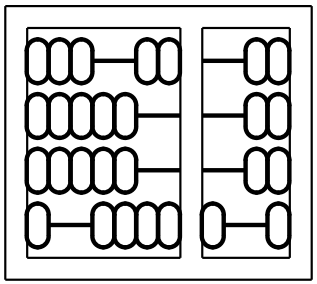
Programação Linear Inteira e Programação Lógica por Restrições para Problemas de Rearranjo de Genomas

Victor de Abreu Iizuka

Orientador: Zanoni Dias
Instituto de Computação - UNICAMP



1. Introdução
2. Ordenação por Reversões
3. Ordenação por Transposições
4. Ordenação por Reversões e Transposições
5. Programação Linear Inteira
   - Modelo
   - Metodologias
6. Programação Lógica por Restrições
   - Formulações
   - Metodologias
7. Objetivos
8. Cronograma

Introdução

- Rearranjo de genomas é uma área que vem recebendo crescente atenção de pesquisadores no decorrer da última década.
- Rearranjos são eventos que melhor caracterizam a distância evolutiva entre duas éspécies do que o estudo de mutações pontuais.
- No modelo de rearranjo de genomas, um genoma é representado por uma seqüência de números inteiros, onde cada inteiro representa um gene ou um grupo de genes.

Introdução

- A comparação de seqüências é o método mais usual de se caracterizar a ocorrência de mutações pontuais, sendo um dos problemas mais abordados em Biologia Computacional.
- A distância de edição é o número mínimo de operações de inserção, remoção e substituição necessárias que transformam uma seqüência em outra.
- Estima a distância evolutiva entre duas cadeias, mas não informa quais operações globais foram utilizadas para transformar uma seqüência em outra.

Introdução

- Estas operações globais são os chamados Rearranjos de Genomas, que podem ser, por exemplo, Reversões, Transposições, Fissões e Fusões.

Definição:

Para fins computacionais, um genoma é representado por uma n-tupla de genes, quando não há genes repetidos essa n-tupla é chamanda de permutação. Uma permutação é representada como $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, onde π_i, para todo $1 \leq i \leq n$, representa um gene (ou um grupo de genes) e os vários eventos de rearranjo ρ são aplicados a π de uma maneira específica.

Ordenação por Reversões

Evento de Reversão:

O evento de reversão ocorre quando um bloco do genoma é invertido.

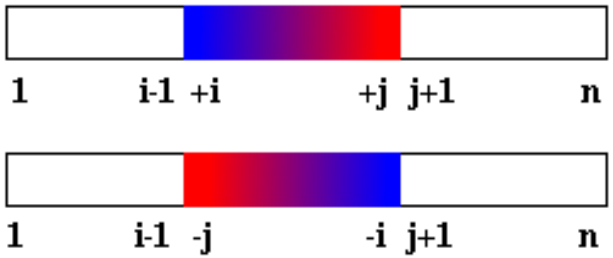


Figura: Reversão em uma permutação orientada

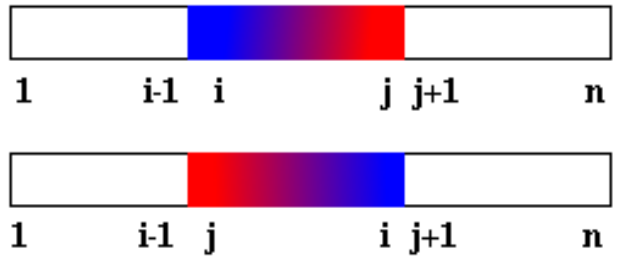


Figura: Reversão em uma permutação não orientada

Ordenação por Reversões

- Em um estudo inicial, Bafna e Pevzner (1993) apresentaram um algoritmo de aproximação com razão de 1.5 quando a orientação dos genes é conhecida e 1.75 caso contrário.
- Conhecer a orientação dos genes em um genoma é importante no problema de reversão, pois existem algoritmos polinomiais caso a orientação seja conhecida.
- Se não se conhece a orientação dos genes este problema pertence a classe dos problemas NP-Difíceis, demonstrado por Caprara, Lancia e Ng (1997).

Ordenação por Reversões

- O primeiro algoritmo polinomial para o problema de reversão com orientação conhecida foi criado por Hannenhalli e Pevzner (1995).
- A estratégia usada por Hannenhalli e Pevzner foi simplificada por Bergeron (2001).
- Existe um algoritmo com complexidade sub-quadrática apresentado por Tannier e Sagot (2004).
- Bader Moret e Yan (2001) apresentaram um algoritmo linear para calcular o valor da distância de reversão.

Ordenação por Reversões

- Meidanis, Walter e Dias (2000) mostraram que toda teoria sobre reversões desenvolvida para genomas lineares pode ser adaptada facilmente para genomas circulares.
- Quando a orientação dos genes não é conhecida existem algoritmos de aproximação que se seguiram ao de Bafna e Pevzner citado anteriormente.
- Berman, Hannenhalli e Karpinski (2002) apresentaram um algoritmo com razão de aproximação de 1.375.

Ordenação por Transposições

Evento de Transposição:

O evento de transposição ocorre quando dois blocos adjacentes no genoma trocam de posição.

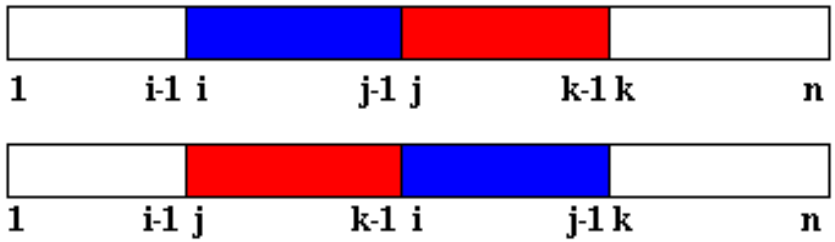


Figura: Transposição aplicada em uma permutação.

Ordenação por Transposições

- O problema de encontrar a distância de transposição, que envolve encontrar o menor número de transposições necessárias para transformar um genoma em outro.
- Bafna e Pevzner (1995) apresentaram um algoritmo de aproximação com razão 1.5, além de derivar um importante limitante inferior para o problema.
- As heurísticas implementadas por Bafna e Pevzner utilizaram os conceitos de *breakpoints* e de grafo de cíclos orientados com arestas de cores alternadas.

Ordenação por Transposições

- Não se conhece provas que este problema pertence a classe dos problemas NP-Difíceis e não se conhece algoritmos polinomiais, o que torna o estudo do problema interessante.
- Recentemente, Elias e Hartman (2006) propuseram um novo algoritmo de aproximação com razão 1.375.
- Labarre (2006) apresentou novos limitantes, além de definir classes de permutações em que a distância de transposição pode ser calculada em tempo e espaço lineares.

Ordenação por Transposições

- Uma generalização do problema de transposição é a operação de troca de blocos, cuja solução pode auxiliar na resolução do problema da distância de transposição.
- Christie (1996) apresentou um algoritmo polinomial para o problema de troca de blocos e utilizou conceitos já conhecidos na literatura sobre transposição.

Ordenação por Reversões e Transposições

- Hannenhalli e co-autores (1995) analisaram a evolução de genomas por diferentes eventos, em especial reversões e transposições.
- Gu, Peng e Sudborough (1996) criaram um algoritmo de aproximação com razão 2 para computar a distância entre dois genomas com a orientação dos genes conhecida.
- O algoritmo permite operações de reversão, transposição e reversão e transposição simultaneamente.

Ordenação por Reversões e Transposições

- Walter, Dias e Meidanis (1998) apresentaram um algoritmo de aproximação para a distância de reversão e transposição.
- Também forneceram limitantes para o diâmetro de reversão e transposição em permutações orientadas que foram melhorados em Meidanis, Walter e Dias (2002).
- Dias e de Souza (2007) apresentaram uma modelagem com programação linear inteira para o problema de reversão, transposição ou reversão e transposição simultaneamente.

Programação Linear Inteira

- Modelo para o problema da distância com tamanho polinomial apresentado por Dias e de Souza (2007).
- Modelo específico para os eventos de reversão, transposição ou reversão e transposição simultaneamente.
- Possui um tempo de execução muito grande. O programa retornava *timeout* para instâncias de tamanho 10 (tempo limite de 10 horas).
- Inviável na prática.

Modelo

Primeiramente vamos apresentar as variáveis e restrições que são comuns para todos modelos. A idéia é assegurar que só estamos tratando com permutações válidas.

Variáveis B_{ijk}

Indicam se a i-ésima posição de π possui o valor j depois da k-ésima operação ter sido executada, para todo $1 \leq i, j \leq n$ e todo $0 \leq k < n$.

$$B_{ijk} = \begin{cases} 1, & \text{se } \pi[i] = j \text{ depois da } k\text{-ésima operação} \\ 0, & \text{caso contrário} \end{cases}$$

Modelo - Restrições Comuns

As restrições 1 e 2 garantem que a permutação inicial e a final são corretas.

$$B_{i,\pi[i],0} = 1, \text{para todo } 1 \leq i \leq n. \quad (1)$$

$$B_{i,\sigma[i],n-1} = 1, \text{para todo } 1 \leq i \leq n. \quad (2)$$

A restrição 3 garante que cada posição de uma permutação possui exatamente um valor associado a ela. Já a restrição 4 garante que todo valor esteja associado a uma posição de cada permutação.

$$\sum_{j=1}^{n} B_{ijk} = 1, \text{para todo } 1 \leq i \leq n,\ 0 \leq k < n. \quad (3)$$

$$\sum_{i=1}^{n} B_{ijk} = 1, \text{para todo } 1 \leq i \leq n,\ 0 \leq k < n. \quad (4)$$

Modelo - Transposição

Agora vamos apresentar as variáveis e restrições que são referentes à operação de transposição.

Variáveis Binárias t_{abck}

Indicam quando a k-ésima operação de transposição realiza a troca de lugares dos blocos $\pi[a \cdots b-1]$ e $\pi[b \cdots c-1]$ de π, para todo $1 \leq a < b < c \leq n+1$ e todo $1 \leq k < n$.

$$t_{abck} = \begin{cases} 1, & \text{se } \rho_k = \rho(a,b,c) \\ 0, & \text{caso contrário} \end{cases}$$

Variáveis Binárias t_k

É usada para decidir se a k-ésima operação de transposição modificou a permutação, para todo $1 \leq k < n$.

$$t_k = \begin{cases} 1, & \text{se } \rho_k = \rho(x,y,z) \text{ e } \rho_k\rho_{k-1}\cdots\rho_1\pi \neq \rho_{k-1}\cdots\rho_1\pi \\ 0, & \text{caso contrário} \end{cases}$$

Modelo - Transposição - Restrições

As restrições 5 e 6 são necessárias para identificar as transposições que fazem parte da solução.

$$t_k \leq t_{k-1}, \text{para todo } 1 \leq k < n. \tag{5}$$

$$\sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{n+1} t_{abck} \leq t_k, \text{para todo } 1 \leq k < n. \tag{6}$$

As próximas restrições refletem a modificações na permutação causada pela transposição a cada passo da execução. A análise será dividida em três casos onde, para cada caso, analisamos cada posição i da permutação para verificar seu valor após a operação de transposição $\rho(a, b, c)$ ser completada.

Modelo - Transposição - Restrições

1. $i < a$ ou $i \geq c$:

$$\sum_{a=i+1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{n+1} t_{abck} + \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{i} t_{abck} + (1 - t_k) + B_{i,j,k-1} - B_{ijk} \leq 1,$$

$$\text{para todo } 1 \leq i, j \leq n \text{ e todo } 1 \leq k < n. \quad (7)$$

2. $a \leq i < a + c - b$:

$$t_{abck} + B_{b-a+i,j,k-1} - B_{ijk} \leq 1,$$

$$1 \leq a < b < c \leq n+1, a \leq i < a + c - b, 1 \leq j \leq n, 1 \leq k < n. \quad (8)$$

3. $a + c - b \leq i < c$:

$$t_{abck} + B_{b-c+i,j,k-1} - B_{ijk} \leq 1,$$

$$1 \leq a < b < c \leq n+1, a + c - b \leq i < c, 1 \leq j \leq n, 1 \leq k < n. \quad (9)$$

Modelo - Transposição - Restrições

Usando o limitante superior e o inferior definido por Bafna e Pevzner (1998), podemos obter as seguintes restrições.

$$t_k * n + k - 1 \geq LB(\pi, \sigma), \text{para todo } 1 \leq k \leq n. \quad (10)$$

$$t_k * k \leq UB(\pi, \sigma), \text{para todo } 1 \leq k \leq n. \quad (11)$$

Onde LB e UB são, respectivamente, limitante inferior e limitante superior para ciclos ímpares (facilmente calculados a partir de π e σ).

Modelo - Reversão

Agora vamos apresentar as variáveis e restrições que são referentes à operação de reversão.

Variáveis Binárias r_{abk}

Indicam quando a k-ésima operação de reversão afeta o blocos $\pi[a \cdots b]$ de π, para todo $1 \leq a < b \leq n$ e todo $1 \leq k < n$.

$$r_{abk} = \begin{cases} 1, & \text{se } \rho_k = \rho(a, b) \\ 0, & \text{caso contrário} \end{cases}$$

Variáveis Binárias r_k

É usada para decidir se a k-ésima operação de reversão modificou a permutação, para todo $1 \leq k < n$.

$$r_k = \begin{cases} 1, & \text{se } \rho_k = \rho(x, y) \text{ e } \rho_k \rho_{k-1} \cdots \rho_1 \pi \neq \rho_{k-1} \cdots \rho_1 \pi \\ 0, & \text{caso contrário} \end{cases}$$

Modelo - Reversão - Restrições

As restrições 12 e 13 são necessárias para identificar as reversões que fazem parte da solução.

$$r_k \leq r_{k-1}, \text{para todo } 1 \leq k < n. \tag{12}$$

$$\sum_{a=1}^{n-1} \sum_{b=a+1}^{n} r_{abk} \leq r_k, \text{para todo } 1 \leq k < n. \tag{13}$$

As próximas restrições lidam com as modificações na permutação causada pela reversão a cada passo da execução. A análise será dividida em dois casos onde, para cada caso, analisamos cada posição i da permutação para verificar seu valor após a operação de reversão $\rho(a, b)$ ser completada.

Modelo - Reversão - Restrições

1. $i < a$ ou $i > b$:

$$\sum_{a=i+1}^{n-1} \sum_{b=a+1}^{n} r_{abk} + \sum_{a=1}^{n-1} \sum_{b=a+1}^{i-1} r_{abk} + (1 - r_k) + B_{i,j,k-1} - B_{ijk} \leq 1,$$

$$\text{para todo } 1 \leq i, j \leq n \text{ e todo } 1 \leq k < n. \qquad (14)$$

2. $a \leq i \leq b$:

$$r_{abk} + B_{b+a-i,j,k-1} - B_{ijk} \leq 1,$$

$$1 \leq a < b \leq n, a \leq i \leq b, 1 \leq j \leq n, 1 \leq k < n. \qquad (15)$$

Modelo - Reversão e Transposição

Para formular o problema da distância de reversão e transposição vamos utilizar todas as variáveis definidas anteriormente, adicionando um novo conjunto de variáveis ao problema.

Variáveis Binárias z_k

É usada para decidir se a k-ésima operação de reversão ou transposição modificou a permutação π. Portanto para todo $1 \leq k < n$:

$$z_k = \begin{cases} 1, & \text{se } \rho_k = \rho(x,y) \text{ ou } \rho_k = \rho(x,y,z) \text{ e } \rho_k\rho_{k-1}\cdots\rho_1\pi \neq \rho_{k-1}\cdots\rho_1\pi \\ 0, & \text{caso contrário} \end{cases}$$

Modelo - Reversão e Transposição - Restrições

Usaremos todas as restrições definidas anteriormente, com excessão das restrições 5, 10, 11 e 12 que serão substituidas pelas restrições 16 e 17.

Restrições

A restrição 16 garante que se não ocorreu operações em um passo da execução então não ocorrerá nenhuma operação nos passos seguintes. A restrição 17 garante que no máximo uma operação é executada a cada passo.

$$z_k \leq z_{k-1}, \text{para todo } 1 \leq k < n. \quad (16)$$

$$r_k + t_k = z_k, \text{para todo } 1 \leq k < n. \quad (17)$$

Modelo - Função Objetivo

Funções Objetivos

Transposição :
$$\omega_t = \min \sum_{k=1}^{n-1} t_k \quad (18)$$

Reversão :
$$\omega_r = \min \sum_{k=1}^{n-1} r_k \quad (19)$$

Reversão e Transposição :
$$\omega_{rt} = \min \sum_{k=1}^{n-1} z_k \quad (20)$$

Metodologias

- Aprimorar o modelo já existente usando técnicas de Relaxação Lagrangeana. Restrições complicadoras servirão como penalização na função objetivo do problema.
- Criação de um novo modelo, não necessariamente polinomial, com o objetivo de aplicar técnicas como:
  - Geração de Colunas: se o problema possuir um número não-polinomial de variáveis.
  - *Branch-and-Cut*, se o problema possuir um número não-polinomial de restrições.

Programação Lógica por Restrições

- Modelo específico para o eventos de transposição apresentado por Dias e Dias (2009).
- Duas formulações: uma baseada na teoria do Problema de Satisfação de Restrições (CSP) e a outra baseada na teoria do Problema de Otimização com Restrições (COP).
- Não utiliza algumas propriedades do problema que podem melhorar a formulação.

Formulação CSP

Primeiramente vamos apresentar a formulação baseada na teoria CSP. Uma permutação π é uma lista de elementos $(\pi_1, \pi_2, \ldots, \pi_n)$ onde $\pi_i \in \mathbb{N}$, $0 < \pi \leq n$ e $\pi_i \neq \pi_j$ para $i \neq j$. Uma transposição $\rho(i, j, k)$, $0 < i < j < k \leq n$, tem o efeito de separar a lista em quatro sub-listas $C_1 C_2 C_3 C_4$ onde $C_1 = (\pi_1, \ldots, \pi_{i-1})$, $C_2 = (\pi_i, \ldots, \pi_{j-1})$, $C_3 = (\pi_j, \ldots, \pi_{k-1})$ e $C_4 = (\pi_k, \ldots, \pi_n)$, e juntando elas da forma $\rho\pi = C_1 C_3 C_2 C_4$.

permutation/2

$$
\begin{aligned}
&permutation(\pi, N)\text{:-}\\
&\quad length(\pi, N),\\
&\quad \pi :: [1..N],\\
&\quad all\_different(\pi).
\end{aligned}
$$

transposition/5

$$
\begin{aligned}
&transposition(\pi, \sigma, I, J, K)\text{:-}\\
&\quad permutation(\pi, N),\\
&\quad permutation(\sigma, N),\\
&\quad 1 \leq I < J < K \leq N,\\
&\quad split(\pi, I, J, K, C_1, C_2, C_3, C_4),\\
&\quad \sigma = C_1, C_3, C_2, C_4.
\end{aligned}
$$

Formulação CSP

A formulação baseada na teoria CSP possui um número variável desconhecido, pois precisamos do valor da distância $d(\pi)$ para atribuir as constantes e as variáveis que representam a permutação. Para isso escolhemos um valor candidato para a distância $T \in [L..U]$, onde L é um valor conhecido para o limitante inferior e U é um valor conhecido para o limitante superior.

distance/3

$$
\begin{aligned}
&distance(\iota, 0, \_Model).\\
&distance(\pi, T, Model):-\\
&\quad bound(\pi, Model, LowerBound, UpperBound),\\
&\quad T :: [LowerBound..UpperBound],\\
&\quad indomain(T),\\
&\quad transposition(\pi, \sigma, \_I, \_J, \_K),\\
&\quad distance(\sigma, T-1, Model).
\end{aligned}
$$

Formulação COP

A formulação baseada na teoria COP, precisa de um valor para o limitante superior e algumas modificações nos predicados definidos anteriormente. Usamos as variáveis binárias B para indicar quando uma transposição modificou a permutação.
O predicado *transposition_cop*/6 é criado para suprir a necessidade de indicar quando a transposição modificou ou não a permutação. Se $(i, j, k) = (0, 0, 0)$, então $\rho\pi = \pi$. Caso contrário, a permutação foi modificada e então o valor da variável B para este caso é 1.

transposition_cop/6

transposition_cop$(\iota, \iota, 0, 0, 0, 0)$.

transposition_cop$(\pi, \sigma, I, J, K, 1)$:-*transposition*$(\pi, \sigma, I, J, K)$.

Formulação COP

O predicado *distance_cop*/3 atribui valores as variáveis binárias B utilizando o limitante superior e as restrições fazendo a permutação $\pi_k = \rho_k \pi_{k-1}$.

distance_cop/3

$$
\begin{aligned}
&distance\_cop(\pi, N, Model):\text{-}\\
&\quad bound(\pi, Model, LowerBound, UpperBound),\\
&\quad length(B, UpperBound),\\
&\quad upperbound\_constraint(\pi, B, UpperBound),\\
&\quad sum(B, Cost),\\
&\quad Cost \geq LowerBound,\\
&\quad minimize(Cost, N).
\end{aligned}
$$

Formulação COP

O predicado *upperbound_constraint*/3 recupera os valores de B para cada transposição ρ_k e insere os efeitos de ρ_k sobre a seqüência de permutações.

upperbound_constraint/3

$$
\begin{aligned}
&upperbound\_constraint(\iota, [\,], \_UpperBound).\\
&upperbound\_constraint(\pi, [B|Bs], UpperBound) :-\\
&\quad transposition\_cop(\pi, \sigma, \_I, \_J, \_K, B),\\
&\quad bound(\pi, Model, LowerBound, \_UpperBound),\\
&\quad UpperBound \geq LowerBound,\\
&\quad upperbound\_constraint(\pi, Bs, UpperBound - 1).
\end{aligned}
$$

Metodologias

- Adicionar predicados com informações não utilizadas pelas formulações citadas anteriormente.
- Podemos citar como exemplo, o grafo de cíclos da permutação que pode auxiliar na escolha de qual conjunto de transposições vai ser ramificado primeiro.

Objetivos

- Procurar um método para melhorar o tempo de execução do modelo de programação linear inteira apresentado por Dias e de Souza (2007).
- Aprimorar o modelo de programação por restrições apresentado por Dias e Dias (2009).
- Criar modelos de programação por restrições para o evento de reversão.

Cronograma

1. Cumprimento das disciplinas obrigatórias do programa de mestrado.
2. Revisão Bibliográfica dos artigos clássicos sobre rearranjo de genomas.
3. Escrita do projeto de mestrado e exame de qualificação.
4. Estudo e desenvolvimento sobre o modelo de programação linear inteira para o problema de rearranjo de genomas.
5. Estudo e desenvolvimento sobre o modelo de programação por restrições para o problema de rearranjo de genomas.

Cronograma

6. Testes dos modelos desenvolvidos.
7. Escrita da dissertação (a escrita começará a ser feita a partir dos primeiros resultados).
8. Revisão final do texto da dissertação.
9. Defesa da dissertação.

Cronograma

Ativ.	2009										2010	
	03	04	05	06	07	08	09	10	11	12	01	02
1	x	x	x	x	x	x	x	x	x	x		
2	x	x	x	x	x	x	x	x	x	x		
3					x	x	x					
4											x	x

Tabela: Cronograma de atividades (mar/2009 - fev/2010)

Cronograma

Ativ.	2010										2011	
	03	04	05	06	07	08	09	10	11	12	01	02
4	x	x	x	x								
5					x	x	x	x	x	x		
6		x	x	x				x	x	x		
7		x	x	x				x	x	x		
8											x	
9												x

Tabela: Cronograma de atividades (mar/2010 - fev/2011)