Instituto de Computação
Universidade Estadual de Campinas

Programação por Restrições aplicada a Problemas de Rearranjo de Genomas

Victor de Abreu Iizuka

Este exemplar corresponde à redação da Dissertação apresentada para a Banca Examinadora antes da defesa da Dissertação.

Campinas, 27 de julho de 2012.

Zanoni Dias (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Instituto de Computação
Universidade Estadual de Campinas

Programação por Restrições aplicada a Problemas de Rearranjo de Genomas

Victor de Abreu Iizuka[1]

Julho de 2012

Banca Examinadora:

- Zanoni Dias (Orientador)
- ?
- ?
- ?
- ?

[1]Suporte financeiro de: Bolsa da CNPq (processo 134380/2009-6) 2009–2011

Resumo

O processo evolutivo foi o principal responsável pela diferenciação entre os seres vivos e uma das teorias contemporâneas acerca do modo como ocorre esse processo afirma que, durante o curso da evolução, mudanças genéticas aconteceram, criando diferentes espécies de seres vivos. Muitas dessas mudanças são devido a mutações pontuais que alteram a cadeia de DNA, impedindo que a informação seja expressa, ou que seja expressa de um modo diferente. A comparação de sequência é o método mais usual de se caracterizar a ocorrência de mutações pontuais, sendo um dos problemas mais abordados em Biologia Computacional. O interesse em fazer tal comparação é encontrar a distância de edição [26]. A distância de edição é uma medida capaz de estimar a distância evolutiva entre duas cadeias, mas não possui a informação de quais operações globais foram utilizadas para a transformação de uma sequência em outra. Estas operações globais são os chamados Rearranjos de Genomas, que podem ser, por exemplo, reversões, transposições, fissões e fusões. Um conceito de distância pode ser definido para qualquer classe de rearranjo como sendo o menor número de operações pertencentes a essa classe que são necessários para transformar um genoma em outro. Por exemplo, chama-se a distância de reversão o menor número de reversões necessárias para transformar um genoma em outro [6] e a distância de transposição é o menor número de transposições [7]. Este trabalho segue a linha de pesquisa utilizada por Dias e Dias [13] e nós apresentaremos modelos de Programação por Restrições para ordenação por reversões e ordenação por reversões e transposições, baseados na teoria do Problema de Satisfação de Restrições e na teoria do Problema de Otimização com Restrições. Nós fizemos comparações com os modelos de Programação por Restrições para ordenação por transposições, descrito por Dias e Dias [13], ordenação por reversões e ordenação por reversões e transposições com os modelos de Programação Linear Inteira para os mesmos problemas descritos em Dias e Souza [14].

Abstract

The evolutionary process was the main responsible for the differentiation between living beings and one of contemporary theories about the way as this process occurs states that, during the course of evolution, genetic changes occurred, creating different kinds of living beings. Many of these changes are due to point mutation that modify the DNA sequence, preventing that information is expressed, or it is expressed in another way. The sequence comparison is the most common method to characterize the occurrence of point mutation, and is one of the most discussed problems in Computational Biology. The interest in making such comparison is to find the edit distance [26]. The edit distance is a measure capable of estimating the evolutionary distance between to sequences, but lacks the information of which global operations were used to change one sequence in another. These global operations are called the Genome Rearrangements, which may be, for example, reversals, transpositions, fissions and fusions. The distance can be defined for any rearrangement class as the smallest number of operations required to change one sequence into another. For example, the reversal distance is the smallest number of reversals required to change one sequence into another [6] and the transposition distance is the smallest number of transpositions [7]. This work follows the research line used by Dias e Dias [13] and we introduce Constraint Logic Programming models for sorting by reversals and sorting by reversals and transpositions, based on Constraint Satisfaction Problems theory and Constraint Optimization Problems theory. We made a comparison between the Constraint Logic Programming models for sorting by transpositions, described in Dias and Dias [13], sorting by reversals and sorting by reversals and transpositions, with the Integer Linear Programming for the same problems described in Dias and Souza [14].

Agradecimentos

Gostaria de agradecer aos meus pais que fizeram tudo de possível para eu conseguir chegar até aqui, dando apoio e aquele "puxão de orelha" sempre quando precisou, ao meu irmão que sempre me ajuda quando preciso e a todos os familiares que me apoiaram durante esse período.

Agradeço também ao professor Zanoni por ter auxiliado em todo o momento em que estive com dúvidas e pela paciência dispensada durante a orientação.

Finalmente, agradeço a todos os meus amigos, porque sem eles certamente não seria possível continuar esse trabalho.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O processo evolutivo foi o principal responsável pela diferenciação entre os seres vivos e uma das teorias contemporâneas acerca do modo como ocorre esse processo afirma que, durante o curso da evolução, mudanças genéticas aconteceram, criando diferentes espécies de seres vivos.

Muitas dessas mudanças são devido a mutações pontuais que alteram a cadeia de DNA, impedindo que a informação seja expressa, ou que seja expressa de um modo diferente. Tais alterações debilitam, na maioria dos casos, o organismo portador ou proporcionam vantagens no processo de seleção natural.

A comparação de sequência é o método mais usual de se caracterizar a ocorrência de mutações pontuais, sendo um dos problemas mais abordados em Biologia Computacional. O interesse em fazer tal comparação é encontrar a distância de edição [26], que é o número mínimo de operações de inserções, remoções e substituições necessárias que transformam uma sequência em outra.

A distância de edição é uma medida capaz de estimar a distância evolutiva entre duas cadeias, mas não possui a informação de quais operações globais foram utilizadas para a transformação de uma sequência em outra. Estas operações globais são os chamados Rearranjos de Genomas, que podem ser, por exemplo, reversões, transposições, fissões e fusões.

Um conceito de distância pode ser definido para qualquer classe de rearranjo como sendo o menor número de operações pertencentes a essa classe que são necessários para transformar um genoma em outro. Por exemplo, chama-se a distância de reversão o menor número de reversões necessárias para transformar um genoma em outro [6] e a distância de transposição é o menor número de transposições [7].

Estudos mostram que os rearranjos de genomas são mais apropriados que mutações pontuais quando se deseja comparar os genoma de duas espécies [25]. Nesse contexto, a distância evolutiva entre dois genomas pode ser estimada pelo conceito de distância para

uma classe de rearranjo definido no parágrafo anterior.

Neste trabalho trataremos os casos em que os eventos de transposição e reversão ocorrem de forma isolada e os casos quando os dois eventos ocorrem ao mesmo tempo.

O evento de transposição ocorre quando dois blocos adjacentes no genoma trocam de posição. O problema da distância de transposição é encontrar o número mínimo de transposições necessárias que transforma um genoma em outro. Este problema pertence à classe dos problemas NP-Difíceis e a prova foi apresentada por Bulteau, Fertin e Rusu [10]. O melhor algoritmo de aproximação conhecido possui razão de 1.375 apresentado por Elias e Hartman [15].

O evento de reversão ocorre quando um bloco do genoma é invertido. O problema da distância de reversão é encontrar o número mínimo de reversões necessárias que transforma um genoma em outro. Neste problema é importante saber se a orientação dos genes é conhecida, pois existem algoritmos polinomiais para este caso. Entretanto, se a orientação dos genes não é conhecida o problema da distância de reversão pertence à classe de problemas NP-Difíceis, com a prova apresentada por Caprara [11], e o melhor algoritmo de aproximação conhecido possui razão de 1.375 apresentado por Berman, Hannenhalli e Karpinski [9].

Na natureza um genoma não sofre apenas eventos de reversão ou de transposição, ele está exposto a diversos eventos diferentes. Para esta situação, iremos estudar o caso onde os eventos de reversão e transposição ocorrem simultaneamente sobre um genoma. Os trabalhos de Walter, Dias e Meidanis [24, 28] e Lin e Xue [20] estudaram o problema de encontrar o número mínimo de reversões e transposições necessárias para transformar um genoma em outro.

O trabalho desenvolvido nesta dissertação segue a linha de pesquisa utilizada por Dias e Dias [13] e nós apresentaremos modelos de Programação por Restrições (CP[1]) para ordenação por reversões e ordenação por reversões e transposições, baseados na teoria do Problema de Satisfação de Restrições (CSP[2]) e na teoria do Problema de Otimização com Restrições (COP[3]). Nós fizemos comparações com os modelos de Programação por Restrições para ordenação por transposições, descrito por Dias e Dias [13], ordenação por reversões e ordenação por reversões e transposições com os modelos de Programação Linear Inteira (ILP[4]) para os mesmos problemas descritos em Dias e Souza [14].

O texto da dissertação está dividido da seguinte maneira. O Capítulo 2 apresenta uma série de conceitos básicos necessários para o entendimento deste trabalho. O Capítulo 3 descreve os modelos usados neste trabalho. O Capítulo 4 traz a análise dos resultados obtidos durante o trabalho. O Capítulo 5 apresenta as conclusões da dissertação.

[1]Do inglês *Constraint Programming*.
[2]Do inglês *Constraint Satisfaction Problems*.
[3]Do inglês *Constraint Optimization Problems*.
[4]Do inglês *Integer Linear Programming*.

Capítulo 2

Conceitos Básicos

Neste capítulo faremos uma apresentação dos conceitos básicos necessários para o entendimento e desenvolvimento deste trabalho. Na seção 2.1 mostraremos as definições usadas no decorrer deste trabalho. As seções 2.2, 2.3 e 2.4 descrevem, respectivamente, os problemas de ordenação por reversões, ordenação por transposições e ordenação por reversões e transposições.

2.1 Definições

Para todos os problemas usamos as seguintes definições.

Permutação. Para fins computacionais, um genoma é representado por uma n-tupla de genes, e quando não há genes repetidos essa n-tupla é chamada de permutação. Uma permutação é representada como $\pi = (\pi_1 \ \pi_2 \ \ldots \ \pi_n)$, para $\pi_i \in \mathbb{N}$, $0 < \pi_i \leq n$ e $i \neq j \leftrightarrow \pi_i \neq \pi_j$. A permutação identidade é representada como $\iota = (1\ 2\ 3 \ \ldots \ n)$. Para a demonstração dos eventos, usaremos como base a permutação $\pi = (4\ 7\ 3\ 6\ 2\ 5\ 1)$.

Eventos de rearranjo. Os eventos de rearranjo tratados neste trabalho são os eventos de transposição e reversão quando ocorrem isoladamente e quando ocorrem de forma conjunta. Os eventos são representados por ρ e são aplicados a π de uma maneira específica.

2.2 Ordenação por Reversões

Um evento de reversão ocorre quando um bloco do genoma é invertido. Uma reversão $\rho(i, j)$, para $1 \leq i < j \leq n$, aplicada ao genoma $\pi = (\pi_1 \ \pi_2 \ \ldots \ \pi_n)$ gera a permutação $\rho\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_j \ \pi_{j-1} \ \ldots \ \pi_{i+1} \ \pi_i \ \pi_{j+1} \ \ldots \ \pi_n)$, caso a orientação de π não é conhecida (Figura 2.1), e $\rho\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ -\pi_j \ -\pi_{j-1} \ \ldots \ -\pi_{i+1} \ -\pi_i \ \pi_{j+1} \ \ldots \ \pi_n)$, caso a orientação de π é conhecida (Figura 2.2).

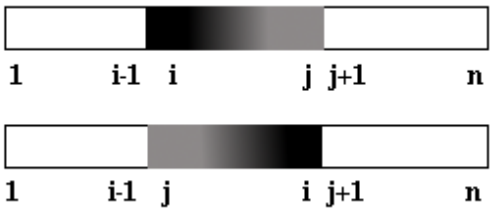

Figura 2.1: Reversão em uma permutação não orientada.

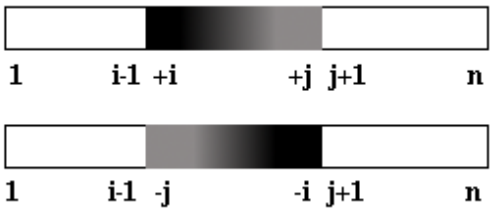

Figura 2.2: Reversão em uma permutação orientada.

A distância de reversão $d_r(\pi, \sigma)$ entre duas permutações π e σ é o número mínimo r de reversões ρ_1, ρ_2, ... , ρ_r tal que $\pi\rho_1\rho_2 \ \ldots \ \rho_r = \sigma$. Note que a distância de reversão entre π e σ é igual à distância de reversão entre $\sigma^{-1}\pi$ e ι. Então, sem perda de generalidade, podemos dizer que o problema da distância de reversão é equivalente ao problema de ordenação por reversões, que é a distância de reversão entre a permutação π e a permutação identidade ι, denotado por $d_r(\pi)$.

Em um estudo inicial sobre este problema, Bafna e Pevzner [6] apresentaram um algoritmo de aproximação com razão de 1.5 quando a orientação de genes é conhecida e 1.75 caso contrário.

Conhecer a orientação dos genes em um genoma é um fator importante no problema de reversão, pois existem algoritmos polinomiais para o caso em que a orientação é conhecida. No caso em que não se conhece a orientação dos genes o problema de encontrar a distância de reversão pertence à classe dos problemas NP-Difíceis [11].

O primeiro algoritmo polinomial para o problema de reversão com orientação conhecida foi criado por Hannenhalli e Pevzner [18] que fez uso de várias operações aplicadas a uma estrutura intermediária conhecida como grafo de *breakpoints*. A estratégia usada por Hannenhalli e Pevzner foi simplificada no trabalho de Bergeron [8]. Atualmente já existe um algoritmo com complexidade sub quadrática [27] e, quando apenas a distância é necessária, um algoritmo linear pode ser usado [5].

Um resultado importante obtido por Meidanis, Walter e Dias [23], mostrou que toda teoria sobre reversões desenvolvida para genomas lineares pode ser adaptada facilmente

para genomas circulares, que são comuns em seres inferiores como vírus e bactérias.

Quando a orientação dos genes não é conhecida existem algoritmos de aproximação que seguiram a ideia do trabalho de Bafna e Pevzner citado anteriormente como, por exemplo, o algoritmo implementado por Berman, Hannenhalli e Karpinski [9] com razão de aproximação de 1.375.

2.2.1 Ferramentas para o Problema de Ordenação por Reversões.

O conceito de grafo de *breakpoints* foi introduzido no trabalho de Bafna e Pevzner [6]. Inicialmente a permutação π é estendida adicionando o elemento $\pi_0 = 0$ e $\pi_{n+1} = n + 1$. Dois elementos consecutivos π_i e π_{i+1}, $0 \leq i \leq n$, são *adjacentes* quando $|\pi_i - \pi_{i+1}| = 1$, e são *breakpoints* caso contrário. Define-se um grafo de arestas coloridas $G(\pi)$ com $n + 2$ vértices $\{0, 1, \ldots, n, n + 1\}$. Unimos os vértices i e j com uma aresta preta se (i, j) for um *breakpoint*. Unimos os vértices i e j com uma aresta cinza se $|i - j| = 1$ e i, j não são consecutivos em π. Denotamos por $b_r(\pi)$ o número de *breakpoints* existentes em π. A Figura 2.3 mostra o grafo de *breakpoints* da permutação $\pi =$ (4 7 3 6 2 5 1).

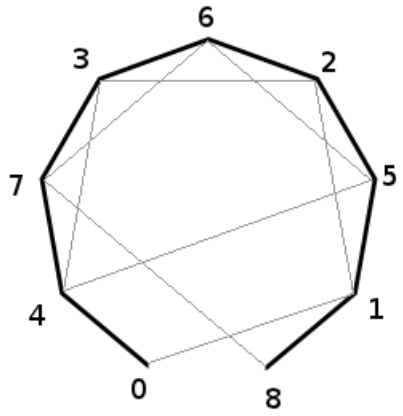

Figura 2.3: Grafo de *breakpoints* da permutação $\pi =$ (4 7 3 6 2 5 1).

Usando o conceito de *breakpoints*, temos que uma reversão atua em dois pontos em uma permutação e, portanto, pode reduzir o número de *breakpoints* em pelo menos um e no máximo dois [6], levando ao Teorema 2.1.

Teorema 2.1 *Para qualquer permutação π,*

$$\frac{1}{2} b_r(\pi) \leq d_r(\pi) \leq b_r(\pi).$$

Um ciclo em $G(\pi)$ é chamado de *alternado* se as cores de duas arestas consecutivas são diferentes ao longo do ciclo. Assim dizemos que todos os ciclos pertencentes ao grafo serão ciclos alternados. O *comprimento* de um ciclo é a sua quantidade de arestas pretas. Um k-ciclo é um ciclo que contem k arestas negras. Um *ciclo longo* é um ciclo de comprimento maior que dois.

Observe que $G(\pi)$ pode ser decomposto em ciclos de arestas disjuntas, pois cada vértice tem o mesmo número de arestas incidentes cinzas e pretas. Logo existem diversas maneiras de realizar a decomposição de ciclos em $G(\pi)$. A Figura 2.4 mostra um exemplo de decomposição em ciclos para o grafo de *breakpoints* da permutação $\pi = (\ 4\ 7\ 3\ 6\ 2\ 5\ 1\ )$.

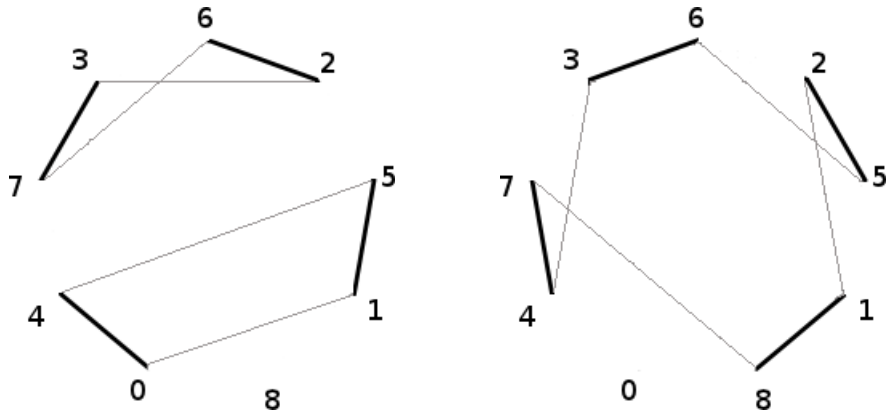

Figura 2.4: Exemplo de decomposição em ciclos de arestas disjuntas para o grafo de *breakpoints* da permutação $\pi = (\ 4\ 7\ 3\ 6\ 2\ 5\ 1\ )$.

Uma reversão atua em duas arestas pretas de $G(\pi)$, se estas arestas representam os *breakpoints* que são separados pela operação de reversão [12]. O Teorema 2.2, demonstrado no trabalho de Christie [12], fornece os limitantes para a distância de reversão usando a quantidade de 2-ciclos na máxima decomposição em ciclos de $G(\pi)$.

Teorema 2.2 *Se $c_2(\pi)$ é o número mínimo de 2-ciclos em qualquer máxima decomposição em ciclos de $G(\pi)$ então:*

$$\frac{2}{3}b_r(\pi) - \frac{1}{3}c_2(\pi) \leq d_r(\pi) \leq b_r(\pi) - \frac{1}{2}c_2(\pi).$$

2.3 Ordenação por Transposições

Um evento de transposição ocorre quando dois blocos adjacentes no genoma trocam de posição. Uma transposição $\rho(i, j, k)$, para $1 \leq i < j < k \leq n + 1$, aplicada ao genoma $\pi = (\ \pi_1\ \pi_2\ \ldots\ \pi_n\ )$ gera a permutação $\rho\pi = (\pi_1\ \ldots\ \pi_{i-1}\ \pi_j\ \ldots\ \pi_{k-1}\ \pi_i\ \ldots\ \pi_{j-1}\ \pi_k\ \ldots\ \pi_n)$ (Figura 2.5).

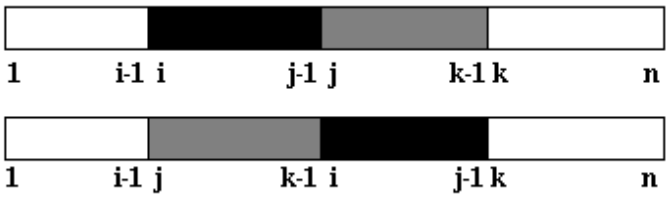

Figura 2.5: Transposição aplicada em uma permutação.

A distância de transposição $d_t(\pi, \sigma)$ entre duas permutações π e σ é o número mínimo t de transposições $\rho_1, \rho_2, \ldots, \rho_t$ tal que $\pi\rho_1\rho_2 \ldots \rho_t = \sigma$. Note que a distância de transposição entre π e σ é igual à distância de transposição entre $\sigma^{-1}\pi$ e ι. Então, sem perda de generalidade, podemos dizer que o problema da distância de transposição é equivalente ao problema de ordenação por transposições, que é a distância de transposição entre a permutação π e a permutação identidade ι, denotado por $d_t(\pi)$.

Este problema foi estudado por Bafna e Pevzner [7], onde apresentaram um algoritmo capaz de fornecer uma resposta aproximada na razão de 1.5, além de derivar um importante limitante inferior para o problema. Introduziram o conceito de *breakpoints* em eventos de transposições, elementos adjacentes em um genoma, mas não no outro, e o conceito de grafo de ciclos, ambos ferramentas importantes utilizadas para encontrar limitantes para o problema. Foram apresentados várias questões em aberto como verificar a complexidade do problema da distância de transposição e o diâmetro, que é a maior distância possível entre duas permutações de tamanho n. O problema do diâmetro foi estudado por Meidanis, Walter e Dias [22].

A complexidade deste problema ficou em aberto por um longo tempo. O trabalho de Bulteau, Fertin e Rusu [10] apresentou a prova de que o problema de ordenação por transposição pertence a classe dos problemas NP-Difíceis. Elias e Hartman [15] apresentaram um algoritmo de aproximação na razão de 1.375. O trabalho de Labarre [19] apresentou novos limitantes, além de definir classes de permutações em que a distância de transposição pode ser calculada em tempo e espaço lineares.

2.3.1 Ferramentas para o Problema de Ordenação por Transposições.

No problema de ordenação por transposições, um *breakpoints* é um par (π_i, π_{i+1}) tal que $\pi_{i+1} \neq \pi_i + 1$. Denota-se por $b_t(\pi)$ como sendo o número de *breakpoints* na permutação π. Sabemos que uma transposição atua em três pontos de uma permutação, logo, pode reduzir o número de *breakpoints* em pelo menos um e no máximo três [7], levando ao

Teorema 2.3.

Teorema 2.3 *Para qualquer permutação π,*

$$\frac{1}{3}b_t(\pi) \leq d_t(\pi) \leq b_t(\pi).$$

O conceito de grafo de ciclos foi introduzido por Bafna e Pevzner [7] e foi usado para obter limitantes melhores para o problema. Um grafo direcionado com arestas coloridas, denotado por $G(\pi)$, é chamado de grafo de ciclos da permutação π se possui um conjunto de vértices $\{0,\ 1,\ \ldots\ ,\ n+1\}$ e seu conjunto de arestas é definido como para todo $1 \leq i \leq n+1$, arestas cinzas são direcionadas de $i-1$ para i e arestas pretas de π_i para π_{i-1}. A Figura 2.6 mostra o grafo de ciclos para a permutação $\pi = (\ 4\ 7\ 3\ 6\ 2\ 5\ 1\ )$.

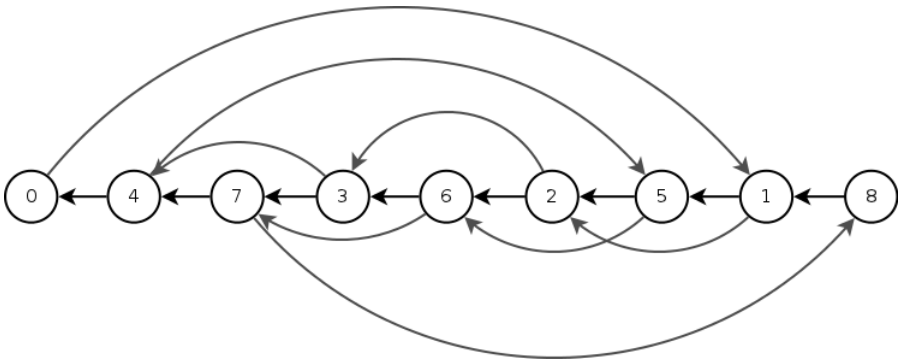

Figura 2.6: Grafo de ciclos para a permutação $\pi = (\ 4\ 7\ 3\ 6\ 2\ 5\ 1\ )$.

Similar ao problema de ordenação por reversões, um ciclo de $G(\pi)$ é chamado de *alternado* se ele for um ciclo direcionado com arestas de cores alternadas. Para todo vértice de $G(\pi)$ toda aresta chegando é unicamente pareada com uma aresta saindo de cor diferente. Isto implica que existe uma decomposição única de ciclos alternados do conjunto de arestas de $G(\pi)$. A seguir o termo ciclo é usado no lugar de ciclos alternados e usamos o termo k-ciclo para definir um ciclo alternado de tamanho $2k$, k-ciclo é longo se $k > 2$, e curto caso contrário. A Figura 2.7 mostra um exemplo de decomposição em ciclos para o grafo de ciclos da permutação $\pi = (\ 4\ 7\ 3\ 6\ 2\ 5\ 1\ )$.

Para melhorar o limitante, Bafna e Pevzner [7] estudou separadamente os ciclos pares e ímpares. Um ciclo é impar se número ímpar de arestas pretas e par caso contrário. Seja $c_{mpar}(\pi)$ o número de ciclos ímpares de $G(\pi)$, para uma permutação π, e $\Delta c_{mpar}(\rho) = c_{mpar}(\pi\rho) - c_{mpar}(\pi)$ a mudança no número de ciclos ímpares devido a transposição ρ, temos que $\Delta c_{mpar} \in \{2, 0, -2\}$, gerando o resultado do Teorema 2.4.

Teorema 2.4 *Para qualquer permutação π,*

$$\frac{1}{2}n + 1 - c_{mpar}(\pi) \leq d_t(\pi) \leq \frac{3}{4}(n + 1 - c_{mpar}(\pi)).$$

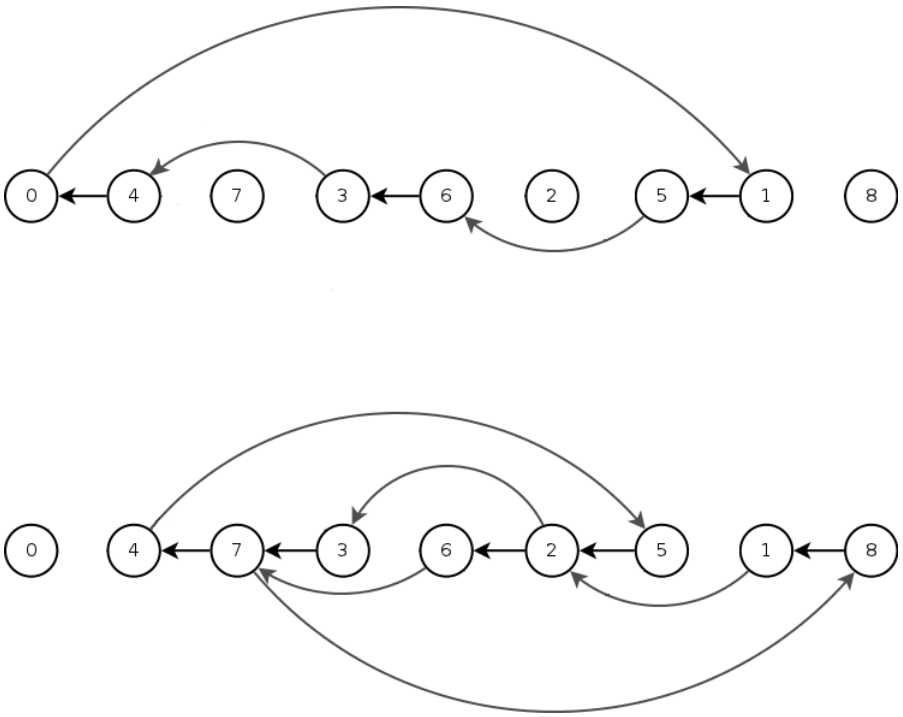

Figura 2.7: Exemplo de decomposição em ciclos de arestas disjuntas para o grafo de ciclos da permutação $\pi = (\ 4\ 7\ 3\ 6\ 2\ 5\ 1\ )$.

2.4 Ordenação por Reversões e Transposições

Na natureza um genoma não sofre apenas eventos de reversão ou de transposição, ele está exposto a diversos eventos diferentes. Para esta situação, iremos estudar o caso onde os eventos de reversão e transposição ocorrem simultaneamente sobre um genoma.

A distância de reversão e transposição $d_{rt}(\pi, \sigma)$ entre duas permutações π e σ é o número mínimo rt de reversões e transposições $\rho_1, \rho_2, \ldots, \rho_{rt}$ tal que $\pi\rho_1\rho_2 \ldots \rho_{rt} = \sigma$. Como no caso em que os eventos ocorrem individualmente, podemos dizer, sem perda de generalidade, que o problema da distância de reversão e transposição é equivalente ao problema de ordenação por reversões e transposições, que é a distância de reversão e transposição entre a permutação π e a permutação identidade ι, denotado por $d_{rt}(\pi)$.

Este problema foi estudado por Hannenhalli e co-autores [17], onde analisaram a evolução de genomas por diferentes tipos de eventos, em especial reversões e transposições.

Em 1998, Walter, Dias e Meidanis [28] apresentaram um algoritmo de aproximação para a distância de reversão e transposição, além de limitantes para o diâmetro de reversão e transposição em permutações orientadas que foram posteriormente melhorados [24].

No trabalho de Gu, Peng e Sudborough [16] é apresentado um algoritmo 2-aproximado para computar a distância entre dois genomas com a orientação dos genes conhecida

usando a operação de reversão e transposição simultaneamente.

Capítulo 3

Modelos

Neste capítulo nós apresentaremos a descrição dos modelos de programação por restrições e programação linear inteira usados para os problemas de ordenação por transposições, ordenação por reversões e ordenação por reversões e transposições.

3.1 Programação por Restrições

O modelo de programação por restrições usado para o problema de ordenação por transposições é o descrito em Dias e Dias [13]. Nós usamos as definições contidas no capítulo 2 para escrever as formulações baseadas nas teorias de Problema de Satisfação de Restrições (CSP) e Problema de Otimização com Restrições (COP). As formulações foram descritas usando a notação *prolog-like* de Marriot [21]. Primeiramente iremos apresentar os predicados que são comum às duas formulações.

Em prolog as variáveis são descritas por *strings* iniciadas com letra maiúscula ou "_" (*underscore*) caso a variável seja anônima. As letras gregas π e σ representam listas nesta notação. A construção $X :: [i \,..\, j]$ significa que X (ou cada elemento de X se X for uma lista) pode assumir um valor do intervalo $[i \,..\, j]$.

A representação da permutação (3.1) e o efeito das operações de reversão (3.2) e transposição (3.3 podem ser vistas da mesma maneira que são descritas pelos problemas. Neste modelo a permutação π é uma lista de elementos $(\pi_1,\ \pi_2,\ \ldots\ ,\ \pi_n)$ onde $\pi_i \in \mathbb{N}$, $0 < \pi_i \leq n$ e $\pi_i \neq \pi_j$ para $i \neq j$.

$$
\begin{aligned}
&\mathit{permutation}(\pi, N)\ \text{:-}\\
&\quad \mathit{length}(\pi, N),\\
&\quad \pi\ ::\ [1\ ..\ N],\\
&\quad \mathit{all\_different}(\pi).
\end{aligned}
\tag{3.1}
$$

Na reversão $\rho(i, j)$, $0 < i < j \leq n$, dividimos a lista em três sublistas $C_1C_2C_3$ onde $C_1 = (\pi_1 \,..\, \pi_{i-1})$, $C_2 = (\pi_i \,..\, \pi_j)$ e $C_3 = (\pi_{j+1} \,..\, \pi_n)$. Depois fazemos a reversão na sublista C_2, resultando na lista R_{C_2}. Então juntamos a nova lista R_{C_2} com as sublistas C_1 e C_3 para formar $\rho\pi = C_1R_{C_2}C_3$.

$$\begin{aligned} &\mathit{reversal}(\pi, \sigma, I, J) \text{ :-} \\ &\quad \mathit{permutation}(\pi, N), \\ &\quad \mathit{permutation}(\sigma, N), \\ &\quad 1 \leq I < J \leq N, \\ &\quad \mathit{split}(\pi, I, J, C_1, C_2, C_3), \\ &\quad \mathit{reverse}(C_2, R_{C_2}), \\ &\quad \sigma = C_1, R_{C_2}, C_3. \end{aligned} \tag{3.2}$$

Na transposição $\rho(i, j, k)$, $0 < i < j < k \leq n$, dividimos a lista em quatro sublistas $C_1C_2C_3C_4$ onde $C_1 = (\pi_1 \,..\, \pi_{i-1})$, $C_2 = (\pi_i \,..\, \pi_{j-1})$, $C_3 = (\pi_j \,..\, \pi_{k-1})$ e $C_4 = (\pi_k \,..\, \pi_n)$. Trocamos de posição os blocos C_2 e C_3 e juntamos elas na ordem C_1, C_3, C_2 e C_4 para formar $\rho\pi = C_1C_3C_2C_4$. Observe que as sublistas C_1 e C_4 podem ser vazias.

$$\begin{aligned} &\mathit{transposition}(\pi, \sigma, I, J, K) \text{ :-} \\ &\quad \mathit{permutation}(\pi, N), \\ &\quad \mathit{permutation}(\sigma, N), \\ &\quad 1 \leq I < J < K \leq N, \\ &\quad \mathit{split}(\pi, I, J, K, C_1, C_2, C_3, C_4), \\ &\quad \sigma = C_1, C_3, C_2, C_4. \end{aligned} \tag{3.3}$$

3.1.1 Modelo CSP

Primeiramente modelaremos o problema usando a teoria CSP, mas o número de variáveis é desconhecido devido ao fato de precisarmos do valor da distância de reversão $d_r(\pi)$ para criar as restrições e variáveis que representam as permutações. Por esta razão, nós escolhemos um valor candidato para a distância R tal que $R \in [LB \,..\, UB]$, onde LB é um limitante inferior conhecido e UB é um limitante superior conhecido para o problema, e tentamos achar uma combinação apropriada de R reversões que solucionam o problema. Se o modelo CSP falha (não existe combinação que soluciona o problema com o valor candidato escolhido) com o candidato R, nós escolhemos outro valor R apenas incrementando seu valor. O valor de R é escolhido usando uma estratégia *bottom-up*, ou seja, a verificação inicia pelo valor do limitante inferior LB e termina quando o valor é maior que o limitante superior UB. Na transposição, o processo é o mesmo que na

reversão, trocando apenas o valor da distância de reversão ($d_r(\pi)$) para o valor da distância de transposição ($d_t(\pi)$).

$$
\begin{aligned}
&\mathit{reversal\_distance}(\iota, 0, \_Model).\\
&\mathit{reversal\_distance}(\pi, R, Model)\ \text{:-}\\
&\quad \mathit{bound}(\pi, Model, LB, UB),\\
&\quad R :: [LB\ ..\ UB],\\
&\quad \mathit{indomain}(R),\\
&\quad \mathit{reversal}(\pi, \sigma, \_I, \_J),\\
&\quad \mathit{reversal\_distance}(\sigma, R-1, Model).
\end{aligned}
\tag{3.4}
$$

$$
\begin{aligned}
&\mathit{transposition\_distance}(\iota, 0, \_Model).\\
&\mathit{transposition\_distance}(\pi, T, Model)\ \text{:-}\\
&\quad \mathit{bound}(\pi, Model, LB, UB),\\
&\quad T :: [LB\ ..\ UB],\\
&\quad \mathit{indomain}(T),\\
&\quad \mathit{transposition}(\pi, \sigma, \_I, \_J, \_K),\\
&\quad \mathit{transposition\_distance}(\sigma, T-1, Model).
\end{aligned}
\tag{3.5}
$$

O predicado *rev_trans_dist/3* (3.6) retorna o valor da distância de reversão e transposição. O predicado *event/2* escolhe o melhor evento entre o predicado *reversal/4* (3.2) e o predicado *transposition/5* (3.3) para minimizar o valor da distância.

$$
\begin{aligned}
&\mathit{rev\_trans\_dist}(\iota, 0, \_Model).\\
&\mathit{rev\_trans\_dist}(\pi, N, Model)\ \text{:-}\\
&\quad \mathit{bound}(\pi, Model, LB, UB),\\
&\quad N :: [LB\ ..\ UB],\\
&\quad \mathit{indomain}(N),\\
&\quad \mathit{event}(\pi, \sigma),\\
&\quad \mathit{rev\_trans\_dist}(\sigma, N-1, Model).
\end{aligned}
\tag{3.6}
$$

O predicado *indomain(X)* em (3.4), (3.5) e (3.6) pega o domínio da variável X e escolhe o menor elemento dele (no caso o valor do limitante inferior). Se o modelo retorna para o predicado *indomain* devido a uma falha, o elemento que originou ela será removido do domínio e um outro valor será escolhido.

Os modelos CSP para os problemas de ordenação possuem a estrutura mostrada acima, trocando apenas os limitantes usados. A seguir apresentaremos os predicados que lidam com a escolha do modelo a ser usado.

Em comum para os todos eventos descritos temos o modelo *def_csp* que não usa nenhum limitante.

$$\begin{aligned} &bound(\pi, def\_csp, LB, UB) \text{ :-} \\ &\quad LB = 0, \\ &\quad length(\pi, UB). \end{aligned} \tag{3.7}$$

Ordenação por reversões:

- *rev_br_csp*: Modelo que usa o conceito de *breakpoints* em reversões para calcular os limitantes descritos no Teorema 2.1.
- *rev_cg_csp*: Modelo que usa o número de 2-ciclos na máxima decomposição em ciclos de $G(\pi)$ para calcular os limitantes descritos no Teorema 2.2.

$$\begin{aligned} &bound(\pi, rev\_br\_csp, LB, UB) \text{ :-} \\ &\quad calc\_breakpoints\_reversal(\pi, B), \\ &\quad LB = B/2 \\ &\quad UB = B. \\ &bound(\pi, rev\_cg\_csp, LB, UB) \text{ :-} \\ &\quad find\_2\_cycle(\pi, B, C), \\ &\quad LB = (2 * B - C)/3, \\ &\quad UB = B - C/2. \end{aligned} \tag{3.8}$$

Ordenação por transposições:

- *tra_br_csp*: Modelo que usa o conceito de *breakpoints* em transposições para calcular os limitantes conforme descrito no Teorema 2.3.
- *tra_cg_csp*: Modelo que usa o conceito de grafo de ciclos em transposições, fazendo a decomposição de ciclos e analisando os ciclos ímpares separadamente para calcular os limitantes conforme descrito no Teorema 2.4.

$$
\begin{aligned}
&bound(\pi, tra\_br\_csp, LB, UB) \text{ :-} \\
&\quad calc\_breakpoints\_transposition(\pi, B), \\
&\quad LB = B/3 \\
&\quad UB = B. \\
&bound(\pi, tra\_cg\_csp, LB, UB) \text{ :-} \\
&\quad calc\_oddcycle\_transposition(\pi, N, C), \\
&\quad LB = (N + 1 - C)/2 \\
&\quad UB = (3 * (N + 1 - C))/4.
\end{aligned}
\tag{3.9}
$$

Ordenação por reversões e transposições:

- *r_t_br_csp*: Melhor limitante superior entre o limitante de *breakpoints* para reversões e o limitante de *breakpoints* para transposições.
- *r_t_bc_csp*: Melhor limitante superior entre o limitante de *breakpoints* para reversões e o limitante do grafo de ciclos para transposições.
- *r_t_cc_csp*: Melhor limitante superior entre o limitante do grafo de ciclos para reversões e o limitante do grafo de ciclos para transposições.

$$
\begin{aligned}
&bound(\pi, r\_t\_br\_csp, 0, UB) \text{ :-} \\
&\quad bound(\pi, rev\_br, \_RLB, RUB), \\
&\quad bound(\pi, tra\_br, \_TLB, TUB), \\
&\quad min(RUB, TUB, UB). \\
&bound(\pi, r\_t\_bc\_csp, 0, UB) \text{ :-} \\
&\quad bound(\pi, rev\_br, \_RLB, RUB), \\
&\quad bound(\pi, tra\_cg, \_TLB, TUB), \\
&\quad min(RUB, TUB, UB). \\
&bound(\pi, r\_t\_cc\_csp, 0, UB) \text{ :-} \\
&\quad bound(\pi, rev\_cg, \_RLB, RUB), \\
&\quad bound(\pi, tra\_cg, \_TLB, TUB), \\
&\quad min(RUB, TUB, UB).
\end{aligned}
\tag{3.10}
$$

O predicado *bound/4* (3.7), (3.8), (3.9), (3.10) recebe na variável *Model* um átomo que representa o modelo a ser usado. Este átomo se conecta com o predicado que retorna o limitante superior e inferior apropriado para o modelo. Observe que o limitante inferior é igual a 0 no caso dos modelos de ordenação por reversões e transposições. Isto ocorre devido ao fato que a cada nova iteração do modelo pode surgir um limitante inferior melhor, simplesmente fazendo a troca entre as operações de reversão e transposição.

3.1.2 Modelo COP

Uma outra alternativa é modelar o problema usando a teoria COP. Os modelos que usam esta abordagem necessitam de um limitante superior, portanto serão feitas algumas alterações nos predicados definidos anteriormente. Nós usamos as variáveis binárias B para indicar quando uma operação de reversão ou de transposição modificou ou não a permutação fornecida.

O primeiro predicado que precisamos criar para o evento de reversão é o *reversal_ cop/5* (3.11). Primeiramente, dado uma reversão $\rho(i, j)$, adicionamos uma nova restrição para permitir $(i, j) = (0, 0)$. Se $(i, j) = (0, 0)$ então $\pi\rho = \pi$. Então, adicionamos um novo argumento ao predicado *reversal_ cop* que recebe a variável B.

$$\begin{aligned} &\mathit{reversal\_\ cop}(\iota, \iota, 0, 0, 0). \\ &\mathit{reversal\_\ cop}(\pi, \sigma, I, J, 1) \text{ :- } \mathit{reversal}(\pi, \sigma, I, J). \end{aligned} \tag{3.11}$$

O predicado equivalente para o evento de transposição é o *transposition_ cop/6* (3.12). Neste caso, dado uma transposição $\rho(i, j, k)$, adicionamos uma nova restrição para permitir $(i, j, k) = (0, 0, 0)$. Se $(i, j, k) = (0, 0, 0)$ então $\pi\rho = \pi$.

$$\begin{aligned} &\mathit{transposition\_\ cop}(\iota, \iota, 0, 0, 0, 0). \\ &\mathit{transposition\_\ cop}(\pi, \sigma, I, J, K, 1) \text{ :- } \mathit{transposition}(\pi, \sigma, I, J, K). \end{aligned} \tag{3.12}$$

Para calcular a distância de reversão nos modelos baseados na teoria COP, implementamos o predicado *reversal_ distance_ cop/3* (3.13), que ajusta as variáveis B usando o valor do limitante superior e restringe as permutações fazendo $\pi_k = \pi_{k-1}\rho_k$. O predicado *length/2*, predicado interno do prolog, é usado para criar uma lista de variáveis não instanciadas com o tamanho dado. A função de custo *Cost* é a soma das variáveis B associadas com cada ρ_k, $Cost = \sum_{k=1}^{UB} B_k$, onde UB é um limitante superior conhecido. A distância de reversão é o valor mínimo da função de custo $d_r = \min Cost$. Para evitar processamento desnecessários, o valor de $Cost$ precisa ser maior ou igual a qualquer limitante inferior. O predicado equivalente para o problema de ordenação por transposições é o *transposition_ distance_ cop/3* (3.14).

$$\begin{aligned} &\mathit{reversal\_\ distance\_\ cop}(\pi, R, Model) \text{ :-} \\ &\quad \mathit{bound}(\pi, Model, LB, UB), \\ &\quad \mathit{length}(B, UB), \\ &\quad \mathit{upperbound\_\ constraint\_\ rev}(\pi, B, Model, UB), \\ &\quad \mathit{sum}(B, Cost), \\ &\quad Cost \geq LB, \\ &\quad \mathit{minimize}(Cost, R). \end{aligned} \tag{3.13}$$

$$
\begin{aligned}
&\mathit{transposition\_distance\_cop}(\pi, T, \mathit{Model}) \text{ :-} \\
&\quad \mathit{bound}(\pi, \mathit{Model}, LB, UB), \\
&\quad \mathit{length}(B, UB), \\
&\quad \mathit{upperbound\_constraint\_trans}(\pi, B, \mathit{Model}, UB), \\
&\quad \mathit{sum}(B, \mathit{Cost}), \\
&\quad \mathit{Cost} \geq LB, \\
&\quad \mathit{minimize}(\mathit{Cost}, T).
\end{aligned}
\tag{3.14}
$$

O predicado equivalente para o modelo de ordenação por reversões e transposições é o *rev_trans_dist_cop/3* (3.15). O predicado *upperbound_constraint_event/4* escolhe o melhor evento entre a reversão, usando o predicado *upperbound_constraint_rev/4* (3.16), e a transposição, usando o predicado *upperbound_constraint_trans/4* (3.17), para minimizar o valor da distância.

$$
\begin{aligned}
&\mathit{rev\_trans\_dist\_cop}(\pi, N, \mathit{Model}) \text{ :-} \\
&\quad \mathit{bound}(\pi, \mathit{Model}, LB, UB), \\
&\quad \mathit{length}(B, UB), \\
&\quad \mathit{upperbound\_constraint\_event}(\pi, B, \mathit{Model}, UB), \\
&\quad \mathit{sum}(B, \mathit{Cost}), \\
&\quad \mathit{Cost} \geq LB, \\
&\quad \mathit{minimize}(\mathit{Cost}, N).
\end{aligned}
\tag{3.15}
$$

O predicado *upperbound_constraint_rev/4* (3.16) aplica na permutação os efeitos de ρ_k e retorna o valor apropriado de B para cada reversão ρ_k. Uma restrição importante é verificar se é possível ordenar a permutação usando o número restante de reversões para evitar processamento desnecessário. O predicado equivalente para o evento de transposição é o *upperbound_constraint_trans/4* (3.17).

$$
\begin{aligned}
&\mathit{upperbound\_constraint\_rev}(\iota, [\,], \_\mathit{Model}, \_UB). \\
&\mathit{upperbound\_constraint\_rev}(\pi, [B|Bt], \mathit{Model}, UB) \text{ :-} \\
&\quad \mathit{reversal\_cop}(\pi, \sigma, \_I, \_J, B), \\
&\quad \mathit{bound}(\pi, \mathit{Model}, LB, \_UB), \\
&\quad UB \geq LB, \\
&\quad \mathit{upperbound\_constraint\_rev}(\sigma, Bt, \mathit{Model}, UB - 1).
\end{aligned}
\tag{3.16}
$$

$$
\begin{aligned}
&upperbound\_constraint\_trans(\iota, [\,], \_Model, \_UB).\\
&upperbound\_constraint\_trans(\pi, [B|Bt], Model, UB) :-\\
&\quad transposition\_cop(\pi, \sigma, \_I, \_J, \_K, B),\\
&\quad bound(\pi, Model, LB, \_UB),\\
&\quad UB \geq LB,\\
&\quad upperbound\_constraint\_trans(\sigma, Bt, Model, UB - 1).
\end{aligned}
\tag{3.17}
$$

Os modelos baseados na teoria COP possuem a estrutura acima, trocando apenas os limitantes usados. Os limitantes são os mesmos usados para os modelos CSP, modificados para os modelos COP. Então temos os seguintes limitantes: *def_cop*, *rev_br_cop*, *rev_cg_cop*, *tra_br_cop*, *tra_cg_cop*, *r_t_br_cop*, *r_t_bc_cop* e *r_t_cc_cop*.

3.2 Programação Linear Inteira

A abordagem utilizada para programação linear inteira é a descrita no trabalho de Dias e de Souza [14]. Neste trabalho, define-se um modelo para o problema da distância com tamanho polinomial em relação ao tamanho da permutação fornecida como entrada. O modelo é específico para os eventos de reversão, transposição ou reversão e transposição quando ocorrem simultaneamente.

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

Gerando permutações válidas a cada iteração. As 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 $0 \leq k < n$.

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

As restrições (3.18) e (3.19) garantem que a permutação inicial e a final são corretas.

$$B_{i,\pi[i],0} = 1, \text{ para todo } 1 \leq i \leq n. \tag{3.18}$$

$$B_{i,\sigma[i],n-1} = 1, \text{ para todo } 1 \leq i \leq n. \tag{3.19}$$

A restrição (3.20) garante que cada posição de uma permutação possui exatamente um valor associado a ela. Já a restrição (3.21) garante que todo valor esteja associado a

uma posição de cada permutação.

$$\sum_{j=1}^{n} B_{ijk} = 1, \text{ para todo } 1 \leq i \leq n,\ 0 \leq k < n. \tag{3.20}$$

$$\sum_{i=1}^{n} B_{ijk} = 1, \text{ para todo } 1 \leq j \leq n,\ 0 \leq k < n. \tag{3.21}$$

Distância de reversão. Para o problema da distância de reversão definimos os seguintes conjuntos de variáveis e restrições. As variáveis binárias r_{abk} indicam quando a k-ésima operação de reversão afeta o blocos $\pi[a\ \ldots\ b]$ de π, para todo $1 \leq a < b \leq n$ e todo $1 \leq k < n$.

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

As variáveis binárias r_k são usadas para decidir se a k-ésima operação de reversão modificou a permutação, para todo $1 \leq k < n$.

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

As restrições (3.22) e (3.23) são necessárias para identificar as reversões que fazem parte da solução. A restrição (3.22) garante que se a k-ésima reversão não alterar a permutação, nenhuma das reversões seguintes poderá alterar. Já a restrição (3.23) garante que no máximo uma reversão poderá ser feita por iteração.

$$r_k \leq r_{k-1}, \text{ para todo } 1 \leq k < n. \tag{3.22}$$

$$\sum_{a=1}^{n-1} \sum_{b=a+1}^{n} r_{abk} \leq r_k, \text{ para todo } 1 \leq k < n. \tag{3.23}$$

As próximas restrições lidam com as modificações na permutação causada pela reversão a cada iteração 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.

1. $i < a$ ou $i > b$: A operação de reversão não modifica estas posições.

$$\sum_{a=i+1}^{n-1} \sum_{b=a+1}^{n} r_{abk} + \sum_{a=1}^{n-1} \sum_{b=a+1}^{i-1} r_{abk} + (1 - r_k) + B_{i,j,k-1} - B_{ijk} \leq 1, \quad \text{para todo } 1 \leq i,\ j \leq n \text{ e todo } 1 \leq k < n. \tag{3.24}$$

2. $a \leq i \leq b$: A operação de reversão altera os elementos guardados nestas posições. Para não ser redundante, a desigualdade precisa ter os dois primeiros termos com valor 1. Neste caso, $B_{b+a-i,j,k-1} = 1$, implicando que o elemento j foi salvo na posição $b + a - i$ antes da reversão ser movida para a posição i ($B_{ijk} = 1$).

$$\begin{aligned} r_{abk} + B_{b+a-i,j,k-1} - B_{ijk} \leq 1, \\ 1 \leq a < b \leq n,\ a \leq i \leq b,\ 1 \leq j \leq n,\ 1 \leq k < n. \end{aligned} \tag{3.25}$$

Distância de transposição. Para o problema da distância de transposição, usaremos os seguintes conjuntos de variáveis e restrições. As variáveis binárias t_{abck} indicam quando a k-ésima operação de transposição realiza a troca de lugares dos blocos $\pi[a\ \ldots\ b-1]$ e $\pi[b\ \ldots\ c-1]$ da permutação π, para todo $1 \leq a < b < c \leq n+1$ e todo $1 \leq k < n$.

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

As variáveis binárias t_k são usadas para decidir se a k-ésima operação de transposição modificou a permutação, para todo $1 \leq k < n$.

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

As restrições (3.26) e (3.27) são necessárias para identificar as transposições que fazem parte da solução. A restrição (3.26) garante que se a k-ésima transposição não alterar a permutação, nenhuma das transposições seguintes poderá alterar. Já a restrição (3.27) garante que no máximo uma transposição poderá ser feita por iteração.

$$t_k \leq t_{k-1}, \text{para todo } 1 \leq k < n. \tag{3.26}$$

$$\sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{n+1} t_{abck} \leq t_k, \text{para todo } 1 \leq k < n. \tag{3.27}$$

As próximas restrições refletem a modificações na permutação causada por uma 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.

1. $i < a$ ou $i \geq c$: A operação de transposição não altera estas posições.

$$\begin{aligned} \sum_{a=i+1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{n+1} t_{abck} + \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{i} t_{abck} + (1 - t_k) + B_{i,j,k-1} - B_{ijk} \leq 1, \\ \text{para todo } 1 \leq i,j \leq n \text{ e todo } 1 \leq k < n. \end{aligned} \tag{3.28}$$

2. $a \leq i < a + c - b$: Após a operação de transposição ser completada, estas posições serão ocupadas pelos elementos que estão nas posições de b a $c - 1$. Para não ser redundante, esta desigualdade precisa ter os dois primeiros termos com o valor 1. Neste caso, temos que $B_{b-a+i,j,k-1} = 1$, implicando que o elemento j foi salvo na posição $b - a + i$ antes da transposição ser movida para a posição i ($B_{ijk} = 1$).

$$\begin{aligned} t_{abck} + B_{b-a+i,j,k-1} - B_{ijk} \leq 1, \\ 1 \leq a < b < c \leq n+1, a \leq i < a + c - b, 1 \leq j \leq n, 1 \leq k < n. \end{aligned} \tag{3.29}$$

3. $a + c - b \leq i < c$: Após a operação de transposição ser completada, estas posições serão ocupadas pelos elementos que estão nas posições de a a $b - 1$. Similar com o caso acima, esta desigualdade é redundante se os valores dos dois primeiros termos não for igual a 1. Isto significa que a k-ésima transposição move $B^{k-1}[a \,..\, b-1]$ para as posições que precedem a posição c. Por definição, i representa uma das posições que receberão um elemento deste subvetor. Então, temos que $B^k[i] = B^{k-1}[b-c+i]$, para todo $i \in [a + c - b \,..\, c - 1]$ e os últimos dois termos se anulam.

$$\begin{aligned} t_{abck} + B_{b-c+i,j,k-1} - B_{ijk} \leq 1, \\ 1 \leq a < b < c \leq n+1, a + c - b \leq i < c, 1 \leq j \leq n, 1 \leq k < n. \end{aligned} \tag{3.30}$$

Distância de reversão e transposição. Para o problema da distância de reversão e transposição usaremos todas as variáveis definidas anteriormente, com a adição das variáveis binárias z_k, que é usada para indicar quando uma k-ésima operação, seja ela uma reversão ou uma transposição, realmente modificou a permutação. Então, para todo $1 \leq k < n$, temos que:

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

Usaremos todas as restrições definidas anteriormente, com exceção das restrições (3.22) e (3.26) que serão substituídas pelas restrições (3.31) e (3.32). A restrição (3.31) garante que se não ocorreu operações em uma iteração então não ocorrerá nenhuma operação nas iterações seguintes. A restrição (3.32) garante que no máximo uma operação é executada a cada iteração.

$$z_k \leq z_{k-1}, \text{para todo } 1 \leq k < n. \tag{3.31}$$

$$r_k + t_k = z_k, \text{para todo } 1 \leq k < n. \tag{3.32}$$

Precisamos modificar as restrições (3.24) e (3.28), substituindo r_k e t_k por z_k, resultando nas restrições (3.33) e (3.34).

$$\sum_{a=i+1}^{n-1}\sum_{b=a+1}^{n} r_{abk} + \sum_{a=1}^{n-1}\sum_{b=a+1}^{i-1} r_{abk} + (1 - z_k) + B_{i,j,k-1} - B_{ijk} \leq 1, \quad \text{para todo } 1 \leq i,\ j \leq n \text{ e todo } 1 \leq k < n. \tag{3.33}$$

$$\sum_{a=i+1}^{n-1}\sum_{b=a+1}^{n}\sum_{c=b+1}^{n+1} t_{abck} + \sum_{a=1}^{n-1}\sum_{b=a+1}^{n}\sum_{c=b+1}^{i} t_{abck} + (1 - z_k) + B_{i,j,k-1} - B_{ijk} \leq 1, \quad \text{para todo } 1 \leq i, j \leq n \text{ e todo } 1 \leq k < n. \tag{3.34}$$

3.2.1 Função Objetivo

Considerando as variáveis e restrições descritas anteriormente para cada um dos três problemas de distâncias, temos a função objetivo $\omega_r = \min \sum_{k=1}^{n-1} r_k$, para o problema da distância de reversão, a função objetivo $\omega_t = \min \sum_{k=1}^{n-1} t_k$, para o problema da distância de transposição, e a função objetivo $\omega_{rt} = \min \sum_{k=1}^{n-1} z_k$, para o problema da distância de reversão e transposição quando ocorrem simultaneamente.

3.2.2 Tamanho do modelo

É fácil observar que o modelo descrito possui tamanho polinomial em relação ao tamanho da permutação fornecida como entrada. A tabela 3.1 mostra o tamanho do modelo para os três problemas de distâncias com relação ao parâmetro n (tamanho da permutação de entrada).

Tabela 3.1: Tamanho dos modelos em relação à n.

Modelo	**Variáveis**	**Restrições**
Distância de Reversão	$O(n^3)$	$O(n^5)$
Distância de Transposição	$O(n^4)$	$O(n^6)$
Distância de Reversão e Transposição	$O(n^4)$	$O(n^6)$

Capítulo 4

Análise dos Resultados

Neste capítulo apresentaremos os resultados obtidos pelos modelos descritos no capítulo 3. A seção 4.1 mostra as características do computador utilizado para executar os testes. A seção 4.2 descreve como os testes foram executados. A seção 4.3 apresenta a análise dos resultados obtidos durante este trabalho.

4.1 Especificações Técnicas

O computador utilizado para executar os testes possui as seguintes características:

- Processador: Intel® Core™ 2 Duo 2.33GHz.
- Memória RAM: 3 GB.
- Sistema Operacional: Ubuntu Linux com kernel 2.6.31.

Todos modelos de programação por restrições foram implementados usando as seguintes ferramentas:

- Sistema de programação de código aberto *ECLiPSe-6.0* [1].
- Pacote proprietário para a linguagem de programação C++ *IBM® ILOG® CPLEX® CP Optimizer v 2.3* [3].

Todas formulações de programação linear inteira foram implementadas usando as seguintes ferramentas:

- Sistema de programação de código aberto *GLPK-4.35* [2].
- Pacote proprietário para a linguagem de programação C++ *IBM® ILOG® CPLEX® Optimizer v 12.1* [4]..

4.2 Descrição dos Testes

Os testes foram separados de acordo com o tamanho das permutações. Uma instância contém o conjunto de permutações com tamanho n, onde $n > 2$ devido ao fato de ser trivial fazer uma das operações de ordenação em uma permutação com tamanho 2. Para cada instância, geramos 50 permutações com tamanho n.

Todas instâncias foram executadas nos softwares indicados na seção 4.1. Para cada instância foi dado o tempo máximo de 25 horas. Fazemos a comparação dos modelos se baseando nos tempos médios usados para resolver cada instância. Como referência usamos os modelos de programação linear inteira descritos na seção 3.2.

4.3 Análise dos Resultados

Tabela 4.1: Tempo médio (em segundos) para o modelo de ordernação por reversões. O caractere "-" significa que o modelo não conseguiu terminar o conjunto de testes dentro do limite de 25 horas. O caractere "*" significa que o modelo não conseguir terminar devido ao limite de memória do sistema.

Modelos de Ordenação por Reversões														
size	**CP**												**ILP**	
	ECLiPSe						**ILOG CP**						**GLPK**	**ILOG CPLEX**
	CSP			**COP**			**CSP**			**COP**				
	def	**rev_br**	**rev_cg**	**def**	**rev_br**	**rev_cg**	**def**	**rev_br**	**rev_cg**	**def**	**rev_br**	**rev_cg**		
3	0.008	0.003	0.013	0.004	0.004	0.003	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
4	0.368	0.339	1.490	0.025	0.004	0.003	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.002
5	31.676	39.424	93.984	0.531	0.014	0.008	0.019	0.018	0.019	0.001	0.006	0.004	0.176	0.039
6	-	-	*	9.001	0.032	0.014	0.144	0.148	0.126	0.001	0.023	0.020	2.118	0.653
7	-	-	-	442.825	0.179	0.072	1.962	1.807	1.331	0.007	0.156	0.225	3.896	21.911
8	-	-	-	-	1.075	0.449	25.831	26.275	30.244	0.001	0.642	2.757	3.006	280.528
9	-	-	-	-	5.015	1.401	503.421	473.005	405.242	0.003	11.416	54.560	-	5.161
10	-	-	-	-	51.800	8.639	-	-	-	*	*	*	-	-
11	-	-	-	-	1664.691	194.266	-	-	-	-	-	-	-	-
12	-	-	-	-	-	671.979	-	-	-	-	-	-	-	-
13	-	-	-	-	-	*	-	-	-	-	-	-	-	-

Tabela 4.2: Tempo médio (em segundos) para o modelo de ordernação por transposições. O caractere "-" significa que o modelo não conseguiu terminar o conjunto de testes dentro do limite de 25 horas.

Modelos de Ordenação por Transposições														
size	**CP**												**ILP**	
	ECLiPSe						**ILOG CP**						**GLPK**	**ILOG CPLEX**
	CSP			**COP**			**CSP**			**COP**				
	def	**tra_br**	**tra_cg**	**def**	**tra_br**	**tra_cg**	**def**	**tra_br**	**tra_cg**	**def**	**tra_br**	**tra_cg**		
3	0.005	0.006	0.004	0.003	0.002	0.003	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
4	0.479	1.557	0.049	0.012	0.004	0.003	0.004	0.001	0.001	0.001	0.001	0.001	0.001	0.005
5	174.445	1275.804	2.922	0.318	0.020	0.005	0.023	0.009	0.004	0.001	0.008	0.008	0.376	0.054
6	-	-	33.311	15.957	0.185	0.009	0.321	0.117	0.072	0.001	0.035	0.041	3.948	2.308
7	-	-	-	394.574	0.519	0.015	1.747	1.313	0.321	0.005	0.112	0.173	5.558	66.497
8	-	-	-	-	7.254	0.069	29.714	45.806	27.883	0.002	0.651	5.895	-	9.506
9	-	-	-	-	99.152	0.365	641.954	838.988	320.615	0.004	2.285	53.673	-	-
10	-	-	-	-	-	5.162	-	-	-	-	-	-	-	-
11	-	-	-	-	-	33.122	-	-	-	-	-	-	-	-
12	-	-	-	-	-	43.893	-	-	-	-	-	-	-	-
13	-	-	-	-	-	521.840	-	-	-	-	-	-	-	-
14	-	-	-	-	-	1223.512	-	-	-	-	-	-	-	-

Tabela 4.3: Tempo médio (em segundos) para o modelo de ordenação por reversões e transposições. O caractere "-" significa que o modelo não conseguiu terminar o conjunto de testes dentro do limite de 25 horas. O caractere "*" significa que o modelo não conseguir terminar devido ao limite de memória do sistema.

Modelos de Ordenação por Reversões e Transposições																		
size	CP																ILP	
	ECLiPSe								ILOG CP								GLPK	ILOG CPLEX
	CSP				COP				CSP				COP					
	def	r_t_br	r_t_bc	r_t_cc	def	r_t_br	r_t_bc	r_t_cc	def	r_t_br	r_t_bc	r_t_cc	def	r_t_br	r_t_bc	r_t_cc		
3	0.035	0.010	0.004	0.006	0.004	0.005	0.004	0.008	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
4	6.434	10.333	0.248	0.687	0.022	0.026	0.035	0.081	0.002	0.002	0.002	0.003	0.001	0.001	0.001	0.001	0.008	0.008
5	-	-	30.824	64.170	0.379	0.414	0.680	2.066	0.019	0.020	0.022	0.023	0.001	0.001	0.001	0.001	0.466	0.059
6	-	-	519.803	-	11.566	12.776	21.264	89.760	0.340	0.327	0.319	0.319	0.001	0.001	0.001	0.001	4.642	1.374
7	-	-	-	-	400.904	441.946	792.422	*	2.052	2.062	2.066	2.088	0.004	0.004	0.002	0.004	5.094	82.241
8	-	-	-	-	-	-	-	-	11.894	11.727	12.367	12.369	0.002	0.004	0.004	0.004	-	82.220
9	-	-	-	-	-	-	-	-	310.012	304.331	331.275	328.956	0.006	0.006	0.004	0.005	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	0.012	0.010	0.010	0.011	-	-

Capítulo 5

Conclusões

TODO: Conclusão do trabalho

Referências Bibliográficas

[1] The ECLiPSe Constraint Programming System. `http://www.eclipseclp.org/`.

[2] Glpk (GNU Linear Programming Kit). `http://www.gnu.org/software/glpk/`.

[3] IBM® ILOG® CPLEX® CP Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/`.

[4] IBM® ILOG® CPLEX® Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[5] D. A. Bader, B. M. E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[6] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[7] V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, 1998.

[8] A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. *Discrete Applied Mathematics*, 146:134–145, March 2005.

[9] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002)*, pages 200–210, London, UK, UK, 2002. Springer-Verlag.

[10] L. Bulteau, G. Fertin, and I. Rusu. Sorting by transpositions is difficult. *Computing Research Repository*, abs/1011.1157, 2010.

[11] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the first annual international conference on Computational molecular biology (RECOMB'97)*, pages 75–83, New York, NY, USA, 1997. ACM.

[12] D. A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, pages 244–252, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[13] U. Dias and Z. Dias. Constraint programming models for transposition distance problem. *Lecture Notes on Bioinformatics*, 5676:13–23, 2009.

[14] Z. Dias and C. de Souza. Polynomial-size ILP models for rearrangements distance problems. In *Proceedings of the Brazilian Symposium on Bioinformatics (BSB'2007)*, pages 74–85, 2007.

[15] I. Elias and T. Hartman. A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology Bioinformatics*, 3(4):369–379, 2006.

[16] Q.-P. Gu, S. Peng, and H. Sudborough. A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theor. Comput. Sci.*, 210(2):327–339, January 1999.

[17] S. Hannenhalli, C. Chappey, E. V. Koonin, and P. A. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: a test case. *Genomics*, 30:299–311, 1995.

[18] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing (STOC'95)*, pages 178–189, New York, NY, USA, 1995. ACM.

[19] A. Labarre. New bounds and tractable instances for the transposition distance. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4):380–394, 2006.

[20] G.-H. Lin and G. Xue. Signed genome rearrangement by reversals and transpositions: Models and approximations. *Lecture Notes in Computer Science*, 1627:71–80, 1999.

[21] K. Marriott and P. J. Stuckey. *Programming with constraints: an introduction*. MIT Press, 1998.

[22] J. Meidanis, M. E. M. T. Walter, and Z. Dias. Transposition distance between a permutation and its reverse. In R. Baeza-Yates, editor, *Proceedings of the 4th South American Workshop on String Processing (WSP'97)*, pages 70–79, Valparaiso, Chile, 1997. Carleton Univertity Press.

[23] J. Meidanis, M. E. M. T. Walter, and Z. Dias. Reversal distance of signed circular chromosomes. Technical Report IC-00-23, Institute of Computing, University of Campinas, December 2000.

[24] J. Meidanis, M. E. M. T. Walter, and Z. Dias. A lower bound on the reversal and transposition diameter. *Journal of Computational Biology*, 9(5):743–746, 2002.

[25] J. D. Palmer and L. A. Herbon. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 27:87–97, 1988.

[26] J. C. Setubal and J. Meidanis. *Introduction to Computional Molecular Biology*. PWS Publishing Company, 1997.

[27] E. Tannier and M.-F Sagot. Sorting by reversals in subquadratic time. In *Proceedings of Combinatorial Pattern Matching (CPM'2004)*, pages 1–13, Istambul, Turkey, 2004.

[28] M. E. M. T. Walter, Z. Dias, and J. Meidanis. Reversal and transposition distance of linear chromosomes. In *Proceedings of the String Processing and Information Retrieval (SPIRE'98)*, 1998.