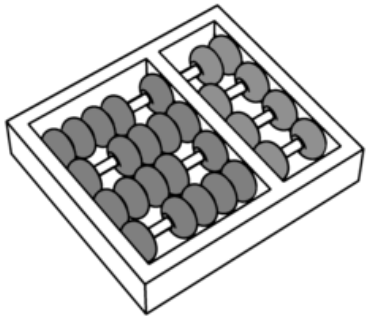
Universidade Estadual de Campinas

Instituto de Computação



Proposta de Dissertação de Mestrado

Problemas de Ordenação de Permutações por Operações Ponderadas pelo Número de Fragmentações

---

Candidato: Alexsandro Oliveira Alexandrino

Orientador: Prof. Dr. Zanoni Dias

Coorientadora: Dra. Carla Negri Lintzmayer

Resumo

Calcular a distância evolucionária entre espécies é um problema importante da área de Biologia Computacional, sendo que para isso consideramos conjuntos de mutações que alteram grandes trechos do genoma, os quais chamamos de rearranjos de genoma. Representa-se um genoma como uma permutação de inteiros, em que cada elemento corresponde a um bloco conservado (região de alta similaridade entre os genomas a serem comparados). Devido a propriedades algébricas das permutações, o problema de transformar um genoma em outro é equivalente ao da ordenação de permutações por operações de rearranjo. A abordagem mais comum considera que todos os rearranjos tem o mesmo custo, assim o objetivo é encontrar uma sequência mínima de rearranjos que ordenem a permutação. Porém, estudos indicam que algumas operações de rearranjo tem maior probabilidade de acontecer do que outras, fazendo com que abordagens em que operações possuem custos diferentes sejam mais realistas. Nessa abordagem ponderada, o objetivo é encontrar a sequência que ordena a permutação, tal que a soma dos custos dos rearranjos dessa sequência seja mínimo. Esta proposta apresenta uma nova versão para o problema da ordenação de permutações por operações ponderadas, em que a função de custo de uma operação corresponde à quantidade de fragmentações que a operação causa na permutação.

1 Introdução

A evolução biológica é o processo no qual as características hereditárias de organismos mudam ao longo de várias gerações, sendo que esse processo acontece por meio de variações genéticas e seleção natural. O genoma de uma espécie é formado por cromossomos, representados como um conjunto ordenado de genes. Um ***bloco conservado*** é uma região de alta similaridade entre dois genomas distintos, podendo ser um gene ou uma sequência de genes. Ao considerar a comparação de genomas, representa-se um genoma como uma sequência de blocos conservados [11]. Assumindo que não existem genes duplicados, essa representação é modelada matematicamente como uma permutação de inteiros. Genes e blocos possuem uma orientação e, quando essa orientação é conhecida, o genoma é representado como uma ***permutação com sinais***, em que o sinal indica a orientação do gene ou bloco. Quando a orientação não

é conhecida, o genoma é representado como uma ***permutação sem sinais***.

Partindo do ***Princípio da Máxima Parcimônia***, em que a explicação mais simples é a mais provável, uma métrica para a comparação de genomas de duas espécies é definida como a sequência mínima de rearranjos necessários para transformar o genoma de uma espécie no da outra, chamada de ***distância evolucionária***. Entende-se por ***rearranjo*** qualquer evento que altere grandes trechos do genoma. Devido a propriedades algébricas das permutações, esse problema é equivalente ao da ordenação de permutações por rearranjos.

Os rearranjos ou operações mais estudados na literatura são as reversões e as transposições. Uma reversão inverte um segmento qualquer da permutação, enquanto uma transposição faz a troca de dois segmentos adjacentes da permutação. Quando essas operações envolvem o primeiro elemento da permutação, elas são chamadas de operações de prefixo. De forma similar, ao envolver o último elemento da permutação elas são chamadas de operações de sufixo.

Uma operação pode fragmentar um genoma (permutação) em posições distintas. Por exemplo, uma reversão fragmenta um genoma em no máximo duas posições, enquanto uma transposição pode fragmentá-lo em até três posições. Reversões de prefixo e sufixo, por sempre envolverem início e fim, respectivamente, fragmentam uma posição a menos de um genoma. O mesmo acontece com as transposições de prefixo e sufixo.

Um ***modelo de rearranjo*** é um conjunto de rearranjos permitidos para o cálculo da distância evolucionária, podendo ser de um ou mais tipos, sendo que cada operação tem um custo correspondente. Normalmente, considera-se que todas operações têm o mesmo custo. Existem outras abordagens nas quais o custo é relacionado com a quantidade de elementos [4, 18, 19] ou ao tipo de operação [2, 13, 16].

Os problemas da Ordenação por Reversões e Ordenação por Transposições são NP-Difíceis [8, 9]. No entanto, em permutações com sinais a Ordenação por Reversões possui algoritmo polinomial, apresentado por Hannenhalli e Pevzner [14]. O problema da Ordenação por Reversões e Transposições ainda possui sua complexidade em aberto.

Todos os problemas citados anteriormente consideram que as operações tem custo unitário. No entanto, estudos apresentam que no decorrer do processo evolutivo

uma operação que envolve muitos segmentos do genoma tem maior probabilidade de debilitar o indivíduo [7]. Assim, uma abordagem com operações ponderadas é mais próxima do processo evolutivo real. Da mesma forma, pressupõe-se que uma operação com maior número de fragmentações também é mais provável de debilitar o indivíduo. Ainda não existem estudos específicos para a análise da frequência das operações e, consequentemente, é desconhecido quais valores de custo são os mais próximos à realidade [7].

Neste trabalho, estamos interessados em uma variação do problema na qual o custo de uma operação é relacionado com a quantidade de fragmentações que ela produz no genoma. Serão consideradas permutações com e sem sinais para os modelos com reversões, transposições, e combinação de reversões e transposições.

2 Rearranjo de Genomas

Um genoma é modelado matematicamente como um permutação com sinais $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, com $\pi_i \in \{-n, -(n-1), \ldots, -1, 1, 2, \ldots, n\}$ e $|\pi_i| \neq |\pi_j| \Leftrightarrow i \neq j$. Cada elemento representa um gene ou uma sequência de genes (bloco conservado), e o sinal de positivo ou negativo representa a orientação do bloco. No caso em que a orientação dos blocos não é conhecida, o genoma é modelado como uma permutação sem sinais $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, com $\pi_i \in \{1, 2, \ldots, n\}$ e $\pi_i \neq \pi_j \Leftrightarrow i \neq j$. Definimos a ***permutação estendida*** de π como a permutação obtida ao adicionar os elementos $\pi_0 = 0$ e $\pi_{n+1} = n + 1$ em π. Em uma permutação com sinais, π_0 e π_{n+1} tem sinais positivos.

A ***composição*** "$\cdot$" de duas permutações π e σ é a operação $\pi \cdot \sigma = (\pi_{\sigma_1}\ \pi_{\sigma_2}\ \ldots\ \pi_{\sigma_n})$. A ***permutação identidade*** ι é a permutação ordenada $(1\ 2\ \ldots\ n)$. Ao considerar permutações com sinais, todos os elementos de ι tem sinal positivo. A ***permutação reversa*** η é a permutação $(n\ \ldots\ 2\ 1)$, e a ***permutação reversa com sinais*** $\overline{\eta}$ é a permutação $(-n\ \ldots\ -2\ -1)$. A ***inversa*** de uma permutação π é π^{-1}, tal que $\pi^{-1}_{\pi_i} = i$, para $1 \leq i \leq n$, e satisfaz $\pi \cdot \pi^{-1} = \pi^{-1} \cdot \pi = \iota$.

Se β é uma operação qualquer, então a aplicação de β em π é denotada por $\pi \cdot \beta$. Para toda operação β, existe um custo associado à aplicação de β em π, e esse custo é denotado pela função $f : M \to \mathbb{R}$, onde M é o modelo de rearranjo considerado.

O custo de uma sequência de operações $\beta_1, \beta_2, \ldots, \beta_m$ é igual a $\sum_{i=1}^{m} f(\beta_i)$.

Dados um modelo de rearranjo M, uma função de custo f e as permutações π e σ, a ***distância*** $c_M^f(\pi, \sigma)$ é o custo de uma sequência de operações $\beta_1, \beta_2, \ldots, \beta_m$ pertencentes a M, tal que $\pi \cdot \beta_1 \cdot \beta_2 \cdot \ldots \cdot \beta_m = \sigma$ e $\sum_{i=1}^{m} f(\beta_i)$ é mínimo. Quando $f(\beta) = 1$ para todo $\beta \in M$, esse problema é equivalente ao de encontrar o número mínimo de operações que transformam π em σ.

Seja γ uma permutação qualquer. A ***distância de ordenação*** de γ é denotada por $c_M^f(\gamma)$, de modo que $c_M^f(\gamma) = c_M^f(\gamma, \iota)$. O problema de transformar π em σ é equivalente ao de ordenar $\gamma = \sigma^{-1} \cdot \pi$, já que $c_M^f(\pi, \sigma) = c_M^f(\sigma^{-1} \cdot \pi, \sigma^{-1} \cdot \sigma) = c_M^f(\gamma, \iota) = c_M^f(\gamma)$.

Dados um modelo de rearranjo M e uma função de custo f, o ***diâmetro de ordenação*** $D_M^f(n)$, com $n \in \mathbb{Z}^+$, é a maior distância de ordenação entre todas as permutações de tamanho n.

2.1 Reversões

Uma ***reversão*** $\rho(i, j)$, com $1 \leq i < j \leq n$, aplicada em uma permutação π é uma operação que inverte o segmento $\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_j$. Uma ***reversão de prefixo*** $\rho_p(j)$ é uma reversão da forma $\rho(1, j)$, para $1 < j \leq n$. Já uma ***reversão de sufixo*** $\rho_s(i)$ é uma reversão $\rho(i, n)$, para $1 \leq i < n$.

Formalmente, uma reversão $\rho(i, j)$ transforma uma permutação sem sinais π em:

$$\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1} \ \pi_j \ \pi_{j+1} \ \ldots \ \pi_n)$$

$$\pi \cdot \rho(i, j) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \pi_{j-1} \ \ldots \ \pi_{i+1} \ \pi_i} \ \pi_{j+1} \ldots \ \pi_n)$$

Uma ***reversão com sinais*** $\overline{\rho}(i, j)$, com $1 \leq i \leq j \leq n$, aplicada em uma permutação com sinais π é uma operação que inverte o segmento $\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_j$ e inverte os sinais dos elementos do segmento afetado. Uma ***reversão de prefixo com sinais*** $\overline{\rho}_p(j)$ é uma reversão com sinais da forma $\overline{\rho}(1, j)$, para $1 \leq j \leq n$. Já uma ***reversão de sufixo com sinais*** $\overline{\rho}_s(i)$ é uma reversão com sinais da forma $\overline{\rho}(i, n)$, para $1 \leq i \leq n$.

Formalmente, uma reversão com sinais $\overline{\rho}(i,j)$ transforma uma permutação com sinais π em:

$$\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1} \ \pi_j \ \pi_{j+1} \ \ldots \ \pi_n)$$
$$\pi \cdot \overline{\rho}(i,j) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{-\pi_j \ -\pi_{j-1} \ \ldots \ -\pi_{i+1} \ -\pi_i} \ \pi_{j+1} \ldots \ \pi_n)$$

Como exemplo, a reversão $\rho(1,3)$ sendo aplicada em $\pi = (2\ 3\ 5\ 4\ 1)$ resulta em $\pi \cdot \rho(1,3) = (5\ 3\ 2\ 4\ 1)$. De forma similar, a reversão $\overline{\rho}(1,3)$ sendo aplicada em $\pi = (+2\ +3\ -5\ +4\ +1)$ resulta em $\pi \cdot \overline{\rho}(1,3) = (+5\ -3\ -2\ +4\ +1)$.

Quando o modelo de rearranjo possui apenas reversões, caracterizam-se o ***Problema de Ordenação de Permutações sem Sinais por Reversões*** e o ***Problema de Ordenação de Permutações com Sinais por Reversões***.

2.2 Transposições

Uma ***transposição*** $\tau(i,j,k)$, com $1 \leq i < j < k \leq n+1$, é uma operação que faz a troca de posição do segmento $\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}$ com o segmento adjacente $\pi_j, \pi_{j+1}, \ldots, \pi_{k-1}$. Uma transposição não afeta sinais dos elementos nos dois segmentos sendo trocados. Uma ***transposição de prefixo*** $\tau_p(j,k)$ é uma transposição da forma $\tau(1,j,k)$, para $1 < j < k \leq n+1$. Uma ***transposição de sufixo*** $\tau_s(i,j)$ é uma transposição $\tau(i,j,n+1)$, para $1 \leq i < j < n+1$.

Formalmente, uma transposição $\tau(i,j,k)$ transforma uma permutação sem sinais π em:

$$\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1} \ \pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1} \ \pi_k \ \ldots \ \pi_n)$$
$$\pi \cdot \tau(i,j,k) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1}} \ \underline{\pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1}} \ \pi_k \ \ldots \ \pi_n)$$

Como exemplo, a transposição $\tau(1,3,5)$ sendo aplicada em $\pi = (1\ 2\ 3\ 4\ 5\ 6)$ resulta em $\pi \cdot \tau(1,3,5) = (3\ 4\ 1\ 2\ 5\ 6)$.

Quando o modelo de rearranjo possui apenas transposições, caracteriza-se o ***Problema de Ordenação de Permutações por Transposições***. Note que, como uma transposição não altera o sinal de elementos da permutação, esse modelo só abrange permutações sem sinais.

Ao permitir tanto reversões quanto transposições no modelo de rearranjo temos o ***Problema de Ordenação de Permutações com Sinais por Reversões e Transposições*** e o ***Problema de Ordenação de Permutações sem Sinais por Reversões e Transposições***.

2.3 Breakpoints

Nesta subseção, mesmo quando não explicitamente citado, as definições apresentadas consideram a versão estendida de uma permutação qualquer π.

Um ***breakpoint de reversão sem sinais*** ocorre entre um par de elementos π_i e π_{i+1}, para $0 \leq i \leq n$, se $|\pi_i - \pi_{i+1}| \neq 1$. O número de *breakpoints* de reversão sem sinais de uma permutação π é denotado por $b_\rho(\pi)$. A variação em $b_\rho(\pi)$ causada por uma operação β é denotada por $\Delta b_\rho(\pi, \beta) = b_\rho(\pi \cdot \beta) - b_\rho(\pi)$. Como exemplo, para $\pi = (3\ 2\ 1\ 4\ 5)$ temos $b_\rho(\pi) = 2$, com *breakpoints* entre os pares $(\pi_0 = 0,\ \pi_1 = 3)$ e $(\pi_3 = 1, \pi_4 = 4)$.

Um ***breakpoint de transposição*** ou ***breakpoint de reversão com sinais*** ocorre entre um par de elementos π_i e π_{i+1}, para $0 \leq i \leq n$, se $\pi_{i+1} - \pi_i \neq 1$. O número de *breakpoints* de transposição (ou reversão com sinais) de uma permutação π é denotado por $b_\tau(\pi)$ (ou $b_{\overline{\rho}}(\pi)$). A variação em $b_\tau(\pi)$ causada por uma operação β é denotada por $\Delta b_\tau(\pi, \beta) = b_\tau(\pi \cdot \beta) - b_\tau(\pi)$. Como exemplo, para $\pi = (1\ 2\ 4\ 3\ 5)$ temos $b_\tau = 3$, com *breakpoints* entre os pares $(\pi_2 = 2, \pi_3 = 4)$, $(\pi_3 = 4, \pi_4 = 3)$ e $(\pi_4 = 3, \pi_5 = 5)$.

A permutação identidade é a única na qual $b_\rho(\iota) = b_\tau(\iota) = 0$, ou seja, a permutação identidade é a única que não possui nenhum *breakpoint* de reversão sem sinais e nenhum *breakpoint* de transposição (ou *breakpoint* de reversão com sinais).

Uma ***strip*** é uma subsequência maximal de π sem *breakpoints* entre os elementos dessa subsequência. Como exemplo, considerando *breakpoints* de transposição, a permutação $\pi = (1\ 2\ 4\ 3\ 5)$ tem as *strips* (1 2), (4), (3), (5). Um ***singleton*** é uma *strip* que possui apenas um elemento. Uma *strip* $(\pi_i, \pi_{i+1}, \ldots, \pi_j)$, com $1 \leq i \leq j \leq n$ e $j - i \geq 1$, é dita crescente se $\pi_k = \pi_{k+1} - 1$, para todo $i \leq k < j$, e é dita decrescente caso contrário. Por definição, um *singleton* é considerado uma *strip* crescente.

Os *breakpoints* são uma das características mais simples da permutação. Tais características permitem definir limites das distâncias de ordenação e vários algoritmos de aproximação se utilizam do conceito de *breakpoints* e *strips*.

2.4 Grafo de Breakpoints

O ***grafo de breakpoints*** de uma permutação sem sinais π é o grafo $G_b(\pi) = (V, E)$, com $V = \{\pi_0, \pi_1, \ldots, \pi_n, \pi_{n+1}\}$ e $E = E_p \cup E_c$, sendo E_p o conjunto de ***arestas pretas*** e E_c o conjunto de ***arestas cinzas***. Existe uma ***aresta preta*** (π_i, π_{i+1}) se existe *breakpoint* entre os elementos π_i e π_{i+1}, para $0 \leq i \leq n$. Existe uma ***aresta cinza*** (π_i, π_j), para algum $0 \leq i < j \leq n+1$, se $\pi_j = \pi_i \pm 1$ e os elementos π_i e π_j não são consecutivos em π. A Figura 1 apresenta o grafo de *breakpoints* da permutação $\pi = (2\ 1\ 3\ 6\ 4\ 5)$.

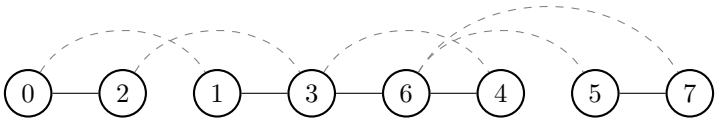


Figura 1: Grafo de *breakpoints* da permutação sem sinais $\pi = (2\ 1\ 3\ 6\ 4\ 5)$. Arestas cinzas são representadas como arcos tracejados e arestas pretas por linhas pretas contínuas.

No grafo de *breakpoints*, as arestas pretas descrevem a ordem dos elementos na permutação π e as arestas cinzas descrevem a ordem dos elementos na permutação identidade.

Um ciclo C no grafo $G_b(\pi)$ é dito um ***ciclo alternante*** se, para todo par de arestas consecutivas (e', e'') de C, as cores de e' e e'' são distintas. O número de ciclos de uma decomposição máxima de $G_b(\pi)$ em ciclos alternantes com arestas disjuntas é definido como $c_b(\pi)$.

3 O Problema da Ordenação de Permutações por Operações Ponderadas

Nesta seção será descrito o problema principal que será tratado neste trabalho e também os problemas relacionados a ele. Além disso, serão apresentados resultados existentes sobre esses problemas.

A nomenclatura dos problemas seguem os acrônimos da forma $O(C|S)(_P|_{PS})(R|T|RT)P(Ta|Ti|F)$, com as seguintes correspondências: O para ordenação; C para permutações com sinais; S para permutações sem sinais; $_P$ para operações de prefixo; $_{PS}$ para operações de prefixo e sufixo; R para reversões; T para transposições; RT para reversões e transposições; P para ponderada; Ta para tamanho; Ti para tipo; e F para fragmentação. Como exemplo, o problema *OSRPTa* é o Problema da Ordenação de Permutações sem Sinais por Reversões Ponderadas pelo Tamanho.

3.1 Ordenação de Permutações

A Ordenação por Reversões é um problema bem estudado na literatura. Para permutações com sinais existem algoritmos polinomiais para o problema, sendo que o primeiro foi apresentado por Hannenhalli e Pevzner [14] e possui complexidade de tempo $O(n^4)$. Porém, existe um algoritmo mais eficiente com complexidade de tempo $O(n^{3/2})$ [21]. Quando se deseja apenas saber o tamanho da sequência de reversões que ordena uma permutação, há também um algoritmo com complexidade de tempo de $O(n)$ [1].

Para permutações sem sinais, o problema foi provado ser NP-Difícil por Caprara [9]. Kececioglu e Sankoff [15] apresentaram o primeiro algoritmo de aproximação para o problema, de fator 2, baseado no conceito de *breakpoints*. Desde então, vários algoritmos com melhores fatores de aproximação foram criados, sendo o melhor deles dado por Berman *et al.* [6] com fator de 1,375.

A Ordenação por Transposições também é um problema NP-Difícil [8]. Como uma transposição não consegue alterar o sinal de elementos da permutação, esse problema só considera permutações sem sinais. O primeiro algoritmo de aproximação para o problema foi apresentado por Bafna e Pevzner [3] com fator de aproximação de 1,5. Desde a criação desse algoritmo, o fator de aproximação só foi melhorado por Elias e Hartman [12], que apresentaram um fator de 1,375. Ainda como contribuição do trabalho, Elias e Hartman [12] introduziram a estrutura grafo de ciclos, que permite a definição de melhores limitantes inferiores e superiores para o problema.

Já a Ordenação por Reversões e Transposições ainda possui sua complexidade

em aberto. Assim como a Ordenação por Reversões, esse problema também possui versões considerando permutações com sinais e sem sinais. Para permutações com sinais, Walter *et al.* [22] apresentaram um algoritmo de aproximação de fator 2. No mesmo trabalho, também foi apresentado um algoritmo de aproximação de fator 3 para permutações sem sinais. Ainda para a versão sem sinais, Rahman *et al.* [20] apresentaram um algoritmo de aproximação de fator $2k$, onde k é o fator de aproximação do algoritmo utilizado para a decomposição máxima de ciclos. O melhor valor de k conhecido [10] é igual a 1,4167 + ϵ, para $\epsilon > 0$. Utilizando esse valor de k, o fator de aproximação torna-se $2,8334 + \epsilon$, para $\epsilon > 0$.

3.2 Ordenação de Permutações por Operações Ponderadas pelo Tamanho

Uma reversão longa tem maior probabilidade de debilitar o indivíduo, e estudos mostram que reversões que aconteceram durante o processo evolutivo tendem a ser curtas [7]. Isso indica que uma abordagem com custo no tamanho da operação tende a ser mais realista do que considerar que todas as operações possuem o mesmo custo.

Nos problemas de ordenação por operações ponderadas pelo tamanho da operação, o custo de uma operação de tamanho l é uma função polinomial $f(l) = l^{\alpha}$. Uma reversão $\rho(i, j)$ tem tamanho $l = j - i + 1$ e uma transposição $\tau(i, j, k)$ tem tamanho $l = k - i$. Note que a abordagem tradicional é equivalente a usar $\alpha = 0$.

Pinter e Skiena [19] apresentaram um algoritmo de aproximação com fator $O(\lg^2 n)$, quando $\alpha = 1$, para o problema *OSRPTa*. Para o mesmo problema, Bender *et al.* [5] apresentaram um fator de aproximação de $O(\lg n)$ para $\alpha = 1$ e um fator de aproximação igual a 2 para $\alpha \geq 2$. Bender *et al.* [5] também concluíram que para $\alpha \geq 3$ o problema se torna polinomial.

Para $\alpha = 1$, Lintzmayer *et al.* [18] apresentam algoritmos de aproximação com fator $O(\lg^2 n)$ para os problemas OSR_PPTa, OST_PPTa, $OSRT_PPTa$, $OSR_{PS}PTa$, $OST_{PS}PTa$, $OSRT_{PS}PTa$, OCR_PPTa, $OCRT_PPTa$, $OCR_{PS}PTa$ e $OCRT_{PS}PTa$.

Lintzmayer [17] fez uma análise dos problemas *OSTPTa* e *OSRTPTa* considerando várias atribuições de valores para α. Os fatores de aproximação apresentados por Lintzmayer [17], para ambos os problemas, foram:

- $O(n^{\alpha})$, para $0 < \alpha < 1$;
- $O(\lg^2 n)$, para $\alpha = 1$;
- $O(\lg n)$, para $1 < \alpha < 2$;
- 2, para $\alpha \geq 2$.

Além disso, Lintzmayer [17] concluiu que para $\alpha \geq 3$ os problemas *OSTPTa* e *OSRTPTa* podem ser resolvidos em tempo polinomial.

3.3 Ordenação de Permutações por Operações Ponderadas pelo Tipo

Em abordagens nas quais a operação é ponderada pelo seu tipo, normalmente uma transposição recebe um custo maior que uma reversão, pois reversões ocorrem em uma proporção muito maior que transposições em cenários reais [2, 7]. Além disso, a maioria dos algoritmos que trabalham com operações não ponderadas utilizam estruturas em que a aplicação de uma transposição é preferível a uma reversão, fazendo com que a sequência de rearranjos devolvida por esses algoritmos tenham uma proporção maior de transposições [2].

Utilizando uma proporção de custo 2:1 (custo transposição : custo reversão), Eriksen [13] introduziu um algoritmo $(1 + \epsilon)$-aproximado, para $\epsilon > 0$. Bader e Ohlebusch [2] consideraram uma terceira operação chamada de transposição reversa. Uma ***transposição reversa*** é uma operação que troca dois segmentos adjacentes de lugar, invertendo um deles. Para qualquer proporção de custos entre 1:1 e 2:1 (custo transposição : custo reversão) e considerando que o custo de uma transposição reversa é igual ao de uma transposição, Bader e Ohlebusch [2] apresentam um algoritmo com fator de aproximação de 1,5.

3.4 Ordenação de Permutações por Operações Ponderadas pelo Número de Fragmentações

Neste trabalho, propomos uma nova versão para o problema da ordenação de permutações por operações ponderadas. Nessa nova versão, o custo de uma operação é

igual a quantidade de fragmentações que a operação causa na permutação. Definimos o custo de β como $f(\beta)$.

Seja $\rho(i,j)$ uma reversão, com $1 \leq i < j \leq n$, e π uma permutação qualquer. Quando $i > 1$ e $j < n$, $\rho(i,j)$ causa duas fragmentações quando aplicada à permutação π e, portanto, $f(\rho(i,j)) = 2$. Quando $i = 1$ e $j < n$, $\rho(i,j)$ é uma reversão de prefixo e $f(\rho(i,j)) = 1$. Quando $i > 1$ e $j = n$, $\rho(i,j)$ é uma reversão de sufixo e $f(\rho(i,j)) = 1$. Quando $i = 1$ e $j = n$, $\rho(i,j)$ é uma reversão que inverte todos os elementos da permutação e $f(\rho(i,j)) = 0$. Os mesmos custos também são válidos para reversões com sinais. Para ilustrar o custo das reversões, podemos observar em (1), (2), (3), (4) e (5) como cada tipo de reversão altera uma permutação sem sinais π.

$$\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1} \ \pi_j \ \pi_{j+1} \ \ldots \ \pi_n) \tag{1}$$

$$\pi \cdot \rho(i,j) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \pi_{j-1} \ \ldots \ \pi_{i+1} \ \pi_i} \ \pi_{j+1} \ \ldots \ \pi_n) \tag{2}$$

$$\pi \cdot \rho(1,j) = (\underline{\pi_j \ \pi_{j-1} \ \ldots \ \pi_{i+1} \ \pi_i \ \pi_{i-1} \ \ldots \ \pi_1} \ \pi_{j+1} \ \ldots \ \pi_n) \tag{3}$$

$$\pi \cdot \rho(i,n) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_n \ \ldots \ \pi_{j+1} \ \pi_j \ \pi_{j-1} \ \ldots \ \pi_{i+1} \ \pi_i}) \tag{4}$$

$$\pi \cdot \rho(1,n) = (\underline{\pi_n \ \ldots \ \pi_{j+1} \ \pi_j \ \pi_{j-1} \ \ldots \ \pi_{i+1} \ \pi_i \ \pi_{i-1} \ \ldots \ \pi_1}) \tag{5}$$

Seja $\tau(i,j,k)$ uma transposição, com $1 \leq i < j < k \leq n+1$, e π uma permutação qualquer. Quando $i > 1$ e $k < n+1$, $\tau(i,j,k)$ causa três fragmentações quando aplicada à permutação π e, portanto, $f(\tau(i,j,k)) = 3$. Quando $i = 1$ e $k < n+1$, $\tau(i,j,k)$ é uma transposição de prefixo e $f(\tau(1,j,k)) = 2$. Quando $i > 1$ e $k = n+1$, $\tau(i,j,k)$ é uma transposição de sufixo e $f(\tau(i,j,n+1)) = 2$. Quando $i = 1$ e $k = n+1$, a transposição $\tau(i,j,k)$ causa apenas uma fragmentação na permutação e $f(\tau(1,j,n+1)) = 1$. Para ilustrar o custo das transposições, podemos observar em (6), (7), (8), (9) e (10) como cada tipo de transposição altera uma permutação sem sinais π.

$$\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1} \ \pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1} \ \pi_k \ \ldots \ \pi_n) \tag{6}$$

$$\pi \cdot \tau(i,j,k) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1}} \ \underline{\pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1}} \ \pi_k \ \ldots \ \pi_n) \tag{7}$$

$$\pi \cdot \tau(1,j,k) = (\underline{\pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1}} \ \underline{\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1}} \ \pi_k \ \ldots \ \pi_n) \tag{8}$$

$$\pi \cdot \tau(i,j,n+1) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1} \ \pi_k \ \ldots \ \pi_n} \ \underline{\pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1}}) \tag{9}$$

$$\pi \cdot \tau(1,j,n+1) = (\underline{\pi_j \ \pi_{j+1} \ \ldots \ \pi_{k-1} \ \pi_k \ \ldots \ \pi_n} \ \underline{\pi_1 \ \ldots \ \pi_{i-1} \ \pi_i \ \pi_{i+1} \ \ldots \ \pi_{j-1}}) \tag{10}$$

Apesar da semelhança dessa abordagem com a de custos por tipo da operação, não existem resultados conhecidos para nenhum problema utilizando custos por fragmentações na permutação. A principal diferença dessa abordagem é a atribuição de diferentes pesos para operações de prefixo e sufixo, o que não acontece nas outras abordagens.

Como exemplo, podemos ordenar a permutação $\pi = (1\ 3\ 8\ 5\ 4\ 7\ 6\ 2)$ aplicando as operações $\rho(3,5)$, $\rho(5,7)$, $\rho(2,7)$ e $\rho(2,8)$, nessa ordem. Essa sequência de operações é uma solução ótima da instância π para o problema de ordenação por reversões não ponderadas. Considerando o problema $OSRPF$, os custos das operações $\rho(3,5)$, $\rho(5,7)$, $\rho(2,7)$ e $\rho(2,8)$ são 2, 2, 2 e 1, respectivamente. Assim, o custo total da sequência é igual a 7. As reversões $\rho(1,7)$, $\rho(7,8)$, $\rho(5,8)$, $\rho(3,8)$, $\rho(1,6)$ e $\rho(4,8)$, aplicadas nessa ordem, também ordenam π. Essa sequência é uma solução ótima da instância π para o problema de ordenação por reversões de prefixo e sufixo não ponderadas. Considerando o problema $OSPRF$, cada uma dessas operações tem custo 1, e assim o custo total da sequência é igual a 6. Porém, a sequência de reversões $\rho(3,5)$, $\rho(1,7)$, $\rho(1,3)$, $\rho(7,8)$ e $\rho(1,8)$ ordena π com custo 5, sendo uma solução ótima da instância π para o problema $OSRPF$.

As Tabelas 1 e 2 apresentam alguns resultados computacionais para permutações pequenas. Ambas possuem as colunas “Problema”, “Limite Superior”, “Prefixo e Sufixo” e “Modelo Ponderado”. A coluna “Problema” indica qual problema e modelo de rearranjo são considerados. A coluna “Limite Superior” foi construída usando, para cada permutação, uma sequência ótima de ordenação quando considera-se que todas as operações tem custo unitário. O custo final foi calculado aplicando a função de custo do problema ponderado pelo número de fragmentações nas sequências obtidas. A coluna “Prefixo e Sufixo” foi construída de forma similar a “Limite Superior”, sendo que foi encontrada, para cada permutação, uma sequência ótima de ordenação quando apenas operações de prefixo e sufixo são permitidas e considerando que elas tem peso unitário. O custo final foi calculado aplicando a função de custo do problema ponderado pelo número de fragmentações nas sequências obtidas. A coluna “Modelo Ponderado” calcula uma sequência ótima considerando a função de custo do problema ponderado.

A Tabela 1 considera a média das colunas “Limite Superior”, “Prefixo e Sufixo”

Problema	Limite Superior	Prefixo e Sufixo	Modelo Ponderado
OSRPF	9,02	7,39	7,25
OCRPF	12,36	9,15	8,78
OSTPF	11,23	9,87	9,29
OSRTPF	8,98	8,32	7,22
OCRTPF	11,21	10,12	8,25

Tabela 1: Distância Média para Problemas de Ordenação por Operações Ponderadas pelo Número de Fragmentações.

Problema	Limite Superior	Prefixo e Sufixo	Modelo Ponderado
OSRPF	15	10	9
OCRPF	18	13	11
OSTPF	16	14	14
OSRTPF	14	12	9
OCRTPF	18	15	11

Tabela 2: Diâmetro para Problemas de Ordenação por Operações Ponderadas pelo Número de Fragmentações.

e “Modelo Ponderado”. A Tabela 2 apresenta o diâmetro das colunas “Limite Superior”, “Prefixo e Sufixo” e “Modelo Ponderado”. Ambas as tabelas consideram todas as permutações com sinais de tamanho 9 e todas as permutações sem sinais de tamanho 10.

4 Objetivos

Este trabalho tem como objetivo estudar limitantes e criar algoritmos de aproximação para os problemas da Ordenação de Permutações por Reversões Ponderadas pelo Número de Fragmentações ($OSRPF$ e $OCRPF$), Ordenação de Permutações por Transposições Ponderadas pelo Número de Fragmentações ($OSTPF$) e Ordenação de Permutações por Reversões e Transposições Ponderadas pelo Número de Fragmentações ($OSRTPF$ e $OCRTPF$).

Inicialmente, estudaremos os fatores de aproximação para os algoritmos já existentes para os problemas relacionados. A partir disso, pretendemos criar novas heurísticas que levem em consideração o custo das operações e analisar se essas mudanças trazem melhores fatores de aproximação. Para essa análise, é essencial o estudo de novos limitantes para os problemas.

5 Metodologia

O trabalho a ser realizado se iniciará com o estudo dos problemas para permutações sem sinais. Acredita-se que grande parte dos resultados para $OSRPF$, $OSTPF$ e $OSRTPF$ ajudarão a conseguir resultados para as versões com sinais ($OCRPF$ e $OCRTPF$). Por ser um trabalho teórico, seu desenvolvimento envolverá o estudo de provas e teoremas para o estabelecimento de limitantes e criação de novos algoritmos. Sempre que necessário, serão desenvolvidos programas para validar os resultados obtidos e comparar diferentes algoritmos que possam ser criados ao longo da pesquisa.

6 Plano de Trabalho

	2017										2018											2019	
	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F
1	*	*	*	*		*	*	*	*															
2	*	*	*	*	*	*				*				*				*						
3					*	*																		
4							*																	
5													*	*	*	*								
6						*	*	*	*															
7										*	*	*	*											
8														*	*	*	*							
9																		*	*	*	*			
10								*	*			*	*			*	*			*	*	*		
11																							*	
12																								*

Tabela 3: Cronograma de atividades.

A Tabela 3 apresenta o cronograma das atividades a serem executadas no decorrer do trabalho, que são listadas a seguir:

1. Obtenção dos créditos obrigatórios em disciplinas do programa de mestrado;
2. Revisão da literatura;
3. Escrita da proposta de mestrado;
4. Exame de qualificação do mestrado;
5. Participação do Programa de Estágio Docente (PED);
6. Adaptação de algoritmos existentes para os problemas estudados;

7. Investigação de novos limitantes para os problemas estudados;
8. Criação de heurísticas e execução de experimentos;
9. Investigação de melhores fatores de aproximação;
10. Escrita da dissertação;
11. Revisão da dissertação;
12. Defesa da dissertação.

O tempo e a ordem alocados para as atividades podem mudar no decorrer do desenvolvimento da pesquisa, uma vez que alguns resultados podem ser mais promissores que os outros, fazendo com que alguma(s) atividade(s) sejam realocada(s).

Referências

[1] 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***, 8:483–491, 2001.

[2] M. Bader and E. Ohlebusch. Sorting by Weighted Reversals, Transpositions, and Inverted Transpositions. ***Journal of Computational Biology***, 14(5):615–636, 2007.

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

[4] C. Baudet, U. Dias, and Z. Dias. Length and Symmetry on the Sorting by Weighted Inversions Problem. In S. Campos, editor, ***Advances in Bioinformatics and Computational Biology***, volume 8826 of ***Lecture Notes in Computer Science***, pages 99–106. Springer International Publishing, Switzerland, 2014.

[5] M. A. Bender, D. Ge, S. He, H. Hu, R. Y. Pinter, S. S. Skiena, and F. Swidan. Improved Bounds on Sorting by Length-Weighted Reversals. ***Journal of Computer and System Sciences***, 74(5):744–774, 2008.

[6] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In R. Möhring and R. Raman, editors, ***Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002)***, volume 2461 of ***Lecture Notes in Computer Science***, pages 200–210. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany, 2002.

[7] M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric Genome Rearrangement. ***Gene***, 172(1):GC11–GC17, 1996.

[8] L. Bulteau, G. Fertin, and I. Rusu. Sorting by Transpositions is Difficult. ***SIAM Journal on Computing***, 26(3):1148–1180, 2012.

[9] A. Caprara. Sorting Permutations by Reversals and Eulerian Cycle Decompositions. ***SIAM Journal on Discrete Mathematics***, 12(1):91–110, 1999.

[10] X. Chen. On Sorting Unsigned Permutations by Double-Cut-and-Joins. ***Journal of Combinatorial Optimization***, 25(3):339–351, 2013.

[11] U. M. Dias. ***Problemas de Comparação de Genomas***. PhD thesis, Institute of Computing, University of Campinas, 2012. In Portuguese.

[12] I. Elias and T. Hartman. A 1.375-Approximation Algorithm for Sorting by Transpositions. ***IEEE/ACM Transactions on Computational Biology and Bioinformatics***, 3(4):369–379, 2006.

[13] N. Eriksen. $(1+\epsilon)$-Approximation of Sorting by Reversals and Transpositions. ***Theoretical Computer Science***, 289(1):517–529, 2002.

[14] S. Hannenhalli and P. A. Pevzner. Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals. ***Journal of the ACM***, 46(1):1–27, 1999.

[15] J. D. Kececioglu and D. Sankoff. Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement. ***Algorithmica***, 13: 180–210, 1995.

[16] Y. C. Lin, C.-Y. Lin, and C. R. Lin. Sorting by Reversals and Block-Interchanges with Various Weight Assignments. ***BMC Bioinformatics***, 10(1):398, 2009.

[17] C. N. Lintzmayer. ***The Problem of Sorting Permutations by Prefix and Suffix Rearrangements***. PhD thesis, University of Campinas, Institute of Computing, 2016. In English.

[18] C. N. Lintzmayer, G. Fertin, and Z. Dias. Approximation Algorithms for Sorting by Length-Weighted Prefix and Suffix Operations. ***Theoretical Computer Science***, 593:26–41, 2015.

[19] R. Y. Pinter and S. Skiena. Genomic Sorting with Length-Weighted Reversals. ***Genome Informatics***, 13:2002, 2002.

[20] A. Rahman, S. Shatabda, and M. Hasan. An Approximation Algorithm for Sorting by Reversals and Transpositions. ***Journal of Discrete Algorithms***, 6(3):449–457, 2008.

[21] E. Tannier and M.-F. Sagot. Sorting by Reversals in Subquadratic Time. In S. C. Sahinalp, S. Muthukrishnan, and U. Dogrusoz, editors, ***Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM'2004)***, volume 3109 of ***Lecture Notes in Computer Science***, pages 1–13, Berlin/Heidelberg, Germany, 2004. Springer Berlin Heidelberg.

[22] M. E. M. T. Walter, Z. Dias, and J. Meidanis. Reversal and Transposition Distance of Linear Chromosomes. In ***Proceedings of the 5th International Symposium on String Processing and Information Retrieval (SPIRE'1998)***, pages 96–102, Los Alamitos, CA, USA, 1998. IEEE Computer Society.