 2000 e utilizado principalmente para descobrir polimorfismos de nucleotídeo único (SNPs - *Single Nucleotide Polymorphism*) [Marçais et al. 2018, Reddy 2020].

Algoritmos de LCS podem ser utilizados para o alinhamento global. Neste contexto, os algoritmos de Wagner-Fischer [Wagner and Fischer 1974] e de Needleman-

Wunsch foram os primeiros a utilizar programação dinâmica. Esses algoritmos são apresentados nas seções 2.1.1 e 2.1.2.

2.1.1. Algoritmo de Wagner-Fischer

O algoritmo de Wagner-Fischer consiste em preencher uma matriz de inteiros $M(m+1 \times n+1)$, onde m e n são os tamanhos das sequências A e B, respectivamente. Cada célula $M_{i,j}$ é preenchida de acordo com a Equação 1.

$$M_{ij} = \begin{cases} M_{i-1,j-1} + 1 & \text{se } A_{i-1} = B_{j-1} \\ max(M_{i-1,j}, M_{i,j-1}) & \text{caso contrário} \end{cases} \quad (1)$$

Como mostrado na Equação 1, se $A_{i-1} = B_{j-1}$, a célula (i, j) recebe o valor da célula $M_{i-1,j-1}$ acrescido de 1. Do contrário, a célula recebe o maior valor entre as células $M_{i-1,j}$ e $M_{i,j-1}$. Células da primeira linha e primeira coluna recebem o valor 0. Uma vez que a matriz M é preenchida, o Algoritmo 1 pode ser utilizado para recuperar a LCS.

Algorithm 1: Recuperação da LCS (Wagner-Fischer)

```
Data: Matriz M, Sequência A
Result: LCS S
i ← m + 1;
j ← n + 1;
while M[i, j] ≠ 0 do
    if M[i, j] ≠ 0 and M[i, j] > M[i − 1, j] and M[i, j] > M[i, j − 1] then
        S ← A[i − 1] + S;
        i ← i − 1;
        j ← j − 1;
    else
        if M[i − 1, j] == M[i, j] then
            i ← i − 1;
        else
            j ← j − 1;
```

O Algoritmo 1 recebe como entrada a sequência A e a matriz M, e percorre M começando da célula $M_{m+1,n+1}$. O valor de uma célula $M_{i,j}$ é comparado com as células $M_{i-1,j}$ e $M_{i,j-1}$. Se $M_{i,j} > M_{i-1,j}$ e $M_{i,j} > M_{i,j-1}$, o elemento A_{i-1} é adicionado à LCS e i e j são decrescidos de 1. Do contrário, o algoritmo move-se para a célula adjacente com maior valor, até chegar na primeira linha ou primeira coluna. $M_{i,j} > M_{i-1,j}$ e $M_{i,j} > M_{i,j-1}$ implica em $A_{i-1} = B_{j-1}$, o que indica que apenas uma das sequências é necessária para a recuperação do LCS. Em nossa implementação, a sequência A foi utilizada. O algoritmo de Wagner-Fischer possui complexidades de tempo e memória $O(mn)$, onde m e n são os tamanhos das duas sequências a serem comparadas.

2.1.2. Algoritmo de Needleman-Wunsch

Um dos algoritmos mais utilizados para alinhamento global é o de Needleman-Wunsch [Needleman and Wunsch 1970]. O mesmo é similar ao algoritmo de Wagner-Fischer, porém a forma de preencher a matriz é diferente. Em contraste, a matriz utilizada por Needleman-Wunsch é $M(n+1 \times m+1)$, onde n e m são os tamanhos das sequências B e A, respectivamente. A Equação 2 descreve o preenchimento da matriz M.

$$M_{ij} = \begin{cases} max(M_{i-1,j-1} + match, M_{i-1,j} + gap, M_{i,j-1} + gap) & \text{se } A_{i-1} = B_{j-1} \\ max(M_{i-1,j-1} + miss, M_{i-1,j} + gap, M_{i,j-1} + gap) & \text{caso contrário} \end{cases} \quad (2)$$

Na Equação 2, $match$, $miss$ e gap são pesos passados como parâmetros para o algoritmo. A célula $M_{0,0}$ recebe valor 0, então cada linha e coluna são preenchidas de acordo com a Equação 2. Uma vez que M é preenchida por completo, o Algoritmo 2 pode ser utilizado para recuperar a LCS.

Algorithm 2: Recuperação da LCS (Needleman-Wunsch)

```
Data: Matriz M, Sequência A, Sequência B
Result: LCS S
i ← n + 1;
j ← m + 1;
while i > 0 and j > 0 do
    if A[j − 1] = B[i − 1] then
        S ← A[j − 1] + S;
        i ← i − 1;
        j ← j − 1;
    else
        upper ← M[i − 1, j];
        left ← M[i, j − 1];
        diag ← M[i − 1, j − 1];
        next ← max(upper, left, diag);
        if next = upper then
            i ← i − 1;
        else if next = left then
            j ← j − 1;
        else
            i ← i − 1;
            j ← j − 1;
```

O Algoritmo 2 recebe como entrada as duas sequências A e B, além da matriz de inteiros M. Para recuperar S através de M, o Algoritmo 2 percorre a matriz a partir da última célula e compara os elementos de A e B referentes à célula atual. Se os elementos são iguais, uma cópia dos mesmos é adicionada ao início da LCS e move-se para a célula

diagonal superior esquerda. Do contrário, nada é adicionado à LCS e move-se para a célula de maior valor à esquerda, acima, ou à diagonal superior esquerda. O algoritmo de Needleman-Wunsch, assim como o de Wagner-Fischer, possui complexidade de tempo e memória $O(mn)$.

2.2. Trabalhos Relacionados

Shikder e colaboradores [Shikder et al. 2019] propuseram implementações utilizando paralelismo para encontrar as LCS em sequências de DNA para obter vantagens em termos de tempo de execução, monetários e custos. Para isso foram propostas três versões de paralelismo, a primeira com memória compartilhada utilizando a OpenMP, a segunda com memória distribuída através do MPI e uma última versão híbrida com memória compartilhada e distribuída. Os resultados experimentais mostraram que a implementação com memória compartilhada obteve um *speedup* absoluto duas vezes mais rápido que a implementação sequencial do algoritmo.

Wei e colaboradores [Wei et al. 2020] trabalharam com a aplicação do *Multiple Longest Common Subsequence* (MLCS) que é a busca da LCS em múltiplas sequências. Para isso o problema do MLCS foi transformado na busca do caminho mais longo de um grafo direcionado acíclico (DAG) o que otimizou o espaço de memória e o tempo de pesquisa. Experimentos empíricos foram realizados em um conjunto padrão de referência de sequências de DNA e sequências de proteínas. Os resultados experimentais demonstraram que o modelo e o algoritmo propostos superaram os algoritmos principais relacionados, principalmente para problemas de MLCS de grande escala.

Issa e Elaziz [Issa and Elaziz 2020] propuseram um modelo que aprimora uma ferramenta de análises de dados, a *Fragmented Local Aligner Technique* (FLAT), para detectar a *Longest Common Consecutive Subsequence* (LCCS) entre pares de dados de sequências biológicas de vírus de COVID-19 e outras viroses para auxiliar em processos bioquímicos e biológicos. Os resultados experimentais demonstram que o modelo proposto conseguiu produzir o melhor LCCS contra outros algoritmos usando LCCS real que usam o algoritmo Smith-Waterman como referência.

Os trabalhos mencionados foram selecionados devido às suas recentes publicações e aplicações utilizando LCS. No entanto, em uma pesquisa exploratória da literatura não foram observados trabalhos que realizaram avaliação de algoritmos clássicos de programação dinâmica aplicados ao problema de LCS, em termos de complexidade, principalmente com avaliação de tempo. Diante disso, o presente trabalho propôs esse tipo de avaliação para auxiliar na seleção de algoritmos com melhor desempenho para comparação de sequências e assim apoiar a tarefa de alinhamento global.

3. Metodologia

A metodologia deste trabalho seguiu duas fases. Na primeira fase, os algoritmos de Wagner-Fischer e Needleman-Wunsch foram implementados na linguagem Python. O código está disponível em repositório do Github[1]. Na segunda fase, uma análise experimental foi realizada com sequências de proteínas e o tempo de execução foi coletado por meio da função *time* do pacote *time* da linguagem Python. A execução dos experimentos

[1]Disponível em https://github.com/mdevino/lcs.git

foi realizada em uma máquina, utilizando um subsistema Linux Ubuntu 20.4 (WSL), com processador Intel Core i9-10900k e 32 GB de RAM.

Para a segunda fase, foram utilizadas proteínas de dois organismos, *Schistossoma mansoni*, um helminto que causa a patologia da esquistossomíase, e *Saccharomyces cerevisiae*, uma levedura utilizada na fabricação de pães e cervejas. O proteoma (conjunto de proteínas) do *Schistossoma mansoni* foi extraído do Uniprot, um banco de dados de sequências de proteínas anotadas com informações funcionais de alta qualidade [Consortium 2020], enquanto que o proteoma da *Saccharomyces cerevisiae* foi extraído do Ensembl, um banco de dados de anotação genômica para apoiar pesquisas em genômica comparativa e disseminar dados genômicos para espécies de vertebrados [Howe et al. 2020]. Os dados são disponibilizados publicamente em ambos os bancos de dados. Os pesos utilizados para o algoritmo Needleman-Wunsch foram 1, -1 e -2 para os parâmetros *match*, *mismatch* e *gap*, respectivamente. A escolha destes valores foi empírica.

4. Resultados

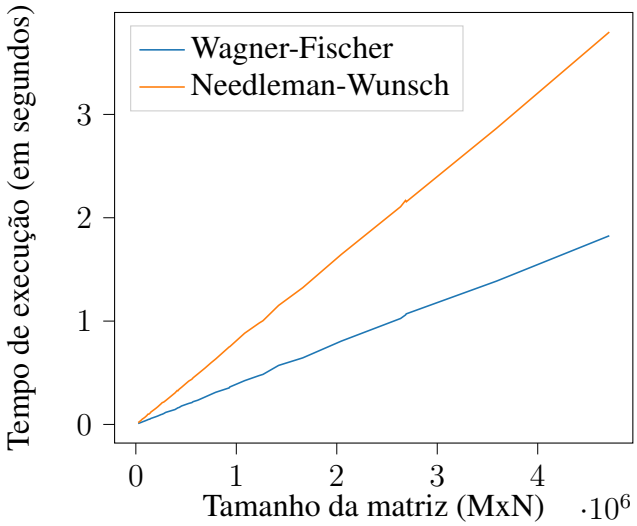
Ao todo, foram selecionadas nove sequências de cada organismo, de tamanhos variados (entre 94 e 2258), gerando matrizes de até $4.712.446$ células. O tamanho da cada proteína utilizada pode ser encontrado na Tabela 1, a qual apresenta o nome do organismo, um identificador para a sequência e o tamanho de cada sequência.

Tabela 1. Sequências de Proteínas

Organismo e Identificador	Tamanho da Sequência
S. mansoni 1	359
S. mansoni 2	187
S. mansoni 3	94
S. mansoni 4	588
S. mansoni 5	796
S. mansoni 6	681
S. mansoni 7	2258
S. mansoni 8	1286
S. mansoni 9	1693
S. cerevisiae 1	334
S. cerevisiae 2	147
S. cerevisiae 3	244
S. cerevisiae 4	732
S. cerevisiae 5	700
S. cerevisiae 6	773
S. cerevisiae 7	1167
S. cerevisiae 8	2087
S. cerevisiae 9	1590

Cada sequência de um organismo foi comparada com todas as sequências do outro, excluindo-se as comparações repetidas, gerando um total de 45 testes para cada algoritmo. Os resultados do experimento podem ser observados na Figura 1.

Figura 1. Comparação de tempo de execução



A Figura 1 ilustra o comportamento do tempo de execução em relação ao tamanho de entrada. No eixo horizontal, a quantidade de células das matrizes são apresentadas, considerando uma ordem de grandeza de 10^6. No eixo vertical, o tempo de execução para cada matriz, em segundos, é apresentado.

Percebe-se que, embora os dois algoritmos analisados possuam complexidade de tempo linear, o tempo de execução do algoritmo de Needleman-Wunsch cresce mais rápido que o de Wagner-Fischer. Os resultados sugerem que, ao menos para a tarefa de encontrar a LCS, o algoritmo de Wagner-Fischer é mais eficiente.

5. Conclusões

Este trabalho mostrou o comportamento dos algoritmos de Wagner-Fischer e Needleman-Wunsch para encontrar a LCS de sequências de proteínas de duas espécias, *Schistossoma mansoni* e *Saccharomyces cerevisiae*. De acordo com os testes realizados, embora os dois algoritmos apresentados possuam complexidade de tempo e memória $O(mn)$, o algoritmo de Wagner-Fischer pareceu ser mais eficiente para a tarefa de LCS. Além disso, o algoritmo de Needleman-Wunsch encontrou LCSes menores que o de Wagner-Fischer. Uma hipótese para tal comportamento é que isso possa ter sido causado pela escolha de pesos atribuídos aos parâmetros do algoritmo Needleman-Wunsch, o que se torna um tema para ser explorado em trabalhos futuros. Outro trabalho futuro refere-se a aplicar a metodologia proposta utilizando os proteomas completos das duas espécies analisadas.

Referências

Consortium, T. U. (2020). UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman.

Howe, K. L., Achuthan, P., Allen, J., Allen, J., Alvarez-Jarreta, J., Amode, M. R., Armean, I. M., Azov, A. G., Bennett, R., Bhai, J., Billis, K., Boddu, S., Charkhchi, M.,

Cummins, C., Da Rin Fioretto, L., Davidson, C., Dodiya, K., El Houdaigui, B., Fatima, R., Gall, A., Garcia Giron, C., Grego, T., Guijarro-Clarke, C., Haggerty, L., Hemrom, A., Hourlier, T., Izuogu, O. G., Juettemann, T., Kaikala, V., Kay, M., Lavidas, I., Le, T., Lemos, D., Gonzalez Martinez, J., Marugán, J. C., Maurel, T., McMahon, A. C., Mohanan, S., Moore, B., Muffato, M., Oheh, D. N., Paraschas, D., Parker, A., Parton, A., Prosovetskaia, I., Sakthivel, M. P., Salam, A., Schmitt, B. M., Schuilenburg, H., Sheppard, D., Steed, E., Szpak, M., Szuba, M., Taylor, K., Thormann, A., Threadgold, G., Walts, B., Winterbottom, A., Chakiachvili, M., Chaubal, A., De Silva, N., Flint, B., Frankish, A., Hunt, S. E., IIsley, G. R., Langridge, N., Loveland, J. E., Martin, F. J., Mudge, J. M., Morales, J., Perry, E., Ruffier, M., Tate, J., Thybert, D., Trevanion, S. J., Cunningham, F., Yates, A. D., Zerbino, D. R., and Flicek, P. (2020). Ensembl 2021. *Nucleic Acids Research*, 49(D1):D884–D891.

Issa, M. and Elaziz, M. A. (2020). Analyzing covid-19 virus based on enhanced fragmented biological local aligner using improved ions motion optimization algorithm. *Applied Soft Computing*, 96:106683.

Kawade, G., Sahu, S., Upadhye, S., Korde, N., and Motghare, M. (2018). An analysis on computation of longest common subsequence algorithm.

Koonin, E. (2005). Orthologs, paralogs, and evolutionary genomics 1. *Annual review of genetics*, 39:309–38.

Marçais, G., Delcher, A. L., Phillippy, A. M., Coston, R., Salzberg, S. L., and Zimin, A. (2018). Mummer4: A fast and versatile genome alignment system. *PLOS Computational Biology*, 14:1–14.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453.

Reddy, B. (2020). *Bioinformatics & Pairwise Sequence Alignment: Local and Global Sequence Alignment Algorithms*. Independently published.

Shafiq, M., Polo, J., Dickov, B., and Hussain, T. (2016). Modeling and performance evaluation of smith-waterman algorithm. In *2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 191–198.

Shikder, R., Thulasiraman, P., Irani, P., and Hu, P. (2019). An openmp-based tool for finding longest common subsequence in bioinformatics. *BMC Research Notes*, 12.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Wei, S., Wang, Y., Yang, Y., and Liu, S. (2020). A path recorder algorithm for multiple longest common subsequences (mlcs) problems. *Bioinformatics (Oxford, England)*, 36.

Zhang, J., Misra, S., Wang, H., and Feng, W.-c. (2017). Eliminating irregularities of protein sequence search on multicore architectures. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 62–71.