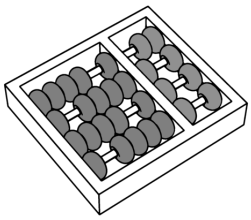
Universidade Estadual de Campinas

Instituto de Computação



Exame de Qualificação Específico de Doutorado

Problemas de Ordenação de Permutações por Reversões e Transposições

Candidato:
Andre Rodrigues Oliveira

Orientador:
Prof. Dr. Zanoni Dias

Coorientador:
Prof. Dr. Ulisses Dias

06 de outubro de 2016

Resumo

Rearranjos de Genomas são eventos que afetam grandes trechos dos genomas. Problemas em Rearranjo de Genomas buscam calcular o número mínimo de eventos de rearranjo necessários para transformar um genoma em outro, baseando-se no princípio que tal número é uma boa aproximação para a distância evolutiva entre eles. Ao representarmos os genomas por meio de permutações e assumirmos que um dos genomas é a permutação identidade, temos o Problema da Ordenação de Genomas. Um modelo de rearranjo determina quais eventos de rearranjo são permitidos para ordenar uma permutação ou transformar uma permutação em outra. Dentre os eventos de rearranjo mais comuns, temos a reversão e a transposição. Modelos considerando os dois eventos citados acima, separadamente, já possuem vasta bibliografia. Entretanto, modelos considerando os dois eventos em conjunto ainda foram pouco explorados. Nossa meta será estudar a versão do problema que permite o uso de reversões e transposições bem como problemas diretamente relacionados. Esta proposta apresenta alguns conceitos iniciais e resultados já existentes sobre estes problemas, bem como especifica o objetivo do nosso trabalho e apresenta os resultados iniciais obtidos.

1 Introdução

Rearranjo de genomas é uma área da Biologia Computacional que aborda o processo de evolução entre espécies. Problemas de rearranjo de genomas buscam, dados dois genomas, encontrar uma sequência de rearranjos que transformam um genoma no outro. Diferente das mutações pontuais, que afetam individualmente moléculas constituintes do genoma, os eventos de rearranjo afetam grandes porções do genoma, tornando esta abordagem mais adequada para comparação de genomas completos.

O genoma de uma espécie é composto de cromossomos representados como um conjunto ordenado de genes. Outra forma de representação dos cromossomos se dá por meio de blocos conservados, sendo esses blocos regiões de alta similaridade entre os dois genomas comparados. Estes blocos podem ser genes ou qualquer subsequência conservada em ambos os genomas. Um problema de rearranjo de genomas consiste então em comparar genomas de dois indivíduos e encontrar a menor sequência de eventos de rearranjo que transformam um

genoma em outro. O tamanho desta sequência pode ser utilizado para estimar a distância evolucionária ou mesmo o grau de parentesco entre os dois genomas comparados e parte do princípio que rearranjos de genomas são eventos relativamente raros e que geralmente causam deficiências nos organismos. Logo, a menor sequência de eventos de rearranjo que explica a transformação entre dois genomas tende a ser a mais provável, com base em uma teoria conhecida como *Princípio da Máxima Parcimônia*.

Um *evento de rearranjo* ocorre com a quebra de um ou mais cromossomos e os segmentos resultantes são unidos, de forma que o conjunto de genes permanece o mesmo, porém a ordem é alterada. Além disso, a orientação destes segmentos também pode ser alterada. Na literatura, foram propostos diversos eventos ou operações de rearranjo de genomas, dentre os quais podemos citar a reversão, quando um segmento do genoma é destacado e inserido no mesmo local, porém com ordem invertida em relação à sua posição original, e a transposição, quando um segmento do genoma é destacado e inserido em outra posição sem alterar a ordem deste segmento. A Figura 1 mostra a aplicação de uma reversão seguida de uma transposição em uma sequência de blocos conservados.

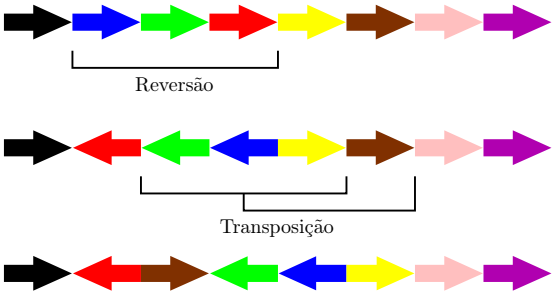


Figura 1: Exemplo de uma reversão e uma transposição aplicadas em uma sequência de blocos conservados.

Um *modelo de rearranjo* determina o conjunto de operações permitidas para transformar um genoma em outro. Estudos iniciais em rearranjos de genomas focaram em modelos de rearranjo que permitiam apenas um tipo de operação, capazes de explicar cenários evolutivos simples. Mais tarde, surgiram modelos que permitiam duas ou mais operações, possibilitando assim a análise de ce-

nários evolutivos mais complexos. Esta proposta foca no estudo de problemas de ordenação de permutações quando o modelo de rearranjo permite que duas operações sejam realizadas: reversões e transposições.

O restante desta proposta está organizado da seguinte forma. A Seção 2 apresenta os conceitos básicos importantes para problemas de rearranjo de genomas. A Seção 3 apresenta o Problema da Ordenação de Permutações por Reversões e Transposições e suas variações. A Seção 4 descreve os objetivos do trabalho a ser realizado. A Seção 5 descreve o trabalho a ser desenvolvido no exterior. A Seção 6 apresenta o cronograma para a realização desta proposta. A Seção 7 apresenta a metodologia que utilizaremos neste trabalho, bem como a forma de análise dos resultados. Por fim, a Seção 8 apresenta os resultados iniciais deste trabalho.

2 Fundamentação Teórica

Em Problemas de Rearranjos de Genomas, um genoma é representado como uma n-tupla cujos elementos representam os genes. Supondo que não haja repetição de genes, esta n-tupla é uma permutação $\pi = (\pi_1\ \pi_2\ ...\ \pi_n)$, com $\pi_i \in \{-n, -(n-1), ..., -2, -1, +1, +2, ..., +(n-1), +n\}$ e $|\pi_i| \neq |\pi_j| \leftrightarrow i \neq j$. Nesse caso, cada elemento π_i possui um sinal, + ou −, que indica a orientação do gene que ele representa, e a permutação é dita *com sinais*. Quando não há informação sobre a orientação dos genes, esse sinal é omitido, e a permutação é dita *sem sinais*.

Note que uma permutação pode representar tanto genomas lineares quanto genomas circulares. Nesta proposta utilizaremos a mesma representação linear para ambos os casos, entretanto, se π representa um genoma circular, temos, além das adjacências (π_i,π_{i+1}), para $1 \leq i < n$, a adjacência (π_n, π_1). Quando não explicitado no texto, estaremos nos referindo a permutações lineares.

Seja ι a *permutação identidade* definida como $\iota = (1\ 2\ ...\ n{-}1\ n)$. Na permutação identidade com sinais, todos os elementos possuem sinal positivo.

Dadas duas permutações π e σ, a *composição* entre elas, denotada por $\pi \cdot \sigma$, é $\pi \cdot \sigma = (\pi_{\sigma_1}\ \pi_{\sigma_2}\ ...\ \pi_{\sigma_n})$, de forma similar à composição de funções.

A operação de composição induz uma estrutura de grupo sobre o conjunto formado por todas as permutações de um determinado tamanho. O grupo

formado por todas as $n!$ permutações sem sinais de tamanho n juntamente com a operação de composição é chamado de *grupo simétrico* e é denotado por S_n. O grupo formado por todas as $n!2^n$ permutações com sinais de tamanho n juntamente com a operação de composição é chamado de *grupo simétrico com sinais* e é denotado por $S_n^{\pm}$.

Dadas duas permutações π e σ e um modelo de rearranjo M, o problema de transformar a permutação π na permutação σ consiste em encontrar a menor sequência de operações $\rho_1, \rho_2, ..., \rho_k$ pertencentes a M tal que $\pi \cdot \rho_1 \cdot \rho_2 \cdot ... \cdot \rho_k = \sigma$. O tamanho desta sequência representa a *distância* entre π e σ com respeito ao modelo M e é denotada por $d_M(\pi, \sigma)$ (neste caso, $d_M(\pi, \sigma) = k$).

Da mesma forma, Problemas de Ordenação por Rearranjos consistem em aplicar a menor sequência de operações $\rho_1, \rho_2, ...\rho_k$ permitidas pelo modelo em uma permutação π tal que $\pi \cdot \rho_1 \cdot \rho_2 \cdot ... \cdot \rho_k = \iota$. A *distância de ordenação* de uma permutação π é o tamanho da menor sequência que transforma π na permutação identidade, e a distância entre elas com respeito a um modelo M é denotada por $d_M(\pi, \iota) = d_M(\pi)$.

Note que $d_M(\pi, \sigma) = d_M(\alpha)$ se tomarmos $\alpha = \sigma^{-1} \cdot \pi$, onde σ^{-1} é a inversa de σ, ou seja, é a permutação tal que $\sigma^{-1} \cdot \sigma = \iota$ De fato, temos que $d_M(\pi, \sigma) = d_M(\sigma^{-1} \cdot \pi, \sigma^{-1} \cdot \sigma) = d_M(\alpha, \iota) = d_M(\alpha)$. Por exemplo, se $\pi = (1\ 3\ 2\ 5\ 4)$ e $\sigma = (2\ 4\ 5\ 1\ 3)$, então $\sigma^{-1} = (4\ 1\ 5\ 2\ 3)$, $\sigma^{-1} \cdot \sigma = (1\ 2\ 3\ 4\ 5) = \iota$, e $\alpha = \sigma^{-1} \cdot \pi = (5\ 1\ 4\ 3\ 2)$. Desta forma, a distância entre π e σ será a mesma que a distância de ordenação da permutação α.

A maior distância possível entre todas as permutações no mesmo grupo simétrico S_n (ou $S_n^{\pm}$ caso π seja uma permutação com sinais) com respeito ao modelo M é chamada de *diâmetro*, e é denotada por $D_M(n)$.

A *permutação estendida* de uma permutação π pode ser obtida a partir de π inserindo-se dois novos elementos: $\pi_0 = 0$ e $\pi_{n+1} = n + 1$. A partir deste momento, a não ser que explicitado, estaremos sempre nos referindo à versão estendida de qualquer permutação, mesmo quando estes elementos extras forem omitidos.

2.1 Breakpoints

Um *breakpoint de reversão* ocorre entre um par de elementos π_i e π_{i+1} de uma permutação π sem sinais, com $0 \leq i \leq n$, se $|\pi_i - \pi_{i+1}| \neq 1$. A permutação identidade é a única permutação que não possui breakpoints de reversão. Por exemplo, se $\pi = (0\ 3\ 4\ 1\ 2\ 5)$, existe um breakpoint entre os elementos $(0, 3)$, $(4, 1)$ e $(2, 5)$. O número de breakpoints de reversão em uma permutação π é denotado por $b_r(\pi)$. No exemplo acima, temos que $b_r(\pi) = 3$.

Um *breakpoint de reversão com sinal* ou um *breakpoint de transposição* ocorre entre um par de elementos π_i e π_{i+1}, com $0 \leq i \leq n$ se $\pi_{i+1} - \pi_i \neq 1$. A permutação identidade é a única permutação que não possui breakpoints de transposição nem breakpoints de reversão com sinal. O número de breakpoints de reversão com sinal em uma permutação com sinais π é denotado por $b_{\overline{r}}(\pi)$. O número de breakpoints de transposições em uma permutação π é denotado por $b_t(\pi)$. Por exemplo, a permutação com sinais $\pi = (+0\ +1\ -2\ +3\ +4\ -5\ +6)$ possui breakpoints de reversão com sinal entre os elementos $(+1, -2)$, $(-2, +3)$, $(+4, -5)$ e $(-5, +6)$ e, desta forma, $b_{\overline{r}}(\pi) = 4$. Já a permutação $\sigma = (0\ 1\ 2\ 4\ 3\ 5\ 6)$ possui breakpoints de transposição entre os elementos $(2, 4)$, $(4, 3)$ e $(3, 5)$ e, desta forma, $b_t(\sigma) = 3$.

2.2 Strips

Dado um tipo de breakpoint, que pode ser de reversão ou de transposição, uma *strip* de uma permutação π é uma sequência de elementos $\pi_i ... \pi_j$, com $0 \leq i \leq j \leq n + 1$, tal que não existe um breakpoint entre os pares (π_k, π_{k+1}), com $i \leq k \leq j - 1$.

Uma strip é dita *crescente* caso seus elementos formem uma sequência crescente e é dita *decrescente* caso contrário. Se a strip contém apenas um elemento, ela é chamada de *singleton* e definida como decrescente. Por exemplo, dada a permutação $\pi = (0\ 3\ 4\ 5\ 2\ 1\ 6)$ e considerando breakpoints de reversão, temos que π possui 4 strips: as strips $\langle 0 \rangle$ e $\langle 6 \rangle$, que são singletons, a strip $\langle 3\,4\,5 \rangle$ que é crescente e a strip $\langle 2\,1 \rangle$ que é decrescente. Se considerarmos breakpoints de transposição temos que π possui 5 strips: as strips $\langle 0 \rangle$, $\langle 2 \rangle$, $\langle 1 \rangle$, e $\langle 6 \rangle$, que são singletons, e a strip $\langle 3\,4\,5 \rangle$, que é crescente.

2.3 Inversões

Dado π, existe uma *inversão* entre dois elementos π_i e π_j se $|\pi_i| > |\pi_j|$ e $i < j$, ou seja, existe um elemento π_j à direita de π_i cujo valor em módulo é menor que ele. Denotamos por $inv(\pi, \pi_i)$ o número de inversões que π_i participa em π. O número de inversões em uma permutação π é dado por $inv(\pi) = \sum_{i=1}^{|\pi|} inv(\pi, \pi_i)$. Por exemplo, os elementos da permutação $\pi = (2\ 5\ 3\ 4\ 1)$ possuem os seguintes números de inversões: $inv(\pi, 2) = 1$, pois o elemento 1 está à direita de 2, $inv(\pi, 5) = 3$, pois os elementos 3, 4 e 1 estão à direita de 5, $inv(\pi, 3) = 1$, pois o elemento 1 está à direita de 3, $inv(\pi, 4) = 1$, pois o elemento 1 está à direita de 4, e $inv(\pi, 1) = 0$, pois não existem elementos à direita de 1 menores que ele. Desta forma, $inv(\pi) = 6$.

2.4 Grafo de Ciclos

O *grafo de ciclos* $G(\pi)$ de uma permutação com sinais π é um grafo formado pelo conjunto de vértices $V(\pi)$, o conjunto de arestas pretas $E_p(\pi)$ e o conjunto de arestas cinzas $E_c(\pi)$ tais que:

- $V(\pi) = \{0, -\pi_1, +\pi_1, -\pi_2, +\pi_2, ..., -\pi_n, +\pi_n, (-n+1)\}$
- $E_c(\pi) = \bigcup_{i=1}^{n+1} \{(+(i-1), -i)\}$
- $E_p(\pi) = \bigcup_{i=1}^{n+1} \{(-\pi_i, +\pi_{i-1})\}$

Nesse caso, todo vértice de $G(\pi)$ é incidente a exatamente uma aresta cinza e uma aresta preta. Isto implica na existência de uma decomposição única das arestas de $G(\pi)$ em ciclos com arestas de cores alternantes, chamados de *ciclos alternantes*.

Quando π é uma permutação sem sinais, podemos mapeá-la para uma permutação π' com sinais atribuindo sinais arbitrários para cada elemento π_i. No final, associamos $G(\pi')$ à π. Note que, por este motivo, podemos mapear π para vários grafos de ciclos diferentes. Como queremos ordenar permutações, o melhor grafo seria aquele que maximiza o número de ciclos, uma vez que a permutação identidade ι é a permutação cujo grafo de ciclos possui o maior número de ciclos possível, com $n + 1$ ciclos. Entretanto, encontrar

o grafo que maximiza o número de ciclos é um problema NP-Difícil [10] e, por este motivo, utilizamos algoritmos aproximados para encontrar grafos de ciclos para permutações sem sinais. O melhor fator de aproximação conhecido é $1.4167 + \epsilon$, para $\epsilon > 0$, proposto por Chen [12]. A Figura 2 mostra dois grafos de ciclos diferentes que podem ser associados à permutação sem sinais $\pi = (2\ 1\ 3\ 6\ 4\ 5)$.

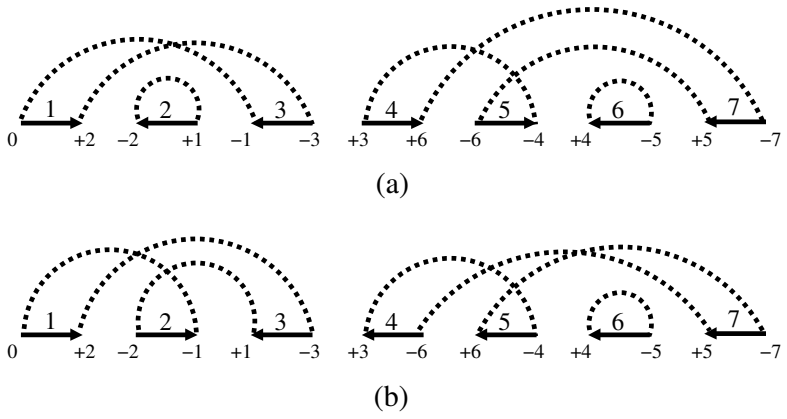


Figura 2: Dois grafos de ciclos diferentes que podem ser mapeados para a permutação sem sinais $\pi = (2\ 1\ 3\ 6\ 4\ 5)$, sendo (a) o grafo de ciclos da permutação com sinais $\pi'_1 = (-2\ -1\ +3\ -6\ +4\ +5)$, e (b) o grafo de ciclos da permutação com sinais $\pi'_2 = (-2\ +1\ +3\ +6\ +4\ +5)$.

Dado o grafo de ciclos $G(\pi)$ de uma permutação π, as arestas pretas são rotuladas de 1 a $n+1$, de modo que a aresta $(-\pi_i, +\pi_{i-1})$ recebe o rótulo i.

Um ciclo c é representado pela listagem dos vértices na ordem em que aparecem no ciclo. Se $c = (v_1, v_2, ..., v_k)$ é um ciclo, assume-se que v_1 é o vértice localizado mais à direita. A representação simplificada lista o ciclo em ordem dos rótulos das arestas pretas $\{(v_1, v_2), (v_3, v_4), ...\}$ e associamos a estes rótulos os sinais '+' e '−' para indicar como as arestas pretas são percorridas. Se uma aresta preta é percorrida da direita para a esquerda, então o sinal '+' é associado a seu rótulo e dizemos que esta aresta preta é positiva. Caso contrário, o sinal '−' é associado a seu rótulo e dizemos que a aresta preta é negativa. A Figura 3 mostra exemplos de arestas pretas positivas e negativas. Dado que v_1 é o vértice mais à direita do ciclo, convencionamos que a aresta (v_1, v_2) é sempre percorrida da direita para a esquerda, e, desta

forma, a primeira aresta preta de um ciclo será sempre positiva. Pelo mesmo motivo, todo 1-ciclo é sempre positivo.

Um ciclo alternante é chamado de *k-ciclo* se ele possui k arestas pretas. Além disso, um k-ciclo é dito *par* se k é par. Caso contrário, o k-ciclo é dito *ímpar*. Seja $c(\pi)$ o número de ciclos de $G(\pi)$, e seja $c_{impar}(\pi)$ o número de ciclos ímpares de $G(\pi)$. A permutação ι é a única cujo grafo de ciclos alternantes possui $n+1$ ciclos, sendo todos ímpares. A Figura 3 mostra o grafo de ciclos da permutação com sinais $\pi = (+6\ +2\ -1\ -5\ +4\ +3)$ contendo os ciclos $c_1 = (7, 5, -6, -3, 2)$ e $c_2 = (4, -1)$, sendo c_1 um ciclo ímpar, já que é um 5-ciclo, e c_2 um ciclo par, já que é um 2-ciclo.

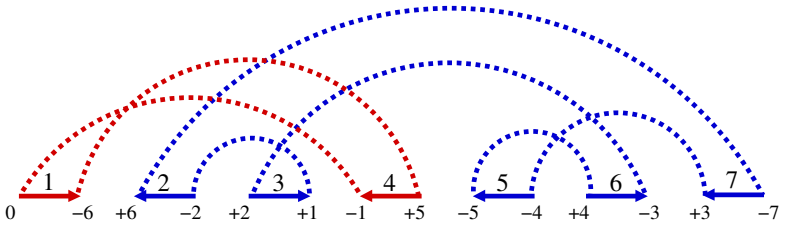


Figura 3: Grafo de ciclos da permutação com sinais $\pi = (+6\ +2\ -1\ -5\ +4\ +3)$, contendo os ciclos $c_1 = (7, 5, -6, -3, 2)$ e $c_2 = (4, -1)$.

Um ciclo do grafo de ciclos é chamado de *curto* caso contenha no máximo 3 arestas pretas, e *longo* caso contrário. Uma permutação é dita *simples* se seu grafo de ciclos possui apenas ciclos curtos. Para o grafo de ciclos da permutação $\pi = (+6\ +2\ -1\ -5\ +4\ +3)$ na Figura 3, temos que o ciclo $c_1 = (7, 5, -6, -3, 2)$ é um ciclo longo, uma vez que é um 5-ciclo, e $c_2 = (4, -1)$ é um ciclo curto, pois é um 2-ciclo. Como este grafo de ciclos possui um ciclo longo, dizemos que π não é simples.

É possível transformar uma permutação não simples π em uma permutação simples $\hat{\pi}$, adicionando novos elementos de forma a quebrar os ciclos longos de π. Esta abordagem de quebrar ciclos longos é amplamente utilizada na bibliografia, tanto na ordenação de permutações por transposições [17], quanto na ordenação de permutações por reversões [25]. Temos então que $\hat{\pi}$ é obtida através da inserção de novos elementos em π e, desta forma, $d(\pi) \leq d(\hat{\pi})$, uma vez que inserir novos elementos em uma permutação pode resultar em uma

permutação que requer mais operações para ser ordenada. A Figura 4 mostra a transformação da permutação $\pi = (+4\ +3\ -2\ +1)$ em uma permutação simples $\hat{\pi}$. Ao rotular $\hat{\pi}$ de acordo com o novo grafo de ciclos obtemos a permutação $\hat{\pi} = (+5\ +4\ -2\ -3\ +1)$. Note que $|\hat{\pi}| = |\pi| + 1$, $G(\hat{\pi})$ possui uma aresta preta a mais que $G(\pi)$ e $c(\hat{\pi}) = c(\pi) + 1$.

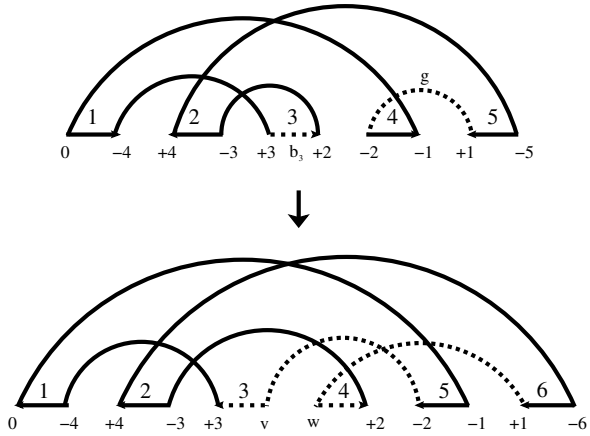


Figura 4: Transformação da permutação $\pi = (+4\ +3\ -2\ +1)$ em uma permutação simples $\hat{\pi} = (+5\ +4\ -2\ -3\ +1)$.

Uma *configuração* C é um grafo contendo arestas pretas e arestas cinzas tal que todo vértice de C é incidente a exatamente uma aresta cinza e uma aresta preta. Ao contrário do grafo de ciclos, uma configuração pode não ter uma permutação com sinais associada a ele. Sendo assim, temos que todo grafo de ciclos é uma configuração, mas nem toda configuração é um grafo de ciclos. A *configuração complementar* $\overline{C}$ é construída a partir de C removendo todas as arestas pretas e inserindo as *arestas complementares*, que ligam os vértices x e $-x$ para todo $x \in \{1...n\}$. Dado um grafo G, dizemos que ele possui um *ciclo hamiltoniano* caso exista um caminho circular passando por todos os vértices de G não repetindo nenhum. Uma configuração C é também um grafo de ciclos se $\overline{C}$ possui apenas um ciclo hamiltoniano. A Figura 5 mostra uma configuração C para a qual não é possível associar uma permutação π, uma vez que $\overline{C}$ possui dois ciclos.

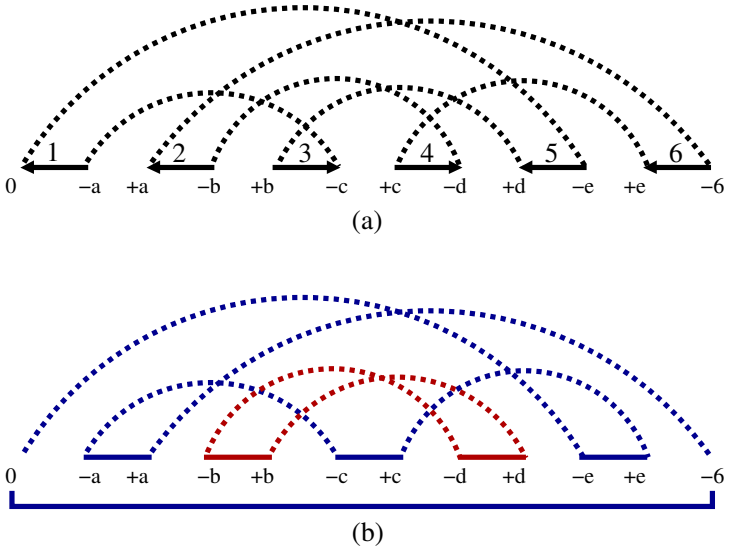


Figura 5: A configuração C, ilustrada em (a), contendo os ciclos $c_1 = (6, -4, 2)$ e $c_2 = (5, -3, 1)$ é tal que $\overline{C}$, ilustrada em (b) possui dois ciclos, destacados em vermelho e azul, e, desta forma, C não possui uma permutação π associada à ela.

2.5 Reversão

Uma reversão $\rho_r(i, j)$ é uma operação que inverte a ordem e a orientação dos elementos entre os pontos i e j de uma permutação.

Definição 1. *Uma reversão $\rho_r(i, j)$, aplicada à permutação com sinais $\pi = (\pi_1 \ ... \ \pi_{i-1} \ \pi_i \ ... \ \pi_j \ \pi_{j+1} \ ... \ \pi_n)$, com $1 \leq i \leq j \leq n$, gera a permutação $\pi \cdot \rho_r(i, j) = (\pi_1 \ ... \ \pi_{i-1} \ \underline{-\pi_j \ ... \ -\pi_i} \ \pi_{j+1} \ ... \ \pi_n)$, ou seja, ocorre a inversão do segmento $\pi_i ... \pi_j$ de π e também a inversão dos sinais dos elementos deste segmento. Uma reversão $\rho_r(i, j)$, aplicada à permutação sem sinais $\pi = (\pi_1 \ ... \ \pi_{i-1} \ \pi_i \ ... \ \pi_j \ \pi_{j+1} \ ... \ \pi_n)$, com $1 \leq i < j \leq n$, gera a permutação $\pi \cdot \rho_r(i, j) = (\pi_1 \ ... \ \pi_{i-1} \ \underline{\pi_j \ ... \ \pi_i} \ \pi_{j+1} \ ... \ \pi_n)$, ou seja, ocorre apenas a inversão do segmento $\pi_i ... \pi_j$ de π.*

Quando não se conhece a orientação dos genes, temos o *Problema de Ordenação por Reversões*, que foi provado ser NP-Difícil por Caprara [10]. Além disso, Berman e Karpinski [6] mostraram que este problema não é aproximável por um fator menor que 1.0008.

Kececioglu e Sankoff [34] em 1995 apresentaram o primeiro algoritmo de aproximação, com fator 2, consistindo basicamente em remover a maior quantidade de breakpoints a cada passo, dando prioridade para as reversões que ainda deixam strips decrescentes. Bafna e Pevzner [2] introduziram a ideia do *grafo de breakpoints* e mostraram um algoritmo de aproximação com fator 1.75. Posteriormente, Christie [13] apresentou um algoritmo de aproximação com fator 1.5. Atualmente, o melhor algoritmo conhecido para o problema possui fator de aproximação 1.375, proposto por Berman e coautores [5] em 2002.

Note que uma reversão $\rho_r(i,j)$ corta dois pontos da permutação: entre (π_{i-1}, π_i) e (π_j, π_{j+1}). Sendo assim, uma reversão pode criar ou remover no máximo 2 breakpoints, bem como deixar o número de breakpoints inalterado e, portanto, $\Delta b_r(\pi, \rho_r) = b_r(\pi \cdot \rho_r) - b_r(\pi)$ e $\Delta b_r(\pi, \rho_r) \in \{-2, -1, 0, 1, 2\}$. Um limitante inferior para a distância, proposto por Bafna e Pevzner [2], pode ser derivado dessa observação: $d_r(\pi) \geq \frac{b_r(\pi)}{2}$. Além disso, Kececioglu e Sankoff [34] mostraram um algoritmo que ordena uma permutação com até $b_r(\pi) - 1$ reversões e, desta forma, temos como limitante superior $d_r(\pi) \leq b_r(\pi) - 1$. Já em relação ao diâmetro, Bafna e Pevzner [2] provaram que $D_r(n) = n - 1$.

Para o caso em que a orientação dos genes é conhecida, denominado *Problema de Ordenação por Reversões com Sinal*, existem algoritmos polinomiais exatos, sendo que o primeiro foi apresentado por Hannenhalli e Pevzner [24], com complexidade $O(n^4)$. Atualmente, o algoritmo mais eficiente disponível na literatura possui complexidade de tempo $O(n^{\frac{3}{2}})$ [44]. Para o caso em que se deseja saber apenas o valor da distância de reversão de permutações, há também um algoritmo com complexidade de tempo $O(n)$ [1].

2.6 Transposição

Uma transposição é um evento de rearranjo que tem a propriedade de trocar dois blocos adjacentes de lugar.

Definição 2. *A transposição $\rho_t(i,j,k)$, com $1 \leq i < j < k \leq n+1$, aplicada à permutação $\pi = (\pi_1 \ ... \ \pi_{i-1} \ \underline{\pi_i \ ... \ \pi_{j-1}} \ \underline{\pi_j \ ... \ \pi_{k-1}} \ \pi_k \ ... \ \pi_n)$, gera a permutação $\pi \cdot \rho_t(i,j,k) = (\pi_1 \ ... \ \pi_{i-1} \ \underline{\pi_j \ ... \ \pi_{k-1}} \ \underline{\pi_i \ ... \ \pi_{j-1}} \ \pi_k \ ... \ \pi_n)$, ou seja, o segmento $\pi_i \pi_{i+1} ... \pi_{j-1}$ é removido e inserido após o segmento*

$\pi_j\pi_{j+1}...\pi_{k-1}$.

O primeiro algoritmo de aproximação para o *Problema de Ordenação por Transposições* foi apresentado por Bafna e Pevzner [3], com complexidade de tempo $O(n^2)$ e fator de aproximação 1.5. Dentre as contribuições do trabalho de Bafna e Pevzner, é possível citar a estrutura de grafo de ciclos introduzida na Seção 2.4. Esta estrutura permite a definição de limitantes inferiores e superiores mais adequados para o problema da distância de transposição.

Christie [14] propôs um algoritmo com fator de aproximação 1.5 mais simples que o de Bafna e Pevzner, mas com complexidade $O(n^4)$. Walter e coautores [47] desenvolveram um algoritmo com complexidade de tempo $O(n^2)$, com fator de aproximação 2.25. Elias e Hartman [17] desenvolveram um algoritmo de aproximação com fator de 1.375 e complexidade $O(n^2)$, um avanço na razão de aproximação que não ocorria desde a publicação do trabalho de Bafna e Pevzner, oito anos antes.

Em 2012, Bulteau e coautores [8] provaram que o problema da distância de transposição é NP-Difícil com uma redução do problema SAT, resolvendo essa questão que permaneceu em aberto durante aproximadamente 15 anos. Além disso, eles provaram que dada uma permutação π, descobrir se é possível ordenar π com exatamente $\frac{b_t(\pi)}{3}$ transposições também é NP-difícil.

Note que uma transposição $\rho_t(i, j, k)$ corta três pontos da permutação: entre (π_{i-1}, π_i), (π_{j-1}, π_j) e (π_{k-1}, π_k). Deste modo, uma transposição pode criar, manter ou remover no máximo 3 breakpoints e, assim, $\Delta b_t(\pi, \rho_t) = b_t(\pi \cdot \rho_t) - b_t(\pi)$ e $\Delta b_t(\pi, \rho_t) \in \{-3, -2, -1, 0, 1, 2, 3\}$. Assim, um limitante inferior para a distância é $d_t(\pi) \geq \frac{b_t(\pi)}{3}$, proposto por Bafna e Pevzner [3]. Além disso, sempre é possível encontrar uma transposição que diminua em pelo menos uma unidade o número de breakpoints se $\pi \neq \iota$. Logo, temos o limite superior para a distância: $d_t(\pi) \leq b_t(\pi)-2$, uma vez que a permutação pode ter $n+1$ breakpoints, onde n é o tamanho da permutação, e a última transposição sempre remove 3 breakpoints. Estes limitantes utilizam apenas o conceito de breakpoints. Limitantes mais precisos, propostos por Bafna e Pevzner [3], consideram o número de ciclos ímpares no grafo de ciclos da permutação π: $\frac{n+1-c_{impar}(G(\pi))}{2} \leq d_t(\pi) \leq \frac{3(n+1-c_{impar}(G(\pi)))}{4}$. Em relação ao diâmetro, Bafna e Pevzner [3] provaram o melhor limitante inferior conhecido, $\lfloor \frac{n+1}{2} \rfloor \leq D_t(n)$,

e Eriksson e coautores [18] provaram o melhor limitante superior conhecido, $D_t(n) \leq \lfloor \frac{2n-2}{3} \rfloor$.

3 Problemas de Ordenação que Consideram Reversões e Transposições

Estes problemas permitem que tanto os eventos de reversão, definidos na Seção 2.5, quanto os eventos de transposição, definidos na Seção 2.6, ocorram durante a ordenação de uma permutação π. As seções abaixo apresentam os problemas que estão diretamente relacionados com esta proposta.

3.1 O Problema da Ordenação de Permutações por Reversões e Transposições

Uma investigação preliminar deste problema foi realizada por Blanchette e coautores [7]. Existem duas versões para este problema: quando considera-se permutações com sinais e quando considera-se permutações sem sinais. Ambas as versões possuem complexidade desconhecida.

Para o caso de permutações com sinais, Walter e coautores [46] apresentaram um algoritmo de aproximação com fator e limitantes para o diâmetro. Meidanis e coautores [38] apresentaram limitantes para a distância. Gu e coautores [23] acrescentaram o evento de transreversão ao problema. No evento de transreversão, um bloco é destacado do genoma e inserido em outra posição, mas com a ordem e a orientação invertidas. Formalmente, usamos $\rho_{rt}(i,j,k)$ para denotar uma transreversão. Existem dois tipos de transreversão, $\pi \cdot \rho_{rt}(i,j,k) = (\pi \cdot \rho_r(i,j-1)) \cdot \rho_t(i,j,k)$ ou $\pi \cdot \rho_{rt}(i,j,k) = (\pi \cdot \rho_r(j,k-1)) \cdot \rho_t(i,j,k)$. Para uma transreversão do segundo tipo, temos $\pi \cdot \rho_{rt}(i,j,k) = (\pi_1 \ ... \ \pi_{i-1} \ \underline{-\pi_{k-1} \ ... \ -\pi_j} \ \underline{\pi_i \ ... \ \pi_{j-1}} \ \pi_k \ ... \ \pi_n)$. Gu e coautores [23] apresentaram um algoritmo de aproximação com fator 2 quando as três operações são permitidas. Lin e Xue [36] apresentaram um algoritmo com fator de aproximação 1.75 quando, além das três operações citadas anteriormente, é adicionada também a operação RevRev, que aplica reversões em dois blocos consecutivos da permutação em apenas uma operação. Mais tarde, Hartman e Sharan [26] melhoraram este fator 1.5. Estas variações também

possuem complexidade desconhecida.

Para o caso de permutações sem sinais, Walter e coautores [46] apresentaram um algoritmo de aproximação com fator 3. Em 2008, Rahman e coautores [42] apresentaram limitantes para a distância e um algoritmo de aproximação com fator $2k$, onde k é o fator de aproximação de um algoritmo utilizado para a decomposição de ciclos. Dado o melhor valor de k conhecido [12], o fator de aproximação deste algoritmo torna-se $2.8334 + \epsilon$, para qualquer $\epsilon > 0$. Lintzmayer e Dias [37] apresentaram limitantes para o diâmetro deste problema.

3.2 O Problema da Ordenação de Permutações por Reversões e Transposições Ponderadas

Os problemas citados até este ponto consideram que todas as operações do modelo possuem o mesmo custo. Entretanto, pesquisas existentes sugerem que, em alguns casos, as reversões ocorridas durante o processo evolutivo tendem a não ser muito longas [7]. Isto sugere que atribuir pesos aos segmentos afetados ao invés de apenas contabilizar o número de operações pode ajudar a explicar melhor alguns processos evolutivos. A partir desta constatação, autores começaram a considerar rearranjos ponderados.

Para o Problema de Ordenação de Permutações por Reversões Ponderadas, considerando permutações sem sinais e com base em seu comprimento, Pinter e Skiena [40] apresentaram um algoritmo aproximado com fator $O(\lg^2 n)$. Bender e coautores [4] apresentaram uma análise para determinados valores de α considerando a função de custo $f(l) = l^\alpha$, com $\alpha \geq 0$, onde l é o número de elementos afetados pela reversão. Quando $\alpha = 1$, os autores conseguiram um fator de aproximação $O(\lg n)$ e, quando $\alpha \geq 2$, um fator de aproximação igual a 2. Swidan e coautores [43] estenderam este trabalho para permutações com sinais, conseguindo a aproximação de $O(\lg n)$ para $\alpha = 1$.

Problemas ponderados que consideram apenas transposições ou que consideram reversões e transposições para permutações com e sem sinais ainda não possuem resultados conhecidos.

3.3 O Problema da Ordenação de Permutações por Reversões e Transposições Curtas e Super Curtas

Uma reversão $\rho_r(i,j)$ é dita uma k-reversão se $k = j - i + 1$. Dizemos que uma reversão é *curta* se $k \leq 3$, e *super curta* se $k \leq 2$. Uma transposição $\rho_t(i,j,k)$ é dita uma (x,y)-transposição se $x = j - i$ e $y = k - j$. Dizemos que uma transposição é *curta* se $x + y \leq 3$, e *super curta* se $x + y = 2$.

Problemas que consideram operações curtas e super curtas possuem importância biológica ao pressupor que eventos de rearranjo afetando grandes porções de um genoma são menos prováveis de ocorrer. De fato, trabalhos anteriores mostram a prevalência de reversões curtas na evolução de alguns genomas bacterianos e eucariotos [15, 35].

Para permutações sem sinais e considerando apenas reversões e transposições super curtas, temos que uma 2-reversão possui o mesmo efeito de uma 2-transposição, logo não faz sentido abordar modelos separados para cada operação. Note que para ordenar uma permutação precisamos remover todas as inversões. Se uma permutação π não está ordenada, então π possui ao menos um par de elementos na forma (π_i, π_{i+1}) tal que $\pi_i > \pi_{i+1}$. Desta forma, sempre é possível remover uma inversão com uma reversão (ou transposição) super curta ao aplicá-la sobre os elementos π_i e π_{i+1}. Assim, temos que ordenar permutações sem sinais por reversões (ou transposições) super curtas pode ser resolvido em tempo polinomial, sendo necessária a aplicação de $inv(\pi)$ reversões (ou transposições) [31].

Heath e Vergara [29] consideraram o problema de ordenar permutações sem sinais por reversões curtas, exibindo um algoritmo com fator de aproximação 2 sendo este o melhor fator de aproximação conhecido. Heath e Vergara [27, 28] também consideraram o problema de ordenar permutações sem sinais por transposições curtas e apresentaram um algoritmo com fator de aproximação $\frac{4}{3}$. Vergara [45] mostrou mais tarde que este algoritmo com fator de aproximação $\frac{4}{3}$ para ordenar permutações sem sinais por transposições curtas é também um algoritmo com fator de aproximação 2 para o problema de ordenar permutações sem sinais por reversões curtas e transposições curtas.

Jiang e coautores [33] apresentaram um algoritmo com fator de aproximação $(1+\epsilon)$ para ordenar permutações sem sinais por transposições curtas, onde

ϵ é o número de elementos divido pelo número de inversões da permutação. Mais tarde, Jiang e coautores [32] apresentaram um algoritmo com fator de aproximação $\frac{5}{4}$ para ordenar permutações sem sinais por transposições curtas.

Galvão e Dias [21] investigaram o problema de ordenar permutações com sinais por reversões curtas e apresentaram algoritmos de aproximação, sendo o melhor deles com fator 9. Mais tarde, Galvão e coautores [22] provaram que ordenar permutações com sinais por reversões super curtas e ordenar permutações com sinais por reversões super curtas e transposições super curtas podem ser resolvidos em tempo polinomial, e apresentaram um algoritmo com fator de aproximação 5 para ordenar permutações com sinais por reversões curtas e um algoritmo com fator de aproximação 3 para ordenar permutações com sinais por reversões curtas e transposições curtas. Além disso, os autores mostraram que o fator de aproximação esperado do algoritmo que ordena permutações por reversões curtas é menor ou igual a 3 para permutações com pelo menos 12 elementos.

Quando consideramos permutações circulares, Jerrum [31] provou que ordenar permutações circulares sem sinais por reversões (ou transposições) super curtas pode ser resolvido em tempo polinomial. Galvão e coautores [20] provaram que ordenar permutações circulares com sinais por reversões super curtas pode ser resolvido em tempo polinomial. Não existem trabalhos disponíveis na literatura abordando a ordenação de permutações circulares com sinais por reversões e transposições super curtas.

3.4 Consolidação do Estado da Arte

A Tabela 1 resume o estado da arte das variações do Problema da Ordenação por Rearranjo revisadas nesta proposta.

4 Objetivos

Esta proposta tem como objetivo estudar variações de problemas de ordenação de permutações quando o modelo permite a aplicação de reversões e transposições, e pode ser dividido em quatro frentes distintas, tal como descrito nas subseções seguintes.

Modelo de Rearranjo	Complexidade	Melhor Solução Existente
Reversões com Sinais	Polinomial [25]	Algoritmo exato $O(n^{\frac{3}{2}})$ [44]
Reversões sem Sinais	NP-Difícil [11]	Algoritmo 1.375-aproximado [5]
Transposições	NP-Difícil [8]	Algoritmo 1.375-aproximado [17]
Reversões e Transposições com Sinais	?	Algoritmo 2-aproximado [46]
Reversões e Transposições sem Sinais	?	Algoritmo $2k$-aproximado [5]
Reversões Curtas com Sinais	?	Algoritmo 5-aproximado [22]
Reversões Curtas sem Sinais	?	Algoritmo 2-aproximado [29]
Transposições Curtas	?	Algoritmo 5/4-aproximado [32]
Reversões e Transposições Curtas com Sinais	?	Algoritmo 3-aproximado [22]
Reversões e Transposições Curtas sem Sinais	?	Algoritmo 2-aproximado [45]
Reversões (ou Transposições) Super Curtas sem Sinais	Polinomial [31]	Algoritmo exato $O(n^2)$ [31]
Reversões (ou Transposições) Super Curtas Circulares sem Sinais	Polinomial [31]	Algoritmo exato $O(n^2)$ [31]
Reversões Super Curtas com Sinais	Polinomial [22]	Algoritmo exato $O(n^3)$ [22]
Reversões Super Curtas Circulares com Sinais	Polinomial [20]	Algoritmo exato $O(n^2)$ [20]
Reversões e Transposições Super Curtas com Sinais	Polinomial [22]	Algoritmo exato $O(n^3)$ [22]
Reversões e Transposições Super Curtas Circulares com Sinais	?	?

Tabela 1: Estado da arte das variações do Problema da Ordenação por Rearranjo.

4.1 O Problema da Ordenação de Permutações por Reversões e Transposições

Este problema vem sendo estudado pelo candidato desde seu mestrado. Inicialmente, foram criadas diversas heurísticas a fim de observar como este problema se comporta ao utilizar uma ou mais métricas clássicas em problemas de ordenação de permutações. Ao final deste estudo, foi constatado que a métrica breakpoints retornou os melhores resultados. Outra parte do mestrado foi dedicada a criar uma heurística que utiliza o grafo de ciclos da permutação para ordená-la, e esta heurística também retornou bons resultados na prática. Assim, foi criada uma nova heurística que ordena permutações com base no número de breakpoints e no grafo de ciclos. Esta heurística ordena permutações da seguinte forma: se a permutação admite uma operação que remova 2 ou 3 breakpoints, esta operação será aplicada, garantindo fatores de aproximação 1 e 1.5, respectivamente. Caso contrário, a heurística verifica se o grafo de ciclos da permutação admite a aplicação de uma operação que aumente em duas unidades o número de ciclos ímpares, ou que não aumente o número de ciclos ímpares, mas garanta que as próximas duas operações aumentam o número de ciclos ímpares em duas unidades cada, garantindo desta forma fatores de aproximação 1 e 1.5, respectivamente.

Caso contrário, a heurística busca configurações geradas a partir do grafo de ciclos da permutação atual em bancos de configurações que admitem uma sequência de operações com fatores de aproximação definidos.

Os bancos de configurações foram divididos de acordo com os fatores de aproximação que as sequências garantem, e a busca por configurações acontece a partir dos bancos de configurações com fatores de aproximação mais baixos para os bancos de configurações com fatores de aproximação mais altos. Além disso, é garantido que existe pelo menos uma configuração gerada a partir do grafo de ciclos da permutação π em um dos bancos de configurações criados.

Comparando a heurística criada com trabalhos existentes na literatura, tanto os que ordenam permutações apenas por reversões ou transposições, quanto os trabalhos que ordenam permutações por reversões e transposições, a heurística retornou na média os melhores resultados, tanto para permutações com sinais quanto para permutações sem sinais. A ideia inicial previa, além da criação da heurística, diminuir o fator de aproximação atual, de 2 para permutações com sinais e $2k$ para permutações sem sinais, onde k é o fator de aproximação do algoritmo utilizado para a decomposição de ciclos, para 1.8 e $1.8k$, respectivamente. Entretanto, a abordagem utilizada na criação e extensão de configurações gerou um volume de configurações proibitivo de serem explorados pelo algoritmo *branch-and-bound* em um intervalo de tempo factível, e, por este motivo, os mesmos fatores de aproximação existentes na literatura foram mantidos. Detalhes da heurística criada foram descritos na dissertação de mestrado do candidato [39]. Como a heurística retorna resultados médios melhores que qualquer outro algoritmo já publicado, para permutações com sinais e sem sinais, submetemos, em setembro de 2016, um artigo com os resultados obtidos por esta heurística para publicação.

Durante o doutorado, continuaremos explorando este problema em busca de algoritmos com fatores de aproximação melhores que os existentes na literatura.

4.2 O Problema da Ordenação de Permutações por Reversões e Transposições Curtas e Super Curtas

Para os problemas onde reversões e transposições curtas são permitidas, pretendemos criar algoritmos com fatores de aproximação melhores que os existentes na literatura, tanto para permutações com sinais quanto para permutações sem sinais. Além disso, pretendemos estudar classes de permutações que podem ser ordenadas em tempo polinomial.

Quando trabalhamos com reversões e transposições super curtas, temos os seguintes problemas:

(i) Ordenação de Permutações sem Sinais por Reversões (ou Transposições) Super Curtas;
(ii) Ordenação de Permutações Circulares sem Sinais por Reversões (ou Transposições) Super Curtas;
(iii) Ordenação de Permutações com Sinais por Reversões Super Curtas;
(iv) Ordenação de Permutações Circulares com Sinais por Reversões Super Curtas;
(v) Ordenação de Permutações com Sinais por Reversões e Transposições Super Curtas;
(vi) Ordenação de Permutações Circulares com Sinais por Reversões e Transposições Super Curtas.

Pela Tabela 1, podemos notar que, com exceção do problema (vi), todos admitem algoritmos polinomiais exatos. Sendo assim, investigaremos se o Problema da Ordenação de Permutações Circulares com Sinais por Reversões e Transposições Super Curtas também admite um algoritmo polinomial exato.

4.3 O Problema da Ordenação de Permutações por Reversões e Transposições Ponderadas

Este problema ainda não possui nenhum algoritmo disponível na literatura. Por este motivo, decidimos também estudá-lo durante o doutorado. Investigaremos se é possível obter algoritmos aproximados para este problema. Além disso, criaremos heurísticas que retornem bons resultados médios.

4.4 Aprendizado de Máquina Aplicado a Problemas de Ordenação

Em aprendizado de máquina, *classificação* é o problema de identificar a qual conjunto de categorias uma nova observação pertence com base em um conjunto de dados de treinamento cujas categorias já são conhecidas. Como exemplo, podemos classificar um email como *spam* ou *não-spam* treinando o classificador com emails confiáveis e não confiáveis, ou mesmo no caso da ordenação de

permutações, classificar operações aplicadas em uma permutação em boas ou ruins, considerando se sua aplicação aproxima ou distancia a permutação resultante da permutação identidade. Esta ideia nunca foi explorada na literatura de rearranjo de genomas.

Queremos criar um algoritmo que, dada uma permutação, teste todas as reversões e transposições aplicáveis e aplique aquela com maior chance de ser uma boa operação. Para fins de classificação, testaremos os seguintes métodos: Regressão Logística, Redes Neurais, Deep Learning e SVM. Trabalhos comparando métodos de classificação abordando problemas gerais são comuns na literatura, como, por exemplo, ao classificar pacientes com problemas coronários [30] ou mesmo ao classificar o risco de crédito de clientes, dentre outras situações reais [41].

Propomos avaliar as técnicas de aprendizado de máquina no problema da ordenação de permutações por reversões e transposições, considerando permutações com sinais e sem sinais.

5 Estágio no exterior

Dentro do escopo deste projeto de doutorado, programamos um período de um ano de estágio no exterior (modalidade doutorado sanduíche). O estágio está programado para iniciar em novembro de 2016 junto ao grupo de pesquisa coordenado pelo Prof. Dr. Guillaume Fertin, no *Laboratoire d'Informatique de Nantes* da Université de Nantes na França.

O candidato já possui uma bolsa de doutorado sanduíche aprovada através do projeto Capes/Cofecub de cooperação entre universidades brasileiras e francesas. Este projeto foi firmado entre o Instituto de Computação da Unicamp, sob coordenação do Prof. Dr. Zanoni Dias, e os laboratórios franceses *Laboratoire D'Informatique de Nantes*, da Université de Nantes, e *Laboratoire d'Informatique Gaspard-Monge*, da Université Paris-Est Marne-la-Vallée, sob coordenação do Prof. Dr. Guillaume Fertin, e prevê a troca de conhecimento e aprofundamento da cooperação em pesquisas na área de Biologia Computacional entre os países. Os recursos deste projeto objetivam favorecer o intercâmbio de pessoas em ambas as direções entre os dois países, o que pode ocorrer mediante as missões trabalho para intercâmbio de pesquisadores e as missões de

estudo para intercâmbio de alunos.

Vale ressaltar que a equipe francesa deste projeto conta com pesquisadores renomados na área de Rearranjo de Genomas, como autores do livro que é a principal referência na área [19], bem como autores de importantes artigos científicos publicados neste campo de pesquisa [8, 9].

Um dos tópicos que pretendemos estudar no exterior é o Problema da Ordenação de Permutações de Reversões e Transposições. A criação dos bancos de configurações deste problema, por exemplo, foi pauta de todas as missões de trabalho já realizadas pelo projeto. Assim, o supervisor francês poderá contribuir com as próximas etapas deste problema. Além disso, pretendemos finalizar a prova do algoritmo polinomial para o Problema da Ordenação de Permutações Circulares com Sinais por Reversões e Transposições Super Curtas. Este problema foi apresentado ao supervisor francês pelo aluno em uma missão de trabalho no Brasil.

6 Cronograma

	2015					2016												2017						
	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J
1	*	*	*	*	*																			
2		*	*	*	*	*	*	*				*	*			*	*	*			*	*	*	
3								*	*	*	*													
4												*	*	*										
5															*									
6																*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*						*	*	*	*	*	*	*	*	*				
8									*	*	*	*	*	*	*	*	*	*						
9						*	*	*	*							*	*	*	*	*	*	*	*	*
10								*	*	*	*													
12												*	*				*	*		*	*			
13			*	*				*	*				*	*				*	*				*	*

	2017					2018												2019						
	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J
2		*	*				*	*				*	*				*	*		*	*			
6	*	*	*																					
9	*	*	*	*	*																			
10											*	*	*	*	*	*	*	*	*	*	*	*		
11					*	*	*	*	*	*	*	*	*	*	*	*								
12					*	*										*	*					*	*	
13			*	*				*	*				*	*			*	*		*	*	*	*	*
14																								*

1. Obtenção dos créditos obrigatórios em disciplinas do programa de dou-

torado;
2. Revisão Bibliográfica;
3. Programa de Estágio Docente (PED);
4. Escrita da proposta de doutorado;
5. Exame de Qualificação Específico (EQE);
6. Realização do Doutorado Sanduíche;
7. Investigação de algoritmo com um melhor fator de aproximação, para as versões do Problema de Ordenação de Permutações por Reversões e Transposições que consideram permutações com sinal e sem sinal;
8. Investigação de algoritmo exato polinomial para O Problema da Ordenação de Permutações com Sinais Circulares por Reversões e Transposições Super Curtas;
9. Aplicação de Aprendizado de Máquina em Ordenação de Permutações por Reversões e Transposições;
10. Investigação de heurísticas para O Problema da Ordenação de Permutações por Reversões e Transposições Ponderadas;
11. Investigação de algoritmo de aproximação com fatores melhores que os existentes para O Problema da Ordenação de Permutações por Reversões e Transposições Curtas;
12. Escrita de artigos;
13. Escrita e revisão da tese;
14. Defesa da tese.

É importante ressaltar que o cronograma apresentado pode ser adaptado à medida que os avanços ocorrerem, uma vez que tais avanços poderão nos levar a resultados mais promissores que outros, o que nos faria dedicar mais tempo em uma ou mais atividades em detrimento de outras.

7 Metodologia

O trabalho a ser realizado é, em grande parte, teórico. O estudo de teoremas e provas dos trabalhos já existentes na literatura são a principal base para o estabelecimento dos limitantes e das classes de permutações específicas. Sempre que possível e necessário, no entanto, programas serão desenvolvidos para

comprovar a validade dos resultados obtidos, bem como comparar os resultados com outros algoritmos disponíveis na literatura.

8 Resultados Iniciais

Abaixo descrevemos o andamento dos estudos envolvendo os problemas apresentados na Seção 4 desde o início do doutorado até o momento.

8.1 O Problema da Ordenação de Permutações por Reversões e Transposições

No segundo semestre de 2015, buscamos um novo limitante inferior para utilizar na criação e extensão de configurações que pudesse evitar que o número de configurações a serem exploradas pelo algoritmo *branch-and-bound* fosse proibitivo, a fim de atacar novamente o problema criando novos bancos de configurações considerando o novo limitante. Em meados deste ano, um novo limitante foi definido, e desde então demos início à construção dos novos bancos de configurações. Atualmente, grande parte das configurações geradas já possuem sequências de operações válidas, sendo que o maior fator de aproximação até o momento é de $\frac{54}{29} \approx 1.8621$. Acreditamos que as configurações sem sequência válida serão resolvidas por um algoritmo de *branch-and-bound* ainda este ano, quando saberemos se de fato todas as configurações admitem uma sequência de operações com fator de aproximação menor que 2.

8.2 O Problema da Ordenação de Permutações por Reversões e Transposições Curtas e Super Curtas

Fizemos um estudo aprofundado sobre este problema e acreditamos que ele também admite uma solução polinomial. Atualmente estamos trabalhando na prova de um algoritmo polinomial exato, e um esboço de prova é apresentado na Seção 9.

8.3 O Problema da Ordenação de Permutações por Reversões e Transposições Ponderadas

No primeiro semestre de 2016, uma investigação preliminar sobre este problema foi feita [16]. Adaptamos heurísticas criadas durante o mestrado do candidato para funcionar também com reversões e transposições ponderadas. A análise dos resultados mostrou que, diferentemente da versão não ponderada onde as heurísticas básicas de breakpoints e a heurística de grafo de ciclos retornaram os melhores resultados médios, as heurísticas ponderadas de breakpoints e inversões retornaram os melhores resultados médios.

Além disso, constatamos que a heurística de grafo de ciclos, que retorna resultados melhores que outros algoritmos existentes na literatura quando se considera apenas o número de operações, obteve resultados piores que as heurísticas básicas e, por este motivo, não é adequada ao considerar reversões e transposições ponderadas. Parte disso se deve ao fato de que os bancos de configurações criados não levam em conta o peso das operações, já que possuem restrições apenas no número de operações aplicadas.

8.4 Aprendizado de Máquina Aplicado a Problemas de Ordenação

Iniciamos a construção de um banco de treinamento para permutações sem sinais de tamanho 10, criando, para cada permutação, um vetor de características para cada operação aplicada e indicando se ela é uma operação boa (diminui a distância até a permutação identidade) ou não. Posteriormente, pretendemos treinar o classificador com as características das permutações de tamanho 10 e validar o treinamento com permutações de tamanhos maiores.

O banco de treinamento criado possui cerca de 30 características para cada operação de cada permutação, dentre elas podemos citar: variação no número de breakpoints, número de strips não unitárias, soma dos tamanhos das strips não unitárias, número de elementos em sua posição correta, número de inversões, número de ciclos ímpares, tamanho do maior ciclo, etc.

9 Ordenação de Permutações Circulares com Sinais por Reversões e Transposições Super Curtas

Esta seção apresenta um esboço de prova de um algoritmo polinomial exato para o Problema da Ordenação de Permutações Circulares com Sinais por Reversões e Transposições Super Curtas. A Subseção 9.1 apresenta definições importantes que serão utilizadas nos lemas apresentados na Subseção 9.2.

9.1 Definições iniciais

Definição 3. *Uma* ***reversão cíclica*** $\rho_r(i,j)$ *é uma operação semelhante à reversão. Entretanto, não existe a restrição de que* $i \leq j$ *e, caso* $i > j$*, o trecho revertido será* $[\pi_j..\pi_n\pi_1..\pi_i]$*, uma vez que em permutações circulares os elementos* π_n *e* π_1 *são adjacentes.*

Definição 4. *Uma* ***transposição cíclica*** $\rho_t(i,j,k)$ *é uma operação semelhante à transposição. Entretanto, a restrição de que* $i < j < k$ *não é unica, sendo permitidas também outras duas restrições:* $j < k < i$*, e* $k < i < j$*. Para a restrição* $j < k < i$*, temos que os blocos afetados pela transposição cíclica serão* $[\pi_i..\pi_n\pi_1..\pi_{j-1}]$ *e* $[\pi_j..\pi_{k-1}]$*. Para a restrição* $k < i < j$*, temos que os blocos afetados pela transposição cíclica serão* $[\pi_i..\pi_{j-1}]$ *e* $[\pi_j..\pi_n\pi_1..\pi_{k-1}]$*.*

Definição 5. *Seja* $S(\pi_i, \pi_{i \pmod n+1})$ *o ato de trocar a posição de dois elementos* π_i *e* $\pi_{i \pmod n+1}$ *em uma permutação* π *não estendida. Se além de trocar a posição destes elementos ocorrer a inversão dos sinais de ambos, temos que* $S(\pi_i, \pi_{i \pmod n+1}) = \rho_r(i, i \pmod n + 1)$*. Se não houver troca de sinal, temos que*
$S(\pi_i, \pi_{i \pmod n+1}) = \rho_t(i, i \pmod n + 1, (i+1) \pmod n + 1)$.

Definição 6. *Dada uma sequência* $W = <\rho_1, \rho_2, ..., \rho_k>$ *de reversões e transposições cíclicas super curtas que ordena uma permutação* $\pi \in S_n^{\pm}$*, denotamos por* $R_W(|\pi_i|)$ *o número de 2-reversões e 2-transposições do tipo* $S(|\pi_i|, \pi_{i \pmod n+1})$ *e denotamos por* $L_W(|\pi_i|)$ *o número de 2-reversões e 2-transposições do tipo* $S(\pi_{(n+i-2) \pmod n+1}, |\pi_i|)$*. Temos que* $R_W(|\pi_i|)$ *denota o número de operações em* W *que troca o elemento* π_i *com o elemento à sua direita, e* $L_W(|\pi_i|)$ *denota o número de operações em* W *que troca o elemento* π_i *com o elemento à sua esquerda.*

Por exemplo, para a sequência de reversões e transposições cíclicas super curtas $W = <\rho_r(2,3), \rho_r(1,2), \rho_r(2,3), \rho_t(5,1,2)>$ que ordena a permutação $\pi = (+3\ +2\ +5\ +4\ +1)$, temos que W resulta na seguinte sequência de troca entre elementos: $<S(2,5), S(3,-5), S(-3,-2), S(1,5)>$. Logo, temos os seguintes valores para L_W e R_W:

- $R_W(|\pi_1|) = |\{S(3,-5), S(-3,-2)\}| = 2$ e $L_W(|\pi_1|) = |\emptyset| = 0$.
- $R_W(|\pi_2|) = |\{S(2,5)\}| = 1$ e $L_W(|\pi_2|) = |\{S(-3,-2)\}| = 1$.
- $R_W(|\pi_3|) = |\emptyset| = 0$ e $L_W(|\pi_3|) = |\{S(2,5), S(3,-5), S(1,5)\}| = 3$.
- $R_W(|\pi_4|) = |\emptyset| = 0$ e $L_W(|\pi_4|) = |\emptyset| = 0$.
- $R_W(|\pi_5|) = |\{S(1,5\}| = 1$ e $L_W(|\pi_5|) = |\emptyset| = 0$.

Definição 7. *O* ***valor deslocamento*** *de um elemento π_i com respeito à W é dado por $d_W(\pi_i) = R_W(|\pi_i|) - L_W(|\pi_i|)$ (1-reversões não alteram o valor de $d_W(\pi_i)$). O* ***vetor deslocamento*** *de π com respeito à W é definido por $d_W(\pi) = (d_W(\pi_1), d_W(\pi_2), ..., d_W(\pi_n))$.*

Para o exemplo anterior, temos:

- $d_W(\pi_1) = R_W(|\pi_1|) - L_W(|\pi_1|) = 2 - 0 = 2$
- $d_W(\pi_2) = R_W(|\pi_2|) - L_W(|\pi_2|) = 1 - 1 = 0$
- $d_W(\pi_3) = R_W(|\pi_3|) - L_W(|\pi_3|) = 0 - 3 = -3$
- $d_W(\pi_4) = R_W(|\pi_4|) - L_W(|\pi_4|) = 0 - 0 = 0$
- $d_W(\pi_5) = R_W(|\pi_5|) - L_W(|\pi_5|) = 1 - 0 = 1$.
- $d_W(\pi) = (2,\ 0,\ -3,\ 0,\ 1)$.

Definição 8. *Seja $x = (x_1, x_2, ..., x_n) \in Z^n$ um vetor deslocamento e seja $\pi \in S_n^{\pm}$ uma permutação. Dizemos que x é um* ***vetor deslocamento válido*** *para π se $\sum_{i=1}^{n} x_i = 0$ e $|\pi_i| - x_i \equiv i \pmod n$.*

Note que podemos obter um valor deslocamento válido com respeito à uma sequência W (não necessariamente ótima) de um elemento pela fórmula $d_W(\pi_i) = |\pi_i| - i$.

Para $\pi = (+3\ +2\ +5\ +4\ +1)$, temos:

- $d_W(\pi_1) = |\pi_1| - 1 = 3 - 1 = 2$
- $d_W(\pi_2) = |\pi_2| - 2 = 2 - 2 = 0$

- $d_W(\pi_3) = |\pi_3| - 3 = 5 - 3 = 2$
- $d_W(\pi_4) = |\pi_4| - 4 = 4 - 4 = 0$
- $d_W(\pi_5) = |\pi_5| - 5 = 1 - 5 = -4$.
- $d_W(\pi) = (2,\ 0,\ 2,\ 0,\ -4)$.

Note que o vetor deslocamento $x = (2,\ 0,\ 2,\ 0,\ -4)$ é um vetor deslocamento válido para $\pi = (+3\ +2\ +5\ +4\ +1)$, uma vez que satisfaz $\sum_{i=1}^{n} x_i = 0$ e $|\pi_i| - x_i \equiv i \pmod{n}$.

Definição 9. *Dado um vetor deslocamento válido $x = (x_1, x_2, ..., x_n) \in Z^n$ e dois inteiros distintos $i, j \in \{1, 2, ..., n\}$, seja $r = i - j$ e $s = (i + x_i) - (j + x_j)$. O* ***crossing number*** *entre i e j, $i \neq j$, com respeito a x é definido por:*

$$c_{ij}(x) = \begin{cases} |\{k \in [r\ ..\ s] : k \equiv 0 \pmod{n}\}|, & \text{if } r \leq s, \\ -|\{k \in [s\ ..\ r] : k \equiv 0 \pmod{n}\}|, & \text{if } r > s. \end{cases} \tag{1}$$

O crossing number do vetor x é definido por $C(x) = \frac{1}{2} \sum_{i \neq j} |c_{ij}(x)|$.

Por exemplo, para a permutação $\pi = (+3\ +2\ +5\ +4\ +1)$, o vetor deslocamento $x = (2,\ 0,\ 2,\ 0,\ -4)$, $i = 1$ e $j = 2$, temos que $r = 1 - 2 = -1$ e $s = (1 + 2) - (2 + 0) = 1$. Então, $c_{12}(x) = |\{k \in [-1\ ..\ 1] : k \equiv 0 \pmod{5}\}| = |\{0\}| = 1$. Abaixo temos a matriz de todos os valores para c_{ij}, onde i é denotado pela linha da matriz e j pela coluna. Note que $c_{ij}(x) = -c_{ji}(x)$.

$$c(x) = \begin{pmatrix} * & 1 & 0 & 0 & 1 \\ -1 & * & 0 & 0 & 1 \\ 0 & 0 & * & 1 & 1 \\ 0 & 0 & -1 & * & 1 \\ -1 & -1 & -1 & -1 & * \end{pmatrix}$$

Assim, o crossing number de x é dado por $C(x) = \frac{1}{2} \sum_{i \neq j} |c_{ij}(x)| = \frac{12}{2} = 6$.

Definição 10. *Seja $\pi \in S_n^{\pm}$ uma permutação e seja $x = (x_1, x_2, ..., x_n) \in Z^n$ um vetor deslocamento válido para π. Dizemos que x induz uma operação ρ envolvendo dois elementos π_i e π_j se $i \neq j$ e $c_{ij}(x) \neq 0$. Além disso, o vetor x não define ρ como uma 2-reversão ou uma 2-transposição, mas sim a existência de uma troca do tipo $S(\pi_i, \pi_j)$ ou $S(\pi_j, \pi_i)$.*

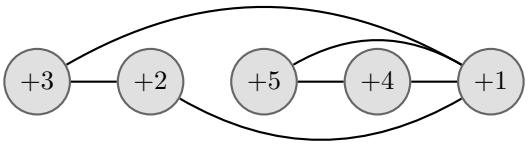
O crossing number $c_{ij}(x)$ indica, na prática, se o vetor x induz uma operação envolvendo os elementos π_i e π_j. Se $c_{ij}(x) = 1$, então x induz a troca $S(\pi_i, \pi_j)$. Se $c_{ij}(x) = -1$, então x induz a troca $S(\pi_j, \pi_i)$. Caso contrário $c_{ij}(x) = 0$, e x não induz uma troca entre os elementos π_i e π_j.

Sabemos que o vetor deslocamento $x = (2,\ 0,\ 2,\ 0,\ -4)$ para a permutação $\pi = (+3\ +2\ +5\ +4\ +1)$ possui $c_{12}(x) \neq 0$, induzindo assim uma operação envolvendo os elementos $\pi_1 = +3$ e $\pi_2 = +2$. Além disso, x não induz uma operação entre os elementos $\pi_1 = +3$ e $\pi_3 = +5$, uma vez que $c_{13}(x) = 0$.

Definição 11. *Dado um vetor deslocamento válido x para uma permutação cíclica $\pi \in S_n^{\pm}$, definimos o* ***grafo de permutação cíclica*** *como um grafo não orientado $G_{\pi,x} = (V, E)$, onde $V = \{\pi_1, \pi_2, ..., \pi_n\}$ e $E = \{(\pi_i, \pi_j) : c_{ij}(x) \neq 0\}$. Um* ***componente*** *de um grafo é um subgrafo maximal de $G_{\pi,x}$ no qual quaisquer dois vértices estão conectados por um caminho. Denotamos o número de componentes de $G_{\pi,x}$ por $c(\pi, x)$. Além disso, dizemos que um componente de $G_{\pi,x}$ é* ***impar*** *se contém um número ímpar de elementos (vértices) negativos, e dizemos que é* ***par*** *caso contrário. O número de componentes ímpares de $G_{\pi,x}$ é denotado por $c_{odd}(\pi, x)$.*

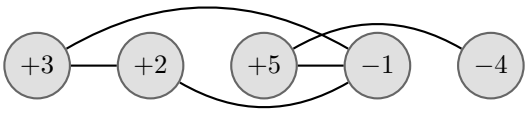
Propriedade 1. *Dado um vetor deslocamento válido x para uma permutação $\pi \in S_n^{\pm}$, temos que o vértice π_i de $G_{\pi,x}$ é incidente a pelo menos $|x_i|$ arestas, e o componente que contém π_i possui $|x_i| + 1$ vértices (o vértice π_i e os $|x_i|$ vértices ligados à π_i).*

Abaixo temos um exemplo de $G_{\pi,x}$ para $\pi = (+3\ +2\ +5\ +4\ +1)$ e $x = (2,\ 0,\ 2,\ 0,\ -4)$. Note que $G_{\pi,x}$ possui apenas um componente, uma vez que quaisquer dois vértices deste grafo estão ligados por um caminho, e $c(\pi, x) = 1$. Além disso, como este componente não possui um número ímpar de elementos negativos, temos que $c_{odd}(\pi, x) = 0$. Note também que $G_{\pi,x}$ possui 6 arestas (crossing number C(x)).



Propriedade 2. *Seja $\pi \in S_n^{\pm}$ uma permutação, $x = (x_1, x_2, ..., x_n) \in Z^n$ um vetor deslocamento válido para π e ρ uma operação super curta sobre os elementos π_i e $\pi_{i \pmod n+1}$ induzida por x. Temos então que o vetor deslocamento resultante x' para $\pi' = \pi \cdot \rho$ é tal que $x'_k = x_k$ para todo $k \neq \{i, i \pmod n + 1\}$, $x'_i = x_{i \pmod n+1} + 1$ e $x'_{i \pmod n+1} = x_i - 1$. Além disso, x' é um vetor deslocamento válido para π', $C(x') = C(x) - 1$ e o grafo $G_{\pi',x'}$ é obtido a partir de $G_{\pi,x}$ removendo-se a aresta que liga os elementos π_i e $\pi_{i \pmod n+1}$ em $G_{\pi,x}$.*

Para $\pi = (+3\ +2\ +5\ +4\ +1)$ e $x = (2,\ 0,\ 2,\ 0,\ -4)$, temos que $\rho_r(4,5)$ é uma operação induzida por x. Assim, $\pi' = \pi \cdot \rho_r(4,5) = (+3\ +2\ +5\ -1\ -4)$ e $x' = (2,\ 0,\ 2,\ -3,\ -1)$. Abaixo ilustramos o efeito desta operação em $G_{\pi',x'}$, que possui uma aresta a menos que o grafo $G_{\pi,x}$.



Propriedade 3. *A transformação $T_{ij}(x) : Z^n \to Z^n$ associa um vetor deslocamento $x \in Z^n$ ao vetor deslocamento x' tal que $x'_k = x_k$, para $k \notin \{i, j\}$, $x'_i = x_i - n$, e $x'_j = x_j + n$.*

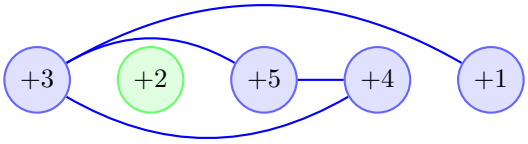
Lema 1 ([31] Teorema 3.9)**.** *Para dois vetores deslocamento x e x' sobre Z^n tais que $x' = T_{ij}(x)$, temos que $C(x') = C(x) + 2 \times (n - x_i + x_j)$.*

Definição 12. *Uma transformação $T_{ij}(x)$ é dita **restritiva** se, e somente se $x_i - x_j \geq n$ (o que implica que $C(x') \leq C(x)$). Uma transformação $T_{ij}(x)$ é dita **estritamente restritiva** se, e somente se $x_i - x_j > n$ e, neste caso, $C(x') < C(x)$. Uma transformação $T_{ij}(x)$ é dita **expansiva** se, e somente se $x_i - x_j < n$ e, neste caso, $C(x') > C(x)$.*

Afirmação 1. *Se um vetor deslocamento x não admite uma transformação estritamente restritiva, então $C(x)$ é mínimo, e $x_i - x_j \leq n$ para quaisquer valores de $1 \leq i, j \leq n$ com $i \neq j$.*

Definição 13. *Dado um vetor deslocamento válido x para uma permutação π, denotamos por $d(\pi, x)$ o número de operações super curtas necessárias para ordenar π aplicando operações induzidas pelo vetor x.*

Dado $\pi = (+3\ +2\ +5\ +4\ +1)$ e $x = (2,\ 0,\ 2,\ 0,\ -4)$, a transformação $x' = T_{15}(x)$ gera $x'_i = 2 - 5 = -3$ e $x'_j = -4 + 5 = 1$. Desta forma, temos $x' = (-3,\ 0,\ 2,\ 0,\ 1)$. A transformação $T_{15}(x)$ é estritamente restritiva, pois $x_1 - x_5 = 2 - (-4) = 6 > 5$, e temos que $C(x') = C(x) - 2 \times (n - x_i + x_j) = 6 - 2 \times (5 - 2 - 4) = 6 - 2 = 4$. Abaixo ilustramos o grafo $G_{\pi,x'}$, que possui 4 arestas, $c(\pi, x') = 2$ componentes (destacados nas cores azul e verde) e $c_{odd}(\pi, x') = 0$.



9.2 Ordenando Permutações Circulares com Sinais por Reversões e Transposições Super Curtas em Tempo Polinomial

Lema 2. *Seja $x \in Z^n$ um vetor deslocamento válido para π e seja x' o vetor deslocamento resultante após a aplicação da operação ρ em π, tal que $x' = x$ e $\pi' = \pi \cdot \rho$. Como o vetor deslocamento não é alterado, ρ deve ser uma 1-reversão e $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) \in \{-1, 1\}$.*

Demonstração. Como $x' = x$, temos que nenhum elemento da permutação mudou de posição, e, desta forma, ρ não muda o número de componentes do grafo ($c(\pi', x') = c(\pi, x)$). Se o componente afetado por ρ em $G_{\pi,x}$ é par, então este componente será ímpar em $G_{\pi',x'}$. Se o componente afetado por ρ em $G_{\pi,x}$ é ímpar, então este componente será par em $G_{\pi',x'}$. Temos então que $\Delta c_{odd} \in \{-1, 1\}$. □

Lema 3. *Seja $x \in Z^n$ um vetor deslocamento válido para π e x' o vetor deslocamento resultante após a aplicação de uma operação ρ induzida por x em π, tal que $x' \neq x$, $C(x') = C(x) - 1$ e $\pi' = \pi \cdot \rho$. Como $x' \neq x$, ρ deve ser uma 2-reversão ou uma 2-transposição, e $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) \in \{0, 2\}$.*

Demonstração. Como $x' \neq x$, temos que ρ não pode ser uma 1-reversão, já que 1-reversões não alteram o vetor deslocamento. Se $c(\pi', x') = c(\pi, x)$, então ρ não muda o número de componentes. Neste caso, se definirmos ρ como uma 2-transposição, então a paridade do componente afetado por ρ em $G_{\pi,x}$

será a mesma em $G_{\pi',x'}$, uma vez que transposições não trocam os sinais dos elementos afetados. Se definirmos ρ como uma 2-reversão, então a paridade do componente afetado por ρ em $G_{\pi,x}$ será a mesma em $G_{\pi',x'}$, uma vez que os sinais de dois elementos dessa componente serão trocados. Temos que em ambos os casos $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) = 0$.

Caso contrário, ρ é aplicado em um componente transformando este componente em dois novos componentes (e, desta forma, $c(\pi, x') = c(\pi, x) + 1$). Se definirmos ρ como uma 2-transposição, e o componente afetado por ρ em $G_{\pi,x}$ é ímpar, os novos componentes de $G_{\pi',x'}$ terão paridades distintas, e $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) = 0$. Caso contrário, o componente afetado por ρ em $G_{\pi,x}$ é par, e os novos componentes em $G_{\pi',x'}$ terão a mesma paridade, e $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) \in \{0, 2\}$. Se definirmos ρ como uma 2-reversão, e o componente afetado por ρ em $G_{\pi,x}$ é ímpar, os dois novos componentes de $G_{\pi',x'}$ terão paridades distintas, e $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) = 0$. Caso contrário, o componente afetado por ρ é par, e os dois novos componentes em $G_{\pi',x'}$ terão a mesma paridade, e, desta forma, $\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) \in \{0, 2\}$. □

Lema 4. *Seja $x \in Z^n$ um vetor deslocamento válido para π. Sempre é possível encontrar uma operação ρ induzida por x tal que x' é um vetor deslocamento válido para $\pi' = \pi \cdot \rho$, $x' \neq x$, $C(x') = C(x) - 1$ e o número de componentes ímpares é mantido ($\Delta c_{odd} = c_{odd}(\pi', x') - c_{odd}(\pi, x) = 0$).*

Demonstração. Se $c(\pi', x') = c(\pi, x)$, então ρ não muda o número de componentes, e pelo Lema 3 sabemos que $c_{odd}(\pi', x') = c_{odd}(\pi, x)$ independentemente de ρ ser uma 2-reversão ou uma 2-transposição.

Se $c(\pi', x') > c(\pi, x)$ e o componente afetado por ρ em $G_{\pi,x}$ é ímpar, sabemos pelo Lema 3 que $c_{odd}(\pi', x') = c_{odd}(\pi, x)$ independentemente de ρ ser uma 2-reversão ou uma 2-transposição. Caso o componente afetado por ρ em $G_{\pi,x}$ seja par, vamos considerar os dois componentes ainda em $G_{\pi,x}$ ao remover a aresta entre os elementos afetados por ρ. Se ambos os componentes são pares, podemos assumir que nenhum novo componente ímpar será criado em $G_{\pi',x'}$ se ρ for uma transposição ao invés de uma reversão. Caso contrário, ambos os componentes são ímpares, e podemos fazer com que ρ seja uma reversão que trocará os sinais de exatamente um elemento de cada componente,

transformando-os em componentes pares em $G_{\pi',x'}$, e nenhum componente ímpar será criado. Temos então que é sempre possível optar por uma reversão ou transposição de modo que o resultado seja que $c_{odd}(\pi', x') = c_{odd}(\pi, x)$. □

Lema 5. *Seja W uma sequência de reversões e transposições cíclicas super curtas que ordena uma permutação $\pi \in S_n^{\pm}$, e seja $x \in Z^n$ um vetor deslocamento válido para π tal que $d_W(\pi) = x$. Se W é uma sequência de tamanho mínimo, então W não possui reversões ou transposições que aumentam o número de componentes ímpares.*

Demonstração. Note que para ordenar uma permutação precisamos atingir um grafo de permutação cíclica com n componentes pares. Dos lemas 2 e 3 temos que apenas 1-reversões podem diminuir o número de componentes ímpares. Assim, W deve conter pelo menos $c_{odd}(\pi, x)$ 1-reversões. Suponha que W é uma sequência de tamanho mínimo que ordena uma permutação π e W possui uma operação ρ que aumenta o número de componentes ímpares.

Se $\rho(i, i)$ é uma 1-reversão, então ρ é aplicada em um componente par resultando em $c_{odd}(\pi, x) + 1$ componentes ímpares. Logo, o número de 1-reversões de W deve ser maior ou igual a $c_{odd}(\pi, x) + 2$ (sendo $c_{odd}(\pi, x) + 1$ 1-reversões aplicadas nos componentes ímpares, somadas à esta operação $\rho(i, i)$). Vamos considerar a sequência $W' = W - \{\rho, \rho'\}$, onde ρ' é uma 1-reversão aplicada no componente ímpar criado por ρ. Note que ρ' deve existir, uma vez que alguma operação de W deve transformar o componente ímpar criado por ρ em um componente par mais tarde.

Se $\rho(i, i+1)$ é uma 2-reversão, então ρ é aplicada em um componente par transformando este componente em dois componentes ímpares, e o número de 1-reversões de W deve ser maior ou igual a $c_{odd}(\pi, x) + 2$. Vamos considerar a sequência W' obtida de W trocando a reversão ρ pela transposição $\rho^*(i, i+1, i+2)$ e removendo $\{\rho', \rho''\}$, onde ρ' e ρ'' são 1-reversões aplicadas nos dois componentes ímpares criados por ρ. Agora, quando ρ^* é aplicada no componente par, os dois novos componentes criados serão pares. Note que ρ' e ρ'' devem existir, uma vez que a aplicação de ρ fez com que um componente par se transformasse em dois componentes ímpares.

Se $\rho(i, i+1, i+2)$ é uma 2-transposição, então ρ é aplicada em um componente par criando dois novos componentes ímpares, e o número de 1-reversões

de W deve ser maior ou igual a $c_{odd}(\pi, x) + 2$. Vamos considerar a sequência W' obtida de W trocando a transposição ρ pela reversão $\rho^*(i, i+1)$ e removendo $\{\rho', \rho''\}$, onde ρ' e ρ'' são 1-reversões aplicadas nos dois componentes ímpares criados por ρ. Agora, quando ρ^* é aplicada no componente ímpar, os dois novos componentes criados serão pares. Note que ρ' e ρ'' devem existir, uma vez que a aplicação de ρ fez com que um componente par se transformasse em dois componentes ímpares.

Temos nos três casos acima uma sequência W' que também ordena π com $|W'| = |W| - 2$, o que contradiz a hipótese de que W é uma sequência mínima. Assim, W não possui operações que aumentam o número de ciclos ímpares. □

Lema 6. *Seja W uma sequência de reversões e transposições cíclicas super curtas de tamanho mínimo que ordena a permutação $\pi \in S_n^{\pm}$ e seja $x \in Z^n$ um vetor deslocamento válido para π. Se $d_W(\pi) = x$, então $d(\pi, x) = C(x) + c_{odd}(\pi, x)$.*

Demonstração. Podemos decompor a sequência W em duas subsequências distintas W_1 e W_2 tal que W_1 contém apenas 1-reversões de W e W_2 contém as 2-reversões e 2-transposições de W. Além disso, podemos assumir sem perda de generalidade que as operações da subsequência W_2 são aplicadas primeiro. Vamos argumentar que $|W_1| = c_{odd}(\pi, x)$ independentemente do tamanho de W_2. Para ver isto, suponha que aplicamos uma 2-reversão $\rho(i, i+1)$ ou uma 2-transposição $\rho(i, i+1, i+2)$ de W_2 em π, obtendo a permutação π'. Além disso, seja W' a sequência resultante após remover a operação aplicada da sequência W. Assumindo que o vetor $x' \in Z^n$ após a aplicação de uma operação induzida por x, temos pelos lemas 4 and 5 que $c_{odd}(\pi', x') = c_{odd}(\pi, x)$. Como $|W_1| = c_{odd}(\pi, x)$ independentemente do tamanho de W_2 e sabemos por Jerrum [31] que $|W_2| \geq C(x)$, podemos concluir que $|W_2| = C(x)$. □

Lema 7. *Seja $x \in Z^n$ um vetor deslocamento válido para π tal que $C(x)$ é mínimo e x é o vetor deslocamento com o menor valor de $c(\pi, x)$ dentre todos os vetores deslocamentos que possuem crossing number mínimo. Seja $\max(x)$ o valor x_i máximo de x e $\min(x)$ o valor x_j mínimo de x. Se $\max(x) - \min(x) = n$, então para qualquer vetor deslocamento x' tal que $C(x') > C(x)$ temos que $d(\pi, x') > d(\pi, x)$, e para o vetor x'' que minimiza $d(\pi, x'')$ temos que $C(x'') = C(x)$.*

Demonstração. Seja $x \in Z^n$ um vetor deslocamento válido para π tal que $C(x)$ é mínimo com $\max(x) - \min(x) = n$ e x é o vetor deslocamento com o menor valor de $c(\pi, x)$ dentre todos os vetores deslocamentos que possuem crossing number mínimo. Suponha que x' é um vetor deslocamento tal que $x' = T_{ij}(x)$ com $C(x') > C(x)$. Pelo Lema 1, $C(x') = C(x) + 2 \times (n - x_i + x_j) \geq C(x) + 2$, uma vez que $n \geq |x_i| + |x_j| + 1$.

Se $c(\pi, x) = 1$, então $c_{odd}(\pi, x) \leq 1$. Neste caso não existe vetor deslocamento x' com $C(x') + c_{odd}(\pi, x') < C(x) + c_{odd}(\pi, x)$, uma vez que temos que $C(x') \geq C(x) + 2$, $c_{odd}(\pi, x') \geq 0$ e $c_{odd}(\pi, x) \in \{0, 1\}$.

Se $c(\pi, x) \geq 2$, seja A o componente com os elementos $\max(x)$ e $\min(x)$. O componente A possui pelo menos $m + 1$ elementos, com $m = \max(\max(x), |\min(x)|)$, e fora de A existem no máximo $n - m - 1$ elementos (m alcança o valor mínimo quando $\max(x) = |\min(x)| = \frac{n}{2}$ e, neste caso, o número de elementos que não estão em A é menor ou igual a $\frac{n}{2} - 1$). O número de componentes de x, sem considerar A é menor ou igual a $n - m - k - 1$, sendo k o maior valor em módulo de um elemento $x_i \notin A$.

Temos então que $c(\pi, x) = 1 + n - m - k - 1 = n - m - k$, e $c_{odd}(\pi, x) \leq c(\pi, x)$. Para $c(\pi, x) \geq 2$, temos que $n - m - k \geq 2$. Logo, $0 \leq k \leq n - m - 2$ e $\frac{n}{2} \leq m \leq n - 2$. Desta forma $d(\pi, x) = C(x) + c_{odd}(\pi, x) \leq C(x) + n - m - k$ e $d(\pi, x') = C(x') + c_{odd}(\pi, x') = C(x) + 2 \times (n - x_i + x_j) + c_{odd}(\pi, x')$.

Para garantir um vetor x' com o menor valor possível para $C(x')$ gerado a partir de uma transformação expansiva, vamos assumir que os valores de k e m vêm de elementos com sinais contrários. Seja então $x_i = k$ e $x_j = -m$, se k é obtido a partir de um elemento $x_i \notin A$ positivo e m é obtido a partir de um elemento $x_i \in A$ negativo, e seja $x_j = -k$ e $x_i = m$, caso contrário. Estes são os valores para x_i e x_j que minimizam o valor de $C(x') > C(x)$ onde $x' = T_{ij}(x)$. Em ambos os casos, assumindo que $c_{odd}(\pi, x') = 0$, temos $d(\pi, x') = C(x) + 2 \times (n - m - k)$. Para que $d(\pi, x') < d(\pi, x)$, é necessário que $C(x) + 2 \times (n - m - k) < C(x) + n - m - k$, resultando em $k > n - m$, o que contradiz a suposição de que $0 \leq k \leq n - m - 2$. Temos então que não existe vetor x' com $C(x') > C(x)$ tal que $d(\pi, x') < d(\pi, x)$. □

Lema 8. *Seja $x \in Z^n$ um vetor deslocamento válido para π tal que $C(x)$ é minimo. Se $\max(x) - \min(x) < n$, então o vetor deslocamento válido para π*

que minimiza a distância de ordenação é x ou é um vetor x' alcançável por uma transformação expansiva em $x(x' = T_{ij}(x)$ e $C(x') > C(x))$.

Demonstração. Se os elementos que geram $\max(x)$ e $\min(x)$ não pertencem ao mesmo componente pela Propriedade 1 temos que existem dois componentes com pelo menos $\max(x)+1$ e $|\min(x)|+1$ elementos, respectivamente. Assim, $c(\pi, x) \leq n - \max(x) + \min(x)$, e $c_{odd}(\pi, x) \leq c(\pi, x)$. O vetor x' com menor valor de $C(x') > C(x)$ é tal que $x' = T_{ij}(x)$ onde $x_i = \max(x)$ e $x_j = \min(x)$. Então, temos:

- $d(\pi, x) = C(x) + c_{odd}(\pi, x) \leq C(x) + n - \max(x) + \min(x)$.
- $d(\pi, x') = C(x) + 2 \times (n - \max(x) + \min(x)) + c_{odd}(\pi, x')$.

Como $(n - \max(x) + \min(x)) > 0$ e $c_{odd}(\pi, x') \geq 0$, temos que $d(\pi, x') > d(\pi, x)$. Segue então que x é o vetor deslocamento que minimiza a distância de ordenação de π $(d(\pi, x) = C(x) + c_{odd}(\pi, x))$.

Caso contrário, $\max(x)$ e $\min(x)$ pertencem ao mesmo componente. Seja $x' = T_{ij}(x)$ com $x_i = \max(x)$ e $x_j = \min(x)$ o vetor deslocamento com menor valor de $C(x') > C(x)$, e seja $x'' = T_{kl}(x')$ com $x'_k = \max(x' \setminus \{x'_j\})$ e $x'_l = \min(x' \setminus \{x'_i\})$ o vetor deslocamento com menor valor para $C(x'') > C(x')$. Temos aqui as seguintes distâncias:

- $d(\pi, x) = C(x) + c_{odd}(\pi, x)$;
- $d(\pi, x') = C(x') + c_{odd}(\pi, x') = C(x) + 2 \times (n - x_i + x_j) + c_{odd}(\pi, x')$;
- $d(\pi, x'') = C(x'') + c_{odd}(\pi, x'') = C(x') + 2 \times (n - x'_k + x'_l) + c_{odd}(\pi, x'') = C(x) + 2 \times (n - x_i + x_j) + 2 \times (n - x'_k + x'_l) + c_{odd}(\pi, x'')$.

Note que $c_{odd}(\pi, x), c_{odd}(\pi, x'), c_{odd}(\pi, x'') \geq 0$, $2 \times (n - x_i + x_j) \geq 2$ e $2 \times (n - x'_k + x'_l) \geq 2$. Suponha que $d(\pi, x) > d(\pi, x'')$.

Para $d(\pi, x) > d(\pi, x'')$, temos que $c_{odd}(\pi, x) - c_{odd}(\pi, x'') > 2 \times (n - x_i + x_j) + 2 \times (n - x_k + x_l)$. Vamos denotar por $comp(x, x_a)$ a componente que o elemento x_a está em x. Neste ponto temos que $comp(x, x_i) = comp(x, x_j)$. Seja $y = \max(x_i, |x_j|)$, e seja $z = \max(x'_k, |x'_l|)$. Então, $2 \times (n - x_i + x_j) = 2 \times (n - 2y) + \epsilon$, com $\epsilon \geq 0$, e $2 \times (n - x'_k + x'_l) = 2 \times (n - 2z) + \epsilon'$, com $\epsilon' \geq 0$, e $c_{odd}(\pi, x) - c_{odd}(\pi, x'') > 2 \times (n - x_i + x_j) + 2 \times (n - x_k + x_l) = 2 \times (n - 2y) + \epsilon + 2 \times (n - 2z) + \epsilon' = 4 \times (n - y - z) + \epsilon + \epsilon'$.

Se $z = x_k$ e $comp(x, x_k) \neq comp(x, x_i)$, ou se $z = |x_l|$ e $comp(x, x_l) \neq comp(x, x_i)$, então $c(\pi, x), c_{odd}(\pi, x) \leq n - y - z$, com $n - y - z > 0$, uma vez que $c(\pi, x) > 0$. Logo, temos que não é possível que $c_{odd}(\pi, x) - c_{odd}(\pi, x'') > 4 \times (n - y - z) + \epsilon + \epsilon'$, já que $c_{odd}(\pi, x) \leq n - y - z$ e $c_{odd}(\pi, x'') \geq 0$. Segue então que x'' não pode ser o vetor deslocamento que minimiza a soma $C(x'') + c_{odd}(\pi, x'')$.

Se $z = x_k$ e $comp(x, x_k) = comp(x, x_i)$, ou $z = |x_l|$ e $comp(x, x_l) = comp(x, x_i)$, então $c(\pi, x), c_{odd}(\pi, x) < n - y$. Neste ponto temos duas situações: $comp(x, x_k) = comp(x, x_l)$ ou $comp(x, x_k) \neq comp(x, x_l)$.

Para $comp(x, x_k) = comp(x, x_l)$, vamos considerar o vetor deslocamento $x^* = T_{kl}(x)$. Esta transformação faz com que os elementos x^*_k e x^*_l, que possuem arestas ligadas apenas com elementos de $comp(x, x_i)$ em $G_{\pi,x}$, possuam arestas com todos os elementos exceto aqueles de $comp(x^*, x^*_i)$ em $G_{\pi,x'}$. Entretanto, como $x_i > x_k$ e $|x_j| > |x_l|$, ainda temos elementos em $comp(x^*, x^*_i)$ possuindo arestas com x^*_k e x^*_l, então este vetor possui $c(\pi, x^*) = 1$, $c_{odd}(\pi, x^*) \leq 1$, e $d(\pi, x^*) \leq C(x) + 2 \times (n - x_k + x_l) + 1$. Como $x_k = x'_k$, $x_l = x'_l$, $c_{odd}(\pi, x'') \geq 0$ e $2 \times (n - x_i + x_j) \geq 2$, segue que $d(\pi, x^*) < d(\pi, x'')$, o que contradiz a suposição de que x'' é o vetor deslocamento que minimiza $d(\pi, x'')$, e x^* é alcançável a partir de uma transformação expansiva em x.

Para $comp(x, x_k) \neq comp(x, x_l)$ um dos seguintes vetores deslocamento x^* terá os mesmos componentes que o vetor deslocamento x'':

- Se $z = x_k$, $x^* = T_{kj}$ e $d(\pi, x^*) = C(x) + 2 \times (n - x_k + x_j) + c_{odd}(\pi, x^*)$;
- Se $z = |x_l|$, $x^* = T_{il}$ e $d(\pi, x^*) = C(x) + 2 \times (n - x_i + x_l) + c_{odd}(\pi, x^*)$.

Como $c_{odd}(\pi, x'') = c_{odd}(\pi, x^*)$ e $(n - x_i + x_l), (n - x_k + x_j) > 0$, temos que $d(\pi, x'') > d(\pi, x^*)$, o que contradiz a suposição de que x'' é o vetor deslocamento que minimiza $d(\pi, x'')$, e x^* é alcançável a partir de uma transformação expansiva em x. □

Lema 9. *Um vetor deslocamento $x \in Z^n$ válido para π e minimiza a soma $C(x) + c_{odd}(\pi, x)$ pode ser encontrado em tempo polinomial.*

Demonstração. Começando a partir de um vetor deslocamento válido x aplicamos uma sequência de transformações estritamente restritivas $T_{ij}(x)$ (onde $x_i - x_j > n$). Seja x agora um vetor deslocamento quando não é possível aplicar

transformações estritamente restritivas. Pela Afirmação 1 temos que x possui o menor valor de $C(x)$ e $\max(x) - \min(x) \leq n$. Se $\max(x) - \min(x) = n$, então, pelo Lema 7, o vetor deslocamento x' que minimiza a distância de ordenação $d(\pi, x')$ possui $C(x') = C(x)$ e Jerrum ([31], Teorema 3.9) provou que, se existe mais de um vetor deslocamento com $C(x)$ mínimo, então todos podem ser encontrados por uma sequência de transformações restritivas, o que pode ser feito em tempo polinomial.

Caso contrário, $\max(x) - \min(x) < n$ e, pelo Lema 8, o vetor deslocamento x' que minimiza a distância de ordenação $d(\pi, x')$ é o próprio vetor x ou é um vetor x' que pode ser obtido por meio de uma transformação expansiva em x ($x' = T_{ij}(x)$ e $C(x') > C(x)$), o que pode ser feito em tempo polinomial. □

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

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

[4] 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.

[5] 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.

[6] P. Berman and M. Karpinski. On Some Tighter Inapproximability Results (Extended Abstract). In J. Wiedermann, P. E. Boas, and M. Nielsen, editors, *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICAL'1999)*, volume 1644 of *Lecture Notes in Computer Science*, pages 200–209. Springer-Verlag, London, UK, 1999.

[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] L. Bulteau, G. Fertin, and I. Rusu. Pancake Flipping is Hard. *Journal of Computer and System Sciences*, 81(8):1556–1574, 2015.

[10] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the 1st Annual International Conference on Computational Molecular Biology (RECOMB'1997)*, pages 75–83, Santa Fe, New Mexico, USA, 1997.

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

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

[13] D. 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.

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

[15] D. A. Dalevi, N. Eriksen, K. Eriksson, and S. G. Andersson. Measuring genome divergence in bacteria: A case study using chlamydian data. *Journal of Molecular Evolution*, 55(1):24–36, 2002.

[16] E. R. Donoso, A. R. Oliveira, U. Dias, and Z. Dias. Problema da Ordenação Ponderada por Reversões e Transposições. Technical Report IC-PFG-16-01, Institute of Computing, University of Campinas, 2016. In Portuguese.

[17] 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.

[18] H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, and J. Wastlund. Sorting a Bridge Hand. *Discrete Mathematics*, 241(1-3):289–300, 2001.

[19] G. Fertin, A. Labarre, I. Rusu, É. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. The MIT Press, London, England, 2009.

[20] G. R. Galvão, C. Baudet, and Z. 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.

[21] G. R. Galvão and Z. 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.

[22] G. R. Galvão, O. Lee, and Z. Dias. Sorting Signed Permutations by Short Operations. *Algorithms for Molecular Biology*, 10(1):1–17, 2015.

[23] Q.-P. Gu, S. Peng, and I. H. Sudborough. A 2-Approximation Algorithm for Genome Rearrangements by Reversals and Transpositions. *Theoretical Computer Science*, 210(2):327–339, 1999.

[24] S. Hannenhalli and P. A. Pevzner. Transforming Men into Mice (Polynomial Algorithm for Genomic Distance Problem). In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'1995)*, pages 581–592, Washington, DC, USA, 1995. IEEE Computer Society Press.

[25] 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.

[26] T. Hartman and R. Sharan. A 1.5-Approximation Algorithm for Sorting by Transpositions and Transreversals. *Journal of Computer and System Sciences*, 70(3):300–320, 2005.

[27] L. S. Heath and J. P. C. Vergara. Sorting by Bounded Block-moves. *Discrete Applied Mathematics*, 88(1-3):181–206, 1998.

[28] L. S. Heath and J. P. C. Vergara. Sorting by Short Block-Moves. *Algorithmica*, 28(3):323–352, 2000.

[29] L. S. Heath and J. P. C. Vergara. Sorting by Short Swaps. *Journal of Computational Biology*, 10(5):775–789, 2003.

[30] M. T. Imran Kurt and A. T. Kurum. Comparing performances of logistic regression classification and regression tree and neural networks for predicting coronary artery disease. *Expert Systems with Applications*, 34:366–374, 2008.

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

[32] H. Jiang, H. Feng, and D. 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.

[33] H. Jiang, D. Zhu, and B. Zhu. A (1+e)-Approximation Algorithm for Sorting by Short Block-Moves. *Theoretical Computer Science*, 437:1–8, 2012.

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

[35] J. F. Lefebvre, N. El-Mabrouk, E. Tillier, , and D. Sankoff. Detection and validation of single gene inversions. *Bioinformatics*, 19(1):190–196, 2003.

[36] G.-H. Lin and G. Xue. Signed Genome Rearrangement by Reversals and Transpositions: Models and Approximations. *Theoretical Computer Science*, 259(1-2):513–531, 2001.

[37] C. N. Lintzmayer and Z. Dias. On the Diameter of Rearrangement Problems. In A.-H. Dediu, C. Martín-Vide, and B. Truthe, editors, *Algorithms for Computational Biology*, volume 8542 of *Lecture Notes in Computer Science*, pages 158–170. Springer International Publishing, Switzerland, 2014.

[38] 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):743–745, 2002.

[39] A. R. Oliveira. O Problema da Ordenação de Permutações por Reversões e Transposições. Master's thesis, Institute of Computing, University of Campinas, 2015. In Portuguese.

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

[41] C. F. R. D. King and A. Sutherland. Statlog-comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995.

[42] 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.

[43] F. Swidan, M. A. Bender, D. Ge, S. He, H. Hu, and R. 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.

[44] E. Tannier, A. Bergeron, and M.-F. Sagot. Advances on Sorting by Reversals. *Discrete Applied Mathematics*, 155(6-7):881–888, 2007.

[45] J. P. C. Vergara. *Sorting by Bounded Permutations*. PhD thesis, Polytechnic Institute & State University, 1998.

[46] 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, Santa Cruz de la Sierra, Bolivia, 1998. IEEE Computer Society.

[47] M. E. M. T. Walter, Z. Dias, and J. 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.