Paralelização da Geração de Consistência em Alinhamentos Múltiplos de Sequências Genéticas

Mario João Jr.[1,3], **Alexandre C. Sena**[2], **Vinod E.F. Rebello**[3]

[1] Laboratório Médico de Pesquisas Avançadas, UERJ – Rio de Janeiro – RJ – Brasil

[2]Instituto de Matemática e Estatística, UERJ – Rio de Janeiro – RJ – Brasil

[3]Instituto de Computação, UFF – Niterói – RJ – Brasil

junior@lampada.uerj.br, asena@ime.uerj.br, vinod@ic.uff.br

Resumo. *O Alinhamento Múltiplo de Sequências genéticas é uma etapa essencial na resolução de vários problemas da área de bioinformática. Devido à sua complexidade exponencial, heurísticas são utilizadas. A que obtém os melhores resultados, mas possui o maior custo computacional, é o Alinhamento baseado em Consistência. Este trabalho apresenta a paralelização da geração da consistência, fase fundamental para esta heurística de alinhamento múltiplo. Os resultados obtidos mostram o desempenho da paralelização proposta, sendo capaz de reduzir o tempo de execução da consistência significativamente.*

1. Introdução

Ao processo de alinhamento de mais de duas sequências genéticas (DNA, RNA ou proteínas) para identificar as similaridades que refletem aspectos relacionados à evolução das espécies e às funcionalidades das estruturas orgânicas [Edgar and Batzoglou 2006] dá-se o nome de Alinhamento Múltiplo de Sequências (*Multiple Sequence Alignment* - MSA). O alinhamento múltiplo é um passo essencial utilizado em diferentes áreas de pesquisa da bioinformática, como por exemplo, evolução das espécies [Thompson et al. 2011], análise de domínios [Thompson et al. 2011], inferência filogenética [Mirarab and Warnow 2011], e predição de funcionalidades das proteínas e/ou de suas estruturas [Notredame et al. 2000].

Devido à complexidade para encontrar o alinhamento múltiplo ótimo ser $O(l^n)$ [Wang and Jiang 1994], onde $n > 2$ é o número de sequências e l é o tamanho médio delas, heurísticas são utilizadas, sendo a mais comum o Alinhamento Progressivo [Feng and Doolittle 1987]. Essa heurística é a base para diversas ferramentas de alinhamento múltiplo, porém possui características que fazem com que decisões tomadas no início do processo possam não levar às melhores soluções. Para mitigar este problema e melhorar a qualidade do MSA gerado [João Jr et al. 2023b], foram criadas heurísticas de Alinhamento Baseado em Consistência. Essas heurísticas partem do princípio que um alinhamento múltiplo é consistente. Isto é, se três sequências A, B e C estão alinhadas, o resíduo A_i está alinhado com o resíduo C_k e o resíduo B_j está alinhado com o resíduo C_k, então A_i deve estar alinhado com B_j. Assim, para alinhar duas sequências A e B, os métodos de Alinhamento Baseado em Consistência buscam evidências em um ou mais tipos de alinhamento par a par que envolvem A ou B e alguma outra sequência e que possam fortalecer o posicionamento dos resíduos no alinhamento final envolvendo A e B. Para que essas evidências possam ser descobertas e utilizadas no alinhamento

múltiplo são necessárias duas etapas: a geração da *Constraints List* (CL) e o alinhamento de *profiles* utilizando a mesma.

A geração da CL é a parte mais custosa ($O(ln^3)$) do MSA, por exemplo, nos experimentos realizados ela chegou a representar mais de 70% do tempo total de execução da ferramenta de geração de múltiplos MSA em desenvolvimento [João Jr et al. 2023a]. O objetivo do presente trabalho é diminuir o tempo necessário para a geração da CL por meio do uso de multiprocessamento. Para isso, essa geração foi paralelizada utilizando a biblioteca OpenMP. Os resultados obtidos mostram claramente o bom desempenho da solução proposta, que não só conseguiu reduzir significativamente o tempo de execução, mas também que ela é escalável ao se aumentar o tamanho do problema e a quantidade de CPUs. Por exemplo, para uma instância grande do problema foi possível reduzir o tempo da geração de $\approx 4,5$ horas para apenas 5 minutos. Como comparação, no trabalho apresentado em [Zola et al. 2007] (PTC), que utiliza trocas de mensagens por MPI, os autores paralelizaram o T-Coffee [Notredame et al. 2000], uma ferramenta estado-da-arte para alinhamento múltiplo baseado em consistência que também serviu de base para este trabalho. Diferentemente do PTC, a abordagem proposta neste trabalho utiliza memória compartilhada, ao invés de troca de mensagens.

O restante desse trabalho está dividido da seguinte forma. A próxima seção descreve a geração da *Constraints List* com o método da sua paralelização sendo apresentado na Seção 3. Em seguida, na Seção 4, é apresentada uma análise dos resultados de desempenho e, ao final, algumas conclusões e comentários sobre trabalhos futuros.

2. Geração da *Constraints List*

Algoritmo 1: Para a geração da *Constraints List* [Notredame et al. 2000]

```
1  Seja pesos[0 até nseq-1];
2  para cada sequência s1 faça
3      para cada resíduo r1 de s1 faça
4          para cada aresta a de s1 envolvendo r1 faça
5              peso = n = 0;
6              para cada aresta b de a.s2 envolvendo a.r2 faça
7                  peso += b.peso;
8                  n++;
9              se n ≠ 0 então adiciona peso/n a pesos[s1];
10 para cada sequência s1 faça
11     para cada resíduo r1 de s1 faça
12         i = 0;
13         para cada aresta a de s1 envolvendo r1 faça
14             adiciona a tupla (s1,r1,a.s2,a.r2,pesos[s1][i++]) à CL;
```

Uma *Constraints List* é uma lista de evidências indicando que, no alinhamento múltiplo final, um resíduo A_i deveria está alinhado com o resíduo B_j. Cada evidência é uma tupla contendo os índices das sequências, os índices, em cada sequência, dos resíduos envolvidos e um peso para a evidência. Neste trabalho, a CL é implementada por uma estrutura de dados de lista ortogonal, como se fosse uma matriz esparsa. A primeira dimensão é indexada pela sequência e a segunda pelo resíduo dela.

Para criar a CL são necessárias duas etapas. Na primeira, as sequências genéticas são alinhadas par a par. Dessa forma, são realizados $\frac{n\times(n-1)}{2}$ alinhamentos par a par independentes. Neste trabalho, o alinhamento é realizado pelo algoritmo de Viterbi e segue o modelo de cadeias de Markov (HMM) [Durbin et al. 1998]. A probabilidade de alinhamento de cada par de resíduos, dada pelo HMM, é utilizada como peso. Cada par de resíduos alinhados é armazenado numa lista auxiliar de arestas (conexão entre resíduos alinhados) juntamente com seu peso.

A segunda etapa é a construção da CL propriamente dita. Nela, para cada aresta, é verificada a existência de outras arestas que envolvam o mesmo resíduo, mas em outras sequências. Se a soma dos pesos dessas arestas ultrapassar um valor pré-definido, a aresta inicial é colocada na CL tendo como peso a média dos pesos das arestas. Como a CL é indexada primeiramente pela sequência, duas inserções são feitas: uma para cada sequência da aresta. Essa etapa é ilustrada no Algoritmo 1 do T-Coffee [Notredame et al. 2000].

3. Paralelização da Geração da *Constraints List*

Para paralelizar a primeira etapa da geração da CL foi necessário disparar paralelamente os alinhamentos par a par utilizando a diretiva OpenMP `parallel for`. Porém, apesar dos alinhamentos serem independentes, existe um trecho de código que pode causar *race conditions*: a inclusão das informações na lista de arestas. A inclusão de um semáforo de exclusão mútua na região crítica limitaria demais o paralelismo. Como se trata de uma lista ortogonal, indexada na primeira dimensão pela sequência, a solução adotada foi a criação de um vetor de semáforos, um para cada sequência, aplicando assim a exclusão mútua a cada uma delas, e não a lista como um todo. Já a paralelização do Algoritmo 1 foi feita paralelizando os dois *loops* mais externos (linhas 2 e 10), uma vez que não há dependência de dados. Mais uma vez, o único local onde poderia acontecer *race conditions* é na inserção na CL, onde a mesma solução da etapa anterior foi adotada.

4. Análise Experimental

Para analisar o desempenho da geração paralela da CL, foram usadas sequências de proteínas da família PF00005 [Hung et al. 1998] pertencentes ao *benchmark* PFAM [Finn et al. 2014]. A PF00005 é a maior família de proteínas presente em muitas bactérias completamente sequenciadas e possuem significativa quantidade de sequências disponíveis Desta família, foram selecionadas sequências pertencentes a um dos três grupos: tamanhos menores que 100 resíduos; entre 200, inclusive, e 300, e; entre 400, inclusive, e 500. Das milhares de sequências que compõem estes grupos, foram selecionadas amostras com 100, 200, 300, 400 e 500 sequências para avaliar o impacto não apenas dos tamanhos das sequências, mas também da sua quantidade, no desempenho da geração da CL. Esta combinação de quantidades e tamanhos de sequências permite avaliar a eficácia da proposta em relação a diferentes tamanhos de problemas de alinhamento múltiplo de proteínas.

As médias dos tempos de três execuções dos cenários utilizados, assim como os *speedups*, são exibidos na Tabela 1. Os experimentos foram executados em instâncias da família C7g da AWS EC2 equipadas com processadores AWS Graviton 3 com clock de 2,6 GHz. Foram utilizadas instâncias com 1, 2, 4, 8, 16, 32 e 64 (a maior quantidade disponível) CPUs, com 2 GiB de memória sendo alocada para cada CPU utilizada.

Tabela 1. Tempos em segundos e Speedups

		Até 100 resíduos					200 a 300 resíduos					400 a 500 resíduos				
		100	200	300	400	500	100	200	300	400	500	100	200	300	400	500
1	Tempo	21,7	105,0	295,0	615,0	1100,3	161,0	686,0	1715,0	3254,0	5401,0	482,7	2027,0	5254,0	9803,7	16147,0
2	Tempo	10,4	50,7	140,0	291,0	497,0	79,0	343,0	826,0	1542,0	2555,0	245,0	1062,0	2553,0	4793,0	7899,0
	Speedup	2,1	2,1	2,1	2,1	2,2	2,0	2,0	2,1	2,1	2,1	2,0	1,9	2,1	2,0	2,0
4	Tempo	5,4	26,2	72,0	150,0	256,0	39,9	172,0	421,0	788,0	1302,0	123,0	535,0	1292,0	2410,0	3967,0
	Speedup	4,0	4,0	4,1	4,1	4,3	4,0	4,0	4,1	4,1	4,1	3,9	3,8	4,1	4,1	4,1
8	Tempo	2,9	13,5	37,2	77,0	132,0	20,7	89,0	215,0	408,0	668,0	62,0	269,0	652,0	1224,0	2013,0
	Speedup	7,5	7,8	7,9	8,0	8,3	7,8	7,7	8,0	8,0	8,1	7,8	7,5	8,1	8,0	8,0
16	Tempo	1,7	7,7	20,4	42,1	72,0	11,2	47,6	117,0	213,0	364,0	32,3	138,0	335,0	628,0	1028,0
	Speedup	12,8	13,6	14,5	14,6	15,3	14,4	14,4	14,7	15,3	14,8	14,9	14,7	15,7	15,6	15,7
32	Tempo	1,0	4,6	11,9	24,3	41,8	6,4	27,0	66,0	119,0	206,0	17,7	73,0	172,0	324,0	545,0
	Speedup	21,7	22,8	24,8	25,3	26,3	25,2	25,4	26,0	27,3	26,2	27,3	27,8	30,5	30,3	29,6
64	Tempo	0,9	3,2	8,1	15,9	27,2	4,8	17,1	40,5	71,0	127,0	11,2	40,9	96,0	181,0	302,0
	Speedup	24,6	32,8	36,4	38,7	40,5	33,5	40,1	42,3	45,8	42,5	43,1	49,6	54,7	54,2	53,5

Analisando a Tabela 1 é possível identificar um desempenho muito próximo do ideal da solução proposta para as instâncias com até 16 CPUs, alcançando *speedups* iguais ou muito próximos do valor máximo possível, mostrando a eficiência da solução proposta. Cabe ressaltar que em alguns casos os *speedups* ultrapassaram seus limites teóricos, devido à execução dos processos de gerenciamento da ferramenta competir com a geração da consistência por um mesmo processador em instâncias com uma única CPU.

O uso da instância com 32 CPUs permitiu reduzir ainda mais o tempo de execução, atingindo um *speedup* máximo de $30,5$, muito próximo do ideal. Apesar de um desempenho menos expressivos para as menores amostras, os resultados mostram que a média da eficiência ($\frac{speedup}{CPUs} \times 100$) para todas as 15 amostras nessa instância, foi de mais de 82%. Por fim, as execuções com 64 CPUs permitiram alcançar os menores tempos de execução. Por exemplo, para a execução com 500 sequências com tamanhos entre 400 e 500 resíduos o tempo de execução foi reduzido de $\approx 4,5$ horas para apenas 302 segundos. Entretanto, para as amostras pequenas, à grande redução do tempo de execução, fez com que o tempo das operações de sincronização, que aumentam com o número de *threads* concorrentes, se tornasse, proporcionalmente, maior, impactando no desempenho. Mesmo assim, a eficiência média foi de mais de 65%, mostrando a escalabilidade da solução quando se aumenta o tamanho do problema.

Assim, os resultados mostraram a alta qualidade da solução paralela implementada, o que permitiu uma significativa redução no tempo de execução. Porém, para as menores amostras, especialmente ao se executar em nuvens computacionais onde muitas vezes o custo pode ter mais ou tanta influência quanto o desempenho, a execução em instâncias com 16 ou 32 CPUs pode levar a uma relação custo/desempenho melhor do que com 64 CPUs.

5. Conclusões e Trabalhos Futuros

Uma vez que a geração da consistência pode ser a parte mais custosa do MSA, este trabalho propôs, implementou e avaliou a paralelização desta etapa. Os resultados mostraram claramente uma redução significativa no tempo de execução, assim como sua escalabili-

dade, especialmente quando foram utilizadas até 32 CPUs. O uso desta solução permite a execução de instâncias grandes do problema em tempos aceitáveis. Como trabalhos futuros, será analisado, no trabalho de doutorado, o impacto desta solução no tempo de execução total da ferramenta de MSA em desenvolvimento, onde não só o escalonamento das tarefas desta etapa do alinhamento tem influência na execução da ferramenta, mas também no alinhamento de *profiles*, momento em que a *Constraints List* é efetivamente utilizada para gerar o alinhamento final.

Agradecimentos

Os autores agradecem o apoio do CNPq através do projeto CNPq/AWS 421828/2022-6.

Referências

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Edgar, R. C. and Batzoglou, S. (2006). Multiple sequence alignment. *Current Opinion in Structural Biology*, 16(3):368–373.

Feng, D.-F. and Doolittle, R. F. (1987). Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees. *Journal of Molecular Evolution*, 25:351–360.

Finn, R. D., Bateman, A., Clements, J., Coggill, P., Eberhardt, R. Y., Eddy, S. R., Heger, A., Hetherington, K., Holm, L., Mistry, J., Sonnhammer, E. L., Tate, J., and Punta, M. (2014). Pfam: the protein families database. *Nucleic acids research*, 42.

Hung, L.-W., Wang, I. X., Nikaido, K., Liu, P.-Q., Ames, G. F.-L., and Kim, S.-H. (1998). Crystal structure of the ATP-binding subunit of an ABC transporter. *Nature*, 396(6712):703–707.

João Jr, M., Sena, A. C., and Rebello, V. E. F. (2023a). Fragmentando o DNA de Ferramentas de Alinhamento Progressivo: uma Metaferramenta Eficiente. In *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 349–360. SBC.

João Jr, M., Sena, A. C., and Rebello, V. E. F. (2023b). On closing the inopportune gap with consistency transformation and iterative refinement. *PLoS ONE*, 18(7):1–24.

Mirarab, S. and Warnow, T. (2011). FastSP: linear time calculation of alignment accuracy. *Bioinformatics*, 27(23):3250–3258.

Notredame, C., Higgins, D. G., and Heringa, J. (2000). T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. *Journal of Molecular Biology*, 302(1):205 – 217.

Thompson, J. D., Linard, B., Lecompte, O., and Poch, O. (2011). A Comprehensive Benchmark Study of Multiple Sequence Alignment Methods: Current Challenges and Future Perspectives. *PLoS ONE*, 6(3).

Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348.

Zola, J., Yang, X., Rospondek, S., and Aluru, S. (2007). Parallel T-Coffee: A Parallel Multiple Sequence Aligner. *ISCA International Conference on Parallel and Distributed Computing and Systems (PDCS)*, pages 248–253.