Universidade de São Paulo
Instituto de Matemática e Estatística

Departamento de Ciência da Computação

Ordenação por reversão: algoritmos e aplicação à biologia computacional

andréa tieme nakasato

Monografia - disciplina MAC-5701

Orientadora: Profa. Dra. Yoshiko Wakabayashi

ORDENAÇÃO POR REVERSÃO: ALGORITMOS E APLICAÇÃO À BIOLOGIA COMPUTACIONAL

ANDRÉA TIEME NAKASATO

Resumo. Nesta monografia apresentaremos alguns dos resultados que estudamos sobre o problema da ordenação por reversão (*sorting by reversals*). Esse problema é de interesse em Biologia Computacional para o tratamento de problemas relativos a rearranjo de genomas. Dados dois genomas representados como permutações de mesmos elementos (isto é, genes), o problema consiste em encontrar a seqüência mais parsimoniosa de reversões que transforma um genoma em outro. Depois que o primeiro algoritmo polinomial para o problema foi obtido por Hannehalli e Pevzner [18] em 1995, muitos trabalhos foram produzidos na tentativa de melhorar e simplificar o algoritmo [13, 2, 6]. Estudamos vários dos algoritmos propostos para este problema e investigamos variantes correlatas. Detalharemos aqui o algoritmo proposto recentemente por Tannier, Bergeron e Sagot [6] para resolver o problema.

1. Introdução

No fim dos anos 80, Jeffrey Palmer *et al.* compararam os genomas mitocondriais de *Brassica oleracea* (repolho) e de *Brassica campestris* (nabo) e descobriram que os genomas dessas espécies têm conjunto de genes idênticos, mas estes diferem na ordem em que ocorrem. Este e muitos outros estudos provaram que esta característica de rearranjo de genomas é uma forma comum de evolução molecular [7, 17, 18].

A análise de rearranjo de genomas permite comparar dados moleculares de espécies que divergem há muito tempo. Entretanto, muito pouco se sabe sobre rearranjos que produziram as variedades existentes das arquiteturas genômicas.

Tradicionalmente para a comparação de dados moleculares é feita a construção de árvores filogenéticas que são reconstruídas baseadas em pontos de mutação de um único gene (ou pequeno número de genes). Mas a comparação de genes às vezes não é muito apropriada em espécies (como no caso do "repolho" e "nabo") em que os pontos de mutação nos genes mitocondriais dessas espécies são tão poucos que são quase idênticos.

Alguns estudos sobre a evolução molecular dos vírus da herpes também contribuíram para incentivar a pesquisa em rearranjos de genomas. Nesse caso, mais perguntas foram levantadas do que respostas foram obtidas. Os genomas desses vírus evoluíram tão rápido que os fenótipos extremos dos dias de hoje podem parecer quase não-relacionados. Assim, os métodos clássicos de comparação de seqüências não são muito úteis para genomas com pouca similaridade, e utilizá-los pode levar até mesmo a contradições, pois genes diferentes podem dar origem a árvores

evolucionárias diferentes. Os vírus da herpes têm entre 70 a 200 genes; todos eles compartilham sete blocos conservados que estão rearranjados nos genomas de diferentes vírus da herpes. Portanto, a análise de tais rearranjos ao nível de *genoma* pode complementar a análise ao nível de *gene*, esta última mais tradicionalmente usada em evolução molecular.

Dobzhansky e Sturtevant [19] foram os pioneiros na análise de rearranjo de genomas em evolução molecular: no final dos anos 30 publicaram um artigo com uma árvore evolucionária apresentando um cenário de rearranjo com 17 reversões para as espécies da *Drosofila* (mosca da fruta).

A idéia de se considerar uma série mínima de eventos de rearranjo é justificada pela *hipótese da parsimônia*, na qual supõe-se que a Natureza sempre encontra caminhos que realizam um mínimo de mudanças. Portanto, se desejarmos investigar como um organismo de uma espécie transformou-se em um organismo de outra espécie devemos tentar encontrar uma série mínima de eventos de rearranjo que possivelmente tenham realizado esta transformação.

Nas próximas seções apresentaremos alguns dos resultados que estudamos sobre o problema da ordenação por reversão. Na Seção 2 definiremos formalmente o problema; e em seguida, na Seção 3, apresentaremos um breve relato dos algoritmos propostos para resolvê-lo. Na Seção 4 será apresentado o algoritmo proposto por Tannier, Bergeron e Sagot [6], bem como a notação utilizada e os teoremas que inspiraram a solução. Alguns problemas relacionados ao problema da ordenação por reversão são descritos na Seção 5, assim como alguns resultados para se resolvê-los. Na Seção 6 mostramos algumas aplicações; e finalmente, na Seção 7 apresentamos algumas conclusões.

2. O problema

Nos rearranjos de genomas, dado um genoma, podemos alterar a ordem dos genes através de operações como a reversão, transposição, fusão e fissão. A operação que vamos estudar é a reversão que também pode alterar a direção de transcrição.

Um genoma é representado como uma permutação em $\{1, \ldots, n\}$, onde cada gene é identificado com um número inteiro acrescido de um sinal de mais $(+)$ ou menos $(-)$ para indicar sua direção. A ordem será representada por uma permutação (sinalizada) π de $\{\pm 1, \ldots, \pm n\}$ tal que $\pi_{-i} = -\pi_i$. Para simplificar, a partir daqui indicaremos o sinal de um elemento em uma permutação apenas quando este for menos $(-)$.

Uma *reversão* $\rho_{i,j}$ do intervalo $[i, j] \subseteq [1, n]$ $(i < j)$ é a permutação

$$\rho_{i,j} = \big(1, \ldots, i, -j, \ldots, -(i+1), j+1, \ldots, n\big).$$

Note que $\pi \cdot \rho_{i,j}$ é a permutação obtida de π revertendo a ordem e invertendo os sinais dos elementos no intervalo $[i, j]$. Se $\rho_1, \ldots, \rho_k$ é uma seqüência de reversões, dizemos que ela *ordena* uma permutação π se $\pi \cdot \rho_1 \ldots \rho_k = Id$, onde Id é a permutação *identidade* $(1, \ldots, n)$. Para cada inteiro $i \in \{0, \ldots, n\}$, o par $\pi_i \pi_{i+1}$ é chamado de *adjacência* se são elementos consecutivos de mesmo sinal dentro da seqüência, ou seja, se $|\pi_i - \pi_{i+1}| = 1$, caso contrário são chamados de *pontos-de-quebra*. O

comprimento da menor seqüência de reversões que ordena π é chamado de *distância genômica* ou *distância de reversão* de π e é denotado por $d(\pi)$.

O *problema da ordenação por reversão* (OR) consiste em encontrar a menor seqüência de reversões que ordena uma dada permutação π. Podemos observar que a permutação identidade não possui pontos-de-quebra, logo, se eliminarmos todos os pontos-de-quebra resolvemos uma solução. Vamos denotar nosso problema por OR(n), indicando que a permutação dada tem comprimento n.

3. Alguns algoritmos propostos

O problema OR(n) tem sido bastante investigado, sendo vasta a literatura a respeito [16, 15, 6]. Em 1995, Hannenhalli e Pevzner [11] apresentaram um algoritmo para esse problema que consome tempo $O(n^4)$. Esses autores provaram um teorema de dualidade que permite calcular a distância de reversão de uma dada permutação como uma soma de três parâmetros combinatórios associados à essa permutação, sendo que um deles é o número de pontos-de-quebra da permutação.

Desde então muitas melhorias foram alcançadas e vários resultados surgiram, dentre os quais citamos o algoritmo de Kaplan, Shamir e Tarjan [13] que consome tempo $O(n^2)$. A melhoria no tempo de execução foi obtida através de uma grande simplificação no algoritmo e na estrutura combinatória necessária para a análise. Em 2001, Bader, Moret e Yan [2] obtiveram um algoritmo linear que apenas determina a distância de reversão (ele não encontra a seqüência de reversões). Esse algoritmo utiliza uma pilha e é simples de ser implementado.

Recentemente, Kaplan e Verbin [10] apresentaramm um algoritmo probabilístico para o problema OR(n). Esse algoritmo executa repetidamente uma rotina que eles chamaram *SBR-RandWalk*. Cada execução dessa rotina ou encontra a menor seqüência de ordenação ou falha. Esses autores também descreveram uma estrutura de dados para representar a permutação de forma a aplicar uma reversão em tempo $O(\sqrt{n \log n})$. Isso permitiu uma implementação do *SBR-RandWalk* cujo tempo de execução é $O(n^{3/2}\sqrt{\log n})$. Adicionalmente, eles também descreveram o primeiro algoritmo paralelo eficiente para o problema.

Na próxima seção apresentaremos o algoritmo proposto por Tannier, Bergeron e Sagot [6], cuja complexidade computacional é a melhor que se conhece até o momento.

4. O algoritmo de Tannier, Bergeron e Sagot

O algoritmo que descreveremos nesta seção foi proposto por Tannier, Bergeron e Sagot [6]. Esse algoritmo utiliza a estrutura de dados apresentada por Kaplan e Verbin [10] e consome tempo $O(n^{3/2}\sqrt{\log n})$. Este resultado responde afirmativamente à pergunta levantada por Ozery-Flato e Shamir [14] quanto à possibilidade de se obter um algoritmo subquadrático para o problema OR(n).

Antes de descrevermos o algoritmo precisamos estabelecer a notação que será utilizada.

Grafo de sobreposição. Seja π uma permutação em $\{1, \ldots, n\}$. Para auxiliar a descrição, fazemos inicialmente a seguinte extensão: acrescentamos à permutação os seguintes elementos: $\pi_0 = 0$ e $\pi_{n+1} = n + 1$. Seja π' a seqüência construída da seguinte maneira: para cada π_i, $1 \leq i \leq n$, associamos dois pontos, π_i^- e π_i^+; para $\pi_0 = 0$ e $\pi_{n+1} = n + 1$ associamos apenas os pontos 0^+ e $(n+1)^-$. Esses pontos são ordenados da seguinte maneira: $\pi_i^- < \pi_i^+$ se π_i é não-negativo, e $\pi_i^+ < \pi_i^-$ caso contrário; e $\pi_i^x < \pi_{i+1}^y$ para quaisquer $x, y \in \{+, -\}$. Por exemplo, para a permutação $\pi = (0,\ 3,\ 1,\ 6,\ 5,\ -2,\ 4,\ 7)$, temos $\pi' = (0^+,\ 3^-,\ 3^+,\ 1^-,\ 1^+,\ 6^-,\ 6^+,\ 5^-,\ 5^+,\ 2^+,\ 2^-,\ 4^-,\ 4^+,\ 7^-)$.

Denotamos por v_i, $i \in \{0, \ldots, n\}$ um arco entre os pontos π_i^+ e π_{i+1}^-. Temos assim $n+1$ arcos, sendo que cada ponto de π' é o ponto extremo de um único arco. (No nosso exemplo o arco v_0 tem pontos extremos em 0^+ e 1^- e cobre os elementos $0^+\ 3^-\ 3^+\ 1^-$.) Esses arcos serão considerados arcos de π'. Dizemos que dois arcos se *sobrepõem* se o intervalo que eles cobrem (isto é, o conjunto de pontos entre os pontos extremos na dada ordem) se intersectam, mas um não contém o outro (é o caso dos arcos v_0 e v_2 do exemplo dado). Um arco v define um intervalo $[i, j]$ em π dos elementos k, tal que k^- e k^+ estão entre os pontos extremos de v. Esse intervalo induz uma reversão que denotaremos por $\rho(v) = \rho_{i,j}$. Dizemos que um arco v_i é *orientado* se π_i e π_{i+1} têm sinais diferentes; e *não-orientado*, caso contrário.

O *grafo de sobreposição* $G_s(\pi) = (V, A)$ é o grafo cujo conjunto de vértices é formado pelos $n+1$ arcos v_i de π', e no qual uma aresta $v_i v_j \in A$ se os arcos v_i e v_j se sobrepõem em π'. Um vértice do grafo $G_s(\pi)$ é *orientado* se o arco que ele representa é orientado e não-orientado, caso contrário. Vértices *isolados* são vértices não-orientados e correspondem às adjacências da permutação.

Um *componente* de π é um componente conexo de $G_s(\pi)$. É *orientado* se um de seus vértices é orientado, e *não-orientado*, caso contrário. Para simplificar, quando nos referirmos a componentes não-orientados de G_s excluiremos os vértices isolados.

O complemento local de um grafo. Seja $G = (V, A)$ um grafo cujos vértices são rotulados como orientados e não-orientados. Seja $W \subseteq V$. Lembramos que o *subgrafo induzido* por W é o subgrafo de G com conjuntos de vértices W e com uma aresta entre dois vértices v e w se, e somente se, a aresta $vw \in A$.

O *complemento local* do subgrafo induzido por W é a operação que consiste em adicionar uma aresta entre $v, w \in W$ se $vw \notin A$, excluir vw se $vw \in A$ e alterar a orientação de todos os vértices de W. Dado um vértice v, denotamos por $N(v)$ a *vizinhança* de v, que é o conjunto dos vértices adjacentes a v, e $N^+(v) = N(v) \cup \{v\}$ a *vizinhança fechada* de v.

Se v é um vértice orientado de G, denotamos por G/v o resultado do complemento local de $N^+(v)$, que chamamos simplesmente de complemento local de v. Note que em G/v, o vértice v é não-orientado e isolado.

A relação entre ordenação por reversão e complemento local é dada pelo seguinte resultado.

Lema 1. *Para uma permutação π, e um vértice orientado v de $G_s(\pi)$, temos que $G_s(\pi \cdot \rho(v)) = G_s(\pi)/v$.*

Como um vértice v em G é não-orientado e isolado em $G_s(\pi)/v$, então pelo lema acima temos que $G_s(\pi \cdot \rho(v))$ contém pelo menos uma adjacência a mais que $G_s(\pi)$ (eliminamos um ponto-de-quebra). Então o problema de ordenação por reversão pode ser resolvido aplicando a operação de complemento local em G_s até que todos os seus vértices sejam isolados. O teorema de Hannehalli e Pevzner [11] citado a seguir garante que sempre teremos vértices orientados em $G_s(\pi)$ se escolhermos um vértice apropriado para aplicarmos o complemento local. Esse teorema é a base da teoria de ordenação por reversão. É explicado em duas partes e a primeira serve como base para um outro teorema que inspirou o algoritmo que descreveremos nesta monografia. Vamos restringir os resultados para permutações sem componentes não-orientados.

Teorema 2. *Se G é um grafo sem componentes não-orientados, então existe um vértice orientado v tal que G/v não tem componentes não-orientados.*

Um vértice orientado v em G tal que G/v não tem componentes não-orientados é chamado *seguro*. O seguinte resultado mostra uma propriedade importante dos vértices seguros.

Teorema 3. *Para uma dada permutação π, se v é um vértice seguro do grafo de sobreposição $G_s(\pi)$, então $d(\pi \cdot \rho(v)) = d(\pi) - 1$.*

A partir desse teorema é possível ter uma idéia para a construção de um algoritmo para ordenar uma permutação π sem componentes não-orientados, basta sabermos encontrar um vértice seguro, que sabidamente existe (pelo Teorema 2). Temos que escolher um vértice orientado v, aplicar o complemento local e testar se há algum componente não-orientado no grafo resultante. Se houver um é porque v não era seguro, então desfazemos o complemento local e tentamos um outro vértice orientado. Essa idéia leva a um algoritmo de complexidade $O(n^3)$ para encontrar um vértice seguro, e foi o fundamento do primeiro algoritmo proposto para resolver o problema.

Uma forma mais eficiente de se encontrar um vértice seguro foi proposta em [14]. Essa técnica permitiu a construção de algoritmos de complexidade $O(n^2)$ para o problema OR(n). O método que veremos aqui não faz um teste explícito para verificar se um vértice orientado escolhido é seguro. Escolhe-se um vértice orientado qualquer e mais tarde faz alguns reparos se em algum ponto foi escolhido um vértice que não era seguro.

Uma outra versão do Teorema 2.

A seguir apresentaremos uma versão alternativa do Teorema 2 que é aplicada ao caso particular de grafos de sobreposição de permutações. Essa nova versão explica como podemos deixar de verificar se um vértice é seguro na hora da escolha de um vértice orientado para a aplicação do complemento local.

A cada passo do procedimento é escolhido um vértice orientado e aplicado o complemento local. Esses vértices são colocados em seqüência e, ao final, queremos que o grafo possua apenas vértices isolados não-orientados, pois como eles representam adjacências, teremos certeza de que a permutação foi ordenada. Como o complemento local sempre atua em um único componente, então podemos supor, sem perda

de generalidade, que o grafo é conexo, pois podemos tratar cada componente separadamente.

Seja G um grafo com n vértices. Uma *seqüência de vértices orientados* de G é uma seqüência $v_1, \ldots, v_k$ tal que, para todo $i \in \{1, \ldots k\}$, v_i é um vértice orientado em $G/v_1/\ldots/v_{i-1}$. Uma tal seqüencia é *maximal* se nenhum vértice de $G/v_1/\ldots/v_k$ é orientado. E é *total* se é maximal e todo vértice de $G/v_1/\ldots/v_k$ é isolado. Vamos provar que sempre é possível construir uma seqüência total de vértices orientados para qualquer grafo conexo com pelo menos um vértice orientado. Esse resultado é equivalente ao Teorema 2

O algoritmo consiste em produzir uma seqüência total de vértices orientados. Constrói-se inicialmente uma seqüência maximal de vértices orientados. Se essa seqüência é não-total então é porque em algum momento foi escolhido um vértice que não era seguro. Não podemos adicionar vértices no final da seqüência, pois ela é maximal, mas o teorema a seguir diz que é possível quebrá-la em duas subseqüências de forma a acrescentar vértices orientados entre elas e obtermos uma seqüência maior de vértices orientados.

Teorema 4. *Se S é uma seqüência maximal, mas não-total de vértices orientados de um grafo G com um único componente, então existe uma seqüência não-vazia S' de vértices de G tal que S pode ser dividida em duas partes $S = S_1, S_2$ tal que S_1, S', S_2 é uma seqüência de vértices orientados de G.*

Demonstração. Seja $S = v_1, \ldots, v_k$ uma seqüência maximal, mas não-total de vértices orientados de G. Vamos denotar por U o conjunto dos vértices não-isolados em $G/v_1/\ldots/v_k$. Sabemos que os vértices de U pertencem a componentes não-orientados de $G/v_1/\ldots/v_k$. Seja v_l um vértice na seqüência e $G_1 = G/v_1/\ldots/v_{l-1}$, tal que todos os vértices de U estão em componentes não-orientados em G_1/v_l, mas não em G_1, ou seja, v_l é um vértice que não era seguro em G_1. Como, por hipótese, G é conexo, sabemos que v_l existe. Sejam $S_1 = v_1, \ldots, v_{l-1}$ e $S_2 = v_l, \ldots, v_k$. Mostraremos que é sempre possível aplicar dois complementos locais em vértices de U no grafo G_1, e que S_2 é uma seqüência de vértices orientados no grafo resultante.

Seja O o conjunto de vértices em componentes de vértices orientados de G_1/v_l, e L o subconjunto de vértices de O adjacentes a v_l em G_1. Em G_1, as seguintes propriedades seguem diretamente da definição de complemento local:

- Um vértice de U é orientado se, e somente se, é adjacente a v_l.
- Todas as arestas possíveis entre $N(v_l) \cap U$ e L estão em G_1.
- Não há nenhuma aresta entre vértices de $U \setminus N(v_l)$ e vértices fora de U.

Existe pelo menos um vértice w_1 em $N(v_l) \cap U$ tal que $N^+(v_l) \neq N^+(w_1)$. De fato, se isto não ocorresse, então o complemento local de qualquer vértice orientado em U teria o mesmo efeito que o complemento local de v_l. Como o complemento local de w_1 tornaria w_1 isolado, temos que w_1 seria isolado em G_1/v_l, e portanto não pertenceria a U; contradizendo a definição de v_l.

Então existe $w_2 \in U$ em $N^+(w_1) \setminus N^+(v_l)$. Temos que w_2 é não-orientado. Se aplicarmos o complemento local de w_1 em G_1, o vértice v_l se torna não-orientado e adjacente a w_2, e w_2 fica orientado e adjacente aos vértices de L. As propriedades

acima continuam valendo (note que $N(v_l)$ é alterado mas L não varia), v_l não tem mais nenhum vizinho em O e o subgrafo induzido por L é o complemento do que era em G_1.

Agora aplicamos o complemento local de w_2 em G_1/w_1. Então v_l volta a ser orientado e adjacente aos vértices de L. As propriedades ainda valem e o subgrafo induzido por L é complementado novamente, então se torna idêntico em $G_1/w_1/w_2$ ao que era em G_1; portanto o subgrafo induzido por $O \cup \{v_l\}$ também é idêntico em $G_1/w_1/w_2$ ao que era em G_1. Temos então que a seqüência S_2 que era uma seqüência de vértices orientados em G_1, continua sendo em $G_1/w_1/w_2$, pois os complementos locais dos vértices de S_2 são aplicados somente ao subgrafo induzido por $O \cup \{v_l\}$.

Portanto podemos adicionar os vértices w_1 e w_2 entre S_1 e S_2 e aumentar a seqüência S. Então S_1, w_1, w_2, S_2 é uma seqüência de vértices orientados de G, e isso conclui a prova. □

4.1. **O algoritmo.** O Teorema 4 e sua prova mostram como construir uma seqüência total de vértices orientados. Primeiro constrói-se uma seqüência maximal e então segue-se aumentando-a até que ela se torne total. Segue abaixo o algoritmo que utiliza a idéia do teorema dado.

Entrada: seqüência π
0. Seja V: conj. dos arcos definidos por π
   $S_1 \leftarrow \emptyset$; $S_2 \leftarrow \emptyset$
1. Enquanto há $v \in V$, v orientado, adicione a reversão $\rho(v)$ no final de S_1:
   $\pi \leftarrow \pi \cdot \rho(v)$
   Atualize V
   Se o primeiro elemento de S_2 é não-orientado, desfaça a última reversão.
2. Se $V = \emptyset$, vá para o passo (3)
   senão suponha $S_1 = \rho_1, \ldots, \rho_k$. (Re)aplique as reversões de S_1 na ordem reversa: $\pi = \pi \cdot \rho_k \cdots \rho_j$,
   até que haja um arco orientado em V.
   Remova $\rho_j, \ldots, \rho_k$ de S_1
   Adicione $\rho_j, \ldots, \rho_k$ no início de S_2
   Vá para o passo (1)
3. A seqüência S_1, S_2 ordena π

4.2. **Complexidade.** Podemos observar que a operação de reversão só é aplicada uma vez no passo 1 e talvez reaplicada uma vez no passo 2, onde é adicionada em S_2, para cada vértice orientado $v \in V$. Depois de aplicada a reversão referente ao vértice v, este se torna isolado e então é retirado do conjunto V. Para entrar no conjunto S_1 o arco referente a reversão tem que estar em V. Uma reversão só é reaplicada no passo 2 se estiver em S_1 e depois que entra no conjunto S_2 a reversão nunca mais é aplicada. Temos então que o algoritmo executa no máximo $n + 1$ iterações. Em cada iteração temos que detectar e escolher um arco orientado e aplicar a reversão na permutação. A complexidade depende então do tempo necessário para executar essas operações. Com uma estrutura de dados clássica, usando um vetor para representar

a permutação corrente, essas operações podem ser executadas em tempo linear. Portanto esse o algoritmo tem complexidade $O(n^2)$, e qualquer outro que também tenha que aplicar uma reversão a cada iteração vai ter a mesma complexidade.

Para diminuir essa complexidade é necessário utilizar uma estrutura de dados mais complexa. Mas na prática, para pequenas permutações, talvez a implementação quadrática seja melhor que a subquadrática, pois a manipulação dessa estrutura mais complexa pode não ser tão eficiente.

Kaplan e Verbin [10] desenvolveram uma estrutura de dados para representar uma permutação que permite a escolha de um arco orientado e a aplicação da respectiva reversão em tempo $O(\sqrt{n \log n})$. Tannier, Bergeron e Sagot [6] fizeram uso dessa estrutura mais inteligente para descrever a implementação do algoritmo acima de maneira mais sofisticada que pode ser executada em tempo $O(n^{3/2}\sqrt{\log n})$.

5. Outros problemas similares

Nesta seção vamos apresentar algumas variantes do problema de ordenação por reversão e alguns resultados já apresentados para resolvê-los.

5.1. **Cromossomos circulares.** *Cromossomos circulares* são permutações circulares de genes. Meidanis, Walter e Dias [12] foram os primeiros a apresentar um algoritmo polinomial para determinar a distância de reversão em cromossomos circulares, dados que são conhecidos a direção de transcrição de seus genes. Basearam-se no algoritmo de Kaplan, Shamir e Tarjan [13] para resolver o problema em cromossomos lineares com sinais.

Foi observado que há menos reversões no caso circular do que no caso linear de mesmo tamanho (uma reversão a menos). Eles mostraram que o *diâmetro de reversão circular* $D^c(n)$ (máximo número de reversões necessárias para transformar uma classe de equivalência em outra) e o *diâmetro de reversão linear* $D(n)$ (distância de reversão linear) é respectivamente n e $n+1$ para cromossomos correspondentes com sinais, onde n é o tamanho da permutação. Isso implica que todos os algoritmos apresentados para ordenar permutações lineares podem ser usados para ordenar os cromossomos circulares.

5.2. **Inserção e exclusão de segmentos.** Existem vários tipos de rearranjo de genomas além da reversão, como a inserção e exclusão de algum gene. A diferença entre dois genomas talvez seja a presença ou ausência de algum elemento da permutação. Os algoritmos de ordenação por reversão não consideram esses casos para o cálculo da distância entre dois genomas, pois o problema OR considera a comparação entre permutações de mesmo elementos.

O caso de reversões e inserções numa permutação é o processo inverso ao caso de reversões e exclusões. Nesses casos apenas uma das permutações possui genes a mais. Mabrouk [7] mostrou como estender o algoritmo de Hannenhalli e Pevzner (HP) para incluir inserções e exclusões de genes, permitindo a comparação de genomas em que ambos contenham genes diferentes.

Considerar essas condições nos cromossomos a serem comparados torna mais realista o cálculo da diferença entre genomas. Uma das idéias para problemas combinatórios é estender os algoritmos apresentados, como o HP, para outros mecanismos de rearranjo, tais como duplicação de genes, ou a transposição, duplicação ou inserção de fragmentos genômicos.

O algoritmo apresentado por Mabrouk ainda pode ser estendido para incluir transposições, fusão e fissão de cromossomos. Isso permite o cálculo de distâncias definidas em genomas multicromossômicos, que incluem essas operações tanto quanto reversões, inserções e exclusões. A complexidade desse método depende da complexidade do algoritmo HP.

Mabrouk ainda testou o método em alguns dados reais de tamanho moderado e concluiu que o número de operações utilizadas foi mínimo.

5.3. **Ordenação por reversão sem sinal.** Utilizamos sinais na representação de genomas quando sabemos sua direção de transcrição. Os biólogos fazem essa transcrição pelo mapeamento físico dos genomas. Mas esses mapas físicos geralmente não informam a direção desses genes o que leva a uma representação não-sinalizada do genoma.

Quando não sabemos a direção de transcrição dos genes temos a versão de ordenação por reversão sem sinal. Apesar da pouca diferença, enquanto a versão sinalizada é polinomial, esse problema foi recentemente provado ser $\mathcal{NP}$-difícil por Caprara [5]. Desde então tem sido um desafio encontrar um algoritmo com uma boa aproximação para o problema. A melhor aproximação até agora encontrada foi dada por Berman, Hannenhalli e Karpinski [4]. Eles desenvolveram uma $1.375-$ aproximação para o problema explorando um algoritmo polinomial de ordenação por reversão com sinal.

5.4. **Problema por reversão de prefixos.** Como o próprio nome já diz, esse problema possui a restrição de que só é possível aplicar a reversão em prefixos da permutação.

Também conhecido como *problema de inversão de panquecas*: Dada uma permutação π, encontrar $d_{pref}(\pi)$, que é o número mínimo de reversões da forma $\rho_{1,i}$ para ordenar π. É um caso especial de ordenação por reversão sem sinal.

Gates e Papadimitriou [9] fizeram a primeira tentativa de resolver esse problema. Eles provaram que o *diâmetro de reversão do prefixo*, $d_{pref}(n) = max_{\pi \in S_n} d_{pref}(\pi)$, é menor ou igual a $\frac{5}{3}n + \frac{5}{3}$ e para infinitos n, $d_{pref}(n) \geq \frac{17}{16}n$. Esse problema ainda continua sem solução.

5.5. **Intervalos conservados.** É o problema de ordenação por reversão com a restrição de não permitir a quebra de blocos conservados de genes usualmente chamados de *intervalos comuns*, que são intervalos de segmentos genômicos que é comum nos dois cromossomos.

A presença de intervalos comuns nos genomas analisados pode ser uma indicação de que esse intervalo de genes permaneceu agrupado durante a evolução. Essa

característica dificulta um pouco o problema de ordenação por reversão, pois um cenário mínimo de reversões pode quebrar segmentos conservados e colocá-los juntos novamente em outra reversão.

Esse problema é objeto de alguns trabalhos bastante recentes [3, 8] que misturam os conceitos de intervalos conservados e ordenação por reversão. Apesar desses dois conceitos não serem sempre compatíveis, Sagot e Tannier [15] descreveram um algoritmo polinomial que decide se existe um cenário mínimo de reversões que transfoma um genoma em outro enquanto mantém os blocos de intervalos comuns juntos.

5.6. **Desempilhamentos em série.** Uma *pop-stack* é uma pilha com uma operação de desempilhamento restrita: uma operação de empilhar é executada de maneira usual, mas uma operação de desempilhar tira todos os elementos da pilha de uma só vez. Essa operação se parece muito com uma reversão.

O problema apresentado por Avis e Newborn [1] considera uma seqüência de pilhas pop-stack em série, tal que uma operação de desempilhar em uma pilha tira todos os elementos dessa pilha um de cada vez e empilha na pilha seguinte. Avis e Newborn analisam, dadas m pilhas em série, quais permutações podem ser ordenadas. E apresentam meios eficientes de decidir quantas pop-stacks em série são necessárias para tornar uma permutação viável.

6. Aplicações

A grande motivação no estudo do problema de ordenação por reversão é a análise de rearranjos de genomas em cromossomos, organelas de plantas, vírus, etc. Tal análise possibilita descobrir relações entre os genomas de duas espécies diferentes e, com isso visualizar a evolução entre elas. O rearranjo de genomas permite comparar dados moleculares de espécies que divergiram há muito tempo. Resultados e complexidades estão fortemente relacionados ao tipos de dados e mutações em nível de genoma considerados.

O grande volume de dados, oriundos de seqüenciamento de genes em Biologia Molecular, vem suscitando um crescente interesse pelo desenvolvimento de algoritmos para comparação de genomas de espécies relacionadas, em termos de mutações ocorrendo em grandes porções dos seus cromossomos.

7. Conclusão

Ordenação por reversão é um problema computacional que tem sido objeto de muito estudo recente. Existem várias variantes que também possuem grande importância em Biologia Computacional. A investigação desse assunto estimula a busca de novos métodos que ajudem a explicar o processo evolutivo. Apresentamos aqui algumas aplicações e vários algoritmos propostos para resolver o problema de ordenação por reversão e também vários problemas similares que podem ser objeto de estudos futuros. Apesar dos resultados já apresentados para esse problema serem bastante conclusivos vemos que ainda existe uma grande área de pesquisa de interesse em Biologia Computacional que inspira muitos problemas combinatórios.

Referências

1. David Avis and Monroe Newborn, *On pop-stacks in series*, Utilitas Mathematica **19** (1981), 129–140.
2. Moret B.M.E. Bader D. A. and Yan M., *A linear-time algorithm for computing inversion distance between signed permutations with an experimental study*, Proceedings of the 7th Workshop on Algorithms and Data Structures, 2001, pp. 365–376.
3. Anne Bergeron, Cedric Chauve, and Sèverine Bérard, *Conservation of combinatorial structures in evolution scenarios*, RECOMB Satellite Workshop on Comparative Genomics, 2004.
4. P. Berman, S. Hannenhalli, and M. Karpinski, *1.375-approximation algorithm for sorting by reversals*, Eletronic Colloquium for Computational Complexity TR01-047, 2001.
5. Alberto Caprara, *Sorting by reversals is difficult*, RECOMB '97: Proceedings of the first annual international conference on Computational molecular biology (New York, NY, USA), ACM Press, 1997, pp. 75–83.
6. Tannier E., Bergeron A., and Sagot M-F., *Advances on sorting by reversals*, Tech. report, Jan 2005.
7. Nadia El-Mabrouk, *Genome rearrangement by reversals and insertions/deletions of contiguous segments*, Lecture Notes in Computer Science, vol. 1848, Jan 2000, p. 222.
8. Martin Figeac and Jean-Stéphane Varré, *Sorting by reversals with common intervals*, Lecture Notes in Computer Science, vol. 3240, 2004, pp. 26–37.
9. Willian H. Gates and Christos H. Papadimitriou, *Bounds for sorting by prefix reversals*, Discrete Mathematics **27** (1979), 47–57.
10. Kaplan H. and Verbin E., *Sorting signed permutations bu reversals, revisited*, Journal of Computer and System Sciences **70** (2005), 321–341.
11. Sridhar Hannenhalli and Pavel Pevzner, *Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals*, STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 1995, pp. 178–189.
12. Meidanis J., Walter M.E., and Dias Z., *Distância de reversão de cromossomos circulares*, Proceedings of SEMISH - XXIV Seminario Integrado de Software e Hardware, Agosto 1997, pp. 119–131.
13. Haim Kaplan, Ron Shamir, and Robert E. Tarjan, *Faster and simpler algorithm for sorting signed permutations by reversals*, SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 1997, pp. 344–351.
14. Ozery-Flato M. and Shamir R., *Two notes on genome rearrangements*, Journal of Bioinformatics and Computational Biology **1** (2003), 71–94.
15. Sagot M-F. and Tannier E., *Perfect sorting by reversals*, Tech. report, Feb 2005.
16. Berman P. and Hannenhalli S., *Fast sorting by reversals*, Lecture Notes in Computer Science, vol. 1075, In Combinatorial Pattern Matching, Proceedings of the 7th Annual Symposium (CPM'96), 1996, pp. 168–185.
17. Pevzner P. and Tesler G., *Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes*, Genome Research **13** (2003), 37–45.
18. Hannenhalli S. and Pevzner P., *Towards a computational theory of genome rearrangements*, Lecture Notes in Computer Science, vol. 1000, 1995, pp. 184–202.
19. Dobzhansky T. and Sturtevant A. H., *Inversions in the chromosomes of drosofila pseudoobscura.*, Genetics **23** (1938), 28–64.

Instituto de Matemática e Estatística — Universidade de São Paulo (USP) – Rua do Matão 1010, 05508–090 São Paulo, SP

Endereços Eletrônicos: tieme@ime.usp.br