# Heurísticas para Problemas de Rearranjo de Genomas

Ulisses Martins Dias

**Resumo**

Rearranjo de genomas é um campo interessado em investigar o parentesco entre organismos calculando o menor número de operações de rearranjo necessárias para transformar um genoma em outro. Dentre os tópicos na área, serão abordadas neste projeto de doutorado heurísticas para os problemas que ainda não possuem solução polinomial. Algumas heurísticas fogem aos conceitos clássicos da área de rearranjo de genomas como, por exemplo, os modelos baseados em Programação Linear Inteira e Programação por Restrições, enquanto outras heurísticas serão aplicados a algoritmos de aproximação existentes para melhorar os resultados apresentados em situações práticas. Outros objetivos serão contemplados: o primeiro deles é o estudo de condições que, se satisfeitas pelos genomas, permitem a obtenção de algoritmos polinomiais. O segundo objetivo é a resolução do problema do diâmetro de transposição, um problema que continua em aberto, apesar de já possuir alguns resultados parciais

## 1 Introdução

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

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

A comparação de seqüências é o método mais usual de se caracterizar a ocorrência de mutações pontuais, sendo um dos problemas mais abordados em biologia computacional. Usualmente o interesse da comparação de seqüências é encontrar a distância de edição [38], que é o número mínimo de operações de inserção, deleção e substituição que transformam uma seqüência em outra.

A distância de edição é uma medida capaz de estimar a distância evolutiva entre duas cadeias. Entretanto, a distância de edição não possui sensibilidade suficiente para capturar operações globais, os chamados rearranjos de genomas, que podem ser de vários tipos como, por exemplo, reversões, transposições, fusões e fissões.

Um conceito de distância pode ser definido para qualquer classe de rearranjo como o menor número de eventos pertencentes a essa classe que são necessários para transformar

um genoma em outro. Por exemplo, chama-se distância de reversão o menor número de reversões necessárias para transformar um genoma em outro [2] e a distância de transposição é, dessa forma, o menor número de transposições [3].

Vários estudos mostram que os rearranjos de genoma são mais apropriados que mutações pontuais quando se deseja comparar os genomas de duas espécies. Os pioneiros foram Dobzhansky e Sturtevant que, em 1938, publicaram um estudo com uma árvore evolucionária que mostrava um cenário de 17 reversões para as espécies *Drosophila pseudoobscura* e *Drosophila miranda* [13].

Em 1988, Palmer e Herbon [37] realizaram a comparação entre os genomas mitocondriais das espécies *Brassica oleracea* e *Brassica campestris* e descobriram que elas são muito similares (muitos genes são entre 99 e 99,9% idênticos). Nesse contexto, a distância evolutiva entre dois genomas pode ser estimada pelo conceito de distância para uma classe de rearranjo definida anteriormente.

Na literatura estão documentados vários avanços no que diz respeito a implementação de métodos computacionais para os vários tipos de rearranjos, a Seção 2 apresenta um breve histórico do que já foi estudado, evidenciando os pontos onde esforços são ainda necessários.

# 2 Rearranjo de Genomas

Os eventos de rearranjo mais comumente estudados são reversões, transposições, translocações, fusões e fissões:

- **Reversões** ocorrem quando um segmento do genoma é destacado e ligado no mesmo lugar com sua ordem invertida em relação a sua posição original.
- **Transposições** envolvem a mudança de posição de dois blocos adjacentes do genoma
- **Translocações** ocorrem quando dois cromossomos $X = (X_1, X_2)$ e $Y = (Y_1, Y_2)$ interagem para formar os cromossomos $X' = (X_1, Y_2)$ e $Y' = (Y_1, X_2)$.
- **Fusões** ocorrem quando dois cromossomos se unem para formar um só cromossomo.
- **Fissões** ocorrem quando um cromossomo se divide e forma dois cromossomos.

Para que seja possível modelar as operações de rearranjo computacionalmente, um genoma é representado como uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, onde $\pi_i$ identifica um gene, ou conjunto de genes, que aparece no genoma. Utilizando esta representação, as operações de reversão e transposição, mencionadas anteriormente e que farão parte do escopo deste projeto de doutorado, são definidadas da seguinte forma:

- **Reversão:** operação em que a ordem de um segmento da permutação é invertida. Dessa forma, ao se aplicar a reversão $\rho(i, j)$, onde $1 \leq i < j \leq n$, a $\pi$ obtém-se: $\rho\pi = (\pi_1\ \ldots\ \pi_{i-1}\ \pi_j\ \pi_{j-1}\ \ldots\ \pi_{i+1}\ \pi_i\ \pi_{j+1}\ \ldots\ \pi_n)$.
- **Transposição:** operação em que um segmento é cortado da permutação e colado em uma posição diferente. Dessa forma, uma transposição $\rho(i, j, k)$, onde $1 \leq i < j < k \leq n + 1$, é uma operação em que dois blocos de genes que estão lado a lado trocam de lugar de modo que o intervalo $[i, j - 1]$ de $\pi$ seja inserido entre $\pi_{k-1}$ e

$\pi_k$. Dessa forma, ao se aplicar a transposição $\rho(i,j,k)$ à permutação $\pi$ obtém-se: $\rho\pi = (\pi_1 \ \ldots \ \pi_{i-1} \ \pi_j \ \ldots \ \pi_{k-1} \ \pi_i \ \ldots \ \pi_{j-1} \ \pi_k \ \ldots \ \pi_n)$.

Os elementos $\pi_i$ são identificados com números inteiros positivos, mas se a orientação dos genes for conhecida, então sinais positivos e negativos são adicionados aos identificadores para indicar essa informação.

Quando comparamos dois genomas, assume-se que um deles é a permutação identidade $\iota = (1 \ 2 \ \ldots \ n)$ e o outro é a permutação $\pi = (\pi_1 \ \pi_2 \ \ldots \ \pi_n)$. A distância será o número mínimo de operações aplicadas à permutação $\pi$ para que ela seja transformada na identidade. Vale ressaltar que dados dois genomas $\pi$ e $\sigma$ diferentes da identidade, a distância entre eles é igual a distância entre $\sigma^{-1}\pi$ e a identidade.

Outro conceito importante na área de rearranjo de genomas é o diâmetro, que pode ser entendido como a maior distância possível entre dois genomas arbitrários e de mesmo tamanho. Até alguns anos acreditava-se que o diâmetro de transposição era igual a distância de ordenação da permutação inversa $\pi = (n \ n-1 \ \ldots \ 2 \ 1)$ [11, 32]. Entretanto, Eriksson e co-autores mostraram que essa conjectura era falsa [15], o que fez com que o problema do diâmetro de transposição permanecesse em aberto.

As seções 2.1 e 2.2 listam uma série de avanços recentes relacionados aos eventos de transposição e reversão respectivamente.

## 2.1 Transposição

Como mencionado anteriormente, transposições são eventos onde dois blocos adjacentes em um mesmo cromossomo trocam de posição. Vale ressaltar que esse evento não troca a orientação dos genes.

O problema da distância de transposição, que envolve encontrar o menor número de transposições necessárias para transformar um genoma em outro, foi estudado por Bafna e Pevzner [3], que apresentou um algoritmo capaz de fornecer uma resposta aproximada na razão de 1.5, além de derivar um importante limitante inferior para o problema.

Dentre as contribuições do trabalho de Bafna e Pevzner [3] é possível citar o conceito de *breakpoints*, elementos adjacentes em um dos genomas, mas não no outro, e uma ferramenta que chamou de grafo direcionado cíclico com arestas de cores alternadas, que fora bastante utilizada em trabalhos posteriores.

Nesse mesmo trabalho, foram apresentados várias questões em aberto como verificar a complexidade do problema da distância de transposição e o diâmetro, que é a maior distância possível entre duas permutações de tamanho $n$. O problema do diâmetro foi estudado por Meidanis, Walter e Dias [32] com a apresentação de alguns resultados parciais.

O algoritmo de Bafna e Pevzner possui o contratempo de ser muito complexo, além de, segundo Christie [11], omitir detalhes técnicos importantes para sua implementação e a complexidade de tempo de $O(n^2)$, afirmada por Bafna e Pevzner, não pode ser atingida na prática.

Nesse contexto, Walter, Dias e Meidanis desenvolveram um algoritmo simples e que executa em $O(n^2)$, porém seu fator de aproximação é de 2.25 [43] e Christie produziu um algoritmo com fator de aproximação 1.5 mais simples de entender que o de Bafna e Pevzner e que executa em $O(n^4)$ [11]

Atualmente não se conhece nenhuma prova de que o problema da distância de transposição seja NP-Difícil, sendo que também não existem evidências da existência de um algoritmo polinomial, o que torna o estudo do problema interessante. Recentemente, Elias e Hartman [14] propuseram um novo algoritmo de aproximação na razão de 1.375, um avanço na razão de aproximação que não ocorria desde a publicação do trabalho de Bafna e Pevzner [5] 8 anos antes.

Caso algumas restrições forem impostas ao problema é possível implementar algoritmos polinomiais ou com um melhor fator de aproximação. Por exemplo, Jerrum apresentou um algoritmo polinomial para transposições que trocam apenas pares de genes adjacentes [27] e Heath e Vergara apresentaram um algoritmo polinomial para transposições onde um dos blocos possui sempre tamanho 1 [24].

Em seu trabalho, Labarre [30] apresentou limitantes utilizando o grafo de ciclos, uma variante do grafo da permutação e as equivalências toroidais [15]. Além disso, o trabalho de Labarre define classes de permutações em que a distância de transposição pode ser calculada em tempo e espaço lineares.

Basicamente, Labarre segue a idéia de que, dada a dificuldade do problema, é mais simples compartimentalizá-lo em várias classes menores que podem ser resolvidas polinomialmente. Essa idéia fora seguida anteriormente por outros pesquisadores e, dentre os resultados importantes, é possível citar Fortuna [17], que mostrou que um subconjunto de permutações, conhecidas como permutações fáceis, pode ser resolvido em tempo polinomial. Entretanto, o escopo desse subconjunto é muito inferior ao número de permutações possíveis.

Uma generalização interessante para o problema de transposição é a operação de troca de blocos, um evento que não ocorre realmente na natureza, mas cuja solução pode auxiliar na resolução do problema da distância de transposição, já que a distância de troca de blocos entre dois genomas é um problema com solução polinomial para qualquer permutação [10].

Dada uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, uma troca de blocos $\rho(i, j, k, l)$, onde $1 \leq i < j \leq k < l \leq n + 1$ é uma operação em que dois blocos $[\pi_i\ \ldots\ \pi_{j-1}]$ e $[\pi_k\ \ldots\ \pi_{l-1}]$ são trocados de posição, em outras palavras, a troca de blocos $\rho(i, j, k, l)$ aplicada ao genoma $\pi$ gera o genoma $\pi' = (\pi_1\ \ldots\ \pi_{i-1}\ \pi_k \ldots \pi_{l-1}\ \pi_j \ldots \pi_{k-1}\ \pi_i \ldots \pi_{j-1}\ \pi_l \ldots \pi_n)$. O caso especial da transposição ocorre quando $j = k$. Christie [10] apresentou um algoritmo polinomial para o problema e utilizou, na implementação do algoritmo, conceitos já conhecidos na literatura sobre transposição.

## 2.2 Reversão

O evento de reversão, nome dado quando um bloco do genoma é invertido, é a classe de rearranjos que possui os melhores resultados. Um estudo inicial acerca do problema da distância de reversão foi apresentado em 1993 por Bafna e Pevzner [2], que criaram um algoritmo de aproximação com razão de 1.5 quando a orientação dos genes é conhecida e 1.75 caso contrário.

No problema de reversão a orientação dos genes é uma informação importante, pois quando não se conhece a orientação há uma prova de que o problema da distância de reversão é NP-Difícil [8], enquanto que se a orientação for conhecida existem algoritmos polinomiais.

O primeiro algoritmo polinomial para o problema de reversão com a orientação dos genes conhecida foi criado por Hannenhalli e Pevzner [20]. Posteriormente, a estratégia de Hannenhalli e Pevzner foi simplificada por Bergeron [6]. Atualmente, já existe um algoritmo com complexidade sub-quadrática [39] e, quando apenas a distância é necessária, um algoritmo linear pode ser usado [1].

Um resultado importante obtido por Meidanis, Walter e Dias [33] mostrou que toda teoria sobre reversões desenvolvida para genomas lineares pode ser adaptada facilmente para genomas circulares, que são comuns em seres inferiores como vírus e bactérias.

Quando a orientação dos genes não é conhecida existem algoritmos de aproximação que se seguiram ao de Bafna e Pevzner como, por exemplo, o algoritmo implementado por Berman, Hannenhalli e Karpinski [7] com razão de aproximação de 1.375.

Como na natureza não é razoável imaginar que um genoma sofrerá apenas eventos de reversões ou de transposições, um estudo acerca da distância entre dois genomas sendo possíveis diferentes tipos eventos é interessante.

Nesse contexto e no que diz respeito apenas aos eventos de reversão e transposição existe o trabalho de Hannenhalli e co-autores [19], que analisaram a evolução de genomas por diferentes eventos, mas particularmente reversões e transposições, e de Gu, Peng e Sudborough [18] que criaram um algoritmo de aproximação para computar a distância entre dois genomas com a orientação dos genes conhecida permitindo operações de reversão, transposição e reversão+transposição simultaneamente.

Em 1998, Walter, Dias e Meidanis [42] apresentaram um algoritmo de aproximação para a distância de reversão e transposição, além de limitantes para o diâmetro de reversão e transposição em permutações com sinal. Posteriormente, Meidanis, Walter e Dias criaram limitantes inferiores melhores para os diâmetros de transposição e reversão [34].

Em geral, quando se lida com vários eventos de rearranjo deve se levar em conta que eles ocorrem com freqüências diferentes na natureza, o que induz a inserção de pesos relacionados à ocorrência destes eventos. O valor que deve ser dados a esses pesos é, então, um novo problema a ser considerado.

## 2.3 Considerações Finais

Os trabalhos em rearranjo de genomas apresentam várias abordagens indo desde a clássica, que segue a linha de raciocínio dos trabalhos de Pevzner [2–4, 20] ao utilizar ferramentas como o grafo de breakpoints, passando pelo formalismo algébrico de Meidanis e Dias [31], criado com o intuito de diminuir a dependência de representações visuais ao se expor argumentos, provas e teoremas em rearranjo de genomas.

Bergeron [6] apresentou um algoritmo mais simples que o anterior de Hannenhalli e Pevzner [20] e utiliza o conceito de pares orientados, reversões induzidas por esses pares e intervalos fechados, que não haviam sido implementados anteriormente. Dessa forma, a abordagem de Bergeron também difere da teoria dita "clássica", podendo ser igualmente chamada de uma abordagem alternativa.

Uma linha de pesquisa que realmente se afasta dos paradigmas já consolidados na área de rearranjo de genomas é a modelagem do problema com programação linear inteira realizada por Dias e Souza [12], que criaram um modelo para as distâncias de reversão e transposição. Entretanto, alguns estudos ainda precisam ser feitos para viabilizar o modelo.

Uma das idéias deste projeto de doutorado é investigar abordagens alternativas já consolidadas em outras áreas da computação para propiciar novos avanços na área de rearranjo de genomas e uma abordagem seria utilizar os conhecimentos e ferramentas disponíveis na área de otimização combinatória.

# 3 Estágio no Exterior

Dentro do escopo do Projeto de Doutorado, programamos um período de um ano de estágio no exterior (bolsa sanduíche a ser solicitada para a CAPES). O estágio tem previsão de início em setembro de 2009 e será desenvolvido junto ao grupo de pesquisa coordenado pelo professor Dr. João Carlos Setubal nas dependências do *Virginia Bioinformatics Institute*, localizado no *Virginia Tech*, Virgínia, Estados Unidos.

O grupo de pesquisa coordenado pelo professor Dr. João Setubal objetiva desenvolver métodos computacionais para problemas em biologia molecular, visando trabalhar problemas computacionais gerados pela extensa quantidade de dados derivada de projetos de sequenciamento, com ênfase em genoma bacterial. No ano de 2008, o grupo trabalhou com três projetos de genomas bacteriais: *Agrobacterium biovars*, *Azotobacter vinelandii* e *Pseudomonas syringae*.

*Agrobacterium* é um gênero da família *Rhizobiaceae*, pertencente à ordem *Rhizobiales*, classe *$\alpha$-proteobacteria*, que inclui a espécie *Agrobacterium tumefaciens*, que causa patogenias em plantas. O objetivo do estudo envolve entender melhor o mecanismo patogênico e evolutivo das espécies pertencentes a esse gênero mediante a comparação de genomas.

A experiência do grupo com bactérias da ordem *Rhizobiales* motivou a proposta de pesquisa "*Ancestral Bacterial Genome Reconstruction*" a ser desenvolvida por Kuan Yang, aluno de doutorado orientado pelo professor Setubal e que participa do programa "*Genetics, Bioinformatics and Computational Biology*" - GBCB.

De uma forma resumida, Kuan Yang pretende desenvolver novos métodos computacionais para reconstrução de genomas ancestrais de espécies de bactérias. A pesquisa visa fornecer um entendimento melhor da base genômica para as diferenças observáveis entre as bactérias.

A proposta utilizará vários métodos, alguns deles baseados em rearranjo de genomas. Dessa forma, o estágio no exterior objetiva associar o conhecimento teórico em rearranjo de genomas desenvolvido durante o doutorado a uma abordagem prática utilizando um conjunto de genomas sequenciado para um grupo particular de bactérias.

De uma forma mais específica, pretende-se participar do projeto de pesquisa que iniciará um estudo para desenvolver novos métodos computacionais visando reconstruir o histórico evolutivo de bactérias tendo como base o conteúdo e a ordem dos genes presentes nos genomas. Em nosso caso, estamos principalmente interessados em métodos que permitam a comparacão da ordem dos genes em diferentes genomas sujeitos a rearranjos. Dado um grupo de genomas completos, encontrar para cada genoma ancestral na árvore filogenética uma ordem para os genes e um conjunto de cenários evolutivos plausíveis.

Inicialmente pretendemos utilizar o trabalho sobre o evento de transposição que estamos desenvolvendo e que já possui resultados parciais (Seção 7), em conjunto com os trabalhos sobre o evento de reversão que pretendemos iniciar, de acordo com os prazos estabelecidos em nosso cronograma (Seção 5).

Este trabalho inicial consiste em utilizar os algoritmos desenvolvidos para calcular uma matriz de distância entre os pares de genomas e utilizar técnicas computacionais como, por exemplo, o algoritmo do vizinho mais próximo para gerar uma árvore filogenética. Tal abordagem não é precisa o suficiente para gerar resultados confiáveis, mas é rápida e fornece várias pistas que serão úteis nos trabalhos seguintes.

Um problema encontrado na abordagem utilizando a matriz de distância é que a mesma pode ser utilizada para gerar a árvore filogenética guia, mas não contribui para reconstruir a ordem dos genes nos genomas ancestrais, o que exigirá a a utilização de métodos para rearranjo múltiplo de genomas, como será visto mais adiante, nesta seção.

Como estamos trabalhando com algoritmos de aproximação (heurísticas aplicadas ao trabalho de Bafna e Pevzner [5]) e com uma abordagem exata baseada em programação em lógica com restrições, pretendemos analisar o comportamento de ambas quando aplicadas a genomas reais, pois até agora estamos utilizando apenas permutações pequenas e selecionadas aleatoriamente.

Dentre outros objetivos, o trabalho pretende responder as seguintes dúvidas:

- A abordagem exata baseada em programação lógica com restrições pode ser aplicada a genomas reais e obter resposta em um tempo razoável?
- Os algoritmos de aproximação geram árvores filogenéticas confiáveis?

Vale ressaltar que até o momento não existem algoritmos polinomias exatos para o problema da distância de transposição e não existe nem uma prova de que o problema é NP-difícil.

Uma segunda abordagem pretende utilizar outras técnicas presentes na literatura para geração de resultados mais confiáveis, o que inclui encontrar estimativas estatísticas para a distância evolutiva, tendo em vista a obtenção de um melhor entendimento do problema.

As estimativas estatísticas são importantes porque a distância em rearranjo de genomas procura o menor número de operações que transformam um genoma em outro, o que tende a subestimar a real distância evolutiva, principalmente se os genomas em questão pertencem a espécies muito divergentes.

Vários modelos estatísticos existem para estimar a distância evolutiva e pretendemos comparar a qualidade das árvores filogenéticas geradas pela primeira abordagem (baseada no algoritmo de aproximação e na distância exata, caso o cálculo seja possível), com as árvores geradas por esta segunda abordagem.

A análise seguinte pretende investigar o histórico evolutivo dos genomais bacteriais e reconstruir os genomas ancestrais utilizando rearranjo múltiplo de genomas. O problema de rearranjo múltiplo de genomas é NP-difícil até mesmo para o caso mais simples envolvendo três genomas e utilizando a distância de reversão. Caprara [9] mostrou que encontrar o genoma ancestral com a soma mínima das distâncias de reversão é NP-difícil.

Com a experiência e os resultados obtidos nas etapas anteriores pretendemos iniciar uma nova etapa que visa utilizar as particularidades dos genomas para gerar métodos específicos para o grupo de bactérias em questão.

# 4 Projeto

As próximas seções apresentam uma visão geral dos estudos que serão realizados no decorrer do projeto.

A Seção 4.1 apresenta o tema principal do projeto, um estudo baseado em heurísticas aplicadas aos problemas de transposição e reversão sem sinal.

A Seção 4.2 apresenta os trabalhos que pretendemos realizar no sentido de encontrar subgrupos de permutações em que algoritmos polinomiais são possíveis para calcular a distância de transposição. Essa linha de pesquisa segue a dos trabalhos de Labarre [30] e Fortuna [17] citados anteriormente.

A Seção 4.3 apresenta alguns objetivos secundários que pretendemos trabalhar aproveitando o embasamento teórico obtido com o cumprimento dos objetivos principais.

## 4.1 Heurísticas

O primeiro tópico de pesquisa desenvolvido neste doutorado pretende investigar heurísticas para os problemas da distância de transposição e da distância de reversão quando não é conhecida a orientação dos genes. Será dada ênfase primeiro no problema da distância de transposição.

As primeiras heurísticas a serem investigadas dizem respeito à utilização de técnicas de Programação Linear Inteira e Programação por Retrição aplicadas ao problema da distância por transposições. Alguns modelos iniciais de Programação por Restrição já foram implementados e os resultados são promissores.

Os modelos de programação por restrição já implementados utilizam os limitantes já conhecidos de trabalhos da literatura como, por exemplo, os limitantes de breakpoints e de ciclos ímpares de Bafna e Pevzner [5] e os limitantes apresentados por Labarre [30].

Todos os modelos possuem solução exata e alguns deles devolvem uma resposta em um intervalo menor do que os modelos utilizando programação linear inteira de Dias e Souza [12]. Um resumo estendido dos modelos foi enviado a um congresso de bioinformática e um resumo dos resultados foi adicionado na Seção 7.

Ainda no tópico de heurísticas, pretende-se complementar o algoritmo de Bafna e Pevzner [5] para que os resultados na prática sejam melhores. Essa proposta segue a linha de pesquisa de Walter e co-autores [44] que consiste em criar mecanismos capazes de aumentar o número de vezes que o algoritmo escolhe uma transposição que realmente pertence a alguma seqüência ótima de transposições.

Nesse contexto, já foi criado um algoritmo preliminar cuja quantidade de acertos é melhor do que a de qualquer outro algoritmo polinomial que conhecemos na literatura. Um artigo relatando as heurísticas implementadas neste algoritmo foi enviado a um congresso de bioinformática e um resumo dos resultados foi adicionado na Seção 7.

## 4.2 Classes de Permutações

Novas classes de permutações com distância de transposição exata em tempo polinomial serão delimitadas seguindo a linha de pesquisa do trabalho de Fortuna [17]. Uma classe promissora é o grupo de permutações que atingem o limitante inferior de *breakpoints*, que normalmente é o limitante mais simples em rearranjo de genomas.

Para o caso da distância de reversão, Kececioglu e Sankoff [29] conjecturaram que determinar se uma permutação pode ser ordenada no limite estabelecido pelo limitante inferior de *breakpoints* é um problema NP-Completo. Entretanto, Irving e Christie [26] e Tran [40] independentemente demonstraram que esta conjectura era falsa, ou seja, é possível determinar em tempo polinomial se uma permutação pode ser ordenada por reversões no limite imposto pelo limitante inferior de *breakpoints*, apesar do problema geral de distância de reversão ser NP-Completo.

Pretendemos resolver o problema análogo para distância de transposição. Algumas propriedades interessantes deste problema foram enunciadas brevemente por Christie [11] e acreditamos que solucionar este problema poderá ajudar a entender a complexidade do problema geral de distância de transposição.

Outro dado importante sobre este problema é o fato de existirem poucas classes de permutações para as quais são conhecidos algoritmos exatos de ordenação por transposição, e a classe das permutações que podem ser ordenadas respeitando o limite inferior de *breakpoints* seria a maior das classes já estudadas.

Além de delimitar a classe acima, pretendemos estudar as permutações cuja distância não é calculada de forma correta pelo algoritmo de Bafna e Pevzner [5], acreditamos que categorizar tais permutações pode auxiliar na criação de um algoritmo com fator de aproximação melhor que 1.5, além de definir as propriedades que fazem com que uma permutação não seja ordenada de forma correta pelo algoritmo de Bafna e Pevzner.

## 4.3 Objetivos Secundários

O problema do diâmetro, definido como a maior distância possível entre duas permutações de tamanho $n$, foi proposto por Bafna e Pevzner [3] e ainda permanece em aberto. Obtivemos resultados parciais importantes anteriormente no Instituto de Computação da UNICAMP [32] para o problema do diâmetro de transposição e pretendemos investigá-lo no decorrer deste projeto de pesquisa.

Um outro tema recorrente em rearranjo de genomas é a implementação de novos algoritmos de aproximação. Um dos objetivos desse projeto é dar seguimento a essa tendência. Atualmente, tanto o problema da transposição quanto o problema de reversão quando a orientação dos genes não é conhecida, possuem algoritmos de aproximação na razão de 1.375 [7, 14].

Por fim, um objetivo que emerge como conseqüência do estudo teórico para descoberta de classes de permutações que podem ser resolvidas em tempo polinomial é adquirir intuição e embasamento adequado a fornecer novas ferramentas que possam contribuir com a resolução do problema da distância de transposição e de reversão quando a orientação dos genes não é conhecida.

# 5 Cronograma

As tabelas 1 e 2 descrevem a distribuição das atividades a serem realizadas durante a execução deste trabalho. Organizou-se o plano de trabalho em etapas, onde cada etapa não possui as outras como pré-requisitos. Nesse caso, a distribuição das atividades nos quatro anos de doutorado segue o raciocínio de que é mais produtivo resolver primeiramente os

objetivos mais simples e, a medida que possuirmos mais embasamento teórico, resolver os objetivos que demandam mais tempo.

**Etapa 1** Voltada para o (1a) cumprimento dos créditos das disciplinas do programa de doutorado do Instituto de Computação e (1b) para a revisão bibliográfica dos artigos clássicos de rearranjo de genomas com o intuito de preparar a (1c) apresentação do Exame de Qualificação Específico (EQE).

**Etapa 2** Estudo de heurísticas para rearranjo de genomas. Inicialmente estudaremos modelos baseados em (2a) Programação Linear Inteira e (2b) Programação por Restrição. No entanto, (2c) outras heurísticas serão acrescentadas nessa etapa no decorrer do projeto. Como mostra a Tabela 1 esta etapa já foi iniciada, inclusive, como relatado anteriormente, dois trabalhos já foram submetidos para um congresso de bioinformática. No que diz respeito às outras heurísticas (2c), por enquanto o foco está em descobrir formas de melhorar o algoritmo de Bafna e Pevzner [5].

| | **2007** | | **2008** | | | **2009** |
|---|---|---|---|---|---|---|
| Etapas | mar-jul | ago-dez | jan-abr | mai-ago | set-dez | jan-abr |
| 1 | a | a,b | b | b | b | b |
| 2 | | | a | a,b | a,b | b,c |
| 3 | | | | | | a |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | a | a |

Tabela 1: Cronograma de atividades para os dois primeiros anos

**Etapa 3** Estudo voltado para a (3a) definição de classes de permutações que poderão ser resolvidas de forma exata em tempo polinomial e (3b) resolução do problema do diâmetro de transposição.

**Etapa 4** Estágio no exterior. Estudo voltado para geração de árvores filogenéticas baseadas na matriz de distância (4a) calculada usando algoritmos de aproximação e abordagem exata baseada em programação lógica com restrições. Utilização de estimativas estatísticas (4b) e rearranjo múltiplo de genomas (4c) para geração de resultados mais precisos e posterior análise com os resultados obtidos com a matriz de distâncias. Utilização das particularidades dos genomas para gerar métodos específicos para o grupo de bactéria em questão (4d).

**Etapa 5** Estudo de algoritmos de aproximação a fim de melhorar o fator de aproximação para o problema da (5a) transposição e (5b) reversão quando não é conhecida a orientação dos genes. É importante ressaltar que, devido ao caráter secundário desse objetivo, sua execução depende dos resultados obtidos nas outras etapas.

**Etapa 6** (6a) Escrita e (6b) defesa da tese. A escrita da tese ocorrerá concomitantemente com as outras etapas, a medida que resultados parciais forem obtidos.

| | 2009 | | 2010 | | | 2011 |
|---|---|---|---|---|---|---|
| Etapas | mai-ago | set-dez | jan-abr | mai-ago | set-dez | jan-fev |
| 1 | b,c | | | | | |
| 2 | b,c | | | | | |
| 3 | a | a | a,b | b | b | |
| 4 | | a,b,c | c | c,d | d | |
| 5 | a,b | a,b | a,b | a,b | a,b | |
| 6 | a | a | a | a | a | b |

Tabela 2: Cronograma de atividades para os dois últimos anos

# 6 Metodologia e Forma de análise dos resultados

Esta seção pretende descrever as abordagens que serão consideradas no desenvolvimento do projeto. Além disso, esta seção define como os resultados serão avaliados e relaciona a forma de análise com os objetivos do projeto mencionados na Seção 4.

## 6.1 Heurísticas

A análise das heurísticas será realizada comparando com outras implementações presentes na literatura. Em se tratando das heurísticas criadas para complementar o algoritmo de Bafna e Pevzner [5], as implementações descritas por Walter [44] serão as primeiras a serem utilizadas como base para comparação.

Os parâmetros utilizados para comparação serão, principalmente, o número de permutações que o algoritmo com as heurísticas resolve corretamente e a complexidade de tempo que cada heurística acrescenta ao algoritmo.

As heurísticas baseadas em modelos de Programação Linear Inteira e Programação por Restrições são um pouco mais difíceis de se comparar com outras da literatura, pois geram um resultado exato e pouco se conhece a utilização de abordagens parecidas. Dessa forma, utilizaremos principalmente o tempo de execução médio para uma permutação ser ordenada. Para isso, utilizaremos como base para a análise o modelo baseado em Programação Linear Inteira de Dias e Souza [12].

## 6.2 Classes de Permutações

No contexto de classes de permutações, a forma de análise dos resultados está relacionada com a abrangência da classe definida e com a complexidade computacional necessária para verificar se uma dada permutação pertence à classe.

O cumprimento desse objetivo pode ser realizado, principalmente, através de duas maneiras. A primeira seria definir uma nova operação semelhante a de rearranjo estudada, mas que possua resolução polinomial e, em seguida, verificar as condições necessárias para que a distância entre dois genomas definida usando essa nova operação seja semelhante a distância original. Uma idéia verificada anteriormente por Fortuna [17] ao utilizar transposições de prefixo para derivar uma classe de instâncias tratáveis, as quais denominou

permutações fáceis.

Uma segunda maneira de definir uma classe é descobrindo características que podem ser analisadas utilizando a própria permutação. Por exemplo, Christie [11] conjecturou que uma condição necessária para que uma permutação seja ordenada por transposição respeitando o limite inferior de *breakpoints* é tal que seu diagrama de ciclos [5] seja composto apenas por ciclos de tamanho 1 ou 3, e toda componente deste diagrama deve poder ser ordenada isoladamente respeitando o mesmo limitante inferior.

## 6.3 Objetivos secundários

Os objetivos secundários dizem respeito a encontrar algoritmos de aproximação para os problemas da distância de transposição e reversão com sinal. Nesses objetivos a análise será realizada de acordo com o fator de aproximação e a complexidade do algoritmo.

# 7 Resultados parciais

Em seções anteriores mencionou-se que dois trabalhos foram submetidos a um congresso de bioinformática. Nesta seção são resumidos os resultados presentes nos dois trabalhos.

O primeiro trabalho intitulado "*Extending Bafna and Pevzner Algorithm*" apresenta uma série de heurísticas aplicadas ao algoritmo de Bafna e Pevzner [5]. Uma análise comparativa com outros resultados publicadas na literatura foi realizada gerando as informações da Tabela 3.

Tabela 3: Quantidade de erros cometidos pelos algoritmos WDM-Walter, Dias e Meidanis [43], Ch-Christie [11] com heurísticas implementado por Walter, Curado e Oliveira [41], H-Hartman [23] implementado por Honda [25], BP-Bafna e Pevzner [5] implementado por Walter et. al. com heurísticas [44], M-Mira et. al. usando formalismo algébrico [35], D-Dias e Dias (trabalho submetido ao congresso) baseado no algoritmo de Bafna and Pevzner.

| Size | WDM | Ch | H | BP | M | D |
|---|---|---|---|---|---|---|
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 6 | 0 | 2 | 0 | 0 | 0 |
| 7 | 72 | 0 | 108 | 0 | 0 | 0 |
| 8 | 1167 | 40 | 1517 | 34 | 124 | 0 |
| 9 | 14327 | 1182 | 25425 | 1007 | 2977 | 0 |
| 10 | - | - | - | - | 69297 | 5907 |

Pela Tabela 3 percebe-se que nossa primeira implementação baseada no algoritmo de Bafna e Pevzner calcula $d(\pi)$, para $|\pi| \leq 9$, sem cometer erros. A análise dos resultados para $|\pi| = 10$ mostrou que a diferença entre $d(\pi)$ e o valor computado pelo nosso programa é sempre 1. Duas situações foram encontradas, $d(\pi) = 4$, mas o valor foi computado 5 e $d(\pi) = 5$, mas o valor computado foi 6.

As heurísticas implementadas no algoritmo consistem basicamente em aproveitar transposições que aumentam o número de ciclos ímpares como *good* 0-*moves* e *valid* 2-*moves* [5] para criar configurações que permitam 2-moves válidos em transposições seguintes. Além disso, Bafna e Pevzner [5] criaram mecanismos para identificar *valid* 2-*moves*, mas dado um conjunto deles não criou um método que permita selecionar o melhor. Para contornar esse problema, criamos uma ordem baseada em resultados experimentais para selecionar quais transposições devem ser aplicadas primeiro.

O segundo trabalho intitulado "*Constraint Programming Models for Transposition Distance Problem*" apresenta uma série de modelos utilizando programação por restrição baseados em limitantes conhecidos. O trabalho visa ser uma demonstração de que a técnica de programação por restrição apresenta bons resultados na prática mesmo com modelos simples.

A Tabela 4 apresenta os resultados dos vários modelos criados. Os modelos com o sufixo "**csp**" utilizam a abordagem de definir o problema como um problema de satisfação de restrições (CSP - do inglês *Constraint Satisfaction Problem*) e os modelos com sufixo "**cop**" utilizam a abordagem de definir o problem como um problema de otimização de restrições (COP - do inglês *Constraint Optimization Problem*). Por exemplo, **def_csp** é um problema de satisfação, enquanto que **def_cop** é um problema de otimização.

Os modelos com prefixo "**def**" utilizam apenas a definição do problema sem quaisquer limitantes e foram criados apenas para verificar a melhoria oferecida pelos outros modelos. Os modelos com prefixo "**br**" e "**cg**" baseiam-se nos limitantes que utilizam o número de *breakpoints* e de ciclos ímpares [5] respectivamente. Os limitantes são usados para "podar" o espaço de busca do problema. O modelo **cg_red_csp** é um problema de restrição que utiliza o limitante de ciclos ímpares e a equivalência por redução apresentada por Christie [11].

É importante ressaltar que nos modelos baseados em problemas de satisfação apenas os limitantes inferiores são importantes, enquanto que nos modelos baseados em problemas de otimização tanto os limitantes inferiores quanto superiores interferem no tempo de execução. Labarre [30] apresentou uma série de limitantes superiores (Apêndice J) com uma estrutura de grafo criada a partir da permutação $\pi$ chamada $\Gamma(\pi)$. Entretanto, Labarre [30] não definiu limitantes inferiores. Dessa forma, criou-se o modelo **gg_cop** que utiliza o limitante inferior do grafo de ciclos de Bafna e Pevzner [5] e o limitante superior de Labarre [30].

A Tabela 4 mostra que o modelo **cg_csp_red** apresentou os melhores resultados. Como esse modelo utiliza o princípio de redução de Christie [11], é mais difícil recuperar uma seqüência ótima de transposições que ordenam a permutação de entrada.

Os modelos de otimização são os que apresentaram o pior resultado. Entretanto, vale ressaltar que não foram criadas quaisquer heurísticas de busca para o problema. Utilizou-se um *solver* genérico para problemas de otimização de restrição que realiza uma busca *branch and bound* em conjunto com a técnica de propagação de restrições. Dessa forma, esforços ainda são necessários para concluir que os modelos CSP representam uma abordagem melhor para o problema da distância de transposição.

Por outro lado, o mecanismo de busca dos modelos COP sempre utilização a estratégia de encontrar uma solução e tentar melhorá-la procurando por novas soluções com custos menores (a função de custo utilizada foi a própria distância). O fato de não existir um mecanismo que, dada uma solução que é uma seqüência de transposições que ordene o

Tabela 4: Média de tempo (em segundos) para calcular a distância de transposição entre 1000 permutações escolhidas ao acaso e a identidade. O símbolo '−' indica que o modelo demorou mais do que 15 horas para terminar a bateria de testes.

| **Size** | **def_csp** | **br_csp** | **cg_csp** | **cg_red_csp** | **def_cop** | **cg_cop** | **gg_cop** |
|---|---|---|---|---|---|---|---|
| 4 | 0.006 | 0.010 | 0.005 | 0.005 | 0.134 | 0.019 | 0.039 |
| 5 | 0.069 | 0.087 | 0.009 | 0.006 | 2.530 | 0.061 | 0.149 |
| 6 | 1.887 | 2.367 | 0.022 | 0.013 | − | 1.842 | 4.421 |
| 7 | 51.69 | 30.707 | 0.045 | 0.024 | − | 3.797 | 39.024 |
| 8 | − | − | 0.233 | 0.104 | − | − | − |
| 9 | − | − | 0.946 | 0.313 | − | − | − |
| 10 | − | − | 6.816 | 2.016 | − | − | − |
| 11 | − | − | 20.603 | 4.212 | − | − | − |

problema, obtenha em tempo polinomial uma outra solução melhor (como ocorre, por exemplo, no algoritmo *simplex* para programação linear), faz com que o mecanismo de busca dos modelos COP utilizem um espaço de busca maior dos que os modelos CSP.

Em outras palavras, seja $A$ um limitante inferior e $B$ um limitante superior para a permutação $\pi$, $d(\pi) = t$, e $A \leq t \leq B$. Os modelos COP produzirão uma série de soluções no intervalo $[t+1..B]$ até encontrar uma solução ótima $t$. Entretanto, o mecanismo de busca precisará ainda provar que não existe solução no intervalo $[A..t-1]$. Os modelos baseados em programação por restrição, por outro lado, realizam uma estratégia *botton-up* verificando se existe solução de custo $A$, em seguida procuram por soluções de custo $A+1$ e assim por diante até encontrar uma solução de custo $t$. Dessa forma, o espaço de busca dos modelos COP são as soluções de custo $cost \in [A..B]$, enquanto que o espaço de busca dos modelos CSP são as soluções com custo associado $cost \in [A..t]$.

Para trabalhos futuros pretende-se melhorar os modelos **cg_csp** e **cg_red_csp** identificando e ordenando as transposições que deverão fazer parte da árvore de busca, ambas as tarefas serão realizadas observando o grafo de ciclos e a variação no número de ciclos ímpares causada por cada transposição. Ainda com o intuito de diminuir a árvore de busca, pretende-se utilizar o conceito de equivalência toroidal de Eriksson [15] (Apêndice I).

Um dado relevante é que nosso modelo com melhores resultados apresentados, **cg_csp_red**, é muito melhor em termos de performance que os modelos baseados em programação linear inteira de Dias e Souza [12], que reportaram testes usando 100 instâncias geradas ao acaso e para $|\pi| = 9$ calculam uma solução com 143.4 segundos em média, enquanto nosso modelo calcula uma solução com 0.313 segundos em média.

# 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. In *Proceedings of the Seventh Workshop on Algorithms and Data Structures (WADS'01)*. Springer Verlag, 2001.

[2] V. Bafna and P. A. Pevzner. Genome Rearrangements and Sorting by Reversals. In IEEE, editor, *Proceedings of the 34th Annual Symposium on Foundations of Comptuer Science (FOCS'93)*, pages 148–157, Palo Alto, USA, November 1993. IEEE Computer Society Press.

[3] V. Bafna and P. A. Pevzner. Sorting by Transpositions. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 614–623, San Francisco, USA, January 1995.

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

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

[6] A. Bergeron. A Very Elementary Presentation of the Hannenhalli-Pevzner Theory. In *Proceedings of the 12th Annual Symposium of the Combinatorial Pattern Matching (CPM'2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 106–117, Berlin, Germany, September 2001. Springer-Verlag.

[7] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In *Proceedings of the 10th European Symposium on Algorithms (ESA'2002)*, Lecture Notes in Computer Science, Rome, Italy, September 2002. Springer.

[8] A. Caprara. Sorting Permutations by Reversals and Eulerian Cycle Decompositions. Technical Report OR-97-4, DEIS - Operations Research Group, University of Bologna, 1997.

[9] A. Caprara. Formulations and Hardness of Multiple Sorting by Reversals. In S. Istrail, P. A. Pevzner, and M. Waterman, editors, *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB'99)*, pages 84–93, Lyon, France, 1999. ACM Press.

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

[11] D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, Glasgow University, 1998.

[12] Z. Dias and C. Souza. Polynomial-sized ILP Models for Rearrangement Dîstance Problems. In *BSB2007 Poster Proceedings*, 2007.

[13] T. Dobzhansky and A. H. Sturtevant. Inversions in the third chromosome of wild races of *Drosophila pseudoobscura*, and their use in the study of the history of the species. *Proceedings of the National Academy of Science*, 22:448–450, July 1936.

[14] I. Elias and T. Hartman. A 1.375-Approximation Algorithm for Sorting by Transpositions. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4):369–379, 2006.

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

[16] J. Feng and D. Zhu. Faster Algorithms for Sorting by Transpositions and Sorting by Block Interchanges. *ACM Trans. Algorithms*, 3(3):25, 2007.

[17] V. J. Fortuna. Distâncias de transposição entre genomas. Master's thesis, Institute of Computing, University of Campinas, 2005.

[18] Qian-Ping Gu, Shietung Peng, and Hal Sudborough. Approximating Algorithms for Genome Rearrangements. In *Proceedings of 7th Workshop on Genome Informatics - GIW'96*, 1996.

[19] S. Hannenhalli, C. Chappey, E. V. Koonin, and P. A. Pevzner. Genome Sequence Comparison and Scenarios for Gene Rearrangements: a Test Case. *Genomics*, 30:299–311, 1995.

[20] S. Hannenhalli and P. A. Pevzner. Transforming Cabbage into Turnip (Polynomial Algorithm for Sorting Signed Permutations by Reversals). In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 178–189, Las Vegas, USA, May 1995.

[21] 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, January 1999.

[22] T. Hartman and R. Shamir. A Simpler and Faster 1.5-approximation Algorithm for Sorting by Transpositions. *Inf. Comput.*, 204(2):275–290, 2006.

[23] T. Hartman and R. Sharan. A Simpler 1.5-approximation Algorithm for Sorting by Transpositions. pages 156–169. Springer, 2003.

[24] L. S. Heath and J. P.Vergara. Sorting by Bounded Block-Moves. *Discrete Applied Mathematics, Second Special Issue on Computational Biology*, 88:181–206, 1998.

[25] M. I. Honda. Implementation of the algorithm of Hartman for the Problem of Sorting by Transpositions. Master's thesis.

[26] R. W. Irving and D. A. Christie. Sorting by Reversals: on a Conjecture of Kececioglu and Sankoff. Technical Report TR-95-12, Department of Computing Science, University of Glasgow, May 1995.

[27] M. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985.

[28] J. Kececioglu and D. Gusfield. Reconstructing a History of Recombinations from a Set of Sequences. In D. D. Sleator, editor, *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 471–480, Arlington, USA, January 1994. ACM Press.

[29] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals. Technical Report 1824, Centre de Recherches mathématiques, Université de Montréal, July 1992.

[30] A. Labarre. New Bounds and Tractable Instances for the Transposition Distance. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(4):380–394, 2006.

[31] J. Meidanis and Z. Dias. An Alternative Algebraic Formalism for Genome Rearrangements. In D. Sankoff and J. H. Nadeau, editors, *Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families*, pages 213–223. Kluwer Academic Publishers, November 2000.

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

[33] J. Meidanis, M. E. M. T. Walter, and Z. Dias. Reversal Distance of Signed Circular Chromosomes. Technical Report IC-00-23, Institute of Computing - University of Campinas, December 2000.

[34] 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), 2002. Submitted: 14/January/2000. Acepted: 16/July/2002.

[35] C. V. G. Mira, Z. Dias, H. P. Santos, G. A. Pinto, and M. E. Walter. Transposition Distance Based on the Algebraic Formalism. In *BSB '08: Proceedings of the 3rd Brazilian symposium on Bioinformatics*, pages 115–126, Berlin, Heidelberg, 2008. Springer-Verlag.

[36] J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. 81:814–818, 1984.

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

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

[39] Eric Tannier and Marie-France Sagot. Sorting by Reversals in Subquadratic Time. In *Proceedings of CPM 2004, LNCS. 2004 . 3109():0-0*, 2004.

[40] N. Q. Tran. An Easy Case of Sorting by Reversals. *Journal of Computational Biology*, 5(4):741–746, 1998.

[41] 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 Fourteenth Annual Symposium on Combinatorial Pattern Matching*, volume 2676 of *Lecture Notes in Computer Science*, pages 372–383. Springer, Berlin, 2003.

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

[43] M. E. M. T. Walter, Z. Dias, and J. Meidanis. A New Approach for Approximating the Transposition Distance. In *Proceedings of the String Processing and Information Retrieval (SPIRE'2000)*, September 2000.

[44] 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. *Discrete Algorithm*, 3:342–361, 2005.

[45] G. A. Watterson, W. J. Ewens, T. E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theorical Biology*, 99:1–7, 1982.

# A Sorting by Transpositions [5]

Este trabalho é um dos primeiros a tratar o problema da distância por transposições, apresentando limitantes para o problema e um algoritmo capaz de fornecer uma resposta aproximada na razão de 1.5.

Representando um genoma por uma seqüência de genes $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, uma transposição $\rho(i,j,k)$ onde $1 \leq i < j < k \leq n+1$ é uma operação em que dois blocos de genes que estão lado a lado trocam de lugar de modo que o intervalo $[i, j-1]$ de $\pi$ seja inserido entre $\pi_{k-1}$ e $\pi_k$. No trabalho de Bafna e Pevzner assume-se que a permutação $\pi$ foi estendida para incluir os elementos $\pi_0 = 0$ e $\pi_{n+1} = n+1$.

Dados dois genomas $\pi$ e $\sigma$, o problema da distância de transposição é encontrar o número mínimo $t$ de transposições $\rho_1, \rho_2, \ldots, \rho_t$ tais que $\pi\rho_1\rho_2\cdots\rho_t = \sigma$. Um ponto a ser mencionado é que a distância de transposição entre $\pi$ e $\sigma$ é igual a distância entre $\sigma^{-1}\pi$ e a permutação identidade. Nesse caso, ordenar por transposição objetiva encontrar a distância $d(\pi)$ entre um genoma $\pi$ e a identidade.

O trabalho aborda duas maneiras para lidar com o problema, uma delas é utilizando o conceito de *breakpoints* e a outra é utilizando uma representação chamada de grafo orientado cíclico com arestas de cores alternadas, ou simplesmente grafo de ciclos. Devido a grande importância dessas abordagens e a freqüência com que aparecem em trabalhos posteriores, ambas serão detalhadas nas próximas seções.

## A.1 Breakpoints

Dada uma permutação estendida $\pi = (\pi_0\ \pi_1\ \cdots\ \pi_n\ \pi_{n+1})$, define-se um *breakpoint* como o par $(\pi_i, \pi_{i+1})$, $\forall i \in [0,n]$ se $\pi_{i+1} \neq \pi_i + 1$. *Breakpoints* podem ser entendidos como os elementos que ainda precisam ser ordenados. Assim, ordenar uma permutação envolve diminuir o número de *breakpoints* até zero, o que corresponderia a uma permutação identidade.

Com o conceito de *breakpoint* é possível definir um limitante inferior para o problema da distância por transposição. Observa-se que em uma dada transposição $\rho(i,j,k)$ as modificações geradas ocorrem apenas em três pontos da seqüência, que são entre os elementos $[\pi_{i-1}, \pi_i]$, $[\pi_{j-1}, \pi_j]$ e $[\pi_{k-1}, \pi_k]$ da permutação. Dessa forma, uma transposição pode no máximo diminuir em três a quantidade de *breakpoints*, o que gera o seguinte limitante:

$$d(\pi) \geq \frac{\#breakpoints(\pi)}{3}$$

## A.2 Grafo orientado cíclico com arestas de cores alternadas

Um grafo $G(\pi)$ é orientado cíclico com arestas de cores alternadas quando, para uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, é associado um conjunto de $n+2$ vértices numerados de 0 a $n+1$ e um conjunto de $2(n+1)$ arestas orientadas e coloridas de duas formas:

**Arestas cinzas:** orientadas do vértice rotulado com $i-1$ ao vértice rotulado com $i$, para todo $i \in [1, n+1]$

**Arestas pretas:** orientadas do $i$-ésimo vértice ao $(i-1)$-ésimo vértice, para todo $i \in [1, n+1]$.

Uma observação a ser feita é que para cada vértice existe uma aresta cinza de entrada pareada com uma aresta preta de saída, o que permite deduzir a possibilidade de realizar uma única decomposição do grafo em ciclos $C$ de cores alternadas. A figura 1 mostra o grafo de ciclos criado para a permutação $\pi = (4\ 2\ 5\ 3\ 1)$. O exemplo da figura não pertence ao trabalho original e foi adicionado para facilitar a compreensão do grafo de ciclos.

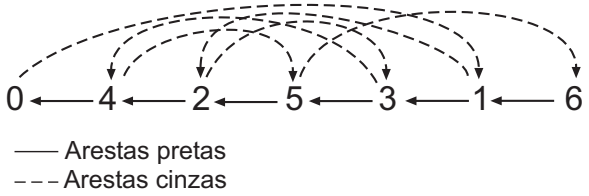


Figura 1: Grafo de ciclos para a permutação $\pi = (4\ 2\ 5\ 3\ 1)$

O tamanho de um ciclo é dado pelo número de arestas pretas que ele possui. Um ciclo com $k$ arestas pretas é dito um $k$-ciclo. Um ciclo pode ser classificado em par ou ímpar de acordo com seu tamanho $k$. Se $k$ for par, então o ciclo é par e, por outro lado, se $k$ for ímpar, então o ciclo é ímpar.

Como existem $2(n+1)$ arestas e a identidade possui $n+1$ ciclos de tamanho 1, conclui-se que ordenar uma permutação implica encontrar um conjunto mínimo de transposições que aumentam o número de ciclos até $n+1$. Dessa forma, denominando $c(\pi)$ o número de ciclos e $\Delta_c(\rho) = c(\pi\rho) - c(\pi)$ a variação sofrida no número de ciclos em decorrência da transposição $\rho$, o trabalho apresenta o teorema 1.

**Teorema 1** *Se após uma transposição há variação no número de ciclos, então essa variação é sempre de* 2*, para mais ou para menos, ou seja,* $\Delta c(\rho) \in \{2, 0, 2\}$.

O teorema 1 permite definir o seguinte limitante inferior.

$$d(\pi) \geq \frac{n + 1 - c(\pi)}{2}$$

Uma observação que pode ser feita é que a identidade possui apenas ciclos ímpares, então um limitante que leve em consideração as alterações no número de ciclos ímpares tende a ser mais próximo da realidade que o limitante obtido pelo teorema 1. Assim, seja $c_{odd}(\pi)$ o número de ciclos ímpares e $\Delta_{c_{odd}}(\rho) = c_{odd}(\pi\rho) - c_{odd}(\pi)$ a variação sofrida no número de ciclos ímpares em decorrência da transposição $\rho$, o trabalho apresenta o teorema 2.

**Teorema 2** *Se após uma transposição há variação no número de ciclos ímpares, então essa variação é sempre de* 2*, para mais ou para menos, ou seja,* $\Delta_{c_{odd}}(\rho) \in \{2, 0, -2\}$.

O teorema 2 permite definir o seguinte limitante inferior, que é mais próximo da distância real que o limitante definido anteriormente.

$$d(\pi) \geq \frac{n + 1 - c_{odd}(\pi)}{2}$$

## A.3 Propriedades do grafo de ciclos

Para representar um ciclo $C$ em um grafo de ciclos $G(\pi)$ numera-se as arestas pretas de 1 até $n+1$ em $G(\pi)$ atribuindo o rótulo $i$ para a aresta preta que vai de $\pi_i$ a $\pi_{i-1}$ e listando os rótulos das arestas pretas na ordem em que aparecem em $C$.

Por exemplo, a figura 1 possui dois ciclos $C_1$ e $C_2$ com 4 e 2 arestas pretas respectivamente, como é mostrado na figura 2. Iniciando a leitura do ciclo $C_1$ pela aresta cinza que sai do elemento $\pi_1 = 4$, temos que os rótulos lidos no grafo $G(\pi)$ são $3, 4, 6, 2$. Dessa forma, $C_1 = (3, 4, 6, 2)$.

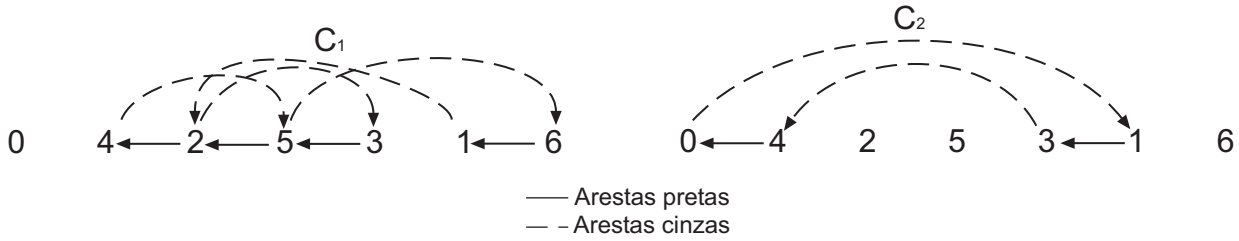


Figura 2: Ciclos $C_1$ e $C_2$ para a permutação $\pi = (4\ 2\ 5\ 3\ 1)$

Como existem $k$ formas de se escrever um $k$-ciclo, dependendo do elemento em que se inicia a leitura do ciclo em $G(\pi)$, assume-se que o primeiro rótulo do ciclo deve representar a aresta preta que está mais à direita em $G(\pi)$. Dessa forma, $C_1 = (6, 2, 3, 4)$ e, de modo análogo, $C_2 = (5, 1)$

O teorema 2 afirma que uma transposição $\rho$ pode aumentar no máximo em 2 o número de ciclos ímpares. Para agilizar a ordenação de uma permutação é desejável maximizar o número dessas transposições. Alguns conceitos sobre ciclos são relevantes para decidir se tais transposições são possíveis.

Um ciclo $C = (i_1, i_2, \ldots, i_k)$ é dito não orientado se $i_1 < i_2 < \ldots < i_k$, caso contrário $C$ é dito orientado. A classificação dos ciclos em orientados e não orientados é importante, como mostram os teoremas 3 e 4.

**Teorema 3** *Se existe um ciclo orientado em $G(\pi)$ então é possível encontrar uma transposição que aumente em 2 o número de ciclos.*

**Teorema 4** *Se existe um ciclo orientado em $G(\pi)$ então ou é possível encontrar uma transposição que aumenta em 2 o número de ciclos ímpares ou é possível encontrar uma seqüência de três transposições que alteram em $(0, 2, 2)$ o número de ciclos ímpares.*

Algumas definições presentes no trabalho de Bafna e Pevzner foram bastante utilizadas em trabalhos posteriores e se tornaram uma nomenclatura básica para a área de rearranjo por transposições.

**Entrelaçamento:** duas seqüências $v = v_1, v_2, \ldots, v_k$ e $w = w_1, w_2, \ldots, w_k$ ordenadas de inteiros se entrelaçam (do inglês, *interleave*) se $v_1 < w_1 < v_2 < w_2 < \ldots < v_k < w_k$ ou $w_1 < v_1 < w_2 < v_2 < \ldots < w_k < v_k$. O trabalho mostra alguns resultados para ciclos não orientados que se entrelaçam.

**Ciclos longos:** Um $k$-ciclo é longo se $k > 2$ e curto caso contrário.

**$k$-moves:** Uma transposição que altera em $k$ o número de ciclos é dita $k$-move.

Quando não existem ciclos orientados não é possível utilizar o teorema 4, mas o trabalho descreve um rigoroso estudo de caso a fim de derivar uma algoritmo capaz de fornecer uma resposta aproximada na razão de 1.5.

A análise deste algoritmo não faz parte do escopo deste resumo, pois existem outros trabalhos na literatura que descrevem o algoritmo de Bafna e Pevzner de uma forma mais simples. Além disso, o trabalho de Bafna e Pevzner omite detalhes técnicos essenciais para a implementação do algoritmo [11].

O resultado que garante o fator de 1.5 do algorimo de Bafna e Pevzner é uma extensão do teorema 4 para qualquer ciclo longo.

**Teorema 5** *Se existe um ciclo longo em $G(\pi)$ então ou é possível encontrar uma transposição que aumenta em 2 o número de ciclos ímpares ou é possível encontrar uma seqüência de três transposições que alteram em $(0, 2, 2)$ o número de ciclos ímpares.*

# B A Very Elementary Presentation of the Hannenhalli-Pevner Theory [6]

O estudo apresentado neste trabalho é um esforço na busca de um algoritmo polinomial mais simples que o apresentado por Hannenhalli e Pevner [21], cuja estratégia para solucionar o problema da ordenação por reversões é baseada em uma série de representações intermediárias que podem ser simplificadas.

Dada uma permutação $\pi = \pi_1, \pi_2, \cdots, \pi_n$, extende-se a notação acrescentando os elementos $\pi_0 = 0$ e $\pi_{n+1} = n + 1$. As definições apresentadas no trabalho partem do princípio de que uma permutação está na forma extendida.

Define-se pares orientados $(\pi_i, \pi_j)$ como pares de inteiros consecutivos com sinais opostos. A identificação desses pares orientados permite prever quais reversões resultariam em um aumento do número de elementos consecutivos. Entretanto, como vários pares orientados podem existir, define-se uma pontuação, $score(\pi_i, \pi_j)$, dada pelo número de pares orientados na permutação resultante. O primeiro algoritmo apresentado no trabalho consiste em selecionar um dos pares orientados de maior *score* e aplicar a reversão por ele induzida.

O problema desse algoritmo é que não retorna uma solução para todos os casos, pois pode ocorrer uma permutação com todos os elementos positivos. Nesse caso, nenhum par orientado existe e são necessários outros meios para que o algoritmo possa prosseguir.

O primeiro procedimento tomado quando ocorre uma situação onde todos os elementos são positivos é realizar uma redução, que significa rearranjar a permutação de modo que ao final não existam elementos consecutivos. Por exemplo, a permutação $\pi = (0, 3, 1, 2, 4)$ possui os elementos consecutivos $\pi_2 = 1$ e $\pi_3 = 2$ e pode ser reduzida para a permutação $\pi' = (0, 2, 1, 3)$, onde os elementos consecutivos foram reunidos em um só e os outros elementos renomeados.

Se $\pi$ é reduzida, então é possível definir um intervalo fechado como um intervalo da forma $i, \pi_{j+1}, \pi_{j+2}, \cdots, \pi_{j+k-1}, i + k$ onde todos os inteiros entre $i$ e $i + k$ pertencem ao

intervalo $[i, i+k]$. Se $\pi$ é reduzida, é possível definir um obstáculo como um intervalo fechado que não contém outro intervalo fechado. Uma observação importante é que essa definição de obstáculo equivale a definição de Hannenhalli e Pevzner [21].

De posse da definição de obstáculos, duas novas operações são introduzidas para complementar o algoritmo básico. A primeira, chamada corte de obstáculo, consiste em reverter o segmento entre $i$ e $i+1$ do obstáculo, onde $i$ é o primeiro elemento do obstáculo. A segunda operação, chamada junção de obstáculo, consiste em realizar uma reversão que atinja dois obstáculos da seguinte forma: dados dois obstáculos de segmentos $i \cdots i+k$ e $i' \cdots i'+k'$, a reversão de junção desses dois obstáculos é $\rho(i+k, i')$.

Em alguns momentos, a operação de corte de um obstáculo gera um novo obstáculo, quando isso acontece o obstáculo original é chamado super-obstáculo. A presença de um super-obstáculo obriga a tomar um certo cuidado ao se escolher o obstáculo para o corte, o algoritmo definido no trabalho segue a estratégia:

1. se uma permutação possui $2k$ obstáculos e $k \geq 2$ então junte dois obstáculos não consecutivos
2. se uma permutação possui $2k+1$ obstáculos e $k \geq 1$ então
   (a) se a permutação possui um obstáculo simples então corte-o
   (b) se não há obstáculos simples então junte dois obstáculos não consecutivos
   (c) se a permutação possui $k = 1$ então junte dois obstáculos consecutivos

O trabalho garante que essa estratégia resolve otimamente o problema.

# C An Alternative Algebraic Formalism for Genome Rearrangements [31]

O trabalho apresenta um novo formalismo capaz de representar problemas em rearranjo de genomas com o intuito de diminuir a dependência de representação visuais para se expor argumentos, provas e teoremas.

No método clássico, representa-se um genoma como uma permutação $\pi = (\pi_1\ \pi_2 \ldots \pi_n)$, significando que $\pi$ é uma função tal que $\pi(1) = \pi_1, \pi(2) = \pi_2$ e assim sucessivamente. O formalismo algébrico propõe que a mesma permutação $\pi$ seja vista como uma função que mapeia um dado elemento da permutação no elemento seguinte, em outras palavras $\pi(\pi_1) = \pi_2, \pi(\pi_2) = \pi_3$ e assim por diante, sendo importante ressaltar que nessa nova interpretação o último elemento $\pi_n$ é mapeando no primeiro, $\pi(\pi_n) = \pi_1$.

A motivação desse novo formalismo é permitir aplicar diretamente alguns resultados deduzidos pela teoria clássica e definir algebricamente, ao invés de graficamente, alguns conceitos importantes como ciclos bons e ruins, arestas pretas e cinzas, breakpoints e grafos de breakpoints.

## C.1 Ciclos

Dado um conjunto base $E$, uma permutação em $E$ é uma função $f : E \to E$ injetora. As permutação em $E$ são compostas de ciclos, por exemplo, dado $E = \{a, b, c, d\}$, a permutação

dada pelo ciclo $\alpha = (a\, b\, c)$ significa que $a$ é mapeado em $b$, que é mapeado em $c$, que é mapeado em $a$. Ao elemento $d$, que não aparece explicitamente, entende-se que é um ciclo unitário $(d)$, em que $d$ é mapeado nele mesmo.

Duas operações com ciclos são de grande importância ao se utilizar o formalismo algébrico: o produto e a conjugação. O produto de dois ciclos $\alpha$ e $\beta$, denotado por $\alpha\beta$, é computado como segue: fixa-se um elemento $x$ do ciclo mais a direita e obtém-se sua imagem $y$, a imagem $y$ é usada como entrada no segundo ciclo mais a direita e obtém-se a resposta $z$, que é para onde $x$ no produto $\alpha\beta$ deve mapear, caso exista mais de dois ciclos esse invariante é repetido até se alcançar o ciclo mais a esquerda.

A conjugação de $\beta$ por $\alpha$, denotada por $\alpha.\beta$ é da forma $\alpha\beta\alpha^{-1}$. Tal operação resulta em uma permutação da mesma estrutura de $\beta$, mas com os elementos "renomeados" por $\alpha$ da seguinte maneira:

$$\alpha\beta\alpha^{-1} = (\alpha(\beta_1)\, \alpha(\beta_2)\, \cdots\, \alpha(\beta_k))$$

## C.2 Formalisto algébrico aplicado a rearranjo de genomas

Para rearranjo de genomas o conjunto base utilizado é $E_n = \{-1, +1, -2, +2, ..., -n, +n\}$ onde $n$ é o número de genes e para cada elemento $+i$, o seu complemento na fita oposta de DNA é chamado $-i$. Denomina-se $\gamma$ a permutação que mapeia cada elemento de $E$ no seu complementar.

$$\gamma = (-1\ +1)(-2\ +2)...(-n\ +n)$$

Para um ciclo $\alpha$, o reverso dado por $\alpha^{-1}$ é obtido invertendo a ordem dos elementos. Assim, se $\alpha = (a\, b\, c)$ então $\alpha^{-1} = (c\, b\, a)$. O complementar reverso de $\alpha$ é, então, dado por $\gamma.(\alpha^{-1})$ ou $(\gamma.\alpha)^{-1}$.

Um conceito importante é o de ciclos admissíveis, que são aqueles ciclos que representam algum trecho de uma fita genômica e, para isso, não podem conter elementos $-i$ e $+i$ ao mesmo tempo. Observa-se que $\gamma$ definido anteriormente não é um ciclo admissível.

Um genoma $\pi$ é o produto de duas fitas complementares reversas $\pi_1$ e $\gamma.(\pi_1^{-1})$. Os problemas de rearranjos aplicados a $\pi$ são em geral definidos da seguinte maneira: dados dois genomas $\pi$, $\sigma$ e uma classe de operações $\rho$, encontre o número mínimo de operações da classe $\rho$ que transformam $\pi$ em $\sigma$. Esse número mínimo é chamado de distância para aquela classe de operações.

Algumas classes de operações são mostradas no trabalho como forma de ilustrar a utilização da notação algébrica. Nas formulações abaixo os literais $u$, $v$, $w$ e $x$ representam elementos em uma mesma fita de $\pi$.

**Reversão com sinal** $\rho = (u\ \gamma\pi v)(v\ \gamma\pi u)$

**Reversão sem sinal** $\rho = (u\ v)(\gamma\pi v\ \gamma\pi u)$

**Transposição sem sinal** $\rho = (u\ v\ w)(\gamma\pi w\ \gamma\pi v\ \gamma\pi u)$

**Transposição com sinal** $\rho = (u\ v\ \gamma\pi w)(w\ \gamma\pi v\ \gamma\pi u)$

**Troca de Blocos** $\rho = (u\ w)(\gamma\pi w\ \gamma\pi u)(vx)(\gamma\pi x\ \gamma\pi v)$

# D Transposition Distance based on the Algebraic Formalism [35]

O trabalho utiliza a abordagem algébrica de Meidanis e Dias (seção C) e apresenta um algoritmo de aproximação para o problema da distância de transposição [5] na razão 1.5, o que exemplifica a viabilidade da abordagem algébrica.

O formalismo algébrico [31] propõe um modo diferente de ver uma permutação e insere novos conceitos como ciclos, conjugação e produto. Dessa forma, recomenda-se fortemente a leitura da seção C antes de iniciar esta seção.

Para uma permutação $\pi$ cujos elementos pertencem ao conjunto $E$, o elemento $x$ é dito fixado quando $\pi(x) = x$. O suporte de $\pi$, $Supp(\pi)$, é o subconjunto de todos os elementos não fixados em $\pi$. Por exemplo, na permutação $\pi = (0\ 3\ 1\ 5\ 4\ 2)$, com elementos pertencentes ao conjunto $E = \{0, 1, 2, 3, 4, 5\}$, os elementos 0 e 4 são fixados e $Supp(\pi) = \{1, 2, 3, 5\}$.

Toda permutação $\pi$ está associada a um conjunto $E$ que contém os elementos de $\pi$. O conjunto $E$ é importante para saber quais elementos existem na permutação $\pi$, pois alguns podem ser omitidos (elementos fixados são normalmente omitidos). Para facilitar a leitura, quando o conjunto $E$ não for mencionado deve-se entender que ele é composto pelos elementos $1 \ldots n$, onde $n$ é o maior elemento que aparece em $\pi$

A órbita de um elemento $x \in E$ em $\pi$ é o conjunto $orb(x, \pi) = \{y | y = \pi^k(x)$ para algum inteiro $k\}$. No exemplo acima, a órbita do elemento 2 é $\{1, 2, 3, 5\}$. O número de órbitas de $\pi$, com os elementos pertencendo ao conjunto $E$, será denominado $\circ(\pi, E)$ e o número de órbitas pares e ímpares de $\pi$ será denominado $\circ_{even}(\pi, E)$ e $\circ_{odd}(\pi, E)$ respectivamente.

Uma órbita é dita trivial quando contém apenas um elemento e não trivial caso contrário. Um ciclo é uma permutação com no máximo uma órbita não trivial e um $k$-ciclo, $k > 1$, é um ciclo cujo órbita não trivial possui tamanho $k$. No exemplo acima, a permutação $\pi$ é um 4-ciclo que pode ser representado por (1 3 5 2). Um $k$-ciclo é dito ímpar se $k$ é ímpar e par, caso contrário.

Para uma permutação $\pi$, a decomposição em ciclos é um produto de órbitas disjuntas que representam $\pi$. Vale ressaltar que elementos fixados são omitidos nesta representação.

Uma transposição utilizando o formalismo algébrico é um 3-ciclo $\gamma = (u\ v\ w)$, com $u, v, w \in E$. A transposição é dita aplicável se $u$, $v$ e $w$ estão no mesmo ciclo na decomposição de $\pi$. Formalmente, diz-se que $w$ segue $v$, $v \rightarrow_{u,\pi} w$ , para o par $(u, \pi)$ quando $v = \pi^{k_1}(u)$, $w = \pi^{k_2}(u)$ e $0 < k_1 < k_2 < |orb(u, \pi)|$.

Assim, o problema da distância de transposição consiste em encontrar um sequência de transposições $\gamma_1, \ldots, \gamma_t$ tal que $\gamma_t \gamma_{t-1} \ldots \gamma_1 \pi = \sigma$, $\gamma_i$ é aplicável a $\gamma_{i-1} \ldots \gamma_1 \pi$, $1 \leq i \leq t$ e t é mínimo.

O produto $\sigma\pi^{-1}$ no formalismo algébrico produz órbitas correspondentes aos ciclos do grafo de ciclos proposto por Bafna e Pevzner [5]. Dessa forma, o trabalho propõe seguir, em linhas gerais, os passos de Bafna e Pevzner em seu algoritmo utilizando este produto.

Dados $\pi$ e $\sigma$ (quando houver referências a $\pi$ e $\sigma$, subentende-se que ambos estão associados ao mesmo conjunto $E$), uma transposição $\gamma$ aplicável a $\pi$ é um $x$-move quando $\circ(\sigma\pi^{-1}\gamma^{-1}) = \circ(\sigma\pi^{-1}) + x$. Em outras palavras, um $x$-move, com $x \in \{-2, 0, 2\}$, aumenta em $x$ o número de ciclos no grafo de ciclos de Bafna e Pevzner [5].

Um $x$-move é um $x$-move válido se a variação de $x$ ocorre no número de ciclos ímpares

do grafo de ciclos de Bafna e Pevzner, ou seja, $\circ_{odd}(\sigma\pi^{-1}\gamma^{-1}) = \circ_{odd}(\sigma\pi^{-1}) + x$.

Abaixo são apresentados alguns lemas que auxiliam na criação do algoritmo de aproximação na razão 1.5. O lemas, bem como as provas (não apresentadas aqui, mas presentes no trabalho), foram enunciados utilizando o formalismo algébrico. Alguns desses resultados já são conhecidos da literatura clássica de rearranjo de genomas, como pode ser visto na seção A.

**Lema 1** *Dados $\pi$ e $\sigma$ e um ciclo não orientado $C = (x \ldots y \ldots z \ldots)$ em $\sigma\pi^{-1}$. Para todo $x, y, z \in C$, o 3-ciclo $\gamma = (x\ y\ z)$, com $y \rightarrow_{z,\pi} x$, transforma $C$ em um ciclo não orientado em $\sigma\pi^{-1}\gamma^{-1}$.*

**Lema 2** *Dados $\pi$ e $\sigma$, ciclos $C = (x \ldots y \ldots)$ e $D = (x' \ldots y' \ldots)$ em $\sigma\pi^{-1}$ e uma transposição $\gamma$ aplicável a $\pi$ formada por três entre os quatro elementos $x$, $y$, $x'$ e $y'$, então:*

1. *Se os ciclos $C$ e $D$ não são ligados, então $\gamma$ cria um ciclo não orientado em $\sigma\pi^{-1}\gamma^{-1}$.*
2. *Se os ciclos $C$ e $D$ são ligados, então $\gamma$ cria um ciclo orientado em $\sigma\pi^{-1}\gamma^{-1}$.*

**Lema 3** *Dados $\pi$ e $\sigma$, se há um ciclo orientado em $\sigma\pi^{-1}$ então há uma transposição aplicável a $\pi$ tal que $\gamma$ é um 2-move válido ou existem as transposições $\gamma_1$, $\gamma_2$ e $\gamma_3$ aplicadas a $\pi$, $\gamma_1\pi$ e $\gamma_2\gamma_1\pi$ respectivamente que são um 0-move válido e dois 2-move válidos respectivamente.*

**Lema 4** *Dados $\pi$ e $\sigma$, um 3-ciclo $\gamma = (u\ v\ w)$ com $v \rightarrow_{u,\pi} w$, $x$, $y$ e $z \in C$, tal que $d(x, y)$ ímpar, $C = (x \ldots y \ldots z \ldots)$ um $k$-ciclo não orientado, $k \geq 3$, em $\sigma\pi^{-1}$ e $\gamma$ e $C$ se cruzam, então $\gamma$ transforma $C$ em um ciclo fortemente orientado em $\sigma\pi^{-1}\gamma^{-1}$.*

**Lema 5** *Dados $\pi$ e $\sigma$, com $y \rightarrow_{x',\pi} y' \rightarrow_{x',\pi} x$, $D = (x \ldots y \ldots)$ e $D' = (x' \ldots y' \ldots)$ dois ciclos não orientados em $\sigma\pi^{-1}$, $D$ e $D'$ não se cruzam, mas estão ligados e $d(x, y)$ é ímpar. Então $\gamma$ formado por três dentre os quatro elementos $x$, $y$, $x'$ e $y'$ tais que $\gamma$ é aplicável a $\pi$, cria um ciclo fortemente orientado.*

**Lema 6** *Dados $\pi$ e $\sigma$. Se em $\sigma\pi^{-1}$ há um ciclos fortemente orientado fortemente cruzado com um ciclo não orientado, então existem dois 2-moves válidos consecutivos.*

**Lema 7** *Dados $\pi$ e $\sigma$ sem ciclos orientados em $\sigma\pi^{-1}$. Se há no mínimo um $k$-ciclo não orientado com $k > 3$ em $\sigma\pi^{-1}$, então existem as transposições $\gamma_1$, $\gamma_2$ e $\gamma_3$ aplicadas a $\pi$, $\gamma_1\pi$ e $\gamma_2\gamma_1\pi$ respectivamente que são um 0-move válido e dois 2-move válidos respectivamente.*

O pseudocódigo do algoritmo que utiliza os lemas acima é mostrado no trabalho, bem como a comparação com outros algoritmos existentes na literatura.

# E Sorting permutations by block-interchanges [10]

O trabalho aborda uma operação em rearranjo de genomas chamada troca de blocos. Uma troca de blocos é um evento que não ocorre realmente na natureza, mas é um problema cuja solução pode auxiliar na resolução do problema da distância da transposição, pois uma troca de blocos é uma generalização de uma transposição, mas que possui solução em

tempo polinomial. Dessa forma, o presente trabalho, apesar de não resolver um problema real, lança luz sobre o problema de transposição.

Dada uma permutação $\pi = \pi_1, \pi_2, \cdots, \pi_n$, uma troca de blocos $\rho(i, j, k, l)$ onde $1 \leq i < j \leq k < l \leq n + 1$ é uma operação em que dois blocos $[\pi_i \cdots \pi_{j-1}]$ e $[\pi_k \cdots \pi_{l-1}]$ são trocados de posição. É importante perceber o caso especial da transposição, que é uma troca de blocos consecutivos onde $j = k$.

O artigo utiliza o conceito de *breakpoints* definido em [3], ou seja, para uma dada permutação $\pi$, acrescenta-se inicialmente dois elementos $\pi_0 = 0$ e $\pi_{n+1} = n + 1$ e, após isso, define-se *breakpoint* como um valor de $i$ tal que $1 \leq i \leq n + 1$ e $\pi_i - \pi_{i-1} \neq 1$. Importante mencionar que a permutação identidade é a única que não possui *breakpoints* e uma operação de troca de blocos pode alterar em 4 o número de *breakpoints*.

O resultado principal do trabalho é apresentar um método que resolve otimamente o problema em tempo polinomial utilizando apenas o conceito de *breakpoint*, apesar de necessitar da estrutura de grafo de ciclos de cores alternadas definida em [3] para provar que o método resolve otimamente o problema. O algoritmo parte do princípio de que, para qualquer permutação $\pi$ que não seja identidade, existe uma troca de blocos que remove no mínimo dois *breakpoints*. Em seguida é provado que um algoritmo que aplica essa troca de blocos ordena $\pi$ com o menor número de operações possível.

Para encontrar uma operação de troca de blocos que remove no mínimo dois *breakpoints* de uma permutação que não é identidade utiliza-se o seguinte procedimento: se $\pi$ não é identidade então existem no mínimo dois elementos $x$ e $y$, tais que $x < y$ mas $\pi = [\cdots y \cdots x \cdots]$. Dessa forma, sendo $x$ o menor valor possível em que isso ocorre então $x - 1$ deve estar a esquerda de $y$ em $\pi$ e, de modo análogo, $y + 1$ deve estar a direita de $x$.

Assim, uma operação de troca de blocos que remove no mínimo dois *breakpoints* é transformar uma permutação $\pi = [1 \cdots x - 1 \cdots y \cdots x \cdots y + 1 \cdots]$ em $\pi' = [1 \cdots x - 1\ x \cdots y\ y + 1 \cdots]$. Isso leva a concluir que a operação $\rho$ de troca de blocos deve possuir os seguintes parâmetros $(\pi^{-1}(x - 1) + 1, \pi^{-1}(y) + 1, \pi^{-1}(x), \pi^{-1}(y + 1))$.

# F Genome Rearrangements and Sorting by Reversals [2]

O trabalho é um dos primeiros a fornecer resultados computacionais para o problema de rearranjo de genomas por reversões, em um período em que havia poucas ferramentas disponíveis aos biólogos sobre o assunto. O trabalho apresenta um algoritmo de aproximação com uma razão de performance de 1.5 quando a orientação dos genes é conhecida e 1.75 caso contrário.

O trabalho, assim como os posteriores, representa um genoma como uma permutação $\pi = \pi_1 \pi_2 \cdots \pi_n$, onde os elementos $\pi_i, \forall 1 \leq i \leq n$ representam os genes ou agrupamentos de genes na ordem que aparecem. Utilizando essa representação, uma reversão $\rho$ de um intervalo $[i, j]$ é a permutação:

$$\rho = (1, 2, \cdots, i - 1, j, j - 1, \cdots, i + 1, i, j + 1, \cdots, n)$$

e dessa forma, $\pi.\rho$ possui o efeito de reverter genes $\pi_i, \pi_i + 1, ..., \pi_j$.

O problema da distância de reversão entre duas permutações $\pi$ e $\sigma$ busca encontrar uma série de reversões $\rho_1, \rho_2, \cdots, \rho_t$ tais que $\pi.\rho_1.\rho_2 \cdots \rho_t = \sigma$ e $t$ é mínimo. É importante notar que a distância de reversão entre $\pi$ e $\sigma$ é igual a distância de reversão entre $\sigma^{-1}\pi$ e a identidade. Assim, ordenar um genoma $\pi$ é o problema de encontrar a distância de reversão $d(\pi)$ entre $\pi$ e a identidade.

## F.1 Grafo de *breakpoints*

O trabalho introduz conceitos de suma importância para que, em trabalhos posteriores, sejam implementados algoritmos polinomiais para o problema. Entre os conceitos introduzidos vale citar a noção de grafo de *breakpoints* e a decomposição do grafo em ciclos.

A definição de *breakpoints* [36, 45] é feita sobre a versão estendida de uma permutação $\pi = \pi_1, \pi_2, \cdots, \pi_n$, que é a própria permutação adicionada dos elementos $\pi_0 = 0$ e $\pi_{n+1} = n + 1$ e é enunciada como segue: escrevendo $i \sim j$ se $|i - j| = 1$, diz se que um par de elementos $(\pi_i, \pi_{i+1})$ é uma adjacência se $\pi_i \sim \pi_{i+1}$ e um *breakpoint* caso contrário.

Como a permutação identidade não possui *breakpoints*, então ordenar por reversão envolve diminuir o número de *breakpoints*. Nesse caso, como uma dada reversão pode remover no máximo apenas dois *breakpoints* então é possível encontrar o seguinte limitante inferior:

$$d(\pi) \geq \frac{b(\pi)}{2}$$

onde $b(\pi)$ é o número de *breakpoints* da permtuação $\pi$ e $d(\pi)$ é a distância de $\pi$ em relação a identidade.

O grafo de *breakpoints* $G(\pi)$ de uma permutação $\pi$ de $n$ elementos é uma construção com $n + 2$ vértices, que correspondem a cada elemento da notação estendida de $\pi$. As arestas são criadas coloridas de preto e cinza da seguinte maneira: une-se os vértices $i$ e $j$ por uma aresta preta se $(i, j)$ é um *breakpoint* de $\pi$ e por uma aresta cinza se $i \sim j$ e $i, j$ não são consecutivos em $\pi$.

No grafo de *breakpoints* é possível definir um ciclo alternante $C$ como um ciclo em que duas arestas consecutivas possuem cores distintas. A máxima decomposição de $G(\pi)$ é o número máximo de ciclos $c(\pi)$ alternantes com arestas disjuntas.

A decomposição em ciclos possui um papel importante para estimar a distância de reversão. Quando se aplica uma reversão em uma permutação há uma mudança no número de *breakpoints* e no número de ciclos em uma máxima decomposição, sendo que há uma forte correlação entre essas duas mudanças.

Define-se $b = b(\pi)$ o número de *breakpoints* de $G(\pi)$ e $c = c(\pi)$ o número de ciclos de uma máxima decomposição de $G(\pi)$. Uma reversão $\rho$ arbitrária gera um novo grafo de *breakpoints* $G' = G(\pi\rho)$, uma quantidade $b' = b(\pi\rho)$ de breakpoints em $\pi\rho$ e $c' = c(\pi\rho)$ o número de ciclos de uma máxima decomposição de $G'$. Sendo $\Delta b = b - b'$ e $\Delta c = c - c'$ o trabalho apresenta os seguintes teoremas.

**Teorema 6** *Para cada permutação $\pi$ e reversão $\rho$, $\Delta b(\pi, \rho) + \Delta c(\pi, \rho) \leq 1$*

**Teorema 7** *Partindo do teorema 6 é possível chegar ao seguinte limitante inferior:*

$$d(\pi) \geq b(\pi) - c(\pi)$$

Além desses teoremas, o trabalho apresenta um algoritmo de aproximação com uma razão de performance de 1.5 para permutações com sinal e 1.7 para permutações sem sinal. Entretanto, entender o algoritmo não é tão proveitoso quanto os conceitos já estudados neste resumo para compreender os trabalhos que foram desenvolvidos posteriormente.

# G Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals [21]

O trabalho apresenta o primeiro algoritmo polinomial para o problema da distância de reversão entre dois genomas quando se conhece a orientação dos genes. Além desse algoritmo polinomial, apresenta-se como resultado importante um teorema para calcular a distância de ordenação de uma permutação com sinal. Esse teorema complementa o resultado de [2] pela inserção de um novo parâmetro denominado obstáculo, que havia sido ignorado.

O problema da distância de reversão entre dois genomas $\pi$ e $\sigma$ é encontrar o número mínimo $n$ de reversões $\rho_1, \rho_2, \cdots, \rho_n$, tal que $\pi\rho_1\rho_2\cdots\rho_n = \sigma$. O problema de ordenar um genoma $\pi$ por reversões é encontrar a distância entre $\pi$ e a permutação identidade.

Como o presente trabalho utiliza estruturas introduzidas em trabalhos anteriores, entre elas os conceitos de *breakpoints* e decomposição em ciclos [2], recomenda-se a leitura do apêndice F para maiores informações. Entretanto, os conceitos mais importantes para o entendimento dos resultados do trabalho de Hannenhalli e Pevzner serão mostrados logo a seguir.

***Breakpoint*** Noção introduzida por [45] e [36] cuja definição é feita sobre a versão estendida de uma permutação $\pi = \pi_1, \pi_2, \cdots, \pi_n$, que é a própria permutação adicionada dos elementos $\pi_0 = 0$ e $\pi_{n+1} = n + 1$. O enunciado segue: sendo $i \sim j$ se $|i - j| = 1$, diz se que um par de elementos $(\pi_i, \pi_{i+1})$ é uma adjacência se $\pi_i \sim \pi_{i+1}$ e um *breakpoint* caso contrário.

**Grafo de *breakpoints*** Representando $G(\pi)$ como o grafo de *breakpoints* em realação a permutação $\pi$ de $n$ elementos. $G(\pi)$ é uma construção com $n + 2$ vértices, que correspondem a cada elemento da notação estendida de $\pi$. As arestas são criadas coloridas de preto e cinza da seguinte maneira: dados dois vértices $\pi_i$ e $\pi_j$, cria-se arestas pretas entre eles se $(\pi_i, \pi_j)$ formam um *breakpoint* em $\pi$ e arestas cinzas se $(i, j)$ formam um *breakpoint* em $\pi^{-1}$.

**Ciclos alternantes em** $G(\pi)$ São ciclos em $G(\pi)$ em que duas arestas consecutivas possuem cores distintas. O tamanho de um ciclo alternante, $l(C)$, equivale ao número de arestas pretas que ele contém, se $l(C) = 2$, então classifica-se $C$ como ciclo curto e se $l(C) > 2$ então classifica-se $C$ como ciclo longo. Uma permutação $\pi$ é simples se não possuir ciclos longos.

**Máxima decomposição em ciclos de** $G(\pi)$ O grafo de *breakpoints* $G(\pi)$ pode ser decomposto em vários ciclos alternantes. Denota-se $c(\pi)$ o número de ciclos de uma decomposição máxima em ciclos alternantes com arestas disjuntas. Encontrar $c(\pi)$ é um problema difícil [28].

## G.1 Grafo de *breakpoints* para permutações com sinal

Em [2] notou-se que o conceito de grafo de *breakpoints* pode ser extendido para permutações com sinal, que são aquelas permutações geradas quando são conhecidas as orientações dos genes. Entretanto, essa questão é bem melhor abordada no presente trabalho.

Para se obter o grafo de *breakpoints* de uma permutação com sinal é preciso transformar a permutação com sinal $\pi$ com $n$ elementos para uma permutação sem sinal $\pi'$ com $2n$ elementos como segue: substituir os elementos positivos $+x$ em $\pi$ por dois elementos $2x-1, 2x$ em $\pi'$ e os elementos negativos $-x$ em $\pi$ por dois elementos $2x, 2x-1$ em $\pi'$.

A permutação sem sinal $\pi'$ é denominada imagem de $\pi$ e o que torna a solução do problema de permutação com sinal mais simples é existir uma única decomposição de $\pi'$ em ciclos alternantes. Observa-se que o efeito de cada reversão que ocorre em $\pi$ pode ser (representado) por uma única reversão em $\pi'$, o que permite afirmar que $d(\pi) = d(\pi')$. O trabalho, então, passa a focar apenas nas permutações sem sinal que são imagens de alguma permutação com sinal, ou seja, de agora em diante todas as referências a uma permutação $\pi$ deixam implícito que $\pi$ é uma permutação sem sinal imagem de alguma permutação com sinal.

Utilizando o grafo de breakpoints é possível analisar se algumas reversões aplicadas a uma permutação $\pi$ são úteis ou não. Nesse contexto, uma reversão $\rho(i, j)$ atua apenas se $(\pi_{i-1}, \pi_i)$ e $(\pi_j, \pi_{j+1})$ forem uma aresta preta de algum ciclo $C$ de $G(\pi)$. Nesse caso, dada uma reversão arbitrária $\rho$ e sendo $\Delta b(\pi, \rho) = b(\pi\rho) - b(\pi)$ e $\Delta c(\pi, \rho) = c(\pi\rho) - c(\pi)$, se $\Delta b(\pi, \rho) - \Delta c(\pi, \rho) = 1$, então $\rho$ é uma reversão própria.

Para caracterizar quando uma reversão é própria é preciso verificar se uma aresta cinza $g$ é orientada ou não orientada. Sendo $(\pi_i, \pi_j)$ uma aresta cinza incidente nas arestas pretas $(\pi_k, \pi_i)$ e $(\pi_j, \pi_l)$, então $(\pi_i, \pi_j)$ é orientada se, e somente se, $i-k = j-l$. A importância de conhecer se uma aresta é orientada é que se $(\pi_i, \pi_j)$ é orientada, então uma reversão agindo nas arestas pretas $(\pi_k, \pi_i)$ e $(\pi_j, \pi_l)$ será uma reversão própria. Dado que um ciclo em $G(\pi)$ possui uma aresta cinza orientada, então diz-se que o ciclo é orientado, caso contrário o ciclo será não orientado.

## G.2 Grafo de entrelaçamento, obstáculos e fortaleza

Dadas duas arestas cinzas $(\pi_i, \pi_j)$ e $(\pi_k, \pi_t)$ em $G(\pi)$, ambas estão entrelaçadas se os intervalos $[i, j]$ e $[k, t]$ estão sobrepostos, mas um não contém o outro. Seguindo essa notação, dois ciclos $C_1$ e $C_2$ são entrelaçados caso existam arestas cinzas $g_1 \in C_1$ e $g_2 \in C_2$ entrelaçadas.

Seja $C_\pi$ o conjunto de ciclos em $G(\pi)$, define-se o grafo de entrelaçamento $H_\pi(C_\pi, I_\pi)$ de $\pi$ com o conjunto de arestas

$$I_\pi = \{(C1, C2) : C1 \text{ e } C2 \text{ são ciclos entrelaçados em } G(\pi)\}$$

Como os vértices de $H_\pi$ são os ciclos em $C_\pi$, define-se que um vértice é orientado caso o ciclo ao qual se relaciona é orientado, caso contrário será não orientado. O grafo $H_\pi$ é formado por vários componentes conexos e para um componente conexo $U$ define-se que é orientado se possuir pelo menos um vértice orientado, caso contrário será não orientado.

Um conceito importante para definição de um limitante inferior e para implementação do algoritmo polinomial é o de obstáculo, que, em poucas palavras, é um componente não

orientado que não separa outros componentes não orientados, sendo que se diz que um componente $U$ separa dois outros componentes $U'$ e $U''$ se existe uma aresta cinza $(\pi_i, \pi_j)$ em $U$ tal que todos os elementos de $U'$ estejam no intervalo $[i, j]$ e não haja elemento de $U''$ no intervalo $[i, j]$.

Denominando $h(\pi)$ o número de obstáculos associados a uma permutação $\pi$, o trabalho apresenta o seguinte teorema, que é um limitante inferior melhor do que o obtido em [2].

**Teorema 8** *Para qualquer permutação $\pi$, têm-se*

$$d(\pi) \geq b(\pi) - c(\pi) + h(\pi)$$

Seja $U_\pi$ o conjunto de componentes não orientados em $H_\pi$, diz-se que um obstáculo $K \in U_\pi$ proteje um não obstáculo $J \in U_\pi$ se ao removermos $K$ de $U_\pi$ o não obstáculo $J$ passa a ser um obstáculo. Um obstáculo em $U_\pi$ é um super-obstáculo se protege um não obstáculo $J \in U_\pi$ e um simples-obstáculo caso contrário.

O trabalho apresenta várias operações para lidar com os obstáculos e super-obstáculos como, por exemplo, junção e corte de obstáculos. Entretanto, a maioria dessas operações, bem como o algoritmo de ordenação, são bastante complexos e vão além do objetivo deste resumo.

Uma última consideração precisa ser feita quanto a um caso bem particular de permutações conhecidas como fortalezas. Uma fortaleza é uma permutação $\pi$ que possui um número ímpar de obstáculos e todos esses obstáculos são super-obstáculos. Nesse caso especial o trabalho apresenta o seguinte teorema.

**Teorema 9** *Se $\pi$ é uma fortaleza, então $d(\pi) \geq b(\pi) - c(\pi) + h(\pi) + 1$*

Com os teoremas 8 e 9 é possível generalizar uma equação para calcular exatamente a distância de ordenação de uma permutação arbitrária. Essa equação é mostrada no próximo teorema.

**Teorema 10** *Para toda permutação $\pi$,*

$$d(\pi) = \begin{cases} b(\pi) - c(\pi) + h(\pi) + 1 & \textit{se } \pi \textit{ é uma fortaleza} \\ b(\pi) - c(\pi) + h(\pi) & \textit{caso contrário} \end{cases}$$

# H Polynomial-sized ILP Models for Rearrangement Distance Problems [12]

O trabalho apresenta um modelo para a distância de rearranjo de genomas usando Programação Linear Inteira (PLI). O modelo é o primeiro conhecido a possuir tamanho polinomial. De uma maneira mais específica, o trabalho é restrito aos problemas das distâncias de reversão e transposição, além da distância quando ambos os eventos são permitidos.

Os conceitos relacionados ao domínio de rearranjo de genomas são fortemente baseados nos trabalhos de Bafna e Pevzner sobre transposições [3] e sobre reversões [2], além de Hannenhalli e Pevzner [21].

Um genoma arbitrário formado por $n$ genes será representado como uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ onde cada elemento de $\pi$ representa um gene. O genoma identidade $\iota$ é definido como $\iota_n = (1\ 2\ \ldots\ n)$.

Utilizando essa notação, define-se os eventos de transposição e reversão da seguinte maneira: uma transposição $\rho(x, y, z)$, onde $1 \leq x < y < z \leq n + 1$, é um rearranjo que transforma $\pi$ em $\rho\pi = (\pi_1\ \ldots\ \pi_{x-1}\ \pi_y\ \ldots\ \pi_{z-1}\ \pi_x\ \ldots\ \pi_{y-1}\ \ \pi_z\ \ldots\ \pi_n)$.

Uma reversão $\rho(x, y)$, onde $1 \leq x < y \leq n$, é um evento de rearranjo que transforma $\pi$ em $\rho\pi = (\pi_1\ \ldots\ \pi_{x-1}\pi_y\ \pi_{y-1}\ \ldots\ \pi_{x+1}\ \pi_x\ \pi_{y+1}\ \ldots\ \pi_n)$. É importante notar a diferença no número de parâmetros entre os eventos de transposição e reversão, respectivamente 3 e 2. Essa diferença será importante para o modelo PLI.

Um modelo PLI é baseado em formulações matemáticas. O trabalho divide essas equações em dois grupos. O primeiro envolve as equações matemáticas que introduzem as variáveis e restrições que são comuns a todos os modelos de distância e cujo papel é garantir que apenas as permutações válidas serão contempladas pelo modelo. A seção H.1 define essas equações.

O segundo grupo envolve algumas restrições específicas para transposição (seção H.2) e reversão (seção H.3). As restrições específicas para os eventos de transposição e reversão serão utilizadas na modelagem da distância quando ambos os os eventos são permitidos, na seção H.4 (exceto quando for explicitado o contrário).

Nas equações seguintes representaremos, por motivos de clareza, um elemento $\pi_i$ por $\pi[i]$.

## H.1 Restrições comuns a todos os modelos

As variáveis binárias $B_{ijk}$ indicam se a $i$-ésima posição de $\pi$ possui valor $j$ após a $k$-ésima operação, onde $1 \leq i,\ j \leq n$ e $0 \leq k < n$:

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

Dadas essas variáveis, as seguintes restrições garantem que a permutação inicial e final são corretas, respectivamente:

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

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

É preciso, no entanto, garantir que as permutaçãos intermediárias serão válidas. Para isso, são estabelecidas duas restrições, a primeira é que qualquer posição de uma permutação possui exatamente um valor associado a ela.

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

A segunda restrição deve garantir que todos os valores no intervalo $[1..n]$ sejam atribuídos a alguma posição da permutação.

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

## H.2 Restrições específicas para transposições

Para as equações que representam uma transposição, utilizou-se as variável binárias $t_{abck}$, definidas para todo $1 \leq a < b < c \leq n+1$ e todo $1 \leq k < n$. A variável $t_{abck}$ indica se a $k$-ésima transposição é um evento de troca entre os blocos $\pi[a \ldots b-1]$ e $\pi[b \ldots c-1]$ de $\pi$.

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

A partir da variável $t_{abck}$ é possível definir a variável binária $t_k$, que é usada para decidir se a $k$-ésima transposição modificou a permutação.

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

Dado que a transposição não modificou uma permutação, as transposições subseqüentes não podem modificá-la também.

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

Na seção H.4 serão criadas restrições para a distância quando os eventos de transposição e reversão são permitidos. É interessante ressaltar que a restrição acima não será incluída naquele conjunto de restrições.

Uma última restrição é necessária para impor que em cada passo no máximo uma transposição seja executada.

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

Com base nas restrições acima, é possível gerar uma nova restrição que reflete as modificações na permutação causadas por eventos de transposição.

$$\sum_{a=i+1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{n+1} t_{abck} + \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=b+1}^{i} t_{abck} + (1 - t_k) + B_{ijk-1} - B_{ijk} \leq 1,$$

para todo $1 \leq i, j \leq n$, e todo $1 \leq k < n$.

Os autores dividem a análise do significado dessa inequação em três casos e apresentam para cada caso a prova de que a fórmula está correta.

1. Os valores nas posições $i < a$ ou $i \geq c$ permanecem inalterados.
2. As posições $a \leq i < a+c-b$ são preenchidas com elementos que estavam em $[b, c-1]$.
3. As posições $a+c-b \leq i < c$ são preenchidos com elementos que estavam em $[a, b-1]$.

Por fim, é possível inserir restrições obtidas a partir de limitantes apresentados por Bafna e Pevzner [3]. Como esses limitantes são específicos para a distância de transposição, as duas restrições abaixo não participarão da modelagem da seção H.4.

$$t_k * n + k - 1 \geq LB(\pi, \sigma), \text{ para todo } 1 \leq \text{k} \leq \text{n}.$$

$$t_k * k \leq UB(\pi, \sigma), \text{ para todo } 1 \leq \text{k} \leq \text{n}.$$

onde $LB(\pi, \sigma)$ e $UB(\pi, \sigma)$ são, respectivamente, os limitantes inferiores e superiores obtidos por Bafna e Pevzner.

O modelo PLI para eventos de transposição descrito nesta seção possui um conjunto de variáveis e restrições de ordem polinomial no tamanho da permutação fornecida como entrada. O número de variáveis é $O(n^4)$ e o número de restrições é $O(n^6)$.

## H.3 Restrições específicas para reversões

Para as equações que representam reversões, utilizou-se a variável binária $r_{abk}$ definidas para todo $1 \leq a < b \leq n$ e todo $1 \leq k < n$. A variável $r_{abk}$ indica se a $k$-ésima reversão afeta o bloco $\pi[a \dots b]$ de $\pi$.

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

A partir da variável $r_{abk}$ é possível definir a variável binária $r_k$, que é usada para decidir se a $k$-ésima reversão modificou a permutação.

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

Dado que a reversão não modificou uma permutação, as transposições subseqüentes não podem modificá-la também.

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

A restrição acima não será utilizada na modelagem da seção H.4.

Uma última restrição é necessária para impor que em cada passo no máximo uma reversão seja executada.

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

Com base nas restrições acima, é possível gerar uma nova restrição que reflete as modificações na permutação causadas por eventos de reversão.

Os autores inserem duas novas restrições que lidam com o problema da distância de reversão propriamente dito:

- Os valores nas posições $i < a$ ou $i > b$ permanecem inalterados:

$$\sum_{a=i+1}^{n-1}\sum_{b=a+1}^{n} r_{abk} + \sum_{a=1}^{n-1}\sum_{b=a+1}^{i-1} r_{abk} + B_{i,j,k-1} + (1 - r_k) - B_{ijk} \leq 1,$$

  para todo $1 \leq i, j \leq n$, e todo $1 \leq k < n$.

- As posições $a \leq i \leq b$ sofrem reversão de seus elementos:

$$r_{abk} + B_{b+a-i,j,k-1} - B_{ijk} \leq 1,$$

  para todo $1 \leq a < b \leq n$, $a \leq i \leq b$, $1 \leq j \leq n$ e $1 \leq k < n$.

O modelo PLI para eventos de reversão descrito nesta seção possui um conjunto de variáveis e restrições de ordem polinomial no tamanho da permutação fornecida como entrada. O número de variáveis é $O(n^3)$ e o número de restrições é $O(n^5)$.

## H.4 Distância de transposição e reversão

Para formular o problema da distância de transposição e reversão usa-se as variáveis e restrições definidas anteriormente, exceto, como já foi dito, aquelas em que, no momento de sua definição, foi ressaltado que eram específicas para o modelo outrora em questão.

Inclui-se a variável $z_k$ para denotar se a $k$-ésima operação, que poderia ser uma reversão ou uma transposição, modifica a permutação:

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

Duas inequações são ainda necessárias. A inequação seguinte indica que se um dado evento de rearranjo não modificou a permutação, então os eventos seguintes não a modificam também.

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

A restrição seguinte garante que apenas um evento de rearranjo (ou reversão ou transposição), serão executados em cada passo.

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

O modelo PLI para eventos de reversão e transposição possui um conjunto de variáveis e restrições de ordem polinomial no tamanho da permutação fornecida como entrada. O número de variáveis é $O(n^4)$ e o número de restrições é $O(n^6)$.

No restante do trabalho, os autores detalham os experimentos computacionais realizados com o modelo. O resultado mais significativo é que o modelo, apesar de polinomial no tamanho, é pouco viável devido o tempo que demora para terminar a execução.

# I Sorting a Bridge Hand [15]

O trabalho apresenta um limitante superior que demonstra ser possível ordenar qualquer permutação de tamanho $n \geq 9$ em $\lfloor 2(n-1)/3 \rfloor$ transposições. Além disso, os autores fornecem uma expressão exata para a distância entre uma permutação e a sua reversa, notadamente $\lceil (n+1)/2 \rceil$. Por fim, o trabalho esclarece que a distância entre uma permutação e sua reversa nem sempre caracteriza o diâmetro, como conjecturava-se anteriormente [11,32], com o primeiro contra-exemplo surgindo para $n = 13$.

O trabalho traz como motivação o problema de rearranjo por transposições em biologia computacional e a ordenação de uma mão de bridge. Entretanto, como se está interessado apenas no primeiro, as várias considerações sobre o segundo foram suprimidas deste resumo.

A definição de transposição, o modo de representar uma permutação e a nomenclatura básica utilizada neste resumo podem ser encontradas na seção A. Apenas serão adicionados dois símbolos $\iota$ e $w_0$ que representarão a permutação identidade e sua reversa respectivamente.

$$\iota = (1\ 2\ \ldots\ n-1\ n) \text{ e } w_0 = (n\ n-1\ \ldots\ 2\ 1)$$

## I.1 Modelo toroidal de permutações

É possível estender uma permutação arbitrária $\pi$ para uma permutação circular $\pi^\circ$ inserindo um elemento 0, que será o predecessor de $\pi_1$ e sucessor de $\pi_n$. Por exemplo, uma permutação original (3 1 2) pode ser estendida para (0 3 1 2) = (3 1 2 0) = (1 2 0 3) = (2 0 3 1). Para obter a permutação original basta remover o elemento 0 e colocar o seu sucessor como o primeiro elemento.

Partindo-se de uma permutação circular $\pi^\circ$, define-se a operação:

$$m + \pi^\circ = (m\ \ m + \pi_1\ \ m + \pi_2 \ldots m + \pi_n)\ (mod\ n+1),$$

onde $m$ é um número inteiro. Por exemplo, dada uma permutação $\pi = (3\ 1\ 2)$ temos,

$\pi^\circ \quad = (0\ 3\ 1\ 2)$
$1 + \pi^\circ \ = (1\ 0\ 2\ 3)$
$2 + \pi^\circ \ = (2\ 1\ 3\ 0)$
$3 + \pi^\circ \ = (3\ 2\ 0\ 1)$

A permutação toroidal $\pi^\circ_\circ$ é a classe de equivalência gerada a partir de $\pi^\circ$. Assim, $(3\ 1\ 2)^\circ_\circ = (2\ 3\ 1)^\circ_\circ = (2\ 1\ 3)^\circ_\circ = (1\ 3\ 2)^\circ_\circ$. É interessante notar que, dadas duas permutações circulares $\pi^\circ$ e $m+\pi^\circ$, se uma seqüência de transposições transforma $\pi^\circ$ em $\iota$, então a mesma seqüência transformará $m + \pi^\circ$ em $m + \iota = \iota$.

O grupo simétrico $S_n$ é o conjunto de todas as permutações de $\{1, 2, \ldots, n\}$. Assim, é possível definir o grafo *Cayley*, com conjunto de vértices igual a $S_n$ e uma aresta orientada $\rho(i, j, k)$ de todo $\pi \in S_n$ para $\pi\rho(i, j, k)$. Uma seqüência de transposições que ordena $\pi$ é um caminho de $\pi$ a $\iota$. Lendo os rótulos do caminho têm-se $\pi\rho_1\rho_2 \ldots \rho_l = \iota$.

A inversa de uma transposição $\rho(i, j, k)$ é uma transposição $\rho(i, j, k)^{-1} = \rho(i, r, k)$, onde $r = i + k - j$. Isso significa que $\pi = \iota\rho_l^{-1} \ldots \rho_2^{-1}\rho_1^{-1}$. Dessa forma, é fácil perceber que, se há um caminho orientado de $\pi$ a $\sigma$, então existe um caminho de $\sigma$ a $\pi$ passando pelos mesmos vértices. Além disso, se $\pi$ e $\sigma$ pertencem a mesma permutação toroidal, então $\pi^{-1}$ e $\sigma^{-1}$ também representarão, entre sí, a uma mesma permutação toroidal.

Assim, unindo qualquer par de arestas opostas no grafo *Cayley* em uma única aresta não orientada, é possível obter um grafo não orientado. Por conseguinte, unindo em um único vértice todos os vértices que representam a mesma permutação toroidal, é obtido o grafo toroidal.

## I.2 Algoritmo ótimo para ordenar $w_0$

**Teorema 11** *Para $n \geq 3$, é possível ordenar $w_0$ em $\lceil (n+1)/2 \rceil$ transposições e essa ordenação é ótima.*

Inicialmente, para permutações de tamanho ímpar $n = 2k+1$, é possível ordenar usando $k+1$ transposições.

Primeiro move-se o bloco $[k+1\ k]$ para o início da permutação, logo em seguida $[k+2\ k-1]$ é inserido no meio do bloco recém movido. Esse invariante continua até que o último bloco a ser movido seja $[n\ 0]$, depois do qual a permutação será $(k+1 \ldots n\ 0 \ldots k)$, um equivalente toroidal da permutação identidade. Para $n = 2k$ é possível utilizar um algoritmo similar.

É necessário, entretanto, verificar que o algoritmo é ótimo. Definindo *descent* em uma permutação $\pi$ como a ocorrência de $[\pi_k\ \pi_{k+1}]$ tal que $\pi_k > \pi_{k+1}$. O trabalho apresenta o lema 8, cuja demonstração não será apresentada neste resumo.

**Lema 8** *O número de descents em uma permutação pode decrescer de no máximo dois por transposição.*

A permutação $w_0$ possui $n-1$ *descents*, enquanto $\iota$ não possui nenhum. É possível perceber que a primeira e a última transposição só poderão diminuir em 1 o número de *descents*, enquanto que as transposições intermediárias do algoritmo alcançam o limite do lema 8. Dessa forma, o número mínimo de transposições intermediárias é $(n-3)/2$.

## I.3 Limitante superior para o problema do diâmetro

O diâmetro é a máxima distância possível entre duas permutações de um dado tamanho $n$. Pode-se também definir o diâmetro como a máxima distância entre $\iota$ e uma permutação $\pi$ arbitrária. Assim, representando o diâmetro por $d(n)$ temos:

$d(n) = \max_{\pi \in S_n} d(\pi)$.

**Teorema 12** *Um limitante superior para o número de transposições necessárias para ordenar uma permutação $\pi$ de tamanho $n$ é $d(n) \leq \lfloor 2(n-1)/3 \rfloor$ para $n \geq 9$.*

A demonstração do limitante superior necessita do lema 9, cuja demonstração não será apresentada neste resumo.

**Lema 9** *Seja $\pi$ uma permutação diferente de $w_0$. Então é possível encontrar transposições $\rho$ e $\sigma$ tais que uma das permutações $\pi\sigma\rho$, $\sigma\pi\rho$ ou $\sigma\rho\pi$ tenha no mínimo três ocorrências de $[x\ x+1]$, onde $0 \leq x < n$.*

Para alcançar o limitante é preciso, inicialmente, provar que $d(\pi) \leq 2 + d(n-3)$ para qualquer $\pi$ com $n \geq 9$. Se $\pi = w_0$ essa inequação é uma conseqüência do que foi exposto na seção anterior, pois $d(w_0) = \lceil (n+1)/2 \rceil$. Então, $d(n-3) \geq \lceil (n+1)/2 \rceil - 2$, ou ainda, fazendo $k = n-3$, $d(k) \geq \lceil k/2 \rceil$, o que é verdade pelo teorema 11.

Se $\pi \neq w_0$ então um dos três casos do lema 9 se aplica. Supondo o caso em que $\sigma\pi\rho$ se aplica, então $d(\sigma\pi\rho) \leq d(n-3)$, pois os elementos $[x\ x+1]$ podem ser tratados como um só. Assim, como $d(\pi) \leq d(\sigma\pi\rho) + 2$, então $d(\pi) \leq 2 + d(n-3)$. Os outros dois casos são similares.

Entretanto, se para uma permutação arbitrária $\pi$ temos $d(\pi) \leq 2 + d(n-3)$, então a definição de $d(n)$ implica que $d(n) \leq 2 + d(n-3)$.

O trabalho mostra que se $9 \leq n \leq 11$ então $d(n) = \lfloor 2(n-1)/3 \rfloor$. Dessa forma, o teorema 12 segue por indução.

## I.4 Conjecturas sobre o diâmetro

Os valores de $d(n)$ para $n \leq 10$ foram calculados por um algoritmo de busca em largura aplicado na construção do grafo *Cayley*. Como o grafo *Cayley* possui $O(n^3 n!)$ arestas, uma outra heurísticas foi utilizada para determinar os valores do diâmetro para $11 \leq n \leq 15$.

Definindo um *k-move* como uma transposição $\rho$ tal que $\pi\rho$ tem $k$ pares $[x\ x+1]$ a mais que $\pi$. Um contra-exemplo mínimo para a conjectura de que $d(n) = \lceil (n+1)/2 \rceil$, ou seja, de que o diâmetro é igual a distância entre $\iota$ e $w_0$, não pode permitir um 2-*move* ou uma 1-*move* seguido por um 3-*move*.

Para $n \leq 13$ foi realizada uma busca em todos as permutações toroidais que satisfazem essa restrição e, para cada uma delas, verificou-se quais podem ser ordenadas em $\lceil (n+1)/2 \rceil$ transposições. Para $n = 11$ e $n = 12$ não foram encontrados contra-exemplos, mas para $n = 13$ foi encontrada a seguinte permutação: (4 3 2 1 5 13 12 11 10 9 8 7 6).

Além dessa, outras quatro permutações que também necessitam de 8 transposições foram encontradas. Observe que $w_0$ pode ser ordenado com 7 transposições.

Para $n = 14$ o teorema 11 mostra que $d(w_0) = 8$, o que corresponde ao limitante superior do diâmetro pelo teorema 12. Dessa forma, nenhum cálculo precisa ser feito.

Para $n = 15$, a permutação (4 3 2 1 5 15 14 13 12 11 10 9 8 7 6) exige a ocorrência de 9 transposições, enquanto $w_0$ necessita apenas de 8.

O trabalho afirma que, baseado no que sugerem os experimentos, os padrões que são especialmente difíceis de ordenar para $n = 13$ e $n = 15$ deixam de ser difíceis para $n$ maiores. Dessa forma, os autores formularam a conjectura 1.

**Conjectura 1** *Para todo $n \geq 3$, exceto para $n = 13$ e $n = 15$, $d(n) = \lceil (n+1)/2 \rceil$.*

# J New Bounds and Tractable Instances for the Transposition Distance [30]

O trabalho tem como objetivo apresentar novas classes de permutações tratáveis para o problema da distância de transposição. Além disso, são apresentados novos limitantes para o problema.

O ponto de partida é o estabelecimento de uma conexão entre uma variante do grafo da permutação (seção J.2) e do grafo de ciclos (seção A). Entretanto, antes de identificar essa conexão é necessário conhecer um pouco da nomenclatura que será usada. Os conceitos serão mostrados supondo conhecimento prévio sobre o exposto na seção A.

## J.1 Conceitos Iniciais

Sendo $G(\pi)$ um grafo orientado cíclico de arestas de cores alternadas criado a partir da permutação $\pi$, como visto na seção A, e $C$ um ciclo que pertence a sua decomposição única, diz-se que $C$ é não-orientado se contém exatamente uma aresta cinza orientada da esquerda para direita e orientado, caso contrário.

Sendo $c(G(\pi))$ o número de ciclos de $G(\pi)$, define-se um *k-move* como uma transposição $\rho$ tal que $c(G(\rho\pi)) = c(G(\pi)) + k$. Bafna e Pevzner [5] mostraram que uma transposição que age em exatamente dois ciclos é uma 0-*move*.

As três próximas definições são relacionadas ao modo como dois ciclos $C_1$ e $C_2$ interagem. Sendo $I_C$ o intervalo definido pelo menor e maior índices dos vértives que pertencem a $C$, temos:

- $C_1$ contém $C_2$ se $I_{C_1} \supset I_{C_2}$ e não há arestas pretas de $C_1$ pertencendo a $I_{C_2}$.
- $C_1$ e $C_2$ se cruzam se $C_1$ não contém $C_2$ e há no mínimo uma aresta preta de $C_1$ pertencente a $C_2$.
- $C_1$ e $C_2$ se intercalam se ao ler as arestas pretas de $C_1$ e $C_2$ da esquerda para a direita, uma aresta de cada ciclo é lida alternadamente.

Uma permutação $\pi$ é reduzida se $b(\pi) = n - 1$, $\pi_1 \neq 1$ e $\pi_n \neq n$, onde $b(\pi)$ representa o número de *breakpoints* de $\pi$ (seção A). Christie [11] mostra que qualquer permutação pode ser unicamente transformada em uma permutação reduzida sem afetar a distância. Encontrar a permutação reduzida $gl(\pi)$ consiste em decompor $\pi$ em $r$ trechos, que são intervalos maximais que não contém *breakpoints*. Depois disso, remove-se o primeiro trecho se ele começar com 1, o $r$-ésimo trecho se ele terminar com $n$ e substitui-se todos os outros trechos pelo seu menor elemento. Por fim, a seqüência resultante é renumerada de modo que o $k$-ésimo menor número seja substituído por $k$.

Um teorema conhecido do trabalho de Christie [11] será útil mais adiante.

**Teorema 13** *Duas permutações $\pi$ e $\sigma$ são equivalentes por redução se $gl(\pi) = gl(\sigma)$. Nesse caso, $d(\pi) = d(\sigma)$.*

Um conceito importante refere-se à classe de equivalência que engloba as chamadas permutações toroidais [15]. As permutações toroidais são definidas na seção I.

Define-se $\overline{x}^m$ como $\overline{x}^m = (x+m)(mod\ n+1)$. Essa operação será usada para representar a operação $m + \pi^\circ$ definida na seção I, sendo $\pi^\circ$ a permutação circular obtida a partir de $\pi$:

$$m + \pi^\circ = \overline{0}^m\ \overline{\pi_1}^m\ \overline{\pi_2}^m\ \ldots\ \overline{\pi_n}^m.$$

A permutação toroidal $\pi^\circ_\circ$ é uma classe formada pelo conjunto de permutações reconstruídas a partir de todas as permutações $m + \pi^\circ$, com $0 < m < n$. Duas permutações $\pi$ e

$\sigma$ pertencem a mesma classe toroidal se $\sigma \in \pi_\circ^\circ$ (ou $\pi \in \sigma_\circ^\circ$). Uma propriedade importante é que se $\pi$ e $\sigma$ pertencem a mesma classe toroidal, então $d(\pi) = d(\sigma)$. Além disso, todo ciclo em $G(\pi)$ corresponde a um ciclo em $G(\sigma)$.

## J.2 O grafo $\Gamma$

O grafo $\Gamma$ de uma permutação $\pi$ é o grafo orientado com o conjunto de vértices ordenados $(\pi_1, \ldots, \pi_n)$ e arestas $(\pi_i, \pi_j)|\pi_i = j$. O grafo $\Gamma$ pode ser decomposto em ciclos, também denominados $C$ em $\Gamma$, onde $C = (i_1, i_2, \ldots, i_k)$ é um $k$-ciclo. Um $k$-ciclo é classificado como ímpar se $k$ é ímpar e, caso contrário, o $k$-ciclo é classificado como par. A figura 3 mostra o grafo $\Gamma$ construído para a permutação $\pi = (4\ 1\ 6\ 2\ 5\ 7\ 3)$.

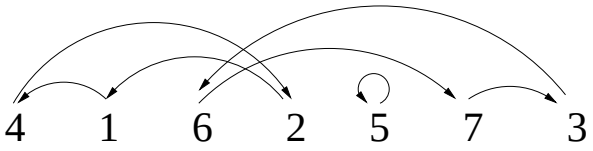


Figura 3: Grafo $\Gamma$ para a permutação $\pi = (4\ 1\ 6\ 2\ 5\ 7\ 3)$

Um $k$-ciclo em $\Gamma$ é crescente se $k \geq 3$ e seus elementos podem ser escritos como uma seqüência crescente. Por outro lado, se $k \geq 3$ e os elementos podem ser escritos como uma seqüência decrescente, então o $k$-ciclo é decrescente. Se o $k$-ciclo é crescente ou decrescente, então o mesmo é dito monotônico e, caso contrário, não monotônico. Por exemplo, na figura 3 o ciclo $(4\ 2\ 1)$ é decrescente, o ciclo $(3\ 6\ 7)$ é crescente e o ciclo $(5)$ é não monotônico.

É importante mencionar que as definições relacionadas ao modo como dois ciclos interagem, vistas anteriormente, podem ser facilmente adaptadas para um ciclo $C$ em $\Gamma$.

## J.3 Permutações $\gamma$

Uma permutação $\gamma$ é uma permutação reduzida cujos elementos pares estão na posição correta. Logo, a permutação $\gamma$ deve possuir um número ímpar de elementos. Assim, $\pi_i = i$ para todo elemento par $\pi_i$. Um exemplo de uma permutação $\gamma$ é $\pi = (3\ 2\ 1\ 4\ 7\ 6\ 9\ 8\ 5)$, cujos grafos $\Gamma$ e $G$ são mostrados na figura 4.

Seja $\pi$ uma permutação $\gamma$ arbitrária, há uma relação entre o número de ciclos pares e ímpares nos grafos $G(\pi)$ e $\Gamma(\pi)$:

$$\begin{cases} c_{even}(G(\pi)) &= 2(c_{even}(\Gamma(\pi))); \\ c_{odd}(G(\pi)) &= 2(c_{odd}(\Gamma(\pi)) - \frac{n-1}{2}). \end{cases}$$

Vale ressaltar que a classificação de $C$ em $G$ em pares e ímpares é de acordo com o número de arestas pretas, enquanto que a classificação de $C$ em $\Gamma$ em pares e ímpares está relacionada ao tamanho do ciclo $k$.

Uma relação útil é que, se dois ciclos $C_1$ e $C_2$ em $G(\pi)$ correspondem a um único $k$-ciclo $C$ em $\Gamma(\pi)$, então $C_1$ e $C_2$ se intercalam. Além disso, ainda é possível encontrar informações mais específicas.

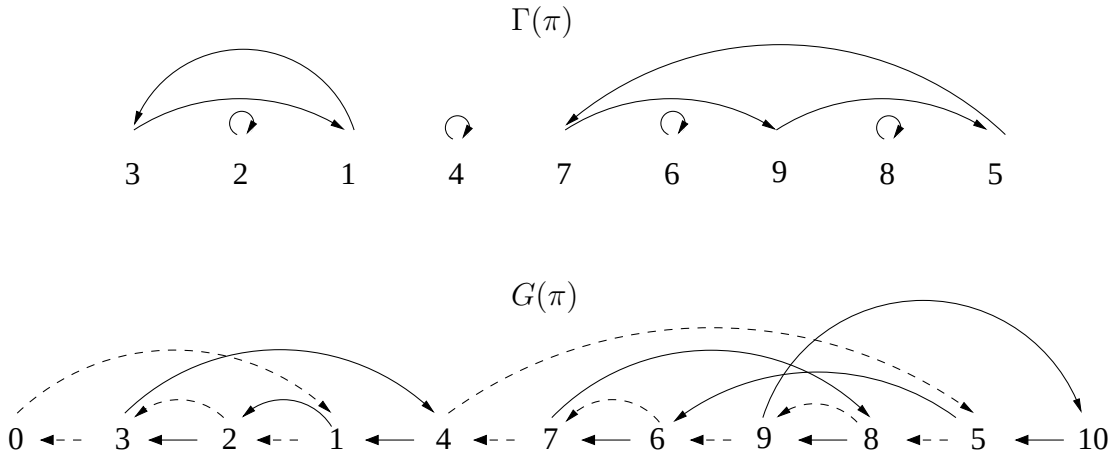


Figura 4: Grafos $\Gamma(\pi)$ e $G(\pi)$ para a permutação $\pi = (3\ 2\ 1\ 4\ 7\ \ 6\ 9\ 8\ )$

1. Se $k = 2$, então $C_1$ e $C_2$ são não orientados
2. Se $C$ é monotônico, então $C_1$ ou $C_2$ são orientados.
3. Se $C$ é não monotônico e $k \geq 4$, então $C_1$ e $C_2$ são orientados.

Com o conjunto de relações acima é possível derivar um limitante inferior para permutações $\gamma$:

$$d(\pi) \geq n - c_{odd}(\Gamma(\pi)).$$

### J.3.1 Permutações $\alpha$ e $\beta$

Uma permutação $\alpha$ é uma permutação reduzida cujos elementos pares estão na posição correta e cujos $\frac{n+1}{2}$ elementos ímpares formam um ciclo monotônico em $\Gamma$, chamado de ciclo principal. Um exemplo de uma permutação $\alpha$ para $n = 7$ é (3 2 5 4 7 6 1).

É importante notar que, fixado $n$, só são possíveis duas permutações $\alpha$. Dessa forma, para $n = 7$, uma outra permutação que atende as restrições é (7 2 1 4 3 6 5).

Uma permutação $\beta$ é uma permutação reduzida cujos elementos pares estão na posição correta e cujos elementos ímpares formam um ciclo não monotônico em $\Gamma$, que, de modo análogo às permutações $\alpha$, será chamado de ciclo principal.

A importância das permutações $\alpha$ e $\beta$ é que, dado que $\pi$ pertence a uma dessas classes de permutações, temos:

$$d(\pi) = n - c_{odd}(\Gamma(\pi)) = |C| - (|C| \ mod\ 2)$$

onde $|C| = \frac{n+1}{2}$ é o número de elementos no ciclo principal.

As classe de permutações $\alpha$ e $\beta$ são restritas, mas o autor mostra que é possível generalizar o limitante acima para todas as permutações $\gamma$, logo, pelo teorema 13, temos que todas as permutações que podem ser reduzidas para uma permutação $\gamma$ apresentam o limitante. Além disso, o trabalho prova o seguinte teorema.

**Teorema 14** *Toda permutação $\sigma$ com um número $n$ ímpar de elementos e cujos elementos ímpares ocupam posições ímpares e formam uma subsequência crescente módulo $n+1$ pode ser transformada em tempo linear em uma permutação $\pi$ redutível à classe $\gamma$, tal que $d(\sigma) = d(\pi) = n - c_{odd}(\Gamma(\pi))$.*

Por fim, o trabalho mostra que essa equação de distância para permutações $\gamma$ nada mais é do que um limite superior para a distância de transposição.

**Teorema 15** *Para todo $\pi$ em $S_n$.*

$$d(\pi) \leq n - c_{odd}(\Gamma(\pi)).$$

Dois aperfeiçoamentos do teorema 15 podem ser gerados: O teorema 16 usa a equivalência toroidal e o teorema 17 usa equivalência por redução.

**Teorema 16** *Para todo $\pi$ em $S_n$.*

$$d(\pi) \leq n - \max_{\sigma \in \pi^{\circ}_{\circ}} c_{odd}(\Gamma(\sigma)).$$

**Teorema 17** *Para todo $\pi \neq \iota$ em $S_n$.*

$$d(\pi) \leq m - \max_{\sigma \in (gl(\pi))^{\circ}_{\circ}} c_{odd}(\Gamma(\sigma)).$$

*Onde $m$ é o número de elementos de $gl(\pi)$.*

O autor afirma que o melhoramento trazido pelo teorema 16 em relação ao teorema 15 é substancial segundo os experimentos, mas é difícil expressar ou avaliar numericamente esse melhoramento devido a dificuldade em prever a evolução de $\Gamma$ sob a equivalência toroidal.

## J.4 Extensões para permutações $\alpha$

As permutações $\gamma$ possuem a restrição de que elementos pares devem estar no lugar correto. Uma dúvida que surge é sobre o que acontece ao sistema quando essas posições fixas são removidas. Uma análise cuidadosa permite melhorar o limitante do teorema 15 no caso das permutações $\alpha$.

Para início de análise, define-se uma permutação *k-perforation* $\pi$ em $S_n$ de uma permutação $\sigma$ em $S_{n+k}$ da classe $\alpha$, removendo $k \geq 1$ elementos pares de $\Gamma(\sigma)$ e renumerando os elementos restantes.

Por exemplo, uma 3-*perforation* da permutação $\sigma$ = (3 2 5 4 7 6 9 8 11 10 1) é (3 $\underline{2}$ 5 4 7 $\underline{6}$ 9 $\underline{8}$ 11 10 1) = (2 4 3 5 6 8 7 1). Os próximos passos do trabalho consistem em analisar a evolução da estrutura de $G$ quando é aplicada uma *k-perforation* em permutações $\alpha$.

Para toda *k-perforation* $\pi$ de uma permutação $\sigma$ da classe $\alpha$:

$$c(G(\pi)) = c_{odd}(G(\pi)) = k$$

e $G(\pi)$ contém apenas ciclos que não cruzam, exceto por um ciclo maior que contém todos os outros. Isso leva a uma fórmula para computar a distância dessas permutações:

$$d(\pi) = n - c_{odd}(\Gamma(\pi)) - k + (|C| \; mod \; 2),$$

onde $|C| = \frac{n+k+1}{2}$ é o número de elementos no ciclo principal.

O trabalho apresenta outros resultados menos genéricos e apresenta questões em aberto como, por exemplo, analisar *k-perforations* em permutações $\beta$ e classificar as permutações cujo grafo $\Gamma$ contém apenas ciclos que se cruzam e não podem ser reduzidas para permutações $\gamma$.

# K Faster Algorithms for Sorting by Transpositions and Sorting by Block Interchanges [16]

O trabalho apresenta uma nova estrutura de dados, chamada de árvore de permutação, com o intuito de melhorar o tempo de execução do algoritmo de ordenação por transposições de Hartman e Shamir [22] de $O(n^{\frac{3}{2}}\sqrt{\log n})$ para $O(n \log n)$ e do algoritmo de ordenação por troca de blocos de Christie [10] de $O(n^2)$ para $O(n \log n)$. No presente resumo não serão mostrados os algoritmos apresentados no trabalho, mas apenas a estrutura de dados criada.

## K.1 Árvore de Permutação

Uma árvore de permutação é uma árvore binária balanceada $T$ com raiz $r$ e onde cada nó da árvore possui um rótulo. Seja $\pi = (\pi_1 \; \pi_2 \; \ldots \; \pi_n)$ a permutação para qual a árvore $T$ foi gerada, então $T$ possui $n$ nós folhas que são rotulados como $\pi_1, \pi_2, \ldots, \pi_n$. Os nós internos da árvore são rotulados com o maior dos rótulos dos filhos. A figura 5 corresponde à árvore de permutação gerada para $\pi = (9\ 6\ 1\ 4\ 7\ 5\ 2\ 3\ 8)$.

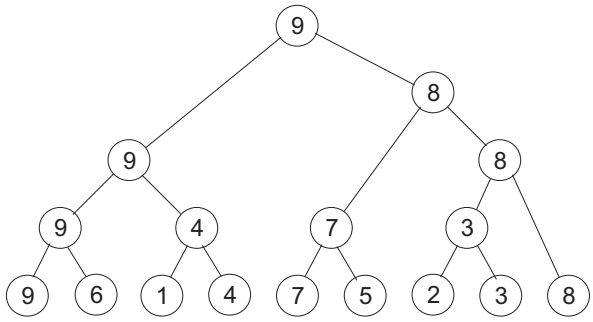


Figura 5: Árvore de permutação para $\pi = (9\ 6\ 1\ 4\ 7\ 5\ 2\ 3\ 8)$

Cada nó interno da árvore representa um intervalo na permutação $\pi$, e o nó raiz $r$ representa a permutação inteira. É interessante notar que não é necessário inserir ou remover nós da árvore de permutação, após a mesma ser criada, pois os efeitos decorrentes de uma transposição podem ser projetados na árvore com a utilização de apenas três operações: *build*, *join* e *split*.

A operação *build* constrói uma árvore a partir de uma permutação e possui uma implementação simples: a árvore é criada das folhas até a raiz, camada por camada. Para a criação dos nós internos usa-se a camada mais abaixo que foi criada no passo anterior.

O algoritmo *build* é mostrado abaixo. Para facilitar a legibilidade, o trabalho omite os cálculos de atribuição do rótulo e da altura dos nós. A complexidade do algoritmo *build* é $O(n)$.

```
BUILD(π)
 1  Criar nós folhas u_i para π_i, 1 ≤ i ≤ n
 2  U ← [u_1, u_2, ..., u_n]
 3  k ← n
 4  While k > 1
 5     do Criar v_i : L(v_i) = u_{2i-1}, R(v_i) = u_{2i}, para 1 ≤ i ≤ ⌊k/2⌋
 6        if k é ímpar
 7           then U ← [v_1, v_2, ..., v_{k/2}]
 8           else  Criar v : L(v) = v_{⌊k/2⌋}, R(v) = u_k
 9                 U ← [v_1, v_2, ..., v_{⌊k/2⌋}, v]
10        k ← ⌊k/2⌋
11  return U
```

A operação *join* recebe duas árvores $T_1$ e $T_2$, representando as permutações $\pi_1$ e $\pi_2$ respectivamente, e gera a árvore $T$ que representa a permutação $\pi$ obtida pela concatenação dos elementos de $\pi_2$ em $\pi_1$.

Vale ressaltar que $T$ deve ser uma árvore balanceada, ou seja, se a altura $H(T_1)$ de $T_1$ for maior que a altura $H(T_2)$ de $T_2$ acrescida de 1, então a árvore resultante não poderá ser construída apenas inserindo um nó raíz $r$ e adicionando $T_1$ e $T_2$ como filhos. O algoritmo abaixo mostra os passos para criar a árvore $T$ quando $H(T_1) \geq H(T_2)$ e omite os cálculos para atribuição do rótulo e altura dos nós. A complexidade do algoritmo *join* é $O(H(T_1) - H(T_2))$.

```
JOIN(T_1, T_2)
 1  if H(T_1) − H(T_2) ≤ 1
 2     then Criar um nó T : L(T) = T_1, R(t) = T_2
 3          return T
 4  if H(T_1) − H(T_2) = 2
 5     then if H(L(T_1)) ≥ H(R(T_1))
 6             then Criar dois nós T' e T
 7                  T' : L(T') = R(T_1), R(T') = T_2
 8                  T : L(T) = L(T_1), R(T) = T'
 9                  return T
10             else  Criar três nós T'', T' e T
11                  T'' : L(T'') = R(R(T_1)), R(T'') = T_2
12                  T' : L(T') = L(T_1), R(T') = L(R(T_1))
13                  T : L(T) = T', R(T) = T''
14                  return T
```

```
15  if H(T1) − H(T2) > 2
16    then T = Join(L(T1), Join(R(T1), T2))
17          return T
```

A operação *split* recebe uma árvore $T$, que representa uma permutação $\pi$ de tamanho $n$, uma posição $m$, onde $1 \leq m \leq n$, e retorna duas árvores $T_r$ e $T_l$. A árvore $T_r$ representa a permutação $\pi_r = (\pi_1 \ \ldots \ \pi_{m-1})$ e a árvore $T_l$ representa a permutação $\pi_l = (\pi_m \ \ldots \ \pi_n)$.

A idéia principal para implementar a operação *join* é encontrar em $T$ o caminho $P = v_0, v_1, \ldots, v_k$ que vai de $\pi_m$ até a raiz $r$, dessa forma $v_0 = \pi_m$ e $v_k = \pi_r$. Calculado o caminho $P$, percorre-se os elementos $v_i$ verificando se o elemento anteror $v_{i-1}$ está no lado esquerdo ou direito de $v_i$. Se o elemento $v_{i-1}$ está do lado esquerdo, então o elemento do lado direito de $v_i$ pertencerá à árvore $T_r$. De modo análogo, se o elemento $v_{i-1}$ está no lado direito de $v_i$, então o elemento do lado esquerdo de $v_i$ pertencerá à árvore $T_l$. O algoritmo abaixo mostra os passos para dividir a árvore. O algoritmo *split* executa com complexidade $O(\log n)$.

```
SPLIT(T, m)
 1  Encontrar o caminho P = v0, v1, ..., vk de πm a r
 2  Tr ← v0
 3  Tl ← null
 4  for i ← 1 to k
 5     do if L(vi) = vi−1
 6           then Tr ← Join(Tr, R(vi))
 7        if R(vi) = vi−1
 8           then Tl ← Join(L(vi), Tl)
 9
10  return Tr, Tl
```

Além das operações acima, para utilizar a árvore de permutação será preciso calcular a posição de um nó de $T$ na permutação $\pi$. Para isso, será preciso utilizar a noção de que cada nó interno de $T$ corresponde a um intervalo na permutação $\pi$. Dessa forma, cada nó $t$ deverá guardar o número de elementos $e(t)$ representados por ele.

Com a informação $e(t)$ armazenada em cada nó, a posição em $\pi$ do nó rotulado com $\pi_i$ pode ser obtida percorrendo o caminho de $\pi_i$ até a raiz da árvore e contando o número de elementos a esquerda de $\pi_i$. Esse cálculo pode ser efetuado em $O(\log n)$ para qualquer elemento da árvore $T$.

# L A 1.375-Approximation Algorithm for Sorting by Transpositions [14]

O trabalho apresenta um algoritmo para o problema de ordenação de permutações por transposições que fornece uma resposta aproximada na razão de 1.375. O trabalho apresenta outros resultados interessantes como, por exemplo, a apresentação de um novo limitante inferior para o problema do diâmetro de transposições, a determinação exata do

diâmetro para permutações simples e 2-permutações.

Muitos dos conceitos preliminares apresentados no trabalho são decorrentes dos trabalhos anteriores e recomenda-se a leitura da seção A, principalmente no que diz respeito a introdução do grafo de ciclos, que no trabalho de Elias e Hartman é chamado de grafo de *breakpoints*.

O trabalho utiliza uma classificação das permutações de acordo com o grafo de ciclos gerado por elas: uma permutação simples contém apenas ciclos simples (seção A) e uma 2-permutação (respectivamente 3-permutação) contém apenas 2-ciclos (3-ciclos).

Uma técnica comum em rearranjo de genomas consiste em transformar permutações com ciclos longos em permutações simples inserindo novos elementos na permutação [23]. Seja $\hat{\pi}$ a permutação obtida inserindo elementos em $\pi$, então $d(\pi) \leq d(\hat{\pi})$. A transformação é dita segura (do inglês *safe*) se ela mantém o limitante inferior do teorema 2. Vale ressaltar que uma transforma segura apenas mantém o limitante inferior e não a distância real.

O algoritmo apresentado no trabalho primeiramente transforma uma permutação $\pi$ em uma permutação simples equivalente $\hat{\pi}$ e, em seguida, ordena $\hat{\pi}$. Dessa forma, a maior parte do trabalho é dedicada a encontrar resultados para permutações simples e ciclos curtos.

## L.1 Diâmetro de transposição

Como o trabalho apresenta resultados para permutações circulares, vale ressaltar que os limitantes apresentados devem ser modificados caso deseje-se trabalhar com permutações lineares. Para isso, basta lembrar que ordenar uma permutação linear de tamanho $n$ é equivalente a ordenar uma permutação circular de tamanho $n+1$.

**Teorema 18** *O limitante superior do diâmetro de transposição $TD(n)$ para o grupo simétrico de tamanho $n$ é dado por:*

$$TD(n) \geq \lfloor \frac{n}{2} + 1 \rfloor$$

**Teorema 19** *O diâmetro de transposição $TD2(n)$ para 2-permutações de tamanho $n$ é dado por:*

$$TD2(n) = \frac{n}{2}$$

**Teorema 20** *O diâmetro de transposição $TDS(n)$ para permutações simples de tamanho $n$ é dado por:*

$$TDS(n) = \lfloor \frac{n}{2} \rfloor$$

O resultado mais importante fornecido pelo trabalho no que ser referere a distância de transposição está relacionado com o limitante superior encontrado para o diâmetro de 3-permutações $TD3(n)$. Esse limitante superior é a base para o algoritmo de apresentação e sua prova exige uma rigorosa análise de casos. Entretanto, como são muitos casos, foi desenvolvido um programa de computador que sistematicamente analisa todos os casos.

O objetivo da análise de casos é mostrar que para toda 3-permutação com no mínimo 8 ciclos existe uma ($x$,$y$)-seqüência tal que $x \leq 11$ e $\frac{x}{y} \leq \frac{11}{8}$. A notação "($x$,$y$)-seqüência" de transposições é uma seqüência de $x$ transposições, tais que no mínimo $y$ delas são 2-moves. Por exemplo, um 0-move seguido de dois 2-moves consecutivos é denominado uma (3,2)-seqüência.

O algoritmo de ordenação usa a idéia de ordenar a permutação aplicando repetidamente (11,8)-seqüências e como $\frac{11}{8} = 1.375$ obtém-se a razão de aproximação.

SORT($\pi$)
1 Transformar a permutação $\pi$ em uma permutação simples $\hat{\pi}$
2 Verificar se há uma (2,2)-seqüência e aplicá-la
3 Enquanto $G(\hat{\pi})$ possuir 2-ciclos, aplicar 2-move
4 Enquanto $G(\hat{\pi})$ contém no mínimo 8 ciclos, aplicar uma (11,8)-seqüência
5 Enquanto $G(\hat{\pi})$ contém um 3-ciclo, aplicar uma (3,2)-seqüência
6 Gerar a permutação $\pi$ usando a ordenação de $\hat{\pi}$

O algoritmo *Sort* possui complexidade tempo de $O(n^2)$.