Alinhamento de seqüências com rearranjos

Augusto Fernandes Vellozo[1]
Orientador: Alair Pereira do Lago[1]

[1]Departamento de Ciência da Computação - Instituto de Matemática e Estatística
Universidade Estadual de São Paulo – SP – Brasil (DCC– IME – USP)

vellozo@ime.usp.br

Abstract. *Sequence comparison done by alignment algorithms is one of the most fundamental tasks in bioinformatics. The evolutive mutations considered in these alignments are insertions, deletions and substitutions of nucleotides. This work treats of generalizations introduced in alignment algorithms in such a way that other mutations known as rearrangements are also considered, more specifically, we consider non-overlapping inversions, duplications and duplications in tandem. The new algorithms presented in this work improve asymptotically the runt time of the algorithms known to solve the same problem. This paper is an extended abstract of [Vellozo 2007].*

Resumo. *Uma das tarefas mais básicas em bioinformática é a comparação de seqüências feita por algoritmos de alinhamento, que modelam as alterações evolutivas nas seqüências biológicas através de mutações como inserção, remoção e substituições de nucleotídeos. Este trabalho trata de generalizações nos algoritmos de alinhamento que levam em consideração outras mutações conhecidas como rearranjos, mais especificamente, inversões não sobrepostas, duplicações e duplicações em* tandem*. Os novos algoritmos apresentados neste trabalho melhoram assintoticamente o tempo de execução dos algoritmos conhecidos para resolver o mesmo problema. Este artigo é um resumo estendido de [Vellozo 2007].*

1. Introdução

Atualmente, devido ao grande número de projetos de seqüenciamentos desenvolvidos e finalizados, temos uma enorme quantidade de dados moleculares disponíveis, principalmente de DNA, RNA e proteína. Processar estes grandes volumes de dados para extrair informações relevantes exige algoritmos eficientes e é atualmente um grande desafio.

Na história da evolução vários eventos biológicos introduzem mudanças nas seqüências do DNA. Quando duas seqüências têm alto grau de semelhança é esperado que elas se diferenciem apenas em alguns trechos. Normalmente as diferenças nestes trechos são conseqüências da ocorrência de alguns eventos biológicos. Em geral, para a visualização destes eventos ocorridos é feito um procedimento de alinhamento entre as duas seqüências. Procedimentos de alinhamento típicos tentam identificar que partes não mudam e, nas partes que mudam, quais os eventos biológicos que ocasionaram as mudanças. Tipicamente os eventos biológicos considerados para um alinhamento são os eventos pontuais de substituição, remoção e inserção de um nucleotídeo.

Além desses eventos pontuais que resultam em alterações em um único símbolo na seqüência biológica, iremos considerar também os rearranjos para a obtenção de um

alinhamento ótimo de duas seqüências. Os rearranjos agem sobre um fragmento da seqüência. Os rearranjos que consideraremos são a inversão e a duplicação de fragmentos. Estes não são os únicos tipos de rearranjos, mas são considerados muito importantes e são largamente estudados, como em [Shaw and Lupski 2004, Verkerk et al. 1991] na associação de doenças humanas à rearranjos no DNA. Os novos resultados comparativos feitos por Sherer et al. [Feuk et al. 2005] nas seqüências de DNA do homem e do chimpanzé, mostram a importância do estudo da inversão ao nível da seqüência de DNA e relataram 83 seqüências reversas que estão contidas dentro de genes, muitas destas ocorrências constituindo-se polimórficas.

O alinhamento de seqüências biológicas é uma das principais técnicas utilizadas por biólogos para comparar seqüências biológicas.

Um alinhamento pode ser associado a um conjunto de operações de edição que transformam uma seqüência em outra. Normalmente, por motivos de performance, as únicas operações de edição consideradas para obtenção de um alinhamento ótimo são a *substituição* de um símbolo em outro, a *inserção* de um símbolo e a *remoção* de um símbolo. Se associamos custos a cada operação de edição, existe um algoritmo clássico de programação dinâmica, que em tempo de execução $O(n^2)$ [Needleman and Wunsch 1970][1][2], computa um conjunto mínimo de operações de edição que tem o custo total ótimo e apresenta o alinhamento associado.

Neste texto, consideraremos que $s = s_1 s_2 \ldots s_n = s[1\,..\,n]$ e $t = t_1 t_2 \ldots t_m = t[1\,..\,m]$ são as duas seqüências a serem comparadas e alinhadas.

Chamaremos uma seqüência de operações de edição que transforma uma seqüência s em uma seqüência t de uma *transformação de s em t*. Consideraremos que cada operação de edição tem uma pontuação associada e que a pontuação de uma transformação é a soma das pontuações de suas operações de edição. Consideraremos que uma transformação de s em t é ótima se não existir outra transformação de s em t com pontuação maior.

Diremos que $\omega_{s,t}((i', j'), (i, j))$ é a pontuação de uma transformação ótima de $s[i' + 1\,..\,i]$ em $t[j' + 1\,..\,j]$ considerando as operações de edição de inserção de um símbolo $t[j]$, remoção de um símbolo $s[i]$ e substituição de um símbolo $s[i]$ por $t[j]$, cujas pontuações são, repectivamente, denotadas por $ins(t[j])$, $rem(s[i])$ e $sub(s[i], t[j])$.

2. Alinhamento com inversões

Quando possibilitamos a operação de edição de inversão para a obtenção de uma transformação diremos que a transformação obtida é uma *transformação com inversões de s em t*. A operação de edição de inversão substitui um fragmento da seqüência pelo seu reverso complementar. Diremos que $\overline{s[i'\,..\,i]}$ é o complemento reverso de $s[i'\,..\,i]$ e ω_{inv} é a pontuação da operação de edição de inversão de um fragmento.

[1]Nas análises das complexidades dos algoritmos, consideraremos neste texto que n é o comprimento da maior seqüência analisada.

[2]O conhecido artigo de Needleman e Wunsch é geralmente considerado a primeira contribuição importante em procedimentos computacionais para a comparação de seqüências. O algoritmo proposto por eles neste artigo, apesar de não ter sido feita uma análise de tempo no artigo, é considerado cúbico, mas normalmente o nome "Algoritmo de Needleman e Wunsch"é utilizado para denominar o algoritmo quadrático clássico de programação dinâmica para alinhamentos [Setubal and Meidanis 1997].

Como pode ser visto em [Chen et al. 2005], o problema de decisão associado ao problema de obter uma transformação ótima com inversões para um alfabeto ilimitado é NP-difícil. Em 1992, Schöniger e Waterman [Schöniger and Waterman 1992] introduziram uma hipótese simplificadora: todas as regiões envolvendo inversões não se sobrepõem. Com esta simplificação eles desenvolveram um algoritmo que obtém um alinhamento ótimo com inversões não sobrepostas em tempo $O(n^6)$. A partir daí, outros trabalhos melhoraram este tempo chegando até $O(n^4)$ [do Lago et al. 2003, Gao et al. 2003].

Propomos dois algoritmos exatos que obtêm uma transformação ótima com inversões não sobrepostas em tempo $O(n^3 \log n)$ [Alves et al. 2005] e $O(n^3)$ [Vellozo et al. 2006].

Os dois algoritmos computam a matriz B, onde $B[i, j]$ é a pontuação do alinhamento ótimo com inversões não sobrepostas de $s[1\,..\,i]$ e $t[1\,..\,j]$ e seguem o pseudocódigo do Algoritmo 1, cuja execução é ilustrada na figura 1.

Entre as linhas 3 a 14 do Algoritmo 1 são consideradas somentes as operações de edição de inserção, remoção e substituição de um símbolo. O laço da linha 18 considera a possibilidade de uma operação de edição de inversão de $s[i' + 1\,..\,i]$, $0 \leq i' \leq i \leq n$.

Os dois algoritmos que propomos diferem somente na forma da construção da matriz $W^i_{i'}$ e do vetor $A^i_{i'}$, onde $W^i_{i'}[j', j] = \omega_{\overline{s},t}((i', j'), (i, j))$ é a pontuação do alinhamento ótimo de $\overline{s[i' + 1\,..\,i]}$ e $t[j'+1\,..\,j]$ considerando somente inserção, remoção e substituição de um símbolo, pois não consideramos inversões sobrepostas no alinhamento de s e t, e $A^i_{i'}[j]$ é o valor máximo de $(B[i', j'] + W^i_{i'}[j', j] + \omega_{inv})$, para todo j', $0 \leq j' \leq j$.

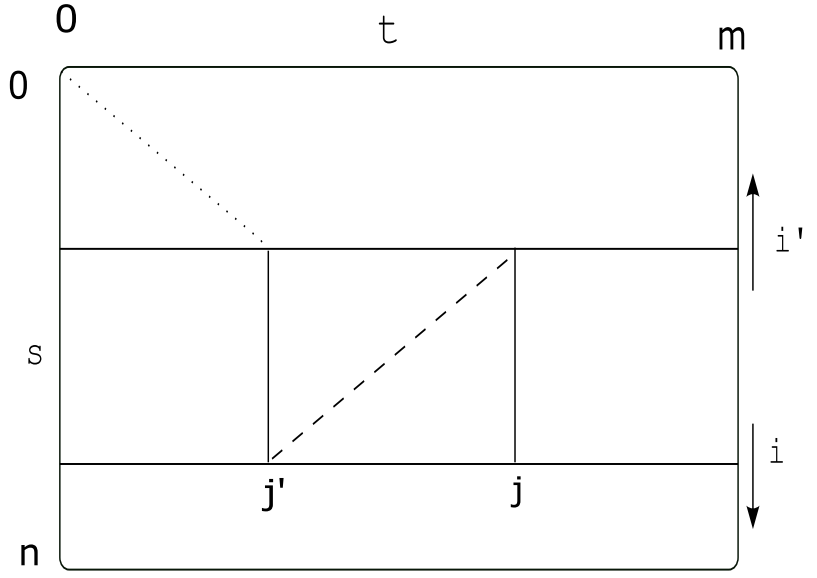

Figura 1. Ilustração da execução do Algoritmo 1. A linha pontilhada representa em um grafo de edição um alinhamento ótimo com inversões não sobrepostas de $s[1\,..\,i] \times t[1\,..\,j]$ e a linha tracejada o alinhamento de $\overline{s[i' + 1\,..\,i]} \times t[j' + 1\,..\,j]$

.

2.1. Alinhamento com inversões não sobrepostas em tempo $O(n^3 \log n)$

Jeanette Schmidt [Schmidt 1998] desenvolveu um algoritmo que constrói uma estrutura de árvores binárias de altura $\log m$ que possibilita obter um valor de $\omega_{s,t}((i', j'), (i, j))$

Algoritmo 1 Algoritmo para obtenção da matriz B tal que $B[i,j]$ é a pontuação do alinhamento ótimo com inversões não sobrepostas de $s[1\,..\,i]$ e $t[1\,..\,j]$

```
INVGERAL(s, t)
 1  para i de 0 até |s| faça
 2        ▷ Obtém B[i,j] considerando somente inserção, remoção e substituição
 3        se i = 0 então
 4              B[0,0] ← 0
 5        senão
 6              B[i,0] ← B[i−1,0] + rem(s[i])
 7        para j de 1 até |t| faça
 8              se i = 0 então
 9                    B[0,j] ← B[0,j−1] + ins(t[j])
10              senão
11                    h ← B[i,j−1] + ins(t[j])
12                    v ← B[i−1,j] + rem(s[i])
13                    d ← B[i−1,j−1] + sub(s[i],t[j])
14                    B[i,j] ← max(h,v,d)
15        ▷ A seguir, obtém B[i,j] considerando também inversões não sobrepostas
16        ▷ W^i_{i'}[j',j] =pontuação do alinhamento ótimo de s[i'+1..i] e t[j'+1..j]
17        Inicializa W^i_{i+1}
18        para i' de i descendo até 0 faça
19              Obtém W^i_{i'} a partir de W^i_{i'+1}
20              Para cada j obtém A^i_{i'}[j] = max_{j'≤j}(B[i',j'] + W^i_{i'}[j',j] + ω_inv)
21              para j de 0 até |t| faça
22                    B[i,j] ← max(B[i,j], A^i_{i'}[j])
23  devolva B
```

em tempo $\log m$. Dada a estrutura de árvores com os valores de $\omega_{s,t}((i',j'),(i-1,j))$ o algoritmo proposto por [Schmidt 1998] precisa de tempo $O(m \log m)$ para construir outras m árvores que contêm os valores de $\omega_{s,t}((i',j'),(i,j))$.

A matriz $W^i_{i'}$ é uma matriz de Monge inversa triangular superior [Aggarwal and Park 1988] e portanto totalmente monotônica, o que possibilita que utilizemos o algoritmo proposto por [Aggarwal et al. 1987] para obtermos $A^i_{i'}[j]$ com $O(m)$ comparações.

Utilizando a estrutura de árvores proposta em [Schmidt 1998] para armazenar os valores de $\omega_{\overline{s},t}((i',j'),(i,j))$, executamos a linha 19 do Algoritmo 1 em tempo $O(m \log m)$ e, utilizando o algoritmo proposto por [Aggarwal et al. 1987], obtemos $A^i_{i'}[j]$ na linha 20 em tempo $O(m \log m)$. Como essas linhas são executadas para todo i e i', $0 \leq i' \leq i \leq n$, o Algoritmo 1 é executado em tempo $O(n^2 m \log m)$. Além disto, o Algoritmo 1 precisa de $O(\max(nm, m^2))$ de memória.

2.2. Alinhamento com inversões não sobrepostas em tempo $O(n^3)$

Dados i, i' e j, $0 \leq i' \leq i \leq n$, $0 \leq j \leq m$, definimos o vetor $\text{hDif}^{i',i,j}$ tal que $\text{hDif}^{i',i,j}[j'] = \omega_{\overline{s},t}((i',j'),(i,j)) - \omega_{\overline{s},t}((i',j'),(i,j-1))$, $0 \leq j' \leq j$.

O vetor hDif$^{i',i,j}$ é não decrescente [Schmidt 1998] e portanto assume no máximo $\psi(i', i, j)$ valores distintos, $1 \leq \psi(i', i, j) \leq j$. Consideraremos que as pontuações das operações de edição pontuais são constantes e inteiras e portanto podemos considerar que $\psi(i', i, j)$ é constante. Por exemplo, se $ins(t[j]) = rem(s[i]) = -g$ e $sub(s[i], t[j]) \in \{a, -g\}$, $a \in \mathbb{N}^*$, $g \in \mathbb{N}^*$, então $\psi(i', i, j) \leq a + 2g + 1 = O(1)$[3].

Seja o vetor crescente $BL^{i',i,j}$ onde são armazenados todos os j' tais que hDif$^{i',i,j}[j'] \neq$ hDif$^{i',i,j}[j' - 1]$ e $j' > 0$. Fixados i e i' e dados os vetores $BL^{i',i-1,j}$, $\forall j \mid 0 \leq j \leq m$, existe um algoritmo na seção 6 de [Schmidt 1998] que obtém em tempo $O(m)$ os vetores $BL^{i',i,j}$ e os valores de hDif$^{i',i,j}[j']$, $\forall j \mid 0 \leq j \leq m$ e $\forall j' \mid j' \in BL^{i',i,j}$.

Portanto, ao invés de calcularmos todos os valores de $W^i_{i'}$ trabalhamos somente com algumas diferenças destes valores para calcularmos $A^i_{i'}$, ou seja, na linha 19 do Algoritmo 1 calculamos somente os valores de hDif$^{i',i,j}[j']$, tais que $0 \leq j \leq m$ e $j' \in BL^{i',i,j}$. Portanto executamos a linha 19 do Algoritmo 1 em tempo $O(m)$.

Dados i e i' , $0 \leq i' \leq\ i \leq n$, dizemos que j' é um *candidato a máximo de* j se não existe outro j'' tal que $j' < j'' \leq j$ e $B[i', j''] + W^i_{i'}[j'', j] \geq B[i', j'] + W^i_{i'}[j', j]$. Colocamos todos os candidatos a máximo de j em uma lista crescente e portanto o primeiro elemento j' desta lista é tal que $B[i', j'] + W^i_{i'}[j', j]$ é máximo para aquele j, ou seja, $A^i_{i'}[j] = B[i', j'] + W^i_{i'}[j', j] + \omega_{inv}$.

Dados i, i', $0 \leq i' \leq\ i \leq n$, e os vetores $BL^{i',i-1,j}$ de cada j, existe um algoritmo em [Landau and Ziv-Ukelson 2001] que constrói em tempo $O(m)$ as listas de candidatos a máximo de cada j, $0 \leq j \leq m$. Utilizamos este algoritmo e conseguimos executar a linha 20 do Algoritmo 1 em tempo $O(m)$. Como as linhas 19 e 20 são executadas $O(n^2)$ vezes, conseguimos executar o Algoritmo 1 em tempo $O(n^2 m)$. Além disto, o algoritmo pode ser executado com $O(nm)$ de memória.

3. Alinhamento com duplicações

A duplicação é um evento biológico que ocorre freqüentemente e que insere uma cópia de um fragmento da seqüência na própria seqüência. Um tipo de duplicação muito estudado é a duplicação encadeada (ou duplicação em *tandem*). Neste tipo de duplicação a cópia é inserida imediatamente após (ou antes) do fragmento copiado. Diremos que *repetição* é a cópia inserida e *seqüência base* é a seqüência de onde a repetição foi copiada.

Em [Benson 1997] é proposto um modelo para o alinhamento de seqüências que considera duplicações em *tandem*. São propostos dois algoritmos exatos para obter um tal alinhamento ótimo. O primeiro algoritmo proposto executa em tempo $O(n^5)$ e espaço $O(n^2)$. O segundo algoritmo proposto executa em tempo $O(n^4)$ e espaço $O(n^3)$. Estamos preparando um artigo onde propomos um modelo mais geral que o de Benson com algoritmos exatos que executam em tempo $O(n^3)$ e espaço $O(n^2)$.

Propomos um modelo para obter um alinhamento ótimo com duplicações que é parecido com o modelo proposto por Benson, porém com as seguintes diferenças:

1. Benson propõe somente duplicações em *tandem*. Nós não mantemos a restrição de ser em *tandem*, apesar de uma derivação do algoritmo que propomos poder ser feita para considerar esta restrição sem alterar as complexidades de tempo e memória do algoritmo que propomos.

[3]Muitos sistemas de pontuação de alinhamento atribuem uma pontuação como neste exemplo.

2. No modelo proposto por Benson, se um trecho de s é uma repetição então a seqüência base a ser comparada (ou alinhada) com esta repetição deve ser um trecho de t, ou seja, ele não considera que a seqüência base da repetição pode estar na própria seqüência. Nós consideramos que a seqüência base pode estar na própria seqüência ou na outra. Com isto, se estamos interessados em obter uma transformação ótima com duplicações de s em t, o modelo proposto por Benson considera que as operações de edição pontuais (inserção, remoção e subsituição) devem ocorrer depois das duplicações na transformação. Nós consideramos que as operações de edição pontuais podem ocorrer antes ou depois das duplicações.
3. No modelo proposto por Benson, a seqüência base deve ser a mesma para repetições contíguas. Nós consideramos que a seqüência base pode ser qualquer trecho das seqüências s ou t, exceto o próprio trecho da repetição.

Utilizando as técnicas utilizadas em [Landau and Ziv-Ukelson 2001], podemos obter um alinhamento ótimo com duplicações em *tandem* segundo o mesmo modelo proposto por Benson, em tempo $O(n^3)$ e com memória $O(n^2)$. Porém, acreditamos que o modelo que propomos é mais fiel à realidade, pois considera mais casos de duplicações.

Vamos considerar que para obtenção de um alinhamento ótimo com duplicações a seqüência base de uma repetição $s[i'+1\,..\,i]$ está em $s[1\,..\,i']s[i+1\,..\,n]$ ou $t[1\,..\,m]$, onde $s[1\,..\,i]s[i'+1\,..\,n]$ é a concatenação das seqüências $s[1\,..\,i]$ e $s[i'+1\,..\,n]$. Vale a pena ressaltar que o alinhamento obtido desta forma pode, em alguns casos, não representar uma transformação. Se estamos querendo garantir que o alinhamento ótimo gerado represente uma transformação ótima de s em t com duplicações então podemos alterar o algoritmo proposto, sem alterar suas compexidades, para considerar que a seqüência base de uma repetição $s[i'+1\,..\,i]$ está em $s[1\,..\,i']$ ou $t[1\,..\,j]$, onde $s[1\,..\,i']$ se transforma em $t[1\,..\,j]$ na transformação ótima sendo obtida. Além disto, em uma transformação com duplicações consideramos a existência da operação de edição de *excisão* que é a operação de edição dual à duplicação, ou seja, uma operação de edição de excisão remove um trecho da seqüência (repetição), tal que a seqüência de símbolos do trecho removido ainda aparece como um fragmento (seqüência base) da seqüência gerada após a excisão. Numa transformação de s em t, podemos associar uma operação de edição de excisão ao evento biológico de uma duplicação na seqüência s que não ocorreu em t ou ao evento biológico de remoção de uma repetição em t que não ocorreu em s.

Dados i e i' tais que $0 \leq i' < i \leq n$, dizemos que $X_s^{i',i}$ é o conjunto de todas as possíveis seqüências que podem ser a seqüência base de $s[i'+1\,..\,i]$. Definimos a matriz R_s, tal que $R_s[i',i] = \omega_{s,x}((i',i),(0,|x|)) + dup(s[i'\,..\,i])$ se $0 \leq i' < i \leq n$ e $-\infty$ caso contrário, onde $x \in X_s^{i',i}$ e não existe um $x' \mid x' \in X_s^{i',i}$ e $\omega_{s,x'}((i',i),(0,|x'|)) > \omega_{s,x}((i',i),(0,|x|))$ e $dup(s[i'\,..\,i])$ é a pontuação da operação de edição de duplicação de $s[i'+1\,..\,i]$. Dizemos que $R_s[i',i]$ é a pontuação da inserção da repetição $s[i'+1\,..\,i]$, $0 \leq i' < i \leq n$. No nosso modelo consideraremos que $X_s^{i',i}$ são todos os possíveis fragmentos de $s[1\,..\,i']s[i+1\,..\,n]$ e todos os possíveis fragmentos de $t[1\,..\,m]$. De forma análoga, definimos a matriz R_t. Utilizando programação dinâmica conseguimos construir as matrizes R_s e R_t em tempo $O(n^3)$ e memória $O(n^2)$.

Seja a matriz M tal que $M[i,j]$ é a pontuação de um alinhamento ótimo com duplicações de $s[1\,..\,i]$ e $t[1\,..\,j]$. Dadas as matrizes R_s e R_t construímos M em tempo $O(n^3)$ e memória $O(n^2)$ utilizando a seguinte recorrência:

$$
\begin{aligned}
&M[0,0] = 0 \\
&M[i,0] = \max \left\{ \begin{array}{l} M[i-1,0] + rem(s[i]), \\ \max_{\forall i' | 0 \leq i' < i} \left(M[i',0] + R_s[i',i] \right) \end{array} \right\}, i > 0 \\
&M[0,j] = \max \left\{ \begin{array}{l} M[0,j-1] + ins(t[j]), \\ \max_{\forall j' | 0 \leq j' < j} \left(M[0,j'] + R_t[j',j] \right) \end{array} \right\}, j > 0 \\
&M[i,j] = \max \left\{ \begin{array}{l} M[i,j-1] + ins(t[j]), \\ M[i-1,j] + rem(s[i]), \\ M[i-1,j-1] + sub(s[i],t[j]), \\ \max_{\forall i' | 0 \leq i' < i} \left(M[i',j] + R_s[i',i] \right), \\ \max_{\forall j' | 0 \leq j' < j} \left(M[i,j'] + R_t[j',j] \right) \end{array} \right\}, i > 0 \text{ e } j > 0
\end{aligned}
$$

4. Conclusão

Para a comparação de duas seqüências, desenvolvemos e apresentamos algoritmos exatos e inéditos, que obtêm alinhamentos que consideram a possibilidade da ocorrência de outros importantes eventos biológicos (inversão e duplicação) além dos eventos de inserção, remoção e substituição de um símbolo, comumente utilizados em algoritmos para obtenção de alinhamentos ótimos. Esperamos com isto, que o alinhamento obtido esteja mais próximo de mostrar o que realmente ocorreu na evolução.

Apesar do problema da obtenção de um alinhamento ótimo com inversões não sobrepostas ser um problema já bem estudado e com alguns algoritmos já publicados, conseguimos desenvolver algoritmos [Alves et al. 2005, Vellozo et al. 2006] com complexidade de tempo significativamente melhor que os já existentes.

Para o alinhamento com duplicações, não conhecemos nenhum outro trabalho que utilize os modelos de duplicações que apresentamos. O modelo mais próximo que conhecemos do modelo que propomos, o qual estamos implementando e preparando para publicação, é o modelo proposto por Benson [Benson 1997]. Porém, os algoritmos propostos por Benson têm complexidades de tempo e memória muito piores que as dos algoritmos que propomos, apesar do nosso espaço de busca das duplicações tratar muito mais casos que os de Benson.

Referências

[Aggarwal et al. 1987] Aggarwal, A., Klawe, M. M., Moran, S., Shor, P., and Wilber, R. (1987). Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(2):195–208.

[Aggarwal and Park 1988] Aggarwal, A. and Park, J. K. (1988). Notes on searching in multidimensional monotone arrays. In *Proc. 29th Symp. Foundations of Computer Science*, pages 497–512. IEEE.

[Alves et al. 2005] Alves, C. E. R., do Lago, A. P., and Vellozo, A. F. (2005). Alignment with non-overlapping inversions in $O(n^3 \log n)$-time. *Electronic Notes in Discrete Mathematics*, 19:365–371.

[Benson 1997] Benson, G. (1997). Sequence alignment with tandem duplication. In *RECOMB '97: Proceedings of the first annual international conference on Computational molecular biology*, pages 27–36, New York, NY, USA. ACM Press.

[Chen et al. 2005] Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., and Jiang, T. (2005). Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2(4):302–315.

[do Lago et al. 2003] do Lago, A. P., Muchnik, I., and Kulikowski, C. (2003). An $O(n^4)$ algorithm for alignment with non-overlapping inversions. In *Second Brazilian Workshop on Bioinformatics, WOB 2003*, Macaé, RJ, Brazil. http://www.ime.usp.br/∼alair/-wob03.pdf.

[Feuk et al. 2005] Feuk, MacDonald, Tang, Carson, Li, Rao, Khaja, and Scherer (2005). Discovery of human inversion polymorphisms by comparative analysis of human and chimpanzee DNA sequence assemblies. *PLoS Genet*, 1(4):e56.

[Gao et al. 2003] Gao, Y., Wu, J., Niewiadomski1, R., Wang, Y., Chen, Z.-Z., and Lin, G. (2003). A space efficient algorithm for sequence alignment with inversions. In *Computing and Combinatorics, 9th Annual International Conference, COCOON 2003*, volume 2697 of *Lecture Notes in Computer Science*, pages 57–67. Springer-Verlag.

[Landau and Ziv-Ukelson 2001] Landau, G. M. and Ziv-Ukelson, M. (2001). On the common substring alignment problem. *J. Algorithms*, 41(2):338–359.

[Needleman and Wunsch 1970] 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(3):443–453.

[Schmidt 1998] Schmidt, J. P. (1998). All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM J. Comput.*, 27(4):972–992 (electronic).

[Schöniger and Waterman 1992] Schöniger, M. and Waterman, M. S. (1992). A local algorithm for DNA sequence alignment with inversions. *Bulletin of Mathematical Biology*, 54(4):521–536.

[Setubal and Meidanis 1997] Setubal, J. and Meidanis, J. (1997). *Introduction to computational molecular biology*. PWS Publishing Company.

[Shaw and Lupski 2004] Shaw, C. and Lupski, J. (2004). Implications of human genome architecture for rearrangement-based disorders: the genomic basis of disease. *Hum. Mol. Genet.*, 13 Spec No 1:57–64.

[Vellozo 2007] Vellozo, A. F. (2007). *Alinhamento de seqüências com rearranjos*. PhD thesis, Universidade de São Paulo - DCC - IME- USP, http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04052007-185842.

[Vellozo et al. 2006] Vellozo, A. F., Alves, C. E. R., and do Lago, A. P. (2006). Alignment with non-overlapping inversions in $o(n^3)$-time. In *6th Workshop on Algorithms in Bioinformatics*. Springer. Lecture Notes in Bioinformatics 4175.

[Verkerk et al. 1991] Verkerk, Pieretti, Sutcliffe, Fu, Kuhl, Pizzuti, Reiner, Richards, Victoria, and Zhang (1991). Identification of a gene (FMR-1) containing a CGG repeat coincident with a breakpoint cluster region exhibiting length variation in fragile X syndrome. *Cell*, 65(5):905–914.