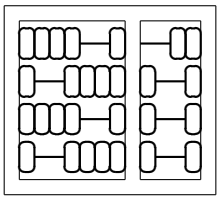
**UNIVERSIDADE ESTADUAL DE CAMPINAS**
INSTITUTO DE COMPUTAÇÃO



**Proposta de Dissertação de Mestrado**

# Uma Ferramenta para Avaliação de Algoritmos de Rearranjo de Genomas e sua Aplicação ao Problema da Ordenação por Transposições de Prefixo

**Aluno**: Gustavo Rodrigues Galvão
**Orientador**: Prof. Dr. Zanoni Dias

CAMPINAS

2011

# Sumário

**1 Introdução** **3**

1.1 Objetivo . . . . . 7

1.2 Organização da Proposta . . . . . 8

**2 A Ferramenta** **8**

**3 Problema da Ordenação por Transposições de Prefixo** **9**

**4 Cronograma e Plano de Trabalho** **13**

**5 Metodologia** **14**

**6 Análise dos Resultados** **15**

**Resumo**

Um dos grandes desafios modernos da ciência é tentar explicar o processo de evolução das espécies. Para superá-lo, pesquisadores da área de Biologia Computacional propõem que a divergência dos genomas ao longo da evolução é majoritariamente fruto de eventos de rearranjo que modificam a ordem de grandes porções de DNA conservadas entre os indivíduos. Assim, o problema de se determinar a distância evolucionária entre os genomas de dois indivíduos reduz-se ao problema de se encontrar a sequência mínima de eventos de rearranjo que transforma um genoma em outro, que pode ser postulado genericamente como o Problema da Ordenação de Genomas ou simplesmente Problema da Ordenação.

Muitos algoritmos de aproximação foram propostos para resolver diferentes variações deste problema. Por essa razão, nós propomos a construção de uma ferramenta para automatizar e padronizar a avaliação quantitativa de algoritmos aproximativos de rearranjo de genomas que virá a facilitar uma análise quantitativa de tais algoritmos. Este tipo de análise é essencial, por exemplo, para comparar algoritmos aproximativos que possuem o mesmo fator de aproximação ou comparar algoritmos aproximativos com heurísticas.

Para demonstrar a aplicação da ferramenta, nós iremos utilizá-la para avaliar algoritmos de aproximação e heurísticas que viermos a desenvolver ao longo deste trabalho para resolver uma variação específica do problema, o Problema da Ordenação por Transposições de Prefixo.

# 1 Introdução

Dentre as diferentes abordagens que se pode propor para explicar o processo de evolução das espécies, uma que vem ganhando destaque nos últimos anos é a dada pela área de *Rearranjo de Genomas*. Esta área da Biologia Computacional visa compreender o mecanismo de evolução comparando indivíduos por meio da análise de eventos de rearranjo que modificam a ordem de grande porções de seus genomas, ao invés de comparar modificações pontuais na sequência de nucleotídeos do DNA destes indivíduos, tal como é feito no processo de Alinhamento de Sequências. Portanto, sob a perspectiva da área de Rearranjo de Genomas, dados dois genomas, representados como uma sequência de genes ou uma sequência de blocos conservados entre eles, o problema consiste-se em encontrar a menor sequência de eventos de rearranjo que levou um genoma a se transformar no outro. Contabilizando-se o tamanho desta sequência, a distância genômica é obtida, sendo esta considerada como a distância evolucionária entre tais indivíduos[1].

O genoma de um indivíduo é constituído por cromossomos, os quais podem ser representados como um conjunto ordenado de genes (ou blocos conservados) orientados, sendo esta orientação determinada pela localização do gene na dupla fita de DNA. Um evento de rearranjo ocorre quando cromossomos são quebrados em segmentos e unidos de tal forma que o conjunto inicial de genes permanece inalterado, mas a ordem ou orientação dos genes possivelmente se alteram.

[1]Assume-se que esta é a distância evolucionária partindo-se do pressuposto que a Natureza é parcimoniosa. Em outras palavras, assume-se que a evolução dos indivíduos é regida pelo princípio da Navalha de Occam.

Diversos eventos de rearranjo ou *operações* foram propostos. Denominando-se como *modelo de rearranjo* o conjunto de operações permitidas para se transformar um genoma em outro, os primeiros estudos voltaram-se à analise de modelos que permitiam apenas um tipo de operação, dentre as quais podemos citar como as mais estudadas: *reversão*, *transposição*, *block-interchange*, *translocação*, *fissão* e a *fusão*. Estes modelos iniciais foram capazes de explicar alguns cenários evolutivos simples, como por exemplo de vírus da herpes (Bafna e Pevzner [4]) e do genoma mitocondrial do repolho e do nabo (Hannenhalli e Pevzner [29]). Posteriormente, iniciaram-se os estudos de modelos que combinavam tais operações, o que possibilitou a análise de cenários evolutivos mais complexos, como por exemplo do rato e do homem (Hannenhalli e Pevzner [28]).

Formalmente, um cromossomo é classicamente representado como uma $n$-tupla cujos elementos representam os genes. Supondo que não haja repetição dos genes, esta $n$-tupla é uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, $1 \leq |\pi_i| \leq n$ e $|\pi_i| \neq |\pi_j| \leftrightarrow i \neq j$. Cada elemento $\pi_i$ possui um sinal, $+$ ou $-$, que indica a orientação do gene que ele representa. Quando não há informação sobre a orientação dos genes, o sinal é omitido.

Dadas duas permutações $\pi$ e $\sigma$ e um modelo de rearranjo $M$, o problema de se transformar a permutação $\pi$ na permutação $\sigma$ consiste-se em encontrar a menor sequências de operações $\rho_1$, $\rho_2$, ..., $\rho_t$ pertencentes a $M$ tal que $(\rho_t \cdots (\rho_2 \cdot (\rho_1 \cdot \pi))) = \sigma$. O tamanho desta sequência representa a distância entre as permutações $\pi$ e $\sigma$ com respeito ao modelo $M$ e é denotada por $d_M(\pi, \sigma)$ (neste caso, $d_M(\pi, \sigma) = t$). A maior distância entre duas permutações de tamanho $n$ com respeito ao modelo $M$ é chamada de diâmetro da distância de rearranjo e é denotada por $D_M(n)$.

Definindo-se a permutação identidade como $I_n = (1\ 2\ \ldots\ n)$, a ordenação de uma permutação $\alpha$ pode ser caracterizada como o processo de transformar a permutação $\alpha$ na permutação identidade $I_n$ e a distância entre elas com respeito a um modelo $M$ é denotada por $d_M(\alpha, I_n) = d_M(\alpha)$. Este processo é equivalente ao processo de se transformar a permutação $\pi$ na permutação $\sigma$ se tomarmos $\alpha = \pi\sigma^{-1}$, pois $d_M(\pi, \sigma) = d_M(\pi\sigma^{-1}, I_n) = d_M(\alpha)$.

Uma reversão é a operação que inverte a ordem dos elementos de uma permutação. Caso a permutação possua sinais, estes também são invertidos. Formalmente, uma reversão pode ser definida como uma operação $r(i, j)$ sobre um intervalo $[i, j]$ de uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ tal que $r(i, j) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}\ \pi_j}\ \pi_{j+1}\ \ldots\ \pi_n) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{-\pi_j\ -\pi_{j-1}\ \ldots\ -\pi_{i+1}\ -\pi_i}\ \pi_{j+1}\ \ldots\ \pi_n)$.

Para o caso de permutações sem sinais, Caprara [8] demostrou que o problema de ordenar uma permutação por reversões é NP-Difícil. Kececioglu e Sankoff [32] foram os primeiros a apresentar um algoritmo de aproximação para o problema, sendo o mesmo uma 2-aproximação. Posteriormente, Bafna e Pevzner [3] construíram um algoritmo de aproximação com fator aproximativo de 1.75. A próxima evolução foi dada por Christie [12] ao desenvolver um algoritmo com fator 1.5. Por fim, o melhor resultado conhecido até o presente momento foi apresentado por Berman, Hannenhalli e Karpinski [6]. Eles apresentaram um algoritmo de aproximação com fator 1.375.

Para o caso em que as permutações possuem sinais, Hannanhalli e Pevzner [28] foram os primeiros

a demonstrar que existe um algoritmo polinomial para o problema. Alguns refinamentos foram feitos neste algoritmo ao longo dos anos até que Tannier, Bergeron e Sagot [43] apresentaram um algoritmo $O(n^{\frac{3}{2}}\sqrt{\log n})$, sendo este o melhor resultado conhecido. Barder, Moret e Yan [2] mostraram como calcular a distância de reversão de uma permutação com sinal em tempo linear sem precisar calcular a sequência de reversões.

A operação de transposição promove a troca de dois trechos adjacentes de uma permutação sem que haja modificação dos sinais dos elementos. Formalmente, ela pode ser definida como uma operação $t(i, j, k)$ sobre uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ tal que $t(i, j, k) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \pi_k\ \ldots\ \pi_n) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n)$, sendo $1 \leq i < j < k \leq n+1$.

O primeiro resultado relevante obtido para o problema de se ordenar permutações (sem sinais) por transposições é devido a Bafna e Pevzner [4]. Neste trabalho, eles apresentaram um algoritmo aproximativo com fator de aproximação 1.5. Elias e Hartman [18] produziram uma algoritmo de aproximação com fator 1.375. Recentemente, Bulteau, Fertin e Rusu [7] demonstraram que o problema é NP-Difícil.

Quando uma reversão/transposição age em uma porção da permutação contendo o primeiro elemento, nós dizemos que ocorreu uma reversão/transposição de prefixo. O problema de ordenar uma permutação sem sinal por reversões de prefixo, também conhecido como *Problema da Ordenação de Panquecas*, foi introduzido por Dweighter [17]. O melhor algoritmo conhecido para resolver este problema foi apresentado por Fischer e Ginzinger [21] e é uma 2-aproximação. Gates e Papadimitriou [25] forneceram um limite superior de $\frac{5n}{3}$ para o problema, sendo $n$ o tamanho da permutação. O melhor limite superior conhecido é $\frac{11n}{8}$ e foi dado por Chitturi *et al.* [9]. Para a versão com sinal do problema, conhecida como *Problema das Panquecas Queimadas*, Cohen e Blum [13] forneceram um limite inferior de $\frac{3n}{2}$ e um limite superior de $2n - 2$ para $n \geq 10$. O problema de ordenar uma permutação por transposições de prefixo foi introduzido por Dias e Meidanis [15]. Ele será visto em detalhes na Seção 3.

A operação de *block-interchange* é uma generalização da operação de transposição, isto é, ela promove a permutação de dois trechos quaisquer de uma permutação. Formalmente, ela pode ser definida como uma operação $bi(i, j, k, l)$ sobre uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ tal que $bi(i, j, k, l) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \ldots\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \pi_k\ \ldots\ \pi_n) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \ldots\ \underline{\pi_i\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n)$, sendo $1 \leq i < j < k < l \leq n+1$. A operação de transposição é o caso especial em que $j = k$.

Este operação foi introduzida por Christie [11] e no mesmo trabalho que ele a apresentou, ele também propôs um algoritmo que resolve o problema de ordenação por *block-interchanges* em tempo quadrático. Feng e Zhu [20] apresentaram uma estrutura de dados que permitiu melhorar a complexidade do algoritmo, tornando-o $O(n\log n)$. O avanço mais recente deve-se a Huang *et al.* [31]. Eles apresentaram um algoritmo $O(n + d\log d)$, sendo $d$ o número mínimo de *block-interchanges* requeridos para ordenar uma permutação.

A operação de translocação promove a troca entre trechos finais (sufixos ou prefixos) de duas

permutações, podendo ser subdividida em dois tipos: translocação prefixo-prefixo e translocação prefixo-sufixo. Dados duas permutações $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_m)$ e $\sigma = (\sigma_1\ \sigma_2\ \ldots\ \sigma_n)$, elas podem ser formalmente definidas como segue:

- Uma translocação prefixo-prefixo $tl_{pp}(\pi, \sigma, i, j)$ aplicada sobre $\pi$ e $\sigma$ produz duas permutações $(\pi_1\ \pi_2\ \ldots\ \pi_{i-1}\ \sigma_j\ \sigma_{j+1}\ \ldots\ \sigma_n)$ e $(\sigma_1\ \sigma_2\ \ldots\ \sigma_{j-1}\ \pi_i\ \pi_{i+1}\ \ldots\ \pi_m)$, sendo $1 < i \leq m$ e $1 < j \leq n$;
- Uma translocação prefixo-sufixo $tl_{ps}(\pi, \sigma, i, j)$ aplicada sobre $\pi$ e $\sigma$ produz duas permutações $(\sigma_n\ \sigma_{n-1}\ \ldots\ \sigma_j\ \pi_i\ \ldots\ \pi_m)$ e $(\sigma_1\ \ldots\ \sigma_{j-1}\ \pi_{i-1}\ \pi_{i-2}\ \ldots\ \pi_1)$, sendo $1 < i \leq m$ e $1 < j \leq n$. Se $\pi$ e $\sigma$ são permutações orientadas, então uma translocação prefixo-sufixo $tl_{ps}(\pi, \sigma, i, j)$ produz duas permutações orientadas $(-\sigma_n\ -\sigma_{n-1}\ \ldots\ -\sigma_j\ \pi_i\ \ldots\ \pi_m)$ e $(\sigma_1\ \ldots\ \sigma_{j-1}\ -\pi_{i-1}\ -\pi_{i-2}\ \ldots\ -\pi_1)$.

Kececioglu e Ravi [33] e Hannenhanlli [27] foram os primeiros a apresentar resultados relevantes para o problema de ordenação por translocações. Os primeiros consideraram o caso em que as permutações não possuem sinal e obtiveram como resultado um algoritmo aproximativo com fator de aproximação 2 e complexidade de tempo $O(n^2)$. O último considerou o caso em que as permutações possuem sinal e construiu um algoritmo polinomial exato com complexidade de tempo $O(n^3)$.

Posteriormente, Bergeron, Mixtacki e Stoye [5] apresentaram uma correção para o algoritmo proposto por Hannenhanlli [27] e propuseram um novo algoritmo, também com complexidade de tempo $O(n^3)$. O melhor resultado conhecido deriva do algoritmo proposto por Ozery-Flato e Shamir [39], que resolve o problema de ordenação por translocação em tempo $O(n^{\frac{3}{2}}\sqrt{\log n})$. Para o caso em que as permutações não possuem sinal, Zhu e Wang [50] demostraram que o problema é NP-Difícil.

As operações de fusão e fissão promovem, respectivamente, a união de duas permutações em uma única e a quebra de uma permutação em duas. Dadas as permutações $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_m)$ e $\sigma = (\sigma_1\ \sigma_2\ \ldots\ \sigma_n)$, a fusão $fu(\pi, \sigma)$ concatena as permutações $\pi$ e $\sigma$, resultando em um única permutação $(\pi_1\ \ldots\ \pi_m\ \sigma_1\ \ldots\ \sigma_n)$; a fissão $fi(\pi, i)$ quebra a permutação $\pi$ em duas permutações $(\pi_1\ \ldots\ \pi_{i-1})$ e $(\pi_i\ \ldots\ \pi_m)$, $2 \leq i \leq m$.

Modelos de rearranjo que consideram apenas operações de fusão ou de fissão são muito limitados. Em contrapartida, quando o número de cromossomos de dois genomas são diferentes, inevitavelmente será necessária a utilização dessas operações para transformar um genoma em outro. Por esse motivo, elas geralmente são combinadas com aquelas descritas anteriormente para formar modelos de rearranjo mais realistas.

Diversos modelos que combinam operações foram estudados e algoritmos para a distância genômica foram construídos a partir deles. Um dos primeiros e mais relevantes foi proposto por Hannenhalli e Pevzner [28], o qual cobria as operações de reversão, translocação, fusão e fissão. Quase 10 anos depois, Yancopoulos *et al.* [49] propuseram um modelo mais completo, pois além das operações anteriores, também permitia a operação de *block-interchange*. Meidanis e Dias [36] foram os primeiros a apresentar um algoritmo polinomial para resolver o problema da distância genômica para um modelo contendo

a operação de transposição. O modelo proposto por eles cobria as operações de transposição, fusão e fissão.

Walter, Dias e Meidanis [45] consideraram o problema da ordenação por reversões e transposições e apresentaram um algoritmo 2-aproximado para o caso em que as permutações possuem sinal e um algoritmo 3-aproximado para o caso em que as permutações não possuem sinal. Rahman, Shatabda e Hasan [40] apresentaram um algoritmo $2k$-aproximado para o caso sem sinal, sendo $k$ o fator de aproximação do algoritmo de decomposição de ciclos. Se as transposições possuem o dobro do peso das reversões, isto é, uma transposição equivale a duas reversões no cálculo da distância genômica, Eriksen [19] mostrou como desenvolver uma $(1 + \varepsilon)$-aproximação para o caso com sinal.

Alguns pesquisadores também consideraram um outro tipo de operação, chamada reversão+transposição ou transreversão. Ela é similar a uma transposição, exceto que um dos segmentos transpostos da permutação é invertido. Formalmente, ela pode ser definida como as operações $tr_a(i, j, k)$ e $tr_b(i, j, k)$, $1 \leq i < j < k \leq n + 1$, sobre uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ tal que $tr_a(i, j, k) \cdot \pi = t(i, j, k) \cdot (r(i, j) \cdot \pi)$ e $tr_b(i, j, k) \cdot \pi = t(i, j, k) \cdot (r(j, k) \cdot \pi)$. Nós dizemos que $tr_a(i, j, k)$ é uma transreversão do tipo A e que $tr_b(i, j, k)$ é uma transreversão do tipo B. Quando uma transreversão age em uma porção da permutação contendo o primeiro elemento, nós dizemos que ela é uma transreversão de prefixo.

Gu *et al.* [26] desenvolveram um algoritmo 2-aproximado para o problema da ordenação de permutações com sinal por reversões, transposições e transreversões do tipo A. Hartman e Sharan [30] apresentaram um algoritmo 1.5-aproximado para o problema da ordenação de permutações com sinal por transposições e transreversões dos tipos A e B.

Recentemente, Sharmina *et al.* [42] propuseram variações do Problema da Ordenação de Panquecas: o problema de ordenação por reversões de prefixos e transposições de prefixos e o problema da ordenação por reversões de prefixo e transreversões (do tipo A) de prefixo. Para este eles apresentaram um algoritmo de aproximação com fator 2 e para aquele, um algoritmo com fator de aproximação 3.

## 1.1 Objetivo

Esta breve apresentação dos problemas clássicos concernentes à área de Rearranjo de Genomas e dos principais avanços obtidos para cada um deles evidencia o quanto ela evoluiu nas últimas duas décadas. Como mostrado, muitos algoritmos aproximativos foram propostos. Por essa razão, nós propomos a construção de uma ferramenta para automatizar e padronizar a avaliação de algoritmos aproximados de rearranjo de genomas que virá a facilitar uma análise quantitativa de tais algoritmos. Este tipo de análise é essencial, por exemplo, para comparar algoritmos aproximativos que possuem o mesmo fator de aproximação ou comparar algoritmos aproximativos com heurísticas, como pode ser averiguado nos trabalhos de Guyer, Heath e Vergara [41], Vergara [44], Walter, Curado e Oliveira [46], Walter *et al.* [47], e Dias e Dias [14].

Nossa ferramenta irá ser focada em algoritmos que resolvem problemas de rearranjo de genomas uni-

cromossomais, limitando-se, portanto, a algoritmos baseados em modelos que não contêm translocações, fusões ou fissões. Para demonstrar sua aplicação, nós iremos utilizá-la para avaliar algoritmos de aproximação e heurísticas que viermos a desenvolver ao longo deste trabalho para resolver o Problema da Ordenação por Transposições de Prefixo. Este problema foi proposto inicialmente na esperança de se obter ideias de como tratar o Problema da Ordenação por Transposições e ainda restam muitas perguntas a serem respondidas a respeito dele, por isso nosso interesse em estudá-lo.

### 1.2 Organização da Proposta

O restante desta proposta está organizada da seguinte forma: a Seção 2 contém detalhes a respeito da ferramenta. A Seção 3 apresenta formalmente o Problema da Ordenação por Transposições de Prefixo e apresenta uma revisão bibliográfica relativa ao mesmo. A Seção 4 apresenta o plano de trabalho e o cronograma. A Seção 5 apresenta a metodologia que será utilizada para o cumprimento de tal plano. Por fim, a Seção 6 discute como os resultados obtidos serão analisados.

## 2 A Ferramenta

Como dito na Introdução, nossa proposta é construir uma ferramenta para automatizar e padronizar a avaliação de algoritmos aproximativos de rearranjo de genomas unicromossomais. Sendo assim, nós precisamos definir dois pontos em relação à ferramenta: como será feita a avaliação dos algoritmos e quais os modelos de rearranjo cobertos.

Algoritmos de rearranjo de genoma têm por objetivo retornar a menor sequência de eventos de rearranjo que ordenam um genoma e, por conseguinte, a distância genômica daquele genoma. Contudo, algoritmos aproximados não garantem que a distância retornada é a distância genômica, mas sim que ela não é maior do que a distância genômica multiplicada por um determinado fator. Portanto, a avaliação deste algoritmos será feita a partir de estatísticas obtidas comparando-se, para todos os genomas que possuem um tamanho não maior do que $N$, a distância retornada pelo algoritmo de aproximação e a distância genômica. As estatísticas devem ser agrupadas pelo tamanho dos genomas e estão listadas a seguir:

- **Distância média**: média aritmética das distâncias retornadas pelo algoritmo;
- **Diâmetro**: valor da maior distância retornada pelo algoritmo;
- **Fator médio**: média aritmética dos fatores de aproximação calculados para cada genoma;
- **Fator máximo**: valor do maior fator de aproximação produzido pelo algoritmo;
- **Igualdade**: porcentagem de genomas para os quais a distância retornada pelo algoritmo é igual a distância genômica.

As distâncias genômicas serão obtidas por meio de um algoritmo de busca em largura, que possui a seguinte estrutura: inicialize uma fila de permutações $F$ com a permutação $I_n$ e defina sua distância como 0. Enquanto $F$ não estiver vazia, remova uma permutação $\pi$ de $F$, imprima $\pi$ e $d(\pi)$, e compute todas as permutações que podem ser obtidas a partir de $\pi$ aplicando sobre ela todas as operações cobertas pelo modelo em consideração. Aquelas que ainda não tiverem sido geradas são adicionadas em $F$ e suas distâncias são definidas como $d(\pi) + 1$.

Dado que um genoma unicromossomal pode ser representado por uma única permutação, seu tamanho equivale ao tamanho da permutação que o representa. Existem $n!$ permutações distintas de tamanho $n$ para o caso em que a orientação dos genes é desconhecida; para o caso em que a orientação dos genes é conhecida, existem $n!2^n$ permutações. Isso significa que o algoritmo de busca em largura deve calcular a distância genômica de $\sum_{i=1}^{N} i!$ e $\sum_{i=1}^{N} i!2^i$ permutações respectivamente. Logo, nós adotaremos $N = 13$ e $N = 10$ para os respectivos casos, pois valores de $N$ maiores do que estes fazem com que o valor das somas ultrapasse as dezenas de bilhões de permutações, tornando impeditivo tanto a execução do algoritmo de busca em largura quanto a avaliação quantitativa dos algoritmos.

A fim de desonerar os usuários de terem que calcular as distâncias genômicas todas as vezes que quiserem utilizar a ferramenta (ou mesmo poupar espaço em disco caso as distâncias genômicas fossem armazenadas após serem calculadas), esta será implementada sob uma arquitetura cliente-servidor. A ideia é armazenar as distâncias genômicas em um servidor, que as disponibilizará como um serviço web. Nós implementaremos um cliente que tornará transparente ao usuário a utilização deste serviço, de tal forma que o único esforço do usuário será direcionado à implementação do algoritmo que deseja avaliar.

Os modelos de rearranjo cobertos pela ferramenta serão aqueles abordados pela literatura (vide Seção 1) que se aplicam apenas à genomas unicrossomais e cujo problema relacionado não possua solução polinomial exata. Isso significa que, para permutações com sinal, consideraremos os modelos que cobrem: reversões; reversões de prefixo; reversões e transposições; reversões e transreversões do tipo A; transposições e transreversões dos tipos A e B. Para permutações sem sinal, consideraremos os modelos que cobrem: reversões; reversões de prefixo; transposições; transposições de prefixo; reversões e transposições. Cabe ressaltar que a ferramenta é totalmente flexível e pode facilmente receber extensões para cobrir qualquer outro modelo que se aplique apenas a genomas unicromossomais.

## 3 Problema da Ordenação por Transposições de Prefixo

A operação de transposição de prefixo é uma transposição em que um dos trechos trocados é um prefixo. Formalmente, ela pode ser definida como um operação $\rho(j, k)$ sobre uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ tal que $\rho(j, k) \cdot (\underline{\pi_1\ \ldots\ \pi_{j-1}}\ \underline{\pi_j\ \ldots\ \pi_{k-1}}\ \pi_k\ \ldots\ \pi_n) = (\underline{\pi_j\ \ldots\ \pi_{k-1}}\ \underline{\pi_1\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n)$, sendo $2 \leq j < k \leq n + 1$.

O problema da ordenação por transposições de prefixo foi introduzido por Dias e Meidanis [15]. Eles apresentaram dois algoritmo aproximativos, um com fator de aproximação 3 e outro com fator de aproximação 2, para resolver tal problema; apresentaram um limite inferior de $\frac{n}{2}$ e um limite superior

de $n-1$ para o diâmetro da distância de transposição de prefixo, denotado por $D_{tp}(n)$; desenvolveram um algoritmo (sem, contudo, apresentar uma prova de corretude) que ordena a permutação reversa (isto é, a permutação identidade invertida) de tamanho $n$, denotada por $R_n$, com $n - \lfloor \frac{n}{4} \rfloor$ transposições de prefixo; conjecturaram que $D_{tp}(n) = d_{tp}(R_n) = n - \lfloor \frac{n}{4} \rfloor$, sendo $d_{tp}(R_n)$ a distância de transposição de prefixo da permutação reversa de tamanho $n$; por fim, exibiram um algoritmo que verifica se uma permutação $\pi$ pode ser ordenada com um número de transposições de prefixo igual ao limite inferior baseado em *breakpoints*.

Para desenvolver um algoritmo 3-aproximado para o problema da ordenação por transposições de prefixo, Dias e Meidanis [15] primeiramente demonstraram que a ação de uma transposição pode ser sempre simulada por, no máximo, duas transposições de prefixo. Com isso, um algoritmo aproximativo com fator de aproximação 3 surge trivialmente a partir do algoritmo aproximativo com fator de aproximação $\frac{3}{2}$ desenvolvido por Bafna e Pevzner [4].

Para desenvolver um algoritmo aproximativo com fator de aproximação 2, os autores primeiramente fixaram um limite inferior para o problema baseando-se no conceito de *breakpoints*. Dada uma permutação $\pi$ de tamanho $n$, um *breakpoint* é definido como uma posição $i$ de $\pi$ tal que $\pi_{i+1} - \pi_i \neq 1$, $2 \leq i \leq n$. Por definição, a posição 1 (início da permutação) é sempre um *breakpoint* e a posição $n+1$ (fim da permutação) é um *breakpoint* se $\pi_n \neq n$. O número de *breakpoints* de uma permutação $\pi$ é denotado por $b(\pi)$.

Demonstrando que uma transposição de prefixo elimina, no máximo, 2 *breakpoints* e que a permutação identidade é a única que possui apenas 1 *breakpoint*, eles mostraram que $d_{tp}(\pi) \geq \lceil \frac{b(\pi)-1}{2} \rceil$ para qualquer permutação $\pi$. Em seguida, eles provaram que é sempre possível aplicar uma transposição de prefixo que elimina pelo menos 1 *breakpoint* de uma permutação $\pi$, concluindo que é possível ordenar uma permutação com $b(\pi) - 2$ transposições de prefixo no pior caso, já que a última transposição de prefixo de uma sequência que ordena $\pi$ sempre elimina 2 *breakpoints*. A partir desses resultados, um algoritmo 2-aproximado para o problema da ordenação por transposições de prefixo é obtido trivialmente.

Para demonstrar que $\frac{n}{2} \leq D_{tp}(n) \leq n-1$, Dias e Meidanis [15] partiram de dois resultados conhecidos. Bafna e Pevzner [4] demonstraram que $\frac{n}{2} \leq D_t(n) \leq \frac{3n}{4}$, sendo $D_t(n)$ o diâmetro da distância de transposição de permutações de tamanho $n$. Dado que a distância de transposição de prefixo de uma permutação é sempre maior ou igual à distância de transposição desta mesma permutação, pois uma transposição de prefixo é uma transposição, temos que $D_{tp}(n) \geq D_t(n)$. Usando o resultado de Aigner e West [1] que diz que o diâmetro da distância de rearranjo sob o modelo que considera apenas inserções do primeiro elemento, isto é, transposições de prefixo do tipo $\rho(2, k)$, é $n-1$, eles concluíram que $D_{tp}(n) \leq n-1$.

Fortuna e Meidanis [23] apresentaram um novo algoritmo, baseado no algoritmo proposto por Dias e Meidanis [15], para ordenar $R_n$, $n \geq 4$, com $\lceil \frac{3n}{4} \rceil$ transposições de prefixo. A partir de sua prova de corretude, demonstrou-se definitivamente que $d_{tp}(R_n) \leq \lceil \frac{3n}{4} \rceil$ para $n \geq 4$. Além da permutação reversa,

considerada uma das permutações mais difíceis de se ordenar por transposições de prefixo, Fortuna [22] estudou permutações *fáceis*. Uma permutação $\pi$ é classificada como fácil se $d_{tp}(\pi) = \lceil \frac{b(\pi)-1}{2} \rceil$. Após identificar e provar algumas propriedades de tais permutações, ele pôde demonstrar que, dada uma permutação fácil $\pi$, $d_{tp}(\pi) = d_t(\pi)$, sendo $d_t(\pi)$ a distância de transposição da permutação $\pi$. Tal resultado representa um ponto de intersecção entre o problema da ordenação por transposições de prefixo e o problema da ordenação por transposições, o que evidencia a relevância de se estudar aquele como meio de se obter respostas para este.

Chitturi e Sudborough [10] melhoraram os limites inferior e superior do diâmetro da distância de transposição de prefixo. Ao invés de utilizar *breakpoints* para delimitar a distância de transposição de prefixo de uma permutação, Chitturi e Sudborough [10] basearam-se em outros conceitos. Definindo-se uma permutação $\pi$ de tamanho $n$ como $\pi = (0\ 1\ \ldots\ n-1)$, dois elementos adjacentes $\pi_i$ e $\pi_{i+1}$ formam uma *adjacência* se $\pi_{i+1} = \pi_i + 1 \pmod n$ e formam uma *antiadjacência* se $\pi_{i+1} = \pi_i - 1 \pmod n$. Seja $(\pi_i, \pi_{i+1}, \ldots, \pi_j)$ uma *sublista* da permutação $(\pi_1, \pi_2, \ldots, \pi_n)$ da posição $i$ até a posição $j$. Um *clã* é uma sublista maximal de antiadjacências e um *bloco* é uma sublista maximal de adjacências. Um *singleton* é um elemento que não forma uma adjacência com seus vizinhos. *Singletons* e blocos também são referidos como *objetos*.

Para se ordenar uma permutação, é preciso tanto diminuir o número de objetos, quanto o número de clãs (a permutação identidade possui um único objeto e nenhum clã). Sabendo que uma transposição de prefixo pode formar até duas adjacências e que o número de objetos é sempre decrescido em unidade quando uma adjacência é formada, temos que seria possível diminuir o número de objetos em duas unidades a cada transposição de prefixo. Contudo, Chitturi e Sudborough [10] observaram que clãs de tamanho maior ou igual a 3 incorrem um *desperdício*, isto é, incorrem a necessidade de se aplicar transposições de prefixo que não formam duas adjacências. Sendo assim, parece razoável pensar que o número mínimo de transposições de prefixo necessárias para ordenar uma permutação é condicionado pelo número de objetos e pela existência de clãs de tamanho maior ou igual a três.

Dada uma permutação $\pi$, seja $\Upsilon(\pi)$ o conjunto de todos os clãs de tamanho pelo menos 3 em $\pi$. Definindo $s(\pi)$ como o número de objetos de $\pi$ e $w(\pi) = \frac{1}{3}(\sum_{C \in \Upsilon(\pi)} |C| - 2)$ como medida do desperdício intrínseco à permutação $\pi$, Chitturi e Sudborough [10] demonstraram que $d_{tp}(\pi) \geq \frac{1}{2}(\mathrm{s}(\pi) + w(\pi))$. Aplicando este resultado à permutação reversa, eles concluíram que $d_{tp}(R_n) \geq \frac{2n}{3}$, logo $D_{tp}(n) \geq \frac{2n}{3}$.

Para demonstrar um novo limite superior para o diâmetro da distância de transposição de prefixo, Chitturi e Sudborough [10] demonstraram que, para qualquer permutação $\pi$, existe uma sequência de $k$ transposições de prefixos que produz pelo menos $k+1$ adjacências, sendo $k \leq \frac{7(n-3)}{8}$. Portanto, no pior caso, temos que são produzidas $\frac{7(n-3)}{8} + 1$ adjacências a cada $\frac{7(n-3)}{8}$ transposições de prefixo. Assim, a ordenação de uma permutação de tamanho $n$ induz a relação de recorrência $T(n) = \frac{7(n-3)}{8} + T(n - (\frac{7(n-3)}{8} + 1)) = n - \log_8 n$. Logo, $D_{tp}(n) \leq n - \log_8 n$.

Labarre [34] foi a último a apresentar resultados para o problema da ordenação por transposições

de prefixo. Para tanto, ao invés de usar alguma das abordagens anteriores para limitar a distância de transposição de prefixo de uma permutação, ele se baseou em conceitos da teoria de grupos de permutação ( [35,37,48]).

Em teoria de grupos de permutação, uma *permutação* é uma função bijetora de um conjunto finito $E$ nele mesmo. O *grupo simétrico* $S_n$ é o grupo formado por todas as permutações sobre $E = \{1, 2, \ldots, n\}$ juntamente com a operação de composição de funções $\circ$, aplicada da direita para a esquerda. O grafo $\Gamma(\pi)$ de uma permutação $\pi$ de tamanho $n$ é o grafo $G(V, E)$ tal que $V = \{1, 2, \ldots, n\}$ e $E = \{(i, j) : \pi_i = j\}$. Tal grafo pode ser decomposto univocamente em ciclos disjuntos, levando à uma representação alternativa de $\pi$ baseada em sua decomposição em ciclos disjuntos. Como exemplificado por Labarre [34], a representação por ciclos disjuntos da permutação $\pi = (4\ 1\ 6\ 2\ 5\ 7\ 3)$ é $\pi = (1, 4, 2)(3, 6, 7)(5)$ (note que, ao contrário das permutações, os elementos de um ciclo são separados por vírgula). O número de ciclos em $\Gamma(\pi)$ é denotado por $c(\Gamma(\pi))$ e o tamanho do ciclo é dado pelo número de elementos que ele contém. Uma permutação $\pi$ é dita *par* se o número de ciclos de tamanho par em $\Gamma(\pi)$ é par. O *grupo alternado* $A_n$ é o grupo formado por todas as permutações pares de $S_n$.

Em um trabalho anterior, Doignon e Labarre [16] introduziram o mapeamento $f : S_n \to A_{n+1} : \pi \mapsto \overline{\pi} = (0, \pi_n, \pi_{n-1}, \ldots, \pi_1) \circ (0, 1, \ldots, n)$. Lançando mão de $f$, Labarre [34] demonstrou que para qualquer permutação $\pi \in S_n$,

$$d_{tp}(\pi) \geq \frac{n + 1 + c(\Gamma(\overline{\pi}))}{2} - c_1(\Gamma(\overline{\pi}) - \begin{cases} 0 & \text{se } \pi_1 = 1 \\ 1 & \text{caso contrário} \end{cases}$$

sendo $c_1(\Gamma(\overline{\pi}))$ o número de ciclos de tamanho 1 em $\Gamma(\overline{\pi})$. Aplicando esse resultado a uma permutação $\pi$ da forma $\pi = (3\ 2\ 1\ 4\ 7\ 6\ 5 \ldots n - 4\ n\ n - 2\ n - 3)$, Labarre [34] demonstrou que $d_{tp}(\pi) \geq \lfloor \frac{3n+1}{4} \rfloor$. Consequentemente, $D_{tp}(n) \geq \lfloor \frac{3n+1}{4} \rfloor$.

Uma estrutura não muito explorada pelos trabalhos citados anteriormente[2], mas que possui um papel fundamental no estudo de problemas de ordenação de genomas, é o *grafo de ciclos*. Introduzido por Bafna e Pevzner [4], o grafo de ciclos de uma permutação $\pi \in S_n$, denotado por $G(\pi)$, é um grafo bicolorido direcionado cujos vértices são dados pelo conjunto $\{0, 1, \ldots, n + 1\}$ e cujas arestas são definidas, para todo $1 \leq i \leq n + 1$, da seguinte maneira: arestas *cinzas* são direcionadas de $i - 1$ a $i$; e arestas *pretas*, de $\pi_i$ a $\pi_{i-1}$. Um *ciclo alternado* de $G(\pi)$ é um ciclo direcionado no qual arestas adjacentes possuem cores distintas. Sabendo que um grafo de ciclos possui no máximo $n + 1$ ciclos alternados e que a única permutação a possuir $n + 1$ ciclos alternados é $I_n$, ordenar uma permutação $\pi$ significa, idealmente, aplicar operações (transposições de prefixo neste caso) que aumentam o número de ciclos alternados de $G(\pi)$.

[2]Ela foi explorada indiretamente por Labarre [34], pois ele demonstrou que $\Gamma(\overline{\pi})$ e $G(\pi)$ são equivalentes para uma mesma permutação $\pi$.

# 4 Cronograma e Plano de Trabalho

O cronograma das atividades é apresentado na Tabela 1 e compreende o período de Agosto de 2010 a Julho de 2012, totalizando 24 meses. Para que o objetivo deste trabalho seja alcançado, as tarefas listadas abaixo devem ser cumpridas. Cabe destacar que o levantamento bibliográfico foi realizado em um período anterior ao ingresso do aluno na pós-graduação, em uma disciplina de estudo dirigido realizada durante a graduação.

| | 2010 | | | | | 2011 | | | | | | | | | | | | 2012 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 | | | | | | | | | | | | | | | | | | | | | | | | • |

Tabela 1: Cronograma das atividades - Setembro/2010 até Fevereiro/2012

1. Obtenção dos créditos obrigatórios e defesa do Exame de Qualificação do Mestrado (EQM).
2. Implementação do algoritmo de busca em largura para calcular as distâncias genômicas.
3. Cálculo das distâncias genômicas relativas ao modelos descritos na Seção 2.
4. Implementação da ferramenta.
5. Estudo de novas famílias de permutações que podem ser ordenadas por transposições de prefixo em tempo polinomial.
6. Obtenção de novos algoritmos aproximativos e heurísticas para o Problema de Ordenação por Transposições de Prefixo.
7. Avaliação, utilizando a ferramenta, dos algoritmos produzidos.
8. Escrita da dissertação.
9. Revisão final do texto da dissertação.
10. Defesa da dissertação.

# 5 Metodologia

Nós já desenvolvemos o algoritmo de busca em largura para calcular as distâncias genômicas referentes aos modelos mencionados na Seção 2. A principal preocupação ao implementá-lo foi como representar as permutações da forma mais concisa possível. A abordagem que nós utilizamos foi converter cada permutação a um número natural único, que podemos chamar de índice. Isso pode ser realizado em tempo linear ao tamanho da permutação graças aos algoritmos de indexação e desindexação apresentados por Myrvold e Ruskey [38]. Um algoritmo de indexação atribui um índice pertencente ao intervalo $[0,\ n! - 1]$ a uma permutação sem sinal de tamanho $n$, enquanto que o algoritmo de desindexação realiza a operação inversa. Para uma permutação com sinal de tamanho $n$, é possível atribuir um índice pertencente ao intervalo $[0,\ n!2^n - 1]$ à ela da seguinte maneira: calcule o seu índice como se ela fosse sem sinal; multiplique o índice calculado por $2^n$ e some o resultado com o valor dos sinais daquela permutação. Nós definimos o valor do sinais de uma permutação com sinal como o número dado pela representação binária dos sinais daquela permutação, sendo que + e − equivalem a 0 e 1 respectivamente. Por exemplo, 1100 é a representação binária dos sinais da permutação ($-1$ $-2$ $+3$ $+4$), portanto o valor dos sinais dessa permutação é 12.

Duas versões do algoritmo foram implementadas em C, utilizando a biblioteca *pthread* para cuidar da criação e gerenciamento de *threads*: uma que representa os índices com 32 bits e outra com 64 bits. A primeira é capaz de indexar até $2^{32}$ permutações, ou seja, capaz de calcular as distâncias genômicas de todas as permutações sem sinal com até 12 elementos e de todas as permutações com sinal com até 10 elementos. A segunda é capaz de indexar até $2^{64}$ permutações, isto é, capaz de calcular as distâncias genômicas de todas as permutações sem sinal com até 20 elementos e de todas as permutações com sinal com até 16 elementos.

Executando a primeira versão implementada em um computador do Instituto de Computação da UNICAMP possuindo 8GB de memória RAM, nós conseguimos calcular as distâncias genômicas de todas as permutações sem sinal com até 12 elementos e de todas as permutações de com sinal com até 9 elementos referentes aos modelos citados na Seção 2. Resultados parciais obtidos dessa execução foram publicados nos anais do *26th Symposium on Applied Computing* da ACM [24]. Para calcular as distâncias genômicas de todas as permutações sem sinal com 13 elementos, nós precisaríamos de um computador com pelo menos 52GB de RAM e para calcular as distâncias genômicas de todas as permutações com sinal com 10 elementos, de um computador com pelo menos 38GB de RAM. Sendo assim, recorremos a um computador localizado na Embrapa para calcular estas e a um computador localizado no *Virginia Bioinformatics Institute* para calcular aquelas. Atualmente, resta apenas calcular as distâncias genômicas das permutações sem sinal de tamanho 13 referentes ao modelo que cobre reversões e transposições. É esperado que este cálculo termine em meados de Abril.

Uma primeira versão da ferramenta deverá ficar pronta até o final de Abril e prentendemos realizar testes nos meses de Maio e Junho. Ela está sendo implementada em Java. Utilizamos o *framework* Apache Axis2 para auxiliar na construção dos serviços web.

Todos os algoritmos que desenvolvermos a partir de agora serão implementados em Java. Nós já implementamos o algoritmo de Dias e Meidanis [15] e o algoritmo de Fortuna e Meidanis [23] para ordenar a permutação reversa. Também já implementamos algoritmos que calculam boa parte das propriedades de uma permutação mencionadas na Seção 3. Juntas, estas implementações constituirão uma ferramenta fundamental para a análise das propriedades de uma permutação que são relevantes ao Problema da Ordenação por Transposições de Prefixo.

A fim de armazenarmos, visualizarmos e tornar público os resultados obtidos ao longo deste trabalho, nós criamos uma aplicação web, que pode ser acessada em:

`http://mirza.ic.unicamp.br:8080/bioinfo`

Ela está em processo de construção e atualmente já é possível visualizar os resultados obtidos pela execução do programa que gera as distância genômicas. Utilizamos o *framework* JSF para desenvolvê-la e o PostgreSQL como Sistema Gerenciador de Banco de Dados. Ela está hospedada em um servidor do Instituto de Computação da UNICAMP rodando o Apache Tomcat 6.0.

Para obtermos novas famílias de permutações que podem ser ordenadas por transposições de prefixo em tempo polinomial, a ideia é primeiramente encontrar famílias simples, como por exemplo a família das permutações reversas, e depois tentar agrupá-las em famílias mais complexas. Um método que pode trazer bons resultados na busca de famílias simples é analisar propriedades em comum entre as permutações. Outra método que parece promissor para tal fim é analisar a sequência de permutações que compõe a transformação de uma permutação à permutação identidade no processo de ordenação e tentar encontrar similaridades ou padrões nesta sequência. A aplicação web possibilitará tal análise.

Uma vez que tivermos encontrado as famílias, a ideia é que elas sirvam de base para construírmos algoritmos de aproximação e heurísticas para o Problema da Ordenação por Transposições de Prefixo. Também pretendemos estudar a fundo o algoritmo 2-aproximado desenvolvido por Dias e Meidanis [15] e tentar melhorá-lo. Adotaremos uma metodologia teórica similar à descrita brevemente na Seção 3 para demonstrar as aproximações que viermos a obter. Os algoritmos de aproximação e heurísticas obtidos serão avaliados pela ferramenta conforme forem sendo desenvolvidos para que possamos verificar se a direção correta está sendo tomada.

## 6 Análise dos Resultados

Uma análise de complexidade será realizada para todos os algoritmos produzidos. Quando for o caso, uma prova da aproximação do algoritmo também será fornecida. Além dessas análises qualitativas, todos os algoritmos produzidos, assim como o algoritmo de Dias e Meidanis [15], serão analisados quantitativamente com o auxílio da ferramenta.

## Referências


[1] M. Aigner and D. West. Sorting by insertion of leading elements. *Journal of Combinatorial Theory*, 45(2):306–309, 1987.

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

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

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

[5] A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. *Journal of Computational Biology*, 13(2):567–578, 2006.

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

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

[8] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the 1st Annual International Conference on Computational Molecular Biology (RECOMB'97)*, pages 75–83, Santa Fe, New Mexico, United States, 1997. ACM Press.

[9] B. Chitturi, W. Fahle, Z. Meng, L. Morales, C. Shields, I. Sudborough, and W. Voit. An (18/11)n upper bound for sorting by prefix reversals. *Theoretical Computer Science*, 410(36):3372–3390, 2009.

[10] B. Chitturi and I. H. Sudborough. Bounding prefix transposition distance for strings and permutations. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS'2008)*, page 468, Waikoloa, Big Island, Hawaii, USA, 2008. IEEE Computer Society.

[11] D. A. Christie. Sorting permutations by block-interchanges. *Information Processing Letters*, 60(4):165–169, 1996.

[12] D. A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pages 244–252, San Francisco, California, United States, 1998. Society for Industrial and Applied Mathematics.

[13] D. S. Cohen and M. Blum. On the problem of sorting burnt pancakes. *Discrete Applied Mathematics*, 61(2):105–120, 1995.

[14] U. Dias and Z. Dias. An improved 1.375-approximation algorithm for the transposition distance problem. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology (BCB'10)*, pages 334–337, Niagara Falls, New York, 2010. ACM Press.

[15] Z. Dias and J. Meidanis. Sorting by prefix transpositions. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE'2002)*, pages 65–76, Lisbon, Portugal, 2002. Springer-Verlag.

[16] J-P. Doignon and L. Labarre. On hultman numbers. *Journal of Integer Sequences*, 10(6), 2007. Article 07.6.2.

[17] H. Dweighter. Problem e2569. *American Mathematical Monthly*, 82:1010, 1975.

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

[19] N. Eriksen. $(1 + \varepsilon)$-approximation of sorting by reversals and transpositions. *Theoretical Computer Science*, 289(1):517–529, 2002.

[20] J. Feng and D. Zhu. Faster algorithms for sorting by transpositions and sorting by block interchanges. *ACM Transactions on Algorithms*, 3(3), 2007. Article No. 25.

[21] J. Fischer and S. W. Ginzinger. A 2-approximation algorithm for sorting by prefix reversals. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA'2005)*, pages 415–425, Mallorca, Spain, 2005. Springer, Heidelberg.

[22] V. J. Fortuna. Transposition distances between genomes. Master's thesis, University of Campinas, 2005. In Portuguese.

[23] V. J. Fortuna and J. Meidanis. Sorting the reverse permutation by prefix transpositions. Technical Report IC-04-04, University of Campinas, April 2004.

[24] G. R. Galvão and Z. Dias. Computing rearrangement distace of every permutation in the symmetric group. In *Proceedings of the 26th ACM Symposium on Applied Computing (SAC'2011)*, pages 106–107, Taichung, Taiwan, 2011. ACM Press.

[25] W. Gates and C. Papadimitriou. Bounds for sorting by prefix reversal. *Discrete Mathematics*, 27:47–57, 1979.

[26] Q. Gu, S. Peng, and H. Sudborough. A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theoritical Computer Science*, 210(2):327–339, 1999.

[27] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71(1-3):137–151, 1996.

[28] 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'95)*, pages 581–592, 1995.

[29] 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.

[30] 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.

[31] Y. Huang, C. Huang, C. Tang, and C. Lu. An improved algorithm for sorting by block-interchanges based on permutation groups. *Information Processing Letters*, 110(8-9):345–350, 2010.

[32] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for the inversion distance between two chromosomes. *Algorithmica*, 13:80–110, 1995.

[33] J. D. Kececioglu and R. Ravi. Of mice and men: algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, pages 604–613, San Francisco, California, United States, 1995. Society for Industrial and Applied Mathematics.

[34] A. Labarre. Edit distances and factorisations of even permutations. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA'08)*, pages 635–646, Karlsruhe, Germany, 2008. Springer-Verlag.

[35] J. Meidanis and Z. Dias. An alternative algebraic formalism for genome rearrangements. In David Sankoff and Joseph Nadeau, editors, *Comparative Genomics*, pages 213–223. Kluwer Academic Publishers, 2000.

[36] J. Meidanis and Z. Dias. Genome rearrangements distance by fusion, fission, and transposition is easy. In *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE'2001)*, pages 250–253, Laguna de San Rafael, Chile, 2001. IEEE Computer Society.

[37] B. Miklos. *Combinatorics of Permutations*. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2004.

[38] W. Myrvold and F. Ruskey. Ranking and unranking permutations in linear time. *Information Processing Letters*, 79(6):281–284, 2001.

[39] M Ozery-Flato and R. Shamir. Sorting by reciprocal translocations via reversals theory. *Journal of Computational Biology*, 14(3):408–422, 2007.

[40] 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.

[41] Guyer S. A., Heath L. S., and Vergara J. P. C. Subsequence and run heuristics for sorting by transpositions. Technical Report TR-97-20, Virginia Polytechnic Institute & State University, 1997.

[42] M. Sharmin, R. Yeasmin, M. Hasan, A. Rahman, and M. S. Rahman. Pancake flipping with two spatulas. *Electronic Notes in Discrete Mathematics*, 36:231–238, 2010.

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

[44] J. P. C. Vergara. *Sorting by bounded permutations*. PhD thesis, Blacksburg, VA, USA, 1998.

[45] M. 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, Bolivia, 1998. IEEE Computer Society.

[46] M. E. M. T. Walter, L. R. A. F. Curado, and A. G. Oliveira. Working on the problem of sorting by transpositions on genome rearrangements. In *Proceedings of the 14th Annual Conference on Combinatorial Pattern Matching (CPM'03)*, pages 372–383, Morelia, Michoacán, Mexico, 2003. Springer-Verlag.

[47] M. E. M. T. Walter, M. C. Sobrinho, E. T. G. Oliveira, L. S. Soares, A. G. Oliveira, T. E. S. Martins, and T. M. Fonseca. Improving the algorithm of bafna and pevzner for the problem of sorting by transpositions: a practical approach. *Journal of Discrete Algorithms*, 3(2-4):342–361, 2005.

[48] H. Wielandt. *Finite Permutation Groups*. Translated from German by R. Bercov. Academic Press, New York, 1964.

[49] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

[50] D. Zhu and L. Wang. On the complexity of unsigned translocation distance. *Theoretical Computer Science*, 352(1):322–328, 2006.