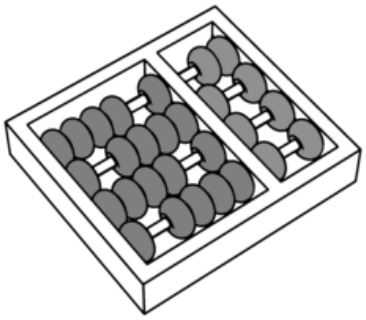
**Universidade Estadual de Campinas**

Instituto de Computação



Proposta de Dissertação de Mestrado

---

# Heurísticas para o Problema de Ordenação de Permutações com Sinais por Reversões e Transposições

---

*Candidato*: Klairton de Lima Brito

*Orientador*: Prof. Dr. Zanoni Dias

**Resumo**

Computar a quantidade de eventos capazes de transformar um genoma em outro é uma tarefa difícil e de grande importância no campo da biologia computacional. O Problema de Ordenação de Genomas busca calcular o número mínimo de eventos de rearranjo necessários para ordenar blocos conservados do genoma, que é representado como uma permutação. Por propriedades algébricas de permutações, o problema de transformar um genoma em outro pode ser declarado como equivalente ao Problema de Ordenação de Genomas. Para ambos os casos, é utilizado um modelo de rearranjo que define quais eventos que podem ser aplicados e consequentemente acabam por alterar o genoma. Dentre os eventos de rearranjo mais comuns, podemos citar a reversão e a transposição. Modelos que utilizam tais eventos separadamente já possuem uma vasta bibliografia. Entretanto, modelos que utilizam ambos ainda são um grande desafio para a Teoria da Computação. Algumas variações desses eventos onde é imposto um tamanho limite nas operações que podem afetar o genoma também são estudadas, sendo essas variações chamadas de operações curtas e super curtas. Vários resultados já foram publicados em estudos relacionados a essas operações, mas algumas questões ainda permanecem em aberto. O trabalho em questão terá como meta estudar a variação do problema que permite eventos de reversão e transposição, tanto na sua versão tradicional, quanto na versão onde são permitidas apenas operações curtas.

# 1 Introdução

O genoma de uma espécie é composto de cromossomos representados como um conjunto ordenado de genes. Blocos conservados são regiões de alta similaridade entre dois genomas quaisquer e essa forma de representação é largamente utilizada na comparação de genomas.

Com o atual avanço no mapeamento genético e devido à imensa quantidade de dados obtidos surge então a necessidade de criar modelos que possam utilizar esses dados como ponto de partida para classificar os seres vivos ou inferir a proximidade entre eles. De imediato, emerge um ponto bastante importante: poder estimar a distância evolutiva ou mesmo determinar qual o grau de parentesco ao comparar dois genomas. Uma área do

conhecimento surgiu para computar essas estimativas tendo como base eventos que afetam grandes porções do genoma, tais eventos são chamados de rearranjos de genomas. Esses eventos são mais raros que as mutações pontuais, o que em tese favorece a geração de estimativas mais precisas. Dentre os eventos estudados na literatura podemos citar fusão, fissão, reversão e transposição. Alguns eventos possuem variações, tipicamente considerando restrições extras, como por exemplo reversões e transposições de prefixos e sufixos, que são operações que afetam o começo ou o final do genoma, respectivamente.

Com a definição desses eventos surgem então os problemas de distância em rearranjos de genomas, que consistem em, dados dois genomas, encontrar uma sequência de rearranjos que transforma um genoma no outro. O objetivo consiste em obter o menor número de eventos de rearranjo possíveis para tal, pois, com base em uma teoria conhecida como *Princípio da Máxima Parcimônia*, a explicação mais simples para um fenômeno é a mais provável. O tamanho da menor sequência de eventos de rearranjo que transforma um genoma no outro é denominado *distância de rearranjo* entre os indivíduos.

Um *modelo de rearranjo* determina qual o conjunto de eventos permitidos para transformar um genoma em outro. Os estudos iniciais apresentaram modelos de rearranjo com a aplicação de apenas um evento, mas com o decorrer do tempo, surgiram modelos que permitiam dois ou mais eventos [1, 2]. Atualmente existem modelos [3, 4] que permitem a aplicação de pesos para cada evento, que atua como uma probabilidade do evento afetar o genoma.

## 2 Eventos de Rearranjos

No problema de rearranjo de genomas, um genoma é representado como uma $n$-tupla, onde cada elemento representa um gene ou bloco de genes. Assumindo que não existem repetições de genes, a $n$-tupla é dada como uma permutação $\pi = (\pi_1\, \pi_2\, \pi_3\, \ldots\, \pi_n)$, onde $\pi_i \in \{-n, -(n-1), \ldots, -2, -1, +1, +2, \ldots, +(n-1), +n\}$ tal que $|\pi_i| \neq |\pi_j| \leftrightarrow i \neq j$. O sinal de positivo ou negativo em um elemento representa a orientação do gene. Existem abordagens que não trabalham com a orientação dos genes e nesse caso, o sinal é omitido.

Definindo-se um modelo de rearranjo M e as permutações $\pi$ e $\sigma$, formalmente o problema consiste em encontrar uma sequência de tamanho mínimo de eventos pertencentes ao conjunto de eventos de M de maneira que $\pi \cdot \rho_1 \cdot \rho_2 \ldots \rho_t = \sigma$. O tamanho da menor sequência de eventos que transforma a permutação $\pi$ em $\sigma$ é denominada de *distância* entre as permutações $\pi$ e $\sigma$ no modelo M, que é denotada como $d_M(\pi, \sigma) = t$. A maior distância entre duas permutações de tamanho $n$ no modelo M é chamada de *diâmetro da distância de rearranjo* e denotada como $D_M(n)$.

Definindo-se a permutação identidade como $\iota_n = (1\,2\,3\,\ldots\,n)$ e considerando uma permutação $\alpha$ qualquer, podemos ordenar $\alpha$ (ou seja, transformar $\alpha$ em $\iota_n$). O tamanho da sequência de operações no modelo M que transforma $\alpha$ em $\iota_n$ é denotado como $d_M(\alpha, \iota_n) = d_M(\alpha)$. Este processo de ordenação é equivalente a transformar a permutação $\pi$ na permutação $\sigma$ se tornarmos $\alpha = \sigma^{-1} \cdot \pi$, onde $\sigma^{-1}$ é a inversa de $\sigma$ tal que $\sigma^{-1} \cdot \sigma = \iota_n$, uma vez que $d_M(\pi, \sigma) = d_M(\sigma^{-1} \cdot \pi, \sigma^{-1} \cdot \sigma) = d_M(\alpha, \iota_n) = d_M(\alpha)$.

## 2.1 Reversões

Uma reversão $\rho(i, j)$, onde $1 \leq i \leq j \leq n$, é um evento que inverte a posição e a orientação dos genes em um bloco do genoma. Quando a orientação dos genes é conhecida, caracteriza-se um Problema de Ordenação de Permutações com Sinais por Reversões, caso contrário, tem-se um Problema de Ordenação de Permutações sem Sinais por Reversões (as vezes chamado apenas de Problema de Ordenação de Permutações por Reversões). O exemplo a seguir mostra uma operação de reversão $\rho(i, j)$ sendo aplicada na permutação $\pi$.

$$\pi = (+\pi_1 + \pi_2 + \pi_3 \ \ldots + \pi_{i-1} + \pi_i \ldots + \pi_j + \pi_{j+1} \ldots + \pi_n)$$

$$\pi \cdot \rho(i, j) = (+\pi_1 + \pi_2 + \pi_3 \ \ldots + \pi_{i-1} \underline{-\pi_j \ \ldots \ - \pi_i} + \pi_{j+1} \ \ldots \ \pi_n)$$

Como exemplo, a reversão $\rho(2, 5)$ sendo aplicada em uma permutação sem sinais $\pi = (1\,2\,3\,4\,5\,6)$ resulta na permutação sem sinais $\pi \cdot \rho(2, 5) = (1\,5\,4\,3\,2\,6)$. De maneira similar, utilizando a reversão $\rho(2, 5)$, mas agora aplicando em uma permutação com sinais $\pi = (+1\,+2\,+3\,+4\,+5\,+6)$, resulta na permutação com sinais $\pi \cdot \rho(2, 5) = (+1\,-5\,-4\,-3\,-2\,+6)$.

## 2.2 Transposições

Uma transposição $\tau(i, j, k)$, onde $1 \leq i < j < k \leq n + 1$, é um evento que afeta dois blocos adjacentes de um mesmo cromossomo trocando-os de posição, mas sem afetar a orientação e a posição dos genes dentro dos blocos. Por tal motivo, em um modelo de rearranjo cujo conjunto de eventos é formado unicamente pelo evento de transposição ou eventos que não alteram a orientação dos genes, elimina-se a necessidade de associar a orientação dos mesmos. O exemplo a seguir mostra uma operação de transposição $\tau(i, j, k)$ sendo aplicada na permutação $\pi$.

$$\pi = (\pi_1\, \pi_2\, \pi_3\, \ldots \pi_{i-1} \pi_i \ldots \pi_{j-1} \pi_j \ldots \pi_{k-1} \pi_k\, \ldots \pi_n)$$

$$\pi \cdot \tau(i, j, k) = (\pi_1\, \pi_2\, \pi_3\, \ldots\, \pi_{i-1}\, \underline{\pi_j\, \ldots\, \pi_{k-1}}\, \underline{\pi_i\, \ldots\, \pi_{j-1}}\, \pi_k\, \ldots\, \pi_n)$$

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

O Problema de Ordenação de Permutações por Transposições caracteriza-se pela utilização unicamente do evento de transposição como modelo de rearranjo, buscando a menor sequência de eventos de transposição que possa transformar um genoma em outro.

## 2.3 Reversões e Transposições

O Problema de Ordenação de Permutações por Reversões e Transposições permite tanto o uso do eventos de reversão quanto eventos de transposição durante a ordenação de uma permutação qualquer $\pi$. Quando a orientação dos genes é conhecida, temos um Problema de Ordenação de Permutações com Sinais por Reversões e Transposições, caso contrário, temos um Problema de Ordenação de Permutações sem Sinais por Reversões e Transposições (as vezes chamado apenas de Problema de Ordenação de Permutações por Reversões e Transposições).

A permutação sem sinais $\pi = (2\,6\,5\,4\,1\,7\,3)$, utilizando apenas reversões, requer o uso de no mínimo quatro operações para ser ordenada ($d_r(\pi) = 4$). Levando em consideração

apenas transposições, a permutação $\pi$ também necessita de no mínimo quatro operações para ser ordenada ($d_t(\pi) = 4$). Por sua vez, considerando ambos os eventos, reversões e transposições, apenas três operações são necessárias para ordenar a permutação $\pi$ ($d_{rt}(\pi) = 3$). Sequências válidas de ordenação para a permutação $\pi$, levando em consideração os três modelos, são apresentadas no exemplo abaixo. Na sequência da esquerda apenas reversões foram utilizadas, na sequência do centro apenas transposições e na sequência da direita, reversões e transposições.

| | | |
|---|---|---|
| $(2\,6\,5\,4\,1\,7\,3) \cdot \rho(6,7)$ | $(2\,6\,5\,4\,1\,7\,3) \cdot \tau(4,7,8)$ | $(2\,6\,5\,4\,1\,7\,3) \cdot \tau(5,7,8)$ |
| $(2\,6\,5\,4\,1\,3\,7) \cdot \rho(2,6)$ | $(2\,6\,5\,3\,4\,1\,7) \cdot \tau(2,3,7)$ | $(2\,6\,5\,4\,3\,1\,7) \cdot \rho(2,6)$ |
| $(2\,3\,1\,4\,5\,6\,7) \cdot \rho(2,3)$ | $(2\,5\,3\,4\,1\,6\,7) \cdot \tau(2,3,6)$ | $(2\,1\,3\,4\,5\,6\,7) \cdot \rho(1,2)$ |
| $(2\,1\,3\,4\,5\,6\,7) \cdot \rho(1,2)$ | $(2\,3\,4\,1\,5\,6\,7) \cdot \tau(1,4,5)$ | $(1\,2\,3\,4\,5\,6\,7)$ |
| $(1\,2\,3\,4\,5\,6\,7)$ | $(1\,2\,3\,4\,5\,6\,7)$ | $d_{rt}(\pi) = 3$ |
| $d_r(\pi) = 4$ | $d_t(\pi) = 4$ | |

## 2.4 Operações Curtas e Super Curtas

Existem variantes de problemas que atribuem um limite no tamanho das operações que podem afetar o genoma. Podemos citar, como exemplos, as chamadas operações (reversões ou transposições) curtas ou super curtas. Como visto anteriormente, um evento de reversão $\rho(i, j)$ inverte a posição e a orientação dos genes dentro de um bloco. Chamamos uma reversão $\rho(i, j)$ de $k$-reversão se $k = j - i + 1$. Uma $k$-reversão é dita curta se $k \leq 3$ e super curta se $k \leq 2$. Da mesma forma, uma transposição $\tau(i, j, k)$ é dita uma $(x, y)$-transposição se $x = j - i$ e $y = k - j$. Uma $(x, y)$-transposição é dita curta se $x + y \leq 3$ e super curta se $x + y = 2$. O Problema de Ordenação de Permutações por Operações Curtas, quando mencionado nesse trabalho, refere-se ao problema onde os eventos de reversão curta e transposição curta podem ser utilizados para ordenar uma permutação. De maneira similar, o Problema de Ordenação de Permutações por Operações Super Curtas refere-se ao problema onde os eventos de reversão super curta e transposição super curta podem ser utilizados para ordenar uma permutação.

# 3 Fundamentação Teórica

Alguns conceitos que serão mencionados ao longo do texto e de suma importância para compreensão do trabalho são apresentados nesta seção, sendo eles: breakpoints, strips, permutação reduzida e inversão.

## 3.1 Breakpoints

Seja a *permutação estendida* que pode ser obtida a partir de $\pi$ inserindo-se dois novos elementos: $\pi_0 = 0$ e $\pi_{n+1} = n+1$. Dada uma permutação sem sinais $\pi$, um *breakpoint de reversão sem sinais* é um par de elementos $\pi_i$ e $\pi_{i+1}$ de $\pi$, com $0 \leq i \leq n$, tal que $|\pi_i - \pi_{i+1}| \neq 1$. Por exemplo, $\pi = (0\,1\,2 \cdot 5\,4\,3 \cdot 6)$, onde " $\cdot$ " representa cada *breakpoint* da permutação estendida. O número de *breakpoints de reversão sem sinais* em uma permutação $\pi$ é denotado de $b_r(\pi)$ e, no exemplo acima, temos que $b_r(\pi) = 2$.

Em uma permutação com sinais $\pi$, um *breakpoint de reversão com sinais* é um par de elementos $\pi_i$ e $\pi_{i+1}$ de $\pi$, com $0 \leq i \leq n$, tal que $\pi_{i+1} - \pi_i \neq 1$. Por exemplo, $\pi = (0 +1 +2 \cdot +5 \cdot +4 \cdot -3 \cdot +6)$, onde "$\cdot$" representa cada *breakpoint* da permutação estendida. O número de *breakpoints de reversão com sinais* em uma permutação $\pi$ é denotado de $b_{\overline{r}}(\pi)$ e, no exemplo acima, temos que $b_{\overline{r}}(\pi) = 4$.

Em uma permutação sem sinais $\pi$, um *breakpoint de transposição* é um par de elementos $\pi_i$ e $\pi_{i+1}$ de $\pi$, com $0 \leq i \leq n$, tal que $\pi_{i+1} - \pi_i \neq 1$. Por exemplo, $\pi = (0\,1\,2 \cdot 5 \cdot 4 \cdot 3 \cdot 6)$, onde " $\cdot$ " representa cada *breakpoint* da permutação estendida. O número de *breakpoints de transposição* em uma permutação $\pi$ é denotado de $b_t(\pi)$ e, no exemplo acima, temos que $b_t(\pi) = 4$.

A permutação identidade é a única permutação que não possui *breakpoints* (de nenhum dos três tipos).

## 3.2 Strips

Os *breakpoints* dividem uma permutação em *strips*, que são intervalos maximais sem *breakpoints*. O processo de obter as *strips* de uma permutação não leva em consideração

os elementos $\pi_0$ e $\pi_{n+1}$, que são inseridos na permutação estendida. Na permutação estendida sem sinais $\pi = (0\,1\,2 \cdot 5\,4\,3 \cdot 6)$, considerando *breakpoints de reversão sem sinais*, temos duas *strips*, sendo elas (1 2) e (5 4 3). No exemplo da permutação estendida com sinais $\pi = (0 +1 +2 \cdot +5 \cdot +4 \cdot -3 \cdot +6)$ com o modelo de *breakpoint de reversão com sinais*, temos quatro *strips*, sendo elas $(+1 + 2), (+5), (+4)$ e $(-3)$.

Considerando permutações sem sinais, uma *strip* é dita crescente caso seus elementos formem uma sequência crescente e é dita decrescente caso contrário. Se uma *strip* contém apenas um único elemento é chamada de *singleton* e definida como crescente.

## 3.3 Permutação Reduzida

Christie [5] mostrou que uma permutação $\pi$ pode ser transformada em uma permutação reduzida $\pi_{red}$, e considerando distância de transposição é verdade que $d_t(\pi) \leq d_t(\pi_{red})$. A transformação é realizada utilizando as *strips* de uma permutação, os passos para realizar a transformação são:

- Remover a primeira *strip*, se ela iniciar com 1;
- Remover a última *strip*, se ela terminar com $n$;
- Substituir cada *strip* pelo menor elemento contido nela;
- Renomear a sequência final de modo a obter uma permutação válida.

Por exemplo, a permutação sem sinais $\pi$ = (1 2 · 8 9 · 5 6 7 · 3 4), considerando *breakpoints de transposição* apresenta as seguintes *strips*: (1 2), (8 9), (5 6 7) e (3 4). Removemos a primeira *strip* pois ela inicia com 1, resultando nas *strips* (8 9), (5 6 7) e (3 4). Em seguida, substituímos cada *strip* pelo menor elemento contido nela, resultando em (8) (5) e (3). Por fim, renomeamos a sequência final para obter uma permutação válida, e obtemos a permutação reduzida $\pi_{red}$ = (3 2 1). Note que a permutação reduzida $\pi_{red}$ = (3 2 1) necessita de duas operações de transposição para transformá-la na permutação identidade $(3\,2\,1) \cdot \tau(1,3,4) \cdot \tau(2,3,4) = \iota_3$, da mesma forma a permutação original $\pi$ = (1 2 · 8 9 · 5 6 7 · 3 4) requer também duas transposições para transformá-la na permutação identidade $(1\ 2\ \cdot\ 8\ 9\ \cdot\ 5\ 6\ 7\ \cdot\ 3\ 4) \cdot \tau(3,8,10) \cdot \tau(5,7,10) = \iota_9$, respeitando que $d_t(\pi) \leq d_t(\pi_{red})$.

### 3.4 Inversões

Dada uma permutação $\pi$, uma inversão ocorre entre um par de elementos $\pi_i$ e $\pi_j$ de $\pi$, com $i > j$, se $|\pi_i| < |\pi_j|$. Por exemplo, na permutação $\pi = (1\,2\,5\,4\,3)$ ocorrem inversões entre os elementos $(\pi_5, \pi_4)$, $(\pi_5, \pi_3)$ e $(\pi_4, \pi_3)$. O número total de inversões de uma permutação $\pi$ é denotado de $inv(\pi)$, e no exemplo anterior, $inv(\pi) = 3$.

## 4 Revisão da Literatura

Essa seção apresenta um levantamento de trabalhos existentes na literatura, bem como os melhores resultados obtidos até então para diferentes tipos de modelos de rearranjo.

### 4.1 Reversões

A primeira versão de um algoritmo exato em tempo polinomial para o Problema de Ordenação de Permutações com Sinais por Reversões foi dada por Hannenhalli e Pevzner [6]. Posteriormente, a estratégia adotada por Hannenhalli e Pevzner [6] foi simplificada por Bergeron [7]. Atualmente, existe um algoritmo com complexidade sub-quadrática proposto por Tannier e coautores [8] e, quando apenas a distância é desejada, sem a necessidade de indicar uma sequência de eventos que transformam um genoma no outro, um algoritmo em tempo linear proposto por Bader e coautores [9] pode ser utilizado.

Por outro lado, a versão do Problema de Ordenação de Permutações sem Sinais por Reversões pertence à classe de problemas NP-Difíceis, tendo a prova sido realizada por Caprara[10]. Com o enquadramento do Problema de Ordenação de Permutações sem Sinais por Reversões na classe NP-Difícil, os estudos na área fortificaram-se em obter algoritmos aproximados. Um dos primeiros estudos foi realizado por Bafna e Pevzner [11], resultando em um algoritmo com um fator de aproximação 1.75. Em seguida, Christie [12] propôs um algoritmo com um fator de aproximação de 1.5. Atualmente, o melhor algoritmo conhecido para o problema possui fator de aproximação 1.375, e foi proposto por Berman e coautores [13].

## 4.2 Transposições

O primeiro algoritmo aproximado para o Problema de Ordenação de Permutações por Transposições foi apresentado por Bafna e Pevzner [14]. Esse algoritmo possui uma alto nível de complexidade na sua implementação e apresenta um fator de aproximação de 1.5 com uma complexidade de tempo de $\mathcal{O}(n^2)$. Ainda como contribuição do trabalho, apresentou-se uma estrutura em grafos chamada de Grafo de Ciclos, que permite a definição de limites inferiores e superiores para o problema. Em seguida, Christie [5] propôs um algoritmo mais simples de ser implementado do que o proposto por Bafna e Pevzner [14] e com o mesmo fator de aproximação de 1.5, mas com uma complexidade de $\mathcal{O}(n^4)$.

Posteriormente, Walter e coautores [15] desenvolveram um algoritmo com a mesma complexidade de tempo do que o proposto por Bafna e Pevzner [14], muito mais simples de ser implementado, mas com um fator de aproximação de 2.25. Atualmente, o melhor algoritmo para o problema de distância de transposição foi proposto por Elias e Hartman [16] e apresenta um fator de aproximação de 1.375, com uma complexidade de $\mathcal{O}(n^2)$, sendo uma melhoria significativa no fator de aproximação que não ocorria desde a publicação do trabalho de Bafna e Pevzner [14]. Rusu [17] mostrou que é possível implementar em tempo $\mathcal{O}(n \log n)$ uma série de algoritmos existentes na literatura utilizando uma estrutura chamada de *log-list*.

Em 2012, Bulteau e coautores [18] provaram que o problema de distância de transposição pertence à classe de problemas NP-Difíceis, resolvendo assim essa questão que permaneceu em aberto por aproximadamente 15 anos.

## 4.3 Reversões e Transposições

Para o Problema de Ordenação de Permutações por Reversões e Transposições Walter e coautores [1] apresentaram um algoritmo 2-aproximado para a variação com sinais do problema e um algoritmo 3-aproximado para a variação sem sinais, e ainda, limitantes para o diâmetro considerando a versão com sinais. Posteriormente, os mesmo autores apresentaram em outro trabalho [19] limitantes para a distância, considerando o caso com sinais.

Em 2008, Rahman e coautores [2] apresentaram limitantes para a distância considerando a variação sem sinais, e um algoritmo $2k$-aproximado, onde $k$ é o fator de aproximação do algoritmo utilizado para decomposição de ciclos. Dado o melhor valor de $k$ conhecido [20], o fator de aproximação deste algoritmo torna-se $2.8334 + \epsilon$, para qualquer $\epsilon > 0$.

## 4.4 Operações Curtas e Super Curtas

Jerrum [21], mostrou que o Problema de Ordenação de Permutação sem Sinais por Reversões Super Curtas pode ser resolvido em tempo polinomial de maneira ótima, tendo a prova sido realizada por Knuth [22]. Health e Vergara [23], apresentaram um trabalho voltado para o Problema de Ordenação de Permutação sem Sinais por Reversões Curtas, onde mostraram um algoritmo 2-aproximado. Em 1998, os mesmos autores [24] apresentaram um trabalho levando em consideração o Problema de Ordenação de Permutações por Transposições, considerando várias restrições em relação aos tamanhos dos blocos que podem ser afetados pelas transposições. Além disso, conjecturaram que o Problema de Ordenação de Permutações por Transposições Curtas pode ser resolvido em tempo polinomial. Em 2000, Heath e Vergara [25] apresentaram um algoritmo $\frac{4}{3}$-aproximado para o Problema de Ordenação de Permutações por Transposições Curtas.

Jiang e coautores [26] apresentaram um algoritmo $(1 + \epsilon)$-aproximado para Problema de Ordenação de Permutações por Transposições Curtas, onde $\epsilon$ é o número de elementos divido pelo número de inversões da permutação. Recentemente, Jiang e coautores [27] apresentaram um algoritmo $\frac{5}{4}$-aproximado para Problema de Ordenação de Permutações por Transposições Curtas. Finalmente, Vergara [28] mostrou que um algoritmo $\frac{4}{3}$-aproximado proposto por Health e ele [25] para o Problema de Ordenação de Permutações por Transposições Curtas, é um algoritmo 2-aproximado para o Problema de Ordenação de Permutações por Operações (Reversões e Transposições) Curtas.

Galvão e Dias [29] apresentaram inicialmente três algoritmos aproximados para o problema de ordenação de permutações com sinais por reversão curtas, sendo o melhor deles um algoritmo 9-aproximado. Posteriormente Galvão e coautores [30] apresentaram um trabalho onde abordaram quatro variações do problema de ordenação de permutações, sendo

eles: $(i)$ ordenação de permutações com sinais por reversões super curtas, $(ii)$ ordenação de permutações com sinais por reversões curtas, $(iii)$ ordenação de permutações com sinais por operações (reversões e transposições) super curtas e $(iv)$ ordenação de permutações com sinais por operações (reversões e transposições) curtas. Para as variações $(i)$ e $(iii)$ os autores apresentam um algoritmo exato em tempo polinomial, enquanto que para as variações $(ii)$ e $(iv)$ foram fornecidos, respectivamente, um algoritmo 5-aproximado e 3-aproximado.

Existe ainda o problema de ordenação de permutações circulares, que representam de forma mais adequada, por exemplo, genomas de vírus e bactérias. Buscando ser mais fiel possível ao cenário evolutivo dessas espécies, opta-se por utilizar operações super curtas. O Problema de Ordenação de Permutações Circulares por Reversões Super Curtas apresenta solução em tempo polinomial para sua versão sem sinais. Em 2015, Galvão e coautores [31] apresentaram um algoritmo em tempo polinomial para a versão com sinais do problema, e em 2016 os mesmo autores disponibilizaram uma ferramenta *web* [32] para inferência filogenética de rearranjo, considerando a comparação de genoma por reversões super curtas.

A Tabela 1 mostra o estado da arte para o problema de ordenação de permutações lineares considerando diferentes tipos de modelos de rearranjo.

| **Modelo de Rearranjo** | **Sinais** | **Complexidade** | **Melhor Solução Teórica** |
|---|---|---|---|
| Reversão | com | Polinomial [6] | Algoritmo exato $\mathcal{O}(n^{\frac{3}{2}})$ [8] |
| | sem | NP-Difícil [10] | Algoritmo 1.375-aproximado [13] |
| Transposição | sem | NP-Difícil [18] | Algoritmo 1.375-aproximado [16] |
| Reversão e Transposição | com | Desconhecida | Algoritmo 2-aproximado [1] |
| | sem | Desconhecida | Algoritmo 2k-aproximado [2] |
| Reversão Curta | com | Desconhecida | Algoritmo 5-aproximado [30] |
| | sem | Desconhecida | Algoritmo 2-aproximado [23] |
| Transposição Curta | sem | Desconhecida | Algoritmo 5/4-aproximado [27] |
| Reversão e Transposição Curta | com | Desconhecida | Algoritmo 3-aproximado [30] |
| | sem | Desconhecida | Algoritmo 2-aproximado [28] |
| Reversão Super Curta | com | Polinomial [30] | Algoritmo exato $\mathcal{O}(n^3)$ [30] |
| | sem | Polinomial [22, p. 108] | Algoritmo exato $\mathcal{O}(n^2)$ [21] |
| Transposição Super Curta | sem | Polinomial [22, p. 108] | Algoritmo exato $\mathcal{O}(n^2)$ [21] |
| Reversão e Transposição Super Curta | com | Polinomial [30] | Algoritmo exato $\mathcal{O}(n^3)$ [30] |
| | sem | Polinomial [22, p. 108] | Algoritmo exato $\mathcal{O}(n^2)$ [21] |

Tabela 1: Estado da arte para diferentes tipos de modelos de rearranjo.

# 5 Objetivos

O objetivo do projeto está focado em dois pontos, sendo eles o Problema de Ordenação de Permutações com Sinais por Reversões e Transposições e o Problema de Ordenação de Permutações por Operações Curtas. As subseções a seguir descrevem quais os objetivos determinados para cada um dos pontos abordados.

## 5.1 Heurísticas para o Problema de Ordenação de Permutações com Sinais por Reversões e Transposições

Em 2010, Dias e Dias [33] apresentaram uma heurística chamada de *look ahead*, o qual foi aplicado ao problema de rearranjo de genomas utilizando eventos de reversão e transposição. A heurística, de forma resumida, funciona aplicando uma busca em largura com uma profundidade definida na permutação atual em que o algoritmo está executando. A partir da busca em largura é escolhida a operação que parece ser mais promissora, que então é aplicada na permutação atual. O uso do *look ahead* apresentou bons resultados para um determinado conjunto de permutações, entretanto a complexidade do algoritmo está diretamente ligada a quantidade de níveis a serem realizados pela busca em largura.

Recentemente, em 2014, Galvão e Dias [34] desenvolveram uma ferramenta chamada *GRAAu*, que auxilia na análise dos resultados de algoritmos aproximados para rearranjo de genomas, essa ferramenta fornece a porcentagem de permutações que o algoritmo consegue ordenar de maneira ótima e calcula a média das soluções fornecidas comparadas com a ordenação ótima. Ainda no mesmo ano, Dias e coautores [35] publicaram um trabalho onde foi apresentado uma heurística chamada de *sliding window*, sendo aplicada nos resultados de algoritmos aproximados para ordenação de permutações sem sinais utilizando eventos de reversão e transposição. A heurística visa reduzir o número de eventos fornecido pelos algoritmos e, para isso, é utilizada uma base de dados que contém a menor sequência de eventos de reversão e transposição necessária para transformar qualquer permutação, de tamanho até dez, em outra. Inicialmente a heurística monta uma sequência de permutações a partir de uma sequência de eventos $S = [S^0, S^1, S^2, S^3, \ldots, S^w]$. O passo

seguinte da heurística é encontrar uma subsequência de permutações de $p$ até $q$, tal que $0 \leq p < q \leq w$, de modo que a permutação $\alpha = S_q^{-1} \cdot S_p$ possa ser reduzida a uma permutação $\alpha_{red}$ de tamanho menor ou igual a doze. Em seguida, é verificada a otimalidade de $S^{'} = [S_p, S_{p+1}, S_{p+2}, S_{p+3}, \ldots, S_q]$, consultando a distância da permutação $\alpha_{red}$ na base de dados de distância de rearranjo. Caso não seja uma solução ótima, $S^{'}$ é substituída pela sequência armazenada na base de dados, reduzindo o tamanho da sequência de eventos fornecida inicialmente. Uma vez que os valores de $p$ e $q$ são definidos o processo de repete, mas agora $p$ passa a valer $p+1$ e $q$ passa a valer $q+1$, dessa forma esse intervalo de tamanho de janela é verificado até o final da sequência. Uma ilustração do funcionamento da heurística pode ser vista na Figura 1.

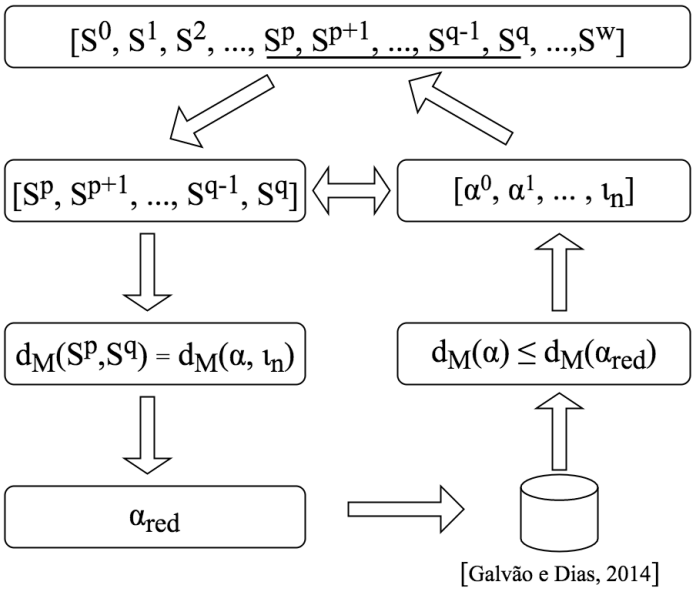



Figura 1: Exemplo da heurística *sliding window*.

No momento estamos desenvolvendo uma adaptação da heurística de *sliding window* para ser aplicada em resultados de algoritmos aproximados para ordenação de permutações com sinais utilizando eventos de reversão e transposição, complementando o trabalho realizado por Dias e coautores [35]. Posteriormente, pretende-se aplicar as heurísticas de *look ahead* e *sliding window* juntas, para que com os resultados possa ser feito um comparativo das duas abordagens utilizadas.

### 5.2 Problema de Ordenação de Permutações por Operações Curtas

Durante essa etapa, pretendemos estudar o problema de ordenação de permutações utilizando operações curtas de reversão e/ou transposição, com o objetivo de desenvolver algoritmos aproximados melhores do que os conhecidos na literatura. Pretendemos também estudar classes de permutações que possam ser ordenadas usando operações curtas em tempo polinomial.

## 6 Plano de Trabalho

| | 2016 | | | | | | | | | | 2017 | | | | | | | | | | | 2018 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 2: Cronograma de atividades.

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 de Mestrado;
5. Participação no Programa de Estágio Docente (PED);
6. Desenvolvimento da heurística de *sliding window* e execução do experimento;
7. Desenvolvimento da heurística de *look ahead* e execução do experimento;

8. Investigação de problemas de ordenação de permutações sem sinais utilizando operações curtas;

9. Investigação de problemas de ordenação de permutações com sinais utilizando operações curtas;

10. Escrita da dissertação;

11. Revisão da dissertação;

12. Defesa da dissertação;

Vale ressaltar que o tempo alocado nas atividades previstas no cronograma pode sofrer alterações no decorrer do desenvolvimento da pesquisa, uma vez que alguns resultados obtidos podem ser mais promissores que outros, fazendo com que mais tempo seja despendido em uma atividade em detrimento de outra.

# 7 Materiais e Métodos

A primeira etapa do trabalho terá um aspecto mais prático. O código fonte para gerar a base de dados de rearranjo, levando em consideração os eventos de reversão e transposição em permutações com e sem sinais foi disponibilizado por Dias e coautores [35]. O desenvolvimento das atividades será focado na implementação de algoritmos aproximados para ordenação de permutações com sinais utilizando eventos de reversão e transposição, bem como a heurística de *sliding window* aplicada em permutações com sinais e a aplicação da heurística de *look ahead* nos algoritmos aproximados, quando possível.

A segunda etapa é, em grande parte, teórica. O estudo de teoremas e provas de trabalhos já existentes na literatura são a principal base para o estabelecimento de limitantes e das classes de permutações específicas, mas sempre que possível e necessário, programas serão desenvolvidos para validar os resultados obtidos.

# 8 Forma de Análise dos Resultados

Os resultados da primeira etapa do projeto podem ser verificados de maneira simples. A etapa, que consiste em aplicar a heurística de *sliding window* em permutações com sinais utilizando eventos de reversão e transposição, pode ter os resultados avaliados ao comparar a quantidade de eventos utilizados pela heurística *slide window* e a quantidade de eventos utilizados pelos algoritmos aproximados. Com os resultados obtidos, é possível realizar uma análise quantitativa, comparando os resultados com outros algoritmos disponíveis na literatura [1, 2]. De maneira similar, pode ser feita a análise dos resultados obtidos pelas heurísticas de *sliding window* e *look ahead* juntas.

Na segunda etapa do projeto, os resultados podem ser comparados com os fatores de aproximação fornecidos pelos trabalhos existentes na literatura, bem como os limitantes para as classes de permutações estudadas.

# Referências


[1] Maria E. M. T. Walter, Zanoni Dias, and João 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, Santa Cruz de La Sierra, Bolivia, 1998. IEEE Computer Society.

[2] Atif Rahman, Swakkhar Shatabda, and Masud Hasan. An Approximation Algorithm for Sorting by Reversals and Transpositions. *Journal of Discrete Algorithms*, 6(3):449–457, 2008.

[3] Ron Y. Pinter and Steven Skiena. Genomic Sorting with Length-Weighted Reversals. *Genome Informatics*, 13:2002, 2002.

[4] Firas Swidan, Michael A. Bender, Dongdong Ge, Simai He, Haodong Hu, and Ron Y. Pinter. Sorting by Length-Weighted Reversals: Dealing with Signs and Circularity. In S. Sahinalp, S. Muthukrishnan, and U. Dogrusoz, editors, *Combinatorial Pattern*

*Matching*, volume 3109 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin Heidelberg, Heidelberg, Germany, 2004.

[5] David A. Christie. *Genome Rearrangement Problems*. PhD thesis, Department of Computing Science, University of Glasgow, 1998.

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

[7] Anne Bergeron. A Very Elementary Presentation of the Hannenhalli-Pevzner Theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.

[8] Eric Tannier, Anne Bergeron, and Marie-France Sagot. Advances on Sorting by Reversals. *Discrete Applied Mathematics*, 155(6-7):881–888, 2007.

[9] David A. Bader, Bernard M. E. Moret, and Mi Yan. A Linear-Time Algorithm for Computing Inversion Distance Between Signed Permutations with an Experimental Study. *Journal of Computational Biology*, 8:483–491, 2001.

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

[11] Vineet Bafna and Pavel A. Pevzner. Genome Rearrangements and Sorting by Reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[12] David A. Christie. A 3/2-Approximation Algorithm for Sorting by Reversals. In H. Karloff, editor, *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'1998)*, pages 244–252, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[13] Piotr Berman, Sridhar Hannenhalli, and Marek 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.

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

[15] Maria E. M. T. Walter, Zanoni Dias, and João Meidanis. A New Approach for Approximating The Transposition Distance. In F. M. Titsworth, editor, *Proceedings of the 7th String Processing and Information Retrieval (SPIRE'2000)*, pages 199–208, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

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

[17] Irena Rusu. Log-Lists and Their Applications to Sorting by Transpositions, Reversals and Block-Interchanges. `http://arxiv.org/abs/1507.01512`, 2015.

[18] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by Transpositions is Difficult. *SIAM Journal on Computing*, 26(3):1148–1180, 2012.

[19] João Meidanis, Maria E. M. T. Walter, and Zanoni Dias. A Lower Bound on the Reversal and Transposition Diameter. *Journal of Computational Biology*, 9(5):743–745, 2002.

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

[21] Mark R. Jerrum. The Complexity of Finding Minimum-length Generator Sequences. *Theoretical Computer Science*, 36(2-3):265–289, 1985.

[22] Donald E. Knuth. *Fundamental Algorithms: The Art of Computer Programming*. 1973.

[23] Lenwood S. Heath and John Paul C. Vergara. Sorting by Short Swaps. *Journal of Computational Biology*, 10(5):775–789, 2003.

[24] Lenwood S. Heath and John Paul C. Vergara. Sorting by Bounded Block-moves. *Discrete Applied Mathematics*, 88(1-3):181–206, 1998.

[25] Lenwood S. Heath and John Paul C. Vergara. Sorting by Short Block-Moves. *Algorithmica*, 28(3):323–352, 2000.

[26] Haitao Jiang, Daming Zhu, and Binhai Zhu. A (1+e)-Approximation Algorithm for Sorting by Short Block-Moves. *Theoretical Computer Science*, 437:1–8, 2012.

[27] Haitao Jiang, Haodi Feng, and Daming Zhu. An 5/4-Approximation Algorithm for Sorting Permutations by Short Block Moves. In H. Ahn and C. Shin, editors, *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC'2014)*, Lecture Notes in Computer Science, pages 491–503. Springer International Publishing, 2014.

[28] John Paul C. Vergara. *Sorting by Bounded Permutations*. PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[29] Gustavo R. Galvão and Zanoni Dias. Approximation Algorithms for Sorting by Signed Short Reversals. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB'2014)*, pages 360–369, New York, NY, USA, 2014. ACM.

[30] Gustavo R. Galvão, Orlando Lee, and Zanoni Dias. Sorting Signed Permutations by Short Operations. *Algorithms for Molecular Biology*, 10(1):1–17, 2015.

[31] Gustavo R. Galvão, Christian Baudet, and Zanoni Dias. Sorting Signed Circular Permutations by Super Short Reversals. In R. Harrison, Y. Li, and I. Măndoiu, editors, *Proceedings of the 11th International Symposium on Bioinformatics Research and Applications (ISBRA'2015)*, Lecture Notes in Computer Science, pages 272–283. Springer International Publishing, Switzerland, 2015.

[32] Gustavo R. Galvão, Christian Baudet, and Zanoni Dias. Sorting Circular Permutations by Super Short Reversals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2016.

[33] Ulisses Dias and Zanoni Dias. Extending Bafna-Pevzner Algorithm. In *Proceedings of the 1st International Symposium on Biocomputing (ISB'2010)*, pages 1–8, New York, NY, USA, 2010. ACM.

[34] Gustavo R. Galvão and Zanoni Dias. An Audit Tool for Genome Rearrangement Algorithms. *Journal of Experimental Algorithmics*, 19:1–34, 2014.

[35] Ulisses Dias, Gustavo R. Galvão, Carla N. Lintzmayer, and Zanoni Dias. A General Heuristic for Genome Rearrangement Problems. *Journal of Bioinformatics and Computational Biology*, 12(3):26, 2014.