Universidade Estadual de Campinas
Instituto de Computação

Alexsandro Oliveira Alexandrino

Variações do Problema de Distância de Rearranjos

CAMPINAS
2024

Alexsandro Oliveira Alexandrino

Variações do Problema de Distância de Rearranjos

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Zanoni Dias
Coorientador: Prof. Dr. Ulisses Martins Dias

Este exemplar corresponde à versão final da Tese defendida por Alexsandro Oliveira Alexandrino e orientada pelo Prof. Dr. Zanoni Dias.

CAMPINAS
2024

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

AL27v

Alexandrino, Alexsandro Oliveira, 1995-
Variações do problema de distância de rearranjos / Alexsandro Oliveira Alexandrino. – Campinas, SP : [s.n.], 2024.

Orientador: Zanoni Dias.
Coorientador: Ulisses Martins Dias.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Rearranjo de genomas. 2. Biologia computacional. 3. Algoritmos de aproximação. 4. Ordenação (Computadores). I. Dias, Zanoni, 1975-. II. Dias, Ulisses Martins, 1983-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: On variants of the genome rearrangement distance problem
Palavras-chave em inglês:
Genome rearrangements
Computational biology
Approximation algorithms
Sorting (Electronic computers)
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Zanoni Dias [Orientador]
Marie-France Sagot
Maria Emília Machado Telles Walter
Orlando Lee
Guilherme Pimentel Telles
Data de defesa: 27-03-2024
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-6320-9747
- Currículo Lattes do autor: http://lattes.cnpq.br/4111593464868789

Universidade Estadual de Campinas
Instituto de Computação

Alexsandro Oliveira Alexandrino

Variações do Problema de Distância de Rearranjos

Banca Examinadora:

- Prof. Dr. Zanoni Dias
  Universidade Estadual de Campinas
- Profa. Dra. Marie-France Sagot
  Institut National de Recherche en Sciences et Technologies du Numérique
- Profa. Dra. Maria Emília Machado Telles Walter
  Universidade de Brasília
- Prof. Dr. Orlando Lee
  Universidade Estadual de Campinas
- Prof. Dr. Guilherme Pimentel Telles
  Universidade Estadual de Campinas

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 27 de março de 2024

Quando a educação não é libertadora,
o sonho do oprimido é ser o opressor.
(Paulo Freire)

Agradecimentos

Esta tese representa o maior marco de uma jornada que começou em 2016, quando comecei o meu trabalho de conclusão de curso da graduação explorando problemas de Rearranjo de Genomas. No início dessa jornada, obtive a ajuda dos professores Criston e Lucas Ismaily, que foram os meus orientadores de TCC. Também gostaria de agradecer às professoras Paulyne e Carla, que me incentivaram a entrar na carreira acadêmica.

Ao ingressar na Unicamp, tive a sorte de ser orientado pelo Zanoni Dias e, além disso, tive a oportunidade de trabalhar como monitor ao seu lado em disciplinas e cursos da Unicamp. Agradeço ao Zanoni por tudo que me ensinou e por sua confiança, e também pela amizade construída ao longo dos anos. Também tive o prazer de ser coorientado pelos professores Carla Lintzmayer e Ulisses Dias no mestrado e doutorado, respectivamente. Gostaria de agradecer a ambos pela colaboração acadêmica e pelo que me ensinaram.

Gostaria de agradecer ao André, Klairton e Gabriel, pela colaboração na pesquisa e pelas conversas semanais. Além deles, gostaria de agradecer a todos os membros do LOCo e todos os funcionários do IC/Unicamp.

Alguns amigos fizeram parte dessa jornada desde a graduação na UFC até o doutorado na Unicamp. Gostaria de agradecer a todos e, em especial, à Ana Paula, Daiane, Décio, Italos e Leodécio.

Durante 2022, tive a oportunidade de viajar para a França e trabalhar com os professores Guillaume Fertin e Géraldine Jean, que me ajudaram muito na minha experiência de intercâmbio. Além disso, fiz grandes amigos em Nantes e gostaria de agradecer a todos eles.

Por fim, e mais importante, gostaria de agradecer a todos da minha família, que sempre estiveram ao meu lado desde 2013, quando saí da minha cidade para ingressar na graduação. Agradeço aos meus pais, Pedro e Marileide, e também à Mara, Alex, Katyeudo e Katyenne.

Esta tese foi realizada com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e do Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - Processo 202292/2020-7.

Resumo

Considerando um par de genomas de organismos de espécies relacionadas, os problemas de distância de rearranjos têm como objetivo estimar quão distante um deles está em relação ao outro em termos de rearranjos de genomas, que são eventos mutacionais capazes de modificar o material genético ou a posição relativa de segmentos de um genoma. Considerando o Princípio da Máxima Parcimônia, o termo *distância*, ou ainda *distância de rearranjos*, é definido como o número mínimo de rearranjos de genomas necessários para transformar um genoma no outro.

Os primeiros trabalhos que estudaram a distância de rearranjos assumiram que os genomas comparados possuem o mesmo conjunto de genes (genomas balanceados) e, além disso, apenas a ordem relativa dos genes e suas orientações, quando conhecidas, são utilizadas na representação matemática dos genomas. Essas restrições implicam que é possível transformar um genoma em outro usando apenas rearranjos que não alteram a quantidade de material genético no genoma (rearranjos conservativos). Nesse caso, os genomas são representados como permutações, o que deu origem aos problemas de Ordenação de Permutações por Rearranjos. Os principais problemas de Ordenação de Permutações por Rearranjos consideram DCJs, reversões, transposições, ou a combinação de reversões e transposições, sendo que eles possuem complexidade conhecida. Além desses, foram estudados outros problemas que combinam transposições com um ou mais dos seguintes rearranjos: transposições inversas, revrevs e reversões. Apesar de existirem algoritmos de aproximação na literatura para esses problemas, as complexidades deles permaneciam em aberto. Um dos resultados desta tese é a prova de complexidade desses problemas que combinam transposições com transposições inversas, revrevs e reversões. Além disso, apresentamos um novo algoritmo de 1.375-aproximação para a Ordenação de Permutações por Transposições que possui melhor complexidade de tempo.

Com o avanço da área, novos trabalhos começaram a considerar genomas desbalanceados e a incorporar a distribuição dos tamanhos das regiões intergênicas. Ao considerar genomas desbalanceados, é necessário o uso de inserções e deleções para transformar um genoma em outro. Nesta tese, estudamos tanto o problema de Distância de Rearranjos em genomas desbalanceados considerando apenas a sequência de genes e suas orientações (quando conhecidas), quanto o problema de Distância de Rearranjos Intergênicos em genomas desbalanceados, que incorpora os tamanhos das regiões intergênicas na representação dos genomas, além do uso da sequência de genes e suas orientações (quando conhecidas). Apresentamos novas estruturas e conceitos para problemas que envolvem reversões, transposições e a combinação de reversões e transposições, que são usados em provas de complexidade e algoritmos de aproximação. Além disso, realizamos experimentos em genomas sintéticos e em genomas reais, evidenciando a aplicabilidade dos nossos algoritmos.

Abstract

Considering a pair of genomes from individuals of related species, the goal of genome rearrangement distance problems is to estimate how distant these genomes are from each other based on genome rearrangements, which are mutational events that modify the genetic material or the relative position from segments of a genome. Using the Principe of Parsimony, the term *distance*, or *rearrangement distance*, refers to the minimum number of rearrangements necessary to transform one genome into the other.

Seminal works in genome rearrangements assumed that both genomes being compared have the same set of genes (balanced genomes) and, furthermore, only the relative order of genes and their orientations, when they are known, are used in the mathematical representation of the genomes. These restrictions imply that it is possible to transform one genome into the other using only conservative rearrangements, which are rearrangements that do not alter the genetic material from a genome. In this case, the genomes are represented as permutations, originating the Sorting Permutations by Rearrangements problems. The main problems of Sorting Permutations by Rearrangements considered DCJs, reversals, transpositions, or the combination of both reversals and transpositions, and these problems have their complexity known. Besides these problems, other ones were studied involving the combination of transpositions with one or more of the following rearrangements: transreversals, revrevs, and reversals. Although there are approximation results for these problems, their complexity remained open. Some of the results of this thesis are the complexity proofs for these problems. Furthermore, we present a new 1.375-approximation algorithm, which has better time complexity, for the Sorting Permutations by Transpositions.

As the field has progressed, new works started to consider unbalanced genomes and to incorporate the size distribution of intergenic regions. When considering unbalanced genomes, it is necessary to use insertions and deletions to transform one genome into another. In this thesis, we studied Rearrangement Distance problems on unbalanced genomes considering only gene order and their orientations (when they are known), as well as Intergenic Rearrangement Distance problems, which incorporate the information regarding the size distribution of intergenic regions, besides the use of gene order and their orientations (when they are known). We present new structures and concepts for problems that include reversals, transpositions, and the combination of reversals and transpositions. These structures and concepts are used in complexity proofs and approximation algorithms. Furthermore, we performed experiments in simulated and real genomes, showing the applicability of our algorithms.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Rearranjos de genomas são mutações que alteram grandes trechos de um genoma. Na genômica comparativa, uma das formas mais aceitas para estimar a distância evolutiva é com o uso da *distância de rearranjos de genoma*. Nessa distância, cada rearranjo está associado a um custo, que pode ser unitário, quando simplesmente indica a ocorrência de um rearranjo, ou pode ter um valor que indica uma característica do rearranjo, como a quantidade de quebras na sequência de DNA. Para os genomas de origem $\mathcal{G}_o$ e destino $\mathcal{G}_d$, a *distância de rearranjos* entre $\mathcal{G}_o$ e $\mathcal{G}_d$ é definida como o menor custo possível de uma sequência de rearranjos que transforma o genoma origem $\mathcal{G}_o$ no genoma destino $\mathcal{G}_d$.

Nos problemas de rearranjo de genomas, um genoma é usualmente representado a partir da sua sequência de genes e, dependendo da informação disponível sobre cada gene e das características dos genomas estudados, diferentes representações matemáticas podem ser utilizadas. Supondo que não existem genes repetidos e que os genomas possuem o mesmo conjunto de genes, um genoma pode ser representado como uma permutação de números inteiros, onde cada elemento da permutação representa um gene. Quando a orientação dos genes é conhecida, essa informação é representada por um sinal "+" ou "−" e, nesse caso, as permutações são ditas com sinais. Quando a orientação dos genes é desconhecida, permutações sem sinais são utilizadas para a representação dos genomas. Ao utilizar permutações, o problema de Distância de Rearranjos é equivalente ao problema de Ordenação de Permutações por Rearranjos [46].

Um *modelo de rearranjo* é o conjunto de rearranjos (ou operações) permitidos em um dado problema de Distância de Rearranjos. Dentre os rearranjos mais estudados na literatura, temos a *reversão*, que inverte um segmento do genoma, e a *transposição*, que troca as posições relativas de dois segmentos adjacentes do genoma. Assim sendo, um modelo de rearranjo pode ser definido de forma a considerar apenas as reversões, outro modelo pode considerar apenas as transposições, enquanto um terceiro modelo pode ainda considerar tanto reversões quanto transposições.

O estudo dos problemas de Distância de Rearranjo teve início com modelos de rearranjo formados por um único tipo de operação, o que gerou o problema da Ordenação de Permutações por Reversões [36, 54] e o problema da Ordenação de Permutações por Transposições [15]. Com o avanço da área, esses rearranjos foram reunidos em um único modelo, mas sem considerar custos distintos entre as operações [80]. O problema da Ordenação de Permutações com Sinais por Reversões possui algoritmo exato polinomial [51]. Já os

problemas de Ordenação de Permutações sem Sinais por Reversões ou por Transposições pertencem à classe de problemas NP-difíceis [34, 36]. Para permutações com ou sem sinais, o problema que considera o modelo de rearranjo contendo tanto reversões quanto transposições também é NP-difícil [64].

Dentre os algoritmos mais conhecidos desenvolvidos para esses problemas, podemos citar o algoritmo exato polinomial para o problema da Ordenação de Permutações com Sinais por Reversões, desenvolvido por Hannenhalli e Pevzner [51], e o algoritmo de 1.375-aproximação para a Ordenação de Permutações por Transposições, desenvolvido por Elias e Hartman [43]. Recentemente, Silva e coautores [74] mostraram que o fator de aproximação do algoritmo de Elias e Hartman [43] ultrapassa o valor de 1.375 em alguns casos. Silva e coautores [74] também mostraram como solucionar o problema e apresentaram um algoritmo com complexidade de tempo de $O(n^6)$. Nesta tese, apresentamos uma nova versão do algoritmo proposto por Elias e Hartman [43] que garante o fator de aproximação de 1.375 em todos os casos e possui complexidade de tempo de $O(n^5)$.

Outros rearranjos estudados na literatura são as transposições inversas e as revrevs. Assim como as transposições, essas operações agem em dois segmentos adjacentes de um genoma. Uma *transposição inversa* troca as posições relativas dos dois segmentos adjacentes, assim como as transposições, mas essa operação também inverte um dos segmentos. Já uma *revrev* inverte cada um dos dois segmentos adjacentes, mas não troca as posições relativas desses segmentos. A complexidade dos problemas de Ordenação de Permutações por Rearranjos com modelos que possuem transposições inversas e revrevs eram desconhecidas, apesar de existirem algoritmos de aproximação para esses problemas [46]. Nesta tese, apresentamos provas de dificuldade para o problema de Ordenação de Permutações por Rearranjos considerando modelos de rearranjo contendo transposições junto com combinações de reversões, transposições inversas e revrevs.

Quando os genomas de origem e destino possuem conjuntos de genes distintos, dizemos que esses genomas são desbalanceados e usamos strings para representá-los matematicamente. Assim como nas permutações, quando a orientação dos genes é conhecida, essa informação é representada por um sinal "+" ou "−". Já quando a orientação dos genes é desconhecida, apenas usamos strings sem sinais.

As reversões, transposições, transposições inversas, e revrevs são operações conservativas, ou seja, são rearranjos que não alteram a quantidade de material genético do genoma. As operações não conservativas mais estudadas são as inserções e deleções de material genético [83, 42]. Inserções e deleções são coletivamente chamadas de *indels* em trabalhos da área. Nos modelos que possuem *indels*, o conjunto de genes do genoma origem e o conjunto de genes do genoma destino podem ser distintos e, por isso, a representação dos genomas é feita usando strings. Nesta tese, apresentamos provas de dificuldade e algoritmos de aproximação para os problemas de Distância de Rearranjos em genomas desbalanceados, considerando modelos que combinam as operações de reversão, transposição e *indel*.

Além disso, estudamos uma outra operação chamada *block interchange*, que troca a posição relativa de dois segmentos quaisquer do genoma. Note que uma transposição é um tipo específico de *block interchange* em que os segmentos afetados são adjacentes. Nesta tese, também apresentamos algoritmos de aproximação para os problemas de Distância de Rearranjos em genomas desbalanceados, considerando *block interchanges* e a combinação

de *block interchanges* e reversões.

Além da ordem relativa em que os genes aparecem no genoma, estudos recentes incorporaram a distribuição do tamanho das *regiões intergênicas* (quantidade de nucleotídeos entre cada par de genes consecutivos) na representação matemática dos genomas. A incorporação dessa informação na representação dos genomas é motivada por evidências de que as regiões intergênicas possibilitam inferir melhores cenários evolucionários [20, 21]. Os problemas de Distância de Rearranjos Intergênicos existentes na literatura tratam apenas genomas balanceados. Os modelos de rearranjo considerados possuem operações conservativas e *indels*, no entanto, esses *indels* podem inserir ou remover apenas nucleotídeos de regiões intergênicas, restringindo os genomas comparados a terem o mesmo conjunto de genes. Nesta tese estudamos os problemas de Distância de Rearranjos Intergênicos em genomas desbalanceados, considerando modelos que combinam as operações de reversão, transposição, e *indels*, sendo que os *indels* também podem inserir ou remover genes. Para a maioria dos modelos estudados, apresentamos provas de dificuldade e, além disso, para todos os modelos, apresentamos algoritmos de aproximação.

Esta tese está organizada da seguinte forma. O Capítulo 2 apresenta notações e conceitos gerais para os problemas estudados. O Capítulo 3 apresenta o novo algoritmo de 1.375-aproximação para a Ordenação de Permutações por Transposições e provas de dificuldade para problemas de Ordenação de Permutações por Transposições e Outros Rearranjos. O Capítulo 4 considera os problemas de Distância de Rearranjos em genomas desbalanceados e o Capítulo 5 considera os problemas de Distância de Rearranjos Intergênicos em genomas desbalanceados. Nos capítulos 4 e 5 apresentamos provas de dificuldade e algoritmos de aproximação para os modelos estudados. No Capítulo 6, apresentamos resultados experimentais em genomas sintéticos usando os algoritmos dos capítulos 4 e 5. Além disso, evidenciamos a aplicabilidade de um dos nossos algoritmos em uma base de dados de genomas reais, usando as soluções desse algoritmo na construção de uma árvore filogenética. Por fim, no Capítulo 7, apresentamos as considerações finais, sumário dos resultados, e direções de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, apresentamos definições e notações fundamentais para esta tese. As definições apresentadas neste capítulo são gerais para os problemas de distância de rearranjos e são usadas nos capítulos seguintes. No entanto, conceitos específicos para alguma variação de problemas de rearranjos são definidos apenas no capítulo referente a essa variação.

2.1 Representação de Genomas

Nesta seção, apresentamos como os genomas em um problema de Distância de Rearranjos são modelados matematicamente.

2.1.1 Representação da Ordem Relativa dos Genes

Os problemas clássicos da área de rearranjo de genomas utilizam apenas a ordem relativa dos genes para a representação matemática dos genomas. Exceto quando dito expressamente o contrário, nesta tese assumimos que os genomas não possuem genes repetidos, assim como a maior parte da literatura na área.

Quando os genomas a serem comparados possuem o mesmo conjunto de genes, dizemos que os genomas são *balanceados* e podemos utilizar permutações de números inteiros para representar a ordem relativa dos genes em cada genoma. Cada elemento da permutação representa um gene. Se a orientação dos genes é considerada, então utilizamos uma *permutação com sinais*, em que cada elemento possui um sinal "+" ou "−" que indica a orientação do gene. Caso contrário, utilizamos uma *permutação sem sinais* ou, de forma equivalente, consideramos que todos os elementos possuem sinal "+".

Definição 2.1.1. Denotamos uma *permutação com sinais* de tamanho n por $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_{n-1}\ \pi_n)$, tal que $\pi_i \in (\{+1, +2, \ldots, +(n-1), +n\} \cup \{-1, -2, \ldots, -(n-1), -n\})$ e $i \neq j$ se, e somente se, $|\pi_i| \neq |\pi_j|$, para quaisquer i e j.

Definição 2.1.2. Denotamos uma *permutação sem sinais* de tamanho n por $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_{n-1}\ \pi_n)$, tal que $\pi_i \in \{1, 2, \ldots, n-1, n\}$ e $i \neq j$ se, e somente se, $\pi_i \neq \pi_j$, para quaisquer i e j.

Quando os genomas possuem conjuntos distintos de genes, dizemos que os genomas são *desbalanceados* e utilizamos *strings* para representar a ordem relativa dos genes. Cada elemento de uma string representa um gene ou uma sequência contígua de genes exclusiva de um dos genomas. Assim como nas permutações, para o caso em que a orientação dos genes é considerada, cada elemento possui um sinal "+" ou "−" que indica a orientação do gene.

Definição 2.1.3. Dada uma string $A = (A_1\ A_2\ \ldots\ A_m)$, denotamos por $|A| = m$ o *tamanho* da string A e por $|A_i|$ o *rótulo* que corresponde ao elemento A_i desconsiderando o seu sinal.

Definição 2.1.4. O *alfabeto* Σ_A de uma string A é o conjunto de rótulos da string A.

A fim de fazer com que as definições para strings e permutações sejam semelhantes, usamos o alfabeto $\Sigma_A = \{1, 2, \ldots, n\} \cup \{\alpha\}$, onde α é um rótulo usado para elementos que fazem parte do genoma origem $\mathcal{G}_o$, mas não fazem parte do genoma destino $\mathcal{G}_d$.

Uma instância genérica $\mathcal{I} = (\mathcal{G}_o, \mathcal{G}_d)$ para um problema de Distância de Rearranjos consiste em um genoma origem $\mathcal{G}_o$ e um genoma destino $\mathcal{G}_d$.

A *permutação identidade* ou *string identidade* de n elementos é denotada por $\iota^n = (+1\ +2\ \ldots\ +n)$ ou $\iota^n = (1\ 2\ \ldots\ n)$, dependendo se a orientação é considerada ou não na variação do problema.

Representamos a ordem relativa dos genes do genoma destino $\mathcal{G}_d$ usando a permutação (string) identidade ι^n, onde cada elemento ι^n_i representa um gene de $\mathcal{G}_d$, que possui correspondência em ambos genomas, ou um segmento contíguo maximal de genes que não possuem correspondência em $\mathcal{G}_o$.

Para a ordem relativa dos genes do genoma origem $\mathcal{G}_o$, usamos uma string $A = (A_1\ A_2\ \ldots\ A_m)$, onde cada elemento A_i representa um gene de $\mathcal{G}_o$, usando o mesmo mapeamento de rótulos e genes usado na representação de $\mathcal{G}_d$, ou um segmento contíguo maximal de genes sem correspondência em $\mathcal{G}_d$. Se A_i mapeia um gene que possui correspondência em ambos os genomas, então A_i tem sinal "+", se o gene possui a mesma orientação em ambos os genomas, ou A_i tem sinal "−", caso contrário. Para qualquer elemento A_i que mapeia um segmento contíguo de genes sem correspondência em $\mathcal{G}_d$, definimos $A_i = \alpha$, sem qualquer sinal, já que o elemento será removido independentemente do seu conteúdo.

Exemplo 2.1.1. Considere dois genomas com as sequências de genes $\mathcal{G}_o = (a\ b\ d\ e\ i\ f\ g)$ e $\mathcal{G}_d = (a\ c\ d\ h\ f\ g)$, onde cada letra representa um gene. Representamos $\mathcal{G}_d$ como a string $\iota^n = (1\ 2\ 3\ 4\ 5\ 6)$ e $\mathcal{G}_o$ como a string $A = (1\ \alpha\ 3\ \alpha\ 5\ 6)$. Neste exemplo, temos $|A| = 6$, $\Sigma_A \cap \Sigma_{\iota^n} = \{1, 3, 5, 6\}$, $\Sigma_A \setminus \Sigma_{\iota^n} = \{\alpha\}$ e $\Sigma_{\iota^n} \setminus \Sigma_A = \{2, 4\}$. Note que o segmento (e, i) de $\mathcal{G}_o$ é mapeado em um único elemento α em A.

Definição 2.1.5. Dada uma permutação π, denotamos por $-\pi_i$ o elemento π_i com orientação invertida. O mesmo é válido para strings.

Exemplo 2.1.2. Dado $\pi = (+4\ -3\ -2\ +1)$, temos que $-\pi_1 = -4$, $-\pi_2 = +3$, $-\pi_3 = +2$ e $-\pi_4 = -1$.

O conjunto de *rótulos comuns* às strings é definido por $\Sigma_A \cap \Sigma_{\iota^n}$, o conjunto de *rótulos exclusivos* ao genoma destino é definido por $\Sigma_{\iota^n} \setminus \Sigma_A$, e o conjunto de *rótulos exclusivos* ao genoma origem é definido por $\Sigma_A \setminus \Sigma_{\iota^n}$, que ou é vazio ou é igual a $\{\alpha\}$. Um elemento da string (A_i ou ι_i^n) é classificado como *comum* ou *exclusivo* de acordo com a classificação do seu rótulo.

A modelagem dos genomas usando exclusivamente a ordem relativa dos genes é chamada de *representação clássica*. Para genomas desbalanceados, uma *instância clássica* é denotada por $\mathcal{I} = (A, \iota^n)$, sendo que as strings possuem sinais ou não, dependendo da informação disponível em relação à orientação dos genes.

Quando $\mathcal{G}_o$ e $\mathcal{G}_d$ são genomas balanceados, a string A corresponde a uma permutação e, assim, usamos π para representar a ordem relativa dos genes de $\mathcal{G}_o$. Exceto quando dito expressamente o contrário, assumimos que a permutação π tem tamanho n e, portanto, o genoma destino $\mathcal{G}_d$ é representado pela permutação identidade ι^n. Note que em genomas balanceados ambas as sequências de genes possuem o mesmo tamanho. Quando a orientação dos genes é desconhecida, apenas ignoramos os sinais no mapeamento dos genomas $\mathcal{G}_o$ e $\mathcal{G}_d$. Como ι^n pode ser inferido a partir do tamanho de π, uma instância clássica é denotada apenas por $\mathcal{I} = \pi$.

Note que para genomas balanceados, os problemas de distância de rearranjos entre dois genomas são equivalentes aos problemas de ordenação de uma permutação π usando operações de rearranjo.

2.1.2 Representação de Regiões Intergênicas

Regiões intergênicas são sequências de nucleotídeos presentes entre pares de genes contíguos e nas extremidades do genoma. O *tamanho* de uma região intergênica é a quantidade de nucleotídeos que estão nela. Estudos mostram que as regiões intergênicas são mais suscetíveis a mudanças e são quebradas por rearranjos de genomas [20, 21]. Além disso, ao comparar dois genomas, podemos achar correspondência entre os genes desses genomas, o que não é possível com as regiões intergênicas [20, 21]. Esses motivos nos levam a representar regiões intergênicas usando os seus tamanhos ao invés de rótulos, o que já é feito em outros trabalhos [35, 45, 68].

A distribuição do tamanho das regiões intergênicas é modelada matematicamente por uma lista de números inteiros não-negativos. Ao considerar regiões intergênicas, o genoma origem $\mathcal{G}_o$ é modelado pela tupla $(A, \breve{A})$ e o genoma destino $\mathcal{G}_d$ é modelado pela tupla $(\iota^n, \breve{\iota}^n)$. De forma similar ao que é feito nas instâncias em que apenas a ordem relativa dos genes é considerada, ao construir a representação de uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, mapeamos com um único elemento tanto em A quanto em ι^n os segmentos contíguos maximais de genes que não possuem correspondência no outro genoma.

A lista $\breve{A}$ tem tamanho $|\breve{A}| = |A| + 1$ e a lista $\breve{\iota}^n$ tem tamanho $|\breve{\iota}^n| = n + 1$. Temos que $\breve{A}_i$ corresponde ao número de nucleotídeos na região intergênica entre A_{i-1} e A_i, para $2 \leq i \leq |A|$, e os valores $\breve{A}_1$ e $\breve{A}_{|A|+1}$ correspondem ao número de nucleotídeos nas extremidades de $\mathcal{G}_o$. Analogamente, temos que $\breve{\iota}_i^n$ corresponde ao número de nucleotídeos na região intergênica entre ι_{i-1}^n e ι_i^n, para $2 \leq i \leq n$, e os valores $\breve{\iota}_1^n$ e $\breve{\iota}_{n+1}^n$ correspondem ao número de nucleotídeos nas extremidades de $\mathcal{G}_d$.

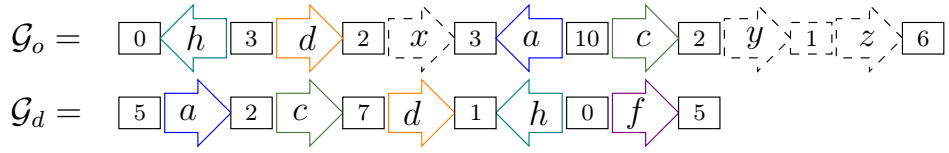


Figura 2.1: Exemplo de dois genomas $\mathcal{G}_o$ e $\mathcal{G}_d$, onde genes são representados por letras dentro de setas, a orientação dos genes é indicada pela orientação das setas, e os tamanhos das regiões intergênicas são representados por números dentro de retângulos. Os genes de $\mathcal{G}_d$ são mapeados da seguinte forma: a é mapeado em $+1$, c é mapeado em $+2$, d é mapeado em $+3$, h é mapeado em $+4$, e f é mapeado em $+5$. Assim, o genoma $\mathcal{G}_d$ é representado por $(\iota^n, \breve{\iota}^n)$, onde $\iota^n = (+1\ +2\ +3\ +4\ +5)$ e $\breve{\iota}^n = (5, 2, 7, 1, 0, 5)$. O gene x e o segmento que vai de y até z em $\mathcal{G}_o$ não estão presentes em $\mathcal{G}_d$. Portanto, ambos são mapeados no elemento α. O genoma $\mathcal{G}_o$ é representado por $(A, \breve{A})$, onde $A = (+4\ +3\ \alpha\ -1\ +2\ \alpha)$ e $\breve{A} = (0, 3, 2, 3, 10, 2, 6)$. Os alfabetos Σ_A e Σ_{ι^n} são os conjuntos $\{1, 2, 3, 4, \alpha\}$ e $\{1, 2, 3, 4, 5\}$, respectivamente.

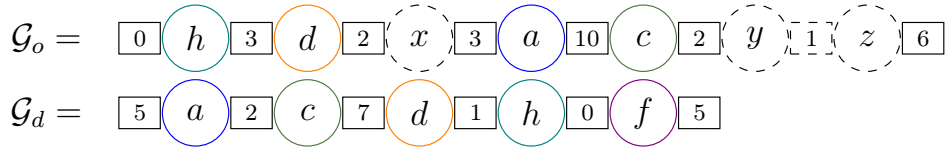


Figura 2.2: Exemplo de dois genomas $\mathcal{G}_o$ e $\mathcal{G}_d$, onde genes são representados por letras dentro de círculos, a orientação dos genes é desconhecida, e os tamanhos das regiões intergênicas são representados por números dentro de retângulos. Os genes de $\mathcal{G}_d$ são mapeados da seguinte forma: a é mapeado em 1, c é mapeado em 2, d é mapeado em 3, h é mapeado em 4, e f é mapeado em 5. Assim, o genoma $\mathcal{G}_d$ é representado por $(\iota^n, \breve{\iota}^n)$, onde $\iota^n = (1\ 2\ 3\ 4\ 5)$ e $\breve{\iota}^n = (5, 2, 7, 1, 0, 5)$. O gene x e o segmento que vai de y até z em $\mathcal{G}_o$ não estão presentes em $\mathcal{G}_d$. Portanto, ambos são mapeados no elemento α. O genoma $\mathcal{G}_o$ é representado por $(A, \breve{A})$, onde $A = (4\ 3\ \alpha\ 1\ 2\ \alpha)$ e $\breve{A} = (0, 3, 2, 3, 10, 2, 6)$. Os alfabetos Σ_A e Σ_{ι^n} são os conjuntos $\{1, 2, 3, 4, \alpha\}$ e $\{1, 2, 3, 4, 5\}$, respectivamente.

A modelagem dos genomas que considera a ordem relativa dos genes e a distribuição do tamanho de regiões intergênicas é chamada de *representação intergênica*. Nos problemas intergênicos estudados neste trabalho, sempre tratamos de genomas desbalanceados. Assim, uma *instância intergênica* é denotada por $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, onde $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$.

Nas figuras 2.1 e 2.2, apresentamos exemplos de como genomas são modelados matematicamente usando a representação intergênica.

2.2 Rearranjos de Genomas

Nesta seção, apresentamos os tipos de rearranjos de genomas que serão estudados no decorrer desta tese, assim como variações desses rearranjos dependendo da representação utilizada para os genomas.

Os rearranjos de genomas podem afetar apenas a ordem relativa de um ou mais segmentos do genoma (*rearranjos conservativos*) ou eles podem alterar o material genético do genoma (*rearranjos não conservativos*). Dentre os rearranjos conservativos, podemos citar as reversões, transposições e *block interchanges*. Os rearranjos não conservativos

considerados neste trabalho são as inserções e deleções. Essas últimas duas operações são chamadas coletivamente de *indels*.

Como uma permutação é um tipo específico de string que possui alfabeto $\Sigma = \{1, 2, \ldots, n\}$, definimos as operações de rearranjo usando strings. Apenas os *indels* não são operações válidas em permutações.

Definição 2.2.1. Para uma operação de rearranjo β, denotamos por $\mathcal{G} \cdot \beta = \mathcal{G}'$ o genoma resultante após β ser aplicado em $\mathcal{G}$. Também temos $A \cdot \beta = A'$ e $\breve{A} \cdot \beta = \breve{A}'$ como resultados da aplicação de β na string A e na lista $\breve{A}$, respectivamente.

Dada uma sequência de operações $S = (\beta_1, \beta_2, \ldots, \beta_k)$ com tamanho $|S| = k$, denotamos por $\mathcal{G} \cdot S = (((\mathcal{G} \cdot \beta_1) \cdot \beta_2) \cdot \ldots \cdot \beta_{k-1}) \cdot \beta_k = \mathcal{G}'$ o genoma resultante da aplicação da sequência S em $\mathcal{G}$. A mesma notação é válida para aplicação de uma sequência de operações em strings e listas de regiões intergênicas.

Agora, apresentamos os rearranjos conservativos e os seus respectivos efeitos na ordem relativa dos genes de um genoma.

Definição 2.2.2. Considerando a representação clássica, dada uma string A de tamanho m, uma *reversão* $\rho(i, j)$, com $1 \leq i \leq j \leq m$, inverte o segmento $(A_i\ A_{i+1}\ \ldots\ A_{j-1}\ A_j)$ e, caso A seja uma string com sinais, inverte todos os sinais dos elementos afetados. Mostramos a seguir o resultado da aplicação de $\rho(i, j)$ quando A é uma string com sinais (1) e quando A é uma string sem sinais (2).

$$(1) \quad A \cdot \rho(i,j) = (A_1\ \ldots\ A_{i-1}\ \underline{-A_j\ -A_{j-1}\ \ldots\ -A_{i+1}\ -A_i}\ A_{j+1}\ \ldots\ A_m)$$
$$(2) \quad A \cdot \rho(i,j) = (A_1\ \ldots\ A_{i-1}\ \underline{A_j\ A_{j-1}\ \ldots\ A_{i+1}\ A_i}\ A_{j+1}\ \ldots\ A_m)$$

Definição 2.2.3. Considerando a representação clássica, dada uma string A de tamanho m, uma *transposição* $\tau(i, j, k)$, com $1 \leq i < j < k \leq m + 1$, troca as posições relativas dos segmentos $(A_i\ A_{i+1}\ \ldots\ A_{j-1})$ e $(A_j\ A_{j+1}\ \ldots\ A_{k-1})$. Por não afetar a orientação dos elementos, o efeito de uma transposição é o mesmo em strings com e sem sinais. Mostramos a seguir o efeito de $\tau(i, j, k)$ em A.

$$A \cdot \tau(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{A_j\ A_{j+1}\ \ldots\ A_{k-1}}\ \underline{A_i\ A_{i+1}\ \ldots\ A_{j-1}}\ A_k\ \ldots\ A_m)$$

Definição 2.2.4. Considerando a representação clássica, dada uma string A de tamanho m, o rearranjo *block interchange* $\mathcal{BI}(i, j, k, l)$, com $1 \leq i \leq j < k \leq l \leq m$, troca as posições relativas dos segmentos $(A_i\ \ldots\ A_j)$ e $(A_k\ \ldots\ A_l)$. O efeito de um *block interchange* é o mesmo para strings com ou sem sinais. Mostramos a seguir o efeito de $\mathcal{BI}(i, j, k, l)$ em A.

$$A \cdot \mathcal{BI}(i,j,k,l) = (A_1\ \ldots\ A_{i-1}\ \underline{A_k\ \ldots\ A_l}\ A_{j+1}\ \ldots\ A_{k-1}\ \underline{A_i\ \ldots\ A_j}\ A_{l+1}\ \ldots\ A_m)$$

Note que uma transposição é um tipo especial de *block interchange* em que os segmentos afetados são adjacentes.

Dados dois segmentos adjacentes $\sigma_1 = (A_i\ A_{i+1}\ \ldots\ A_{j-1})$ e $\sigma_2 = (A_j\ A_{j+1}\ \ldots\ A_{k-1})$, uma *transposição inversa* $\rho\tau$ troca as posições relativas desses dois segmentos adjacentes e

inverte os elementos de um dos dois segmentos afetados, sendo que essa operação também inverte todos os sinais dos elementos presentes no segmento invertido, quando aplicada a uma string com sinais.

Definição 2.2.5. Considerando a representação clássica, dada uma string A de tamanho m, uma transposição inversa Tipo 1 $\rho\tau_1(i,j,k)$ e uma transposição inversa Tipo 2 $\rho\tau_2(i,j,k)$, com $1 \leq i < j < k \leq m+1$, afetam A da seguinte forma, dependendo se A é uma string com sinais (1) ou se A é uma string sem sinais (2):

$$(1)\ A \cdot \rho\tau_1(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{A_j\ A_{j+1}\ \ldots\ A_{k-1}}\ \underline{-A_{j-1}\ \ldots\ -A_{i+1}\ -A_i}\ A_k\ \ldots\ A_m)$$
$$(2)\ A \cdot \rho\tau_1(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{A_j\ A_{j+1}\ \ldots\ A_{k-1}}\ \underline{A_{j-1}\ \ldots\ A_{i+1}\ A_i}\ A_k\ \ldots\ A_m)$$
$$(1)\ A \cdot \rho\tau_2(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{-A_{k-1}\ \ldots\ -A_{j+1}\ -A_j}\ \underline{A_i\ A_{i+1}\ \ldots\ A_{j-1}}\ A_k\ \ldots\ A_m)$$
$$(2)\ A \cdot \rho\tau_2(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{A_{k-1}\ \ldots\ A_{j+1}\ A_j}\ \underline{A_i\ A_{i+1}\ \ldots\ A_{j-1}}\ A_k\ \ldots\ A_m)$$

Dados dois segmentos adjacentes $\sigma_1 = (A_i\ A_{i+1}\ \ldots\ A_{j-1})$ e $\sigma_2 = (A_j\ A_{j+1}\ \ldots\ A_{k-1})$, uma *revrev* $\rho\rho$ inverte esses dois segmentos adjacentes, sendo que essa operação também inverte todos os sinais dos elementos afetados, quando aplicada em uma string com sinais. As revrevs não trocam a posição relativa dos segmentos σ_1 e σ_2.

Definição 2.2.6. Considerando a representação clássica, dada uma string A de tamanho m, uma *revrev* $\rho\rho(i,j,k)$, com $1 \leq i < j < k \leq m+1$, afeta A da seguinte forma, dependendo se A é uma string com sinais (1) ou se A é uma string sem sinais (2):

$$(1)\ A \cdot \rho\rho(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{-A_{j-1}\ \ldots\ -A_{i+1}\ -A_i}\ \underline{-A_{k-1}\ \ldots\ -A_{j+1}\ -A_j}\ A_k\ \ldots\ A_m)$$
$$(2)\ A \cdot \rho\rho(i,j,k) = (A_1\ \ldots\ A_{i-1}\ \underline{A_{j-1}\ \ldots\ A_{i+1}\ A_i}\ \underline{A_{k-1}\ \ldots\ A_{j+1}\ A_j}\ A_k\ \ldots\ A_m)$$

Nesta tese, estudamos também as inserções e deleções (*indels*). Essas operações afetam o alfabeto e o tamanho da string em que são aplicadas, sendo operações essenciais em problemas de rearranjos que comparam genomas desbalanceados, já que rearranjos conservativos não conseguem balancear os genomas comparados. As próximas definições apresentam formalmente os *indels* e seus respectivos efeitos em strings.

Definição 2.2.7. Considerando a representação clássica, dada uma string A de tamanho m, uma *inserção* $\phi(i,\sigma)$ adiciona a string σ após o i-ésimo elemento de A, onde $0 \leq i \leq m$ e σ é uma string não vazia sem repetições. Mostramos a seguir o efeito de $\phi(i,\sigma)$ em A.

$$A \cdot \phi(i,\sigma) = (A_1\ \ldots\ A_i\ \underline{\sigma_1\ \ldots\ \sigma_{|\sigma|}}\ A_{i+1}\ \ldots\ A_m)$$

Note que, em uma inserção $\phi(i,\sigma)$ aplicada na string A, é necessário que σ seja do mesmo tipo (com ou sem sinais) que a string A.

Definição 2.2.8. Considerando a representação clássica, dada uma string A de tamanho m, uma *deleção* $\psi(i,j)$, com $1 \leq i \leq j \leq m$, remove o segmento $(A_i\ \ldots\ A_j)$. Mostramos a seguir o efeito de $\psi(i,j)$ em A.

$$A \cdot \psi(i,j) = (A_1\ \ldots\ A_{i-1}\ A_{j+1}\ \ldots\ A_m)$$

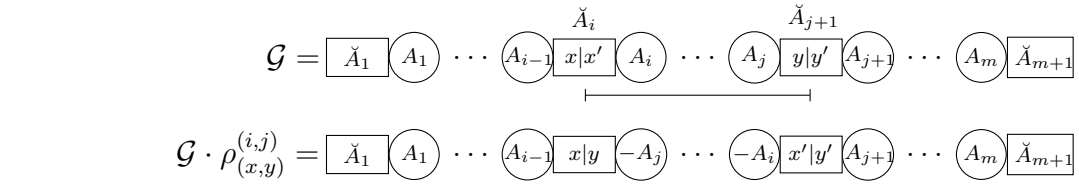
2.2.1 Efeito dos Rearranjos de Genomas em Regiões Intergênicas

Nesta seção, mostramos como os principais rearranjos de genomas (reversões, transposições e *indels*) afetam tanto a ordem relativa dos genes quanto as regiões intergênicas de um genoma.

Definição 2.2.9. Considerando a representação intergênica, dado um genoma $\mathcal{G} = (A, \breve{A})$ com $|A| = m$, uma *reversão intergênica* $\rho^{(i,j)}_{(x,y)}$, com $1 \leq i \leq j \leq m$, $0 \leq x \leq \breve{A}_i$ e $0 \leq y \leq \breve{A}_{j+1}$, quebra as regiões intergênicas $\breve{A}_i$ em (x, x') e $\breve{A}_{j+1}$ em (y, y'), onde $x' = \breve{A}_i - x$ e $y' = \breve{A}_{j+1} - y$, e inverte o segmento $(x'\ A_i\ \breve{A}_{i+1}\ A_{i+1}\ \ldots\ \breve{A}_j\ A_j\ y)$. Além disso, caso A seja uma string com sinais, todos os sinais dos elementos afetados da string A são invertidos. Essa reversão transforma $\mathcal{G}$ em $\mathcal{G} \cdot \rho^{(i,j)}_{(x,y)} = (A', \breve{A}')$, onde:

$$A' = A \cdot \rho^{(i,j)}_{(x,y)} = (A_1\ \ldots\ A_{i-1}\ \underline{-A_j\ \ldots\ -A_i}\ A_{j+1}\ \ldots\ A_m)$$
$$\breve{A}' = \breve{A} \cdot \rho^{(i,j)}_{(x,y)} = (\breve{A}_1, \ldots, \breve{A}_{i-1}, x + y, \underline{\breve{A}_j, \ldots, \breve{A}_{i+1}}, x' + y', \breve{A}_{j+2}, \ldots, \breve{A}_{m+1})$$

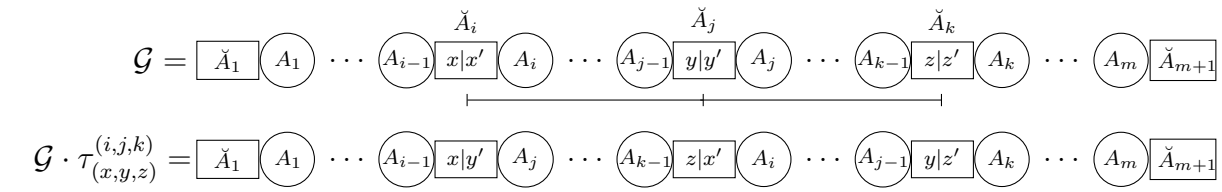
O efeito de uma reversão intergênica é similar quando A é uma string sem sinais, sendo necessário apenas ignorar os sinais da string. Mostramos a seguir o efeito de $\rho^{(i,j)}_{(x,y)}$ em $\mathcal{G} = (A, \breve{A})$ quando A é uma string com sinais.



Definição 2.2.10. Considerando a representação intergênica, dado um genoma $\mathcal{G} = (A, \breve{A})$ com $|A| = m$, uma *transposição intergênica* $\tau^{(i,j,k)}_{(x,y,z)}$, com $1 \leq i < j < k \leq m+1$, $0 \leq x \leq \breve{A}_i$, $0 \leq y \leq \breve{A}_j$ e $0 \leq z \leq \breve{A}_k$, quebra as regiões intergênicas $\breve{A}_i$ em (x, x'), $\breve{A}_j$ em (y, y') e $\breve{A}_k$ em (z, z'), onde $x' = \breve{A}_i - x$, $y' = \breve{A}_j - y$ e $z' = \breve{A}_k - z$, e troca as posições relativas dos segmentos adjacentes $(x'\ A_i\ \breve{A}_{i+1}\ A_{i+1}\ \ldots\ \breve{A}_{j-1}\ A_{j-1}\ y)$ e $(y'\ A_j\ \breve{A}_{j+1}\ \ldots\ \breve{A}_{k-1}\ A_{k-1}\ z)$. Essa transposição transforma $\mathcal{G}$ em $\mathcal{G} \cdot \tau^{(i,j,k)}_{(x,y,z)} = (A', \breve{A}')$, onde:

$$A' = (A_1\ \ldots\ A_{i-1}\ \underline{A_j\ \ldots\ A_{k-1}}\ \underline{A_i\ \ldots\ A_{j-1}}\ A_k\ \ldots\ A_m)$$
$$\breve{A}' = (\breve{A}_1, \ldots, \breve{A}_{i-1}, x + y', \underline{\breve{A}_{j+1}, \ldots, \breve{A}_{k-1}}, z + x', \underline{\breve{A}_{i+1}, \ldots, \breve{A}_{j-1}}, y + z', \breve{A}_{k+1}, \ldots, \breve{A}_{m+1})$$

Mostramos a seguir o efeito de $\tau^{(i,j,k)}_{(x,y,z)}$ em $\mathcal{G} = (A, \breve{A})$.

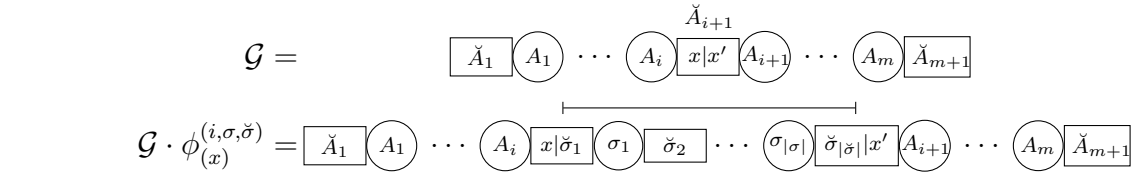


Definição 2.2.11. Considerando a representação intergênica, dado um genoma $\mathcal{G} = (A, \breve{A})$

com $|A| = m$, uma *inserção intergênica* $\phi_{(x)}^{(i,\sigma,\breve{\sigma})}$, tal que $0 \leq i \leq m$, $0 \leq x \leq \breve{A}_{i+1}$, σ é uma string sem repetições e $\breve{\sigma}$ é uma lista de inteiros de tamanho $|\breve{\sigma}| = |\sigma| + 1$, adiciona o segmento $(\breve{\sigma}_1\ \sigma_1\ \ldots\ \breve{\sigma}_{|\sigma|}\ \sigma_{|\sigma|}\ \breve{\sigma}_{|\sigma|+1})$ após o x-ésimo nucleotídeo da região intergênica $\breve{A}_{i+1}$. A inserção $\phi_{(x)}^{(i,\sigma,\breve{\sigma})}$ transforma $\mathcal{G}$ em $\mathcal{G} \cdot \phi_{(x)}^{(i,\sigma,\breve{\sigma})} = (A', \breve{A}')$, onde:

$$
\begin{aligned}
A' &= A \cdot \phi_x^{(i,\sigma,\breve{\sigma})} = (A_1\ \ldots\ A_i\ \underline{\sigma_1\ \ldots\ \sigma_{|\sigma|}}\ A_{i+1}\ \ldots\ A_m) \\
\breve{A}' &= \breve{A} \cdot \phi_x^{(i,\sigma,\breve{\sigma})} = (\breve{A}_1, \ldots, \breve{A}_i, \underline{x + \breve{\sigma}_1, \breve{\sigma}_2, \ldots, \breve{\sigma}_{|\sigma|}, \breve{\sigma}_{|\sigma|+1} + x'}, \breve{A}_{i+2}, \ldots, \breve{A}_{m+1}) \\
x' &= \breve{A}_{i+1} - x
\end{aligned}
$$

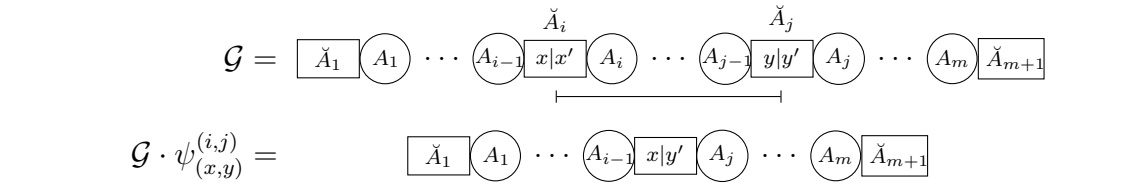
Ao contrário de uma inserção considerando a representação clássica, em uma inserção intergênica a string σ pode ser vazia. Nesse caso, a inserção intergênica $\phi_{(x)}^{(i,\sigma,\breve{\sigma})}$ altera apenas a região intergênica $\breve{A}_{i+1}$. Mostramos a seguir o efeito de $\phi_{(x)}^{(i,\sigma,\breve{\sigma})}$ em $\mathcal{G} = (A, \breve{A})$.



Definição 2.2.12. Considerando a representação intergênica, dado um genoma $\mathcal{G} = (A, \breve{A})$ com $|A| = m$, uma *deleção intergênica* $\psi_{(x,y)}^{(i,j)}$, tal que $1 \leq i \leq j \leq m+1$, $0 \leq x \leq \breve{A}_i$ e $0 \leq y \leq \breve{A}_j$, remove o segmento que inicia após o x-ésimo nucleotídeo de $\breve{A}_i$ e termina no y-ésimo nucleotídeo de $\breve{A}_j$, transformando $\mathcal{G}$ em $\mathcal{G} \cdot \psi_{(x,y)}^{(i,j)} = (A', \breve{A}')$, onde:

$$
\begin{aligned}
A' &= A \cdot \psi_{(x,y)}^{(i,j)} = (A_1\ \ldots\ A_{i-1}\ A_j\ \ldots\ A_m) \\
\breve{A}' &= \breve{A} \cdot \psi_{(x,y)}^{(i,j)} = (\breve{A}_1, \ldots, \breve{A}_{i-1}, x + y', \breve{A}_{j+1}, \ldots, \breve{A}_{m+1}) \\
y' &= \breve{A}_j - y
\end{aligned}
$$

Quando $i = j$, uma deleção $\psi_{(x,y)}^{(i,j)}$ não remove elementos de A e apenas altera a região intergênica $\breve{A}_j$, portanto, essa operação deve atender a restrição $0 \leq x \leq y \leq \breve{A}_j$. Mostramos a seguir o efeito de $\psi_{(x,y)}^{(i,j)}$ em $\mathcal{G} = (A, \breve{A})$.



Neste trabalho, seguimos as restrições apresentadas por Willing e coautores [83]: dada uma instância clássica $\mathcal{I} = (A, \iota^n)$ ou uma instância intergênica $\mathcal{I}^{ig} = ((A, \breve{A}), (\iota^n, \breve{\iota}^n))$, temos que um elemento A_i só pode ser removido se $|A_i| \notin \Sigma_{\iota^n}$, e um elemento x só pode ser inserido em A se $x \in \Sigma_{\iota^n} \setminus \Sigma_A$. Ou seja, adicionamos restrições para prevenir que os *indels* removam e, posteriormente, insiram os mesmos genes (ou vice-versa).

2.3 Problemas de Distância de Rearranjos

Um *modelo de rearranjos* $\mathcal{M}$ é o conjunto de operações permitidas em um problema de Distância de Rearranjos.

Em problemas de distância, podemos considerar que todas as operações utilizadas na solução possuem o mesmo custo (*abordagem não ponderada*) ou que a aplicação de uma determinada operação possui um custo associado a ela (*abordagem ponderada*). A não ser que seja dito expressamente o contrário, assumimos que os problemas de distância de rearranjos consideram uma abordagem não ponderada. A seguir, apresentamos esses problemas considerando cada tipo de instância.

Dados um modelo $\mathcal{M}$ e uma permutação π de tamanho n, a *distância de ordenação* $d_{\mathcal{M}}(\pi)$ é o número mínimo de rearranjos do modelo $\mathcal{M}$ necessários para ordenar π (transformar π em ι^n), ou seja, temos $d_{\mathcal{M}}(\pi) = |S|$ tal que $\pi \cdot S = \iota^n$, a sequência S tem tamanho mínimo e $\beta \in \mathcal{M}$, para todo $\beta \in S$.

Ordenação de Permutações por Rearranjos
Entrada: Uma instância clássica de genomas balanceados $\mathcal{I} = \pi$.
Objetivo: Considerando um modelo de rearranjos $\mathcal{M}$, encontrar uma sequência $S = (\beta_1, \beta_2, \ldots, \beta_{|S|})$ de tamanho mínimo (i.e., $|S| = d_{\mathcal{M}}(\pi)$) que ordena π e é formada apenas de rearranjos pertencentes a $\mathcal{M}$.

Na abordagem ponderada, considerando uma função de custo $w : \mathcal{M} \to \mathbb{R}$, dizemos que um rearranjo β possui custo $w(\beta)$. Para uma sequência $S = (\beta_1, \ldots, \beta_\ell)$, temos que $w(S) = \sum_{i=1}^{\ell} w(\beta_i)$.

Dados um modelo $\mathcal{M}$, uma função de custo w e uma permutação π, a distância de ordenação $d^w_{\mathcal{M}}(\pi)$ é o custo mínimo necessário para ordenar π usando apenas operações de $\mathcal{M}$, ou seja, $d^w_{\mathcal{M}}(\pi) = w(S)$, com $S = (\beta_1, \ldots, \beta_\ell)$, tal que $\beta_i \in \mathcal{M}$, para todo $\beta_i \in S$, $\pi \cdot S = \iota^n$ e o valor de $w(S)$ é mínimo.

Ordenação de Permutações por Rearranjos Ponderados
Entrada: Uma instância clássica de genomas balanceados $\mathcal{I} = \pi$.
Objetivo: Considerando um modelo de rearranjos $\mathcal{M}$ e uma função de custo w, encontrar uma sequência S de custo mínimo que ordena π e é formada apenas de rearranjos pertencentes a $\mathcal{M}$, ou seja, $w(S) = d^w_{\mathcal{M}}(\pi)$ e $\beta \in \mathcal{M}$, para todo $\beta \in S$.

Dados um modelo $\mathcal{M}$ e uma instância clássica de genomas desbalanceados $\mathcal{I} = (A, \iota^n)$, a distância de rearranjos $d_{\mathcal{M}}(A, \iota^n)$ (ou $d_{\mathcal{M}}(\mathcal{I})$) é o número mínimo de rearranjos do modelo $\mathcal{M}$ necessários para transformar a string A na string ι^n, ou seja, temos $d_{\mathcal{M}}(A, \iota^n) = |S|$ tal que $A \cdot S = \iota^n$, a sequência S tem tamanho mínimo e $\beta \in \mathcal{M}$, para todo $\beta \in S$.

Distância de Rearranjos
Entrada: Uma instância clássica de genomas desbalanceados $\mathcal{I} = (A, \iota^n)$.
Objetivo: Considerando um modelo de rearranjos $\mathcal{M}$, encontrar uma sequência $S = (\beta_1, \beta_2, \ldots, \beta_{|S|})$ de tamanho mínimo (i.e., $|S| = d_{\mathcal{M}}(A, \iota^n)$) que transforma A em ι^n e é formada apenas de rearranjos pertencentes a $\mathcal{M}$.

Dados um modelo $\mathcal{M}$ e uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, a distância de rearranjos intergênicos $d_{\mathcal{M}}(\mathcal{G}_o, \mathcal{G}_d)$ (ou $d_{\mathcal{M}}(\mathcal{I}^{ig})$) é o número mínimo de rearranjos do modelo $\mathcal{M}$ necessários para transformar o genoma origem $\mathcal{G}_o$ no genoma destino $\mathcal{G}_d$, ou seja, temos $d_{\mathcal{M}}(\mathcal{G}_o, \mathcal{G}_d) = |S|$ tal que $A \cdot S = \iota^n$, $\breve{A} \cdot S = \breve{\iota}^n$, a sequência S tem tamanho mínimo e $\beta \in \mathcal{M}$, para todo $\beta \in S$.

Distância de Rearranjos Intergênicos
Entrada: Uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$.
Objetivo: Considerando um modelo de rearranjos $\mathcal{M}$, encontrar uma sequência $S = (\beta_1, \beta_2, \ldots, \beta_{|S|})$ de tamanho mínimo (i.e., $|S| = d_{\mathcal{M}}(\mathcal{G}_o, \mathcal{G}_d)$) que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$ e é formada apenas de rearranjos pertencentes a $\mathcal{M}$.

2.4 Breakpoints

O conceito de *breakpoints* é bastante usado em problemas de distância de rearranjos, sendo útil na definição de limitantes e algoritmos, além de já ter sido utilizado em provas de NP-dificuldade [64, 34]. De forma geral, existe um *breakpoint* entre dois elementos consecutivos do genoma origem se esses dois elementos não são consecutivos no genoma destino. A definição de um *breakpoint* depende do modelo de rearranjos utilizado e da representação dos genomas. A seguir, apresentamos cada tipo de *breakpoint*.

2.4.1 Breakpoints em Permutações

Nesta seção, apresentamos definições de *breakpoints* para uma instância clássica de genomas balanceados $\mathcal{I} = \pi$.

Definição 2.4.1. Dada uma permutação σ de tamanho n, obtemos a *versão estendida* de σ ao adicionar os elementos $\sigma_0 = +0$ e $\sigma_{n+1} = +(n+1)$. Caso σ seja uma permutação sem sinais, temos que $\sigma_0 = 0$ e $\sigma_{n+1} = n+1$. Esses elementos são adicionados nas permutações apenas para facilitar algumas notações e, portanto, eles nunca são afetados por rearranjos de genomas.

Nas próximas definições, assumimos que π e ι^n estão nas suas versões estendidas.

Definição 2.4.2. Um *breakpoint de reversões sem sinais* existe entre um par de elementos consecutivos (π_i, π_{i+1}) se $|\pi_{i+1} - \pi_i| \neq 1$, para $0 \leq i \leq n$.

Exemplo 2.4.1. Para a permutação sem sinais $\pi = (0\ 4\ 3\ 5\ 1\ 2\ 6\ 7)$, que está na versão estendida, temos os seguintes *breakpoints* de reversões sem sinais (representados pelo símbolo $\circ$):

$$\pi = (0 \circ 4\ 3 \circ 5 \circ 1\ 2 \circ 6\ 7)$$

Definição 2.4.3. Um *breakpoint de transposições* existe entre um par de elementos consecutivos (π_i, π_{i+1}) se $\pi_{i+1} - \pi_i \neq 1$, para $0 \leq i \leq n$.

Exemplo 2.4.2. Para a permutação sem sinais $\pi = (0\ 4\ 3\ 5\ 1\ 2\ 6\ 7)$, que está na versão estendida, temos os seguintes *breakpoints* de transposições (representados pelo símbolo $\circ$):

$$\pi = (0 \circ 4 \circ 3 \circ 5 \circ 1\ 2 \circ 6\ 7)$$

Definição 2.4.4. Um *breakpoint de reversões com sinais* existe entre um par de elementos consecutivos (π_i, π_{i+1}) se $\pi_{i+1} - \pi_i \neq 1$, para $0 \leq i \leq n$.

Exemplo 2.4.3. Para a permutação com sinais $\pi = (+0\ -4\ -3\ +5\ +1\ +2\ -6\ +7)$, que está na versão estendida, temos os seguintes *breakpoints* de reversões com sinais (representados pelo símbolo $\circ$):

$$\pi = (+0 \circ -4\ -3 \circ +5 \circ +1\ +2 \circ -6 \circ +7)$$

Agora, apresentamos qual tipo de *breakpoint* é usado em cada modelo de rearranjos. Dessa forma, quando o modelo de rearranjos está explícito, não precisamos indicar qual o tipo de *breakpoint* está sendo considerado.

Para permutações com sinais, todos os modelos considerados neste trabalho utilizam a definição de *breakpoint* de reversões com sinais e, assim, a permutação identidade com sinais é a única que não possui *breakpoints*.

Note que a *permutação inversa sem sinais* $\eta^n = (n\ (n-1)\ (n-2)\ \ldots\ 2\ 1)$ possui apenas 2 *breakpoints* de reversões sem sinais, enquanto essa mesma permutação possui $n+1$ *breakpoints* de transposições. Essa permutação pode ser transformada em ι^n usando apenas uma reversão ou apenas duas operações, considerando modelos que possuem transposições combinadas com transposições inversas ou revrevs. No entanto, para transformar η^n em ι^n usando apenas transposições, precisamos de $\Theta(n)$ operações [59].

Sendo assim, para permutações sem sinais, os modelos que possuem alguma operação que causa inversão de segmento(s) (e.g., reversão, transposição inversa e revrev) utilizam a definição de *breakpoint* de reversões sem sinais. Por último, o modelo que possui apenas transposições ou *block interchanges* utiliza a definição de *breakpoint* de transposições. Em ambos os casos, a permutação identidade sem sinais é a única que não possui *breakpoints*.

Note que se um modelo de rearranjos possui apenas transposições ou *block interchanges*, então apenas permutações sem sinais podem ser consideradas, já que ambas as operações não alteram sinais de elementos.

Definição 2.4.5. Dado um modelo $\mathcal{M}$, $b_{\mathcal{M}}(\pi)$ denota o número de *breakpoints* em π.

Definição 2.4.6. Dado um modelo $\mathcal{M}$ e um rearranjo (ou sequência de rearranjos) β, $\Delta b_{\mathcal{M}}(\pi, \beta) = b_{\mathcal{M}}(\pi) - b_{\mathcal{M}}(\pi \cdot \beta)$ denota a variação no número de *breakpoints* após β ser aplicado em π.

Definição 2.4.7. Considerando um tipo de *breakpoints*, uma *strip* é uma sequência maximal de elementos sem *breakpoints* entre elementos consecutivos dessa sequência.

Para permutações sem sinais, uma *strip* $(\pi_i\ \pi_{i+1}\ \ldots\ \pi_j)$, com $0 \leq i < j \leq n+1$, é *crescente* se $\pi_{k+1} > \pi_k$, para todo $i \leq k < j$. Caso contrário, a *strip* é *decrescente* e temos que $\pi_{k+1} < \pi_k$, para todo $i \leq k < j$. Um *singleton* é uma *strip* de tamanho um. Um *singleton* é crescente se ele é igual a (π_0) ou (π_{n+1}), caso contrário, ele é dito decrescente. Note que os elementos π_0 e π_{n+1} sempre pertencem a *strips* crescentes.

Para permutações com sinais, uma *strip* é classificada como *positiva* se todos os elementos possuem sinal "+". Caso contrário, essa *strip* é classificada como *negativa*. Note que pelas definições de breakpoints apresentadas nesta seção, todos os elementos de uma *strip* devem possuir o mesmo sinal.

2.4.2 Breakpoints em Genomas Desbalanceados

Nesta seção, apresentamos definições de *breakpoints* para uma instância clássica de genomas desbalanceados $\mathcal{I} = (A, \iota^n)$.

Definição 2.4.8. Dada uma instância (A, ι^n), obtemos as *versões estendidas* de A e ι^n ao adicionar os elementos $A_0 = +0$, $A_{|A|+1} = +(n+1)$, $\iota^n_0 = +0$ e $\iota^n_{n+1} = +(n+1)$. Caso as strings sejam sem sinais, então apenas desconsideramos os sinais dos novos elementos. Assim como em permutações estendidas, os elementos adicionados não são afetados por rearranjos. Além disso, não incluímos os elementos 0 e $n+1$ nos alfabetos Σ_A e Σ_{ι^n}.

Nas próximas definições, assumimos que A e ι^n estão nas suas versões estendidas.

Definição 2.4.9. Dado um elemento x, seja $\texttt{anterior}(x, \mathcal{I})$ e $\texttt{posterior}(x, \mathcal{I})$ definidos como a seguir.

$$\texttt{anterior}(x, \mathcal{I}) = \begin{cases} \max(\{y \in \Sigma_A \cap \Sigma_{\iota^n} | y < x\}), & \text{if } min(\Sigma_A \cap \Sigma_{\iota^n}) < x \leq n+1 \\ 0, & \text{if } x = min(\Sigma_A \cap \Sigma_{\iota^n}) \\ \alpha, & \text{if } x = \alpha \end{cases}$$

$$\texttt{posterior}(x, \mathcal{I}) = \begin{cases} \min(\{y \in \Sigma_A \cap \Sigma_{\iota^n} | y > x\}), & \text{if } 0 \leq x < max(\Sigma_A \cap \Sigma_{\iota^n}) \\ n+1, & \text{if } x = max(\Sigma_A \cap \Sigma_{\iota^n}) \\ \alpha, & \text{if } x = \alpha \end{cases}$$

Exemplo 2.4.4. Para a string estendida $A = (0\ 4\ 3\ \alpha\ 1\ 2\ 6)$ e considerando ι^n com $n = 5$,

temos os seguintes valores:

$$\begin{aligned}
&\texttt{anterior}(A_1 = 4, \mathcal{I}) = 3, && \texttt{posterior}(A_0 = 0, \mathcal{I}) = 1,\\
&\texttt{anterior}(A_2 = 3, \mathcal{I}) = 2, && \texttt{posterior}(A_1 = 4, \mathcal{I}) = 6,\\
&\texttt{anterior}(A_3 = \alpha, \mathcal{I}) = \alpha, && \texttt{posterior}(A_2 = 3, \mathcal{I}) = 4,\\
&\texttt{anterior}(A_4 = 1, \mathcal{I}) = 0, && \texttt{posterior}(A_3 = \alpha, \mathcal{I}) = \alpha,\\
&\texttt{anterior}(A_5 = 2, \mathcal{I}) = 1, && \texttt{posterior}(A_4 = 1, \mathcal{I}) = 2,\\
&\texttt{anterior}(A_6 = 6, \mathcal{I}) = 4, && \texttt{posterior}(A_5 = 2, \mathcal{I}) = 3.
\end{aligned}$$

Definição 2.4.10. Considerando que A e ι^n são strings sem sinais, para $0 \leq i \leq |A|$, um *breakpoint de reversões sem sinais* existe entre um par de elementos consecutivos (A_i, A_{i+1}) se $A_{i+1} \neq \texttt{posterior}(A_i, \mathcal{I})$ e $A_{i+1} \neq \texttt{anterior}(A_i, \mathcal{I})$.

Exemplo 2.4.5. Para $\mathcal{I} = (A, \iota^n)$ tal que $A = (0\ 4\ 3\ \alpha\ 1\ 2\ 6\ 7\ 10)$ e $n = 9$, temos os seguintes *breakpoints* de reversões sem sinais (representados pelo símbolo $\circ$):

$$(0\ \circ\ 4\ 3\ \circ\ \alpha\ \circ\ 1\ 2\ \circ\ 6\ 7\ 10).$$

Definição 2.4.11. Considerando que A e ι^n são strings sem sinais, para $0 \leq i \leq |A|$, um *breakpoint de transposição* existe entre um par de elementos consecutivos (A_i, A_{i+1}) se $A_{i+1} \neq \texttt{posterior}(A_i, \mathcal{I})$.

Exemplo 2.4.6. Para $\mathcal{I} = (A, \iota^n)$ tal que $A = (0\ 4\ 3\ \alpha\ 1\ 2\ 6\ 7\ 10)$ e $n = 9$, temos os seguintes *breakpoints* de transposição (representados pelo símbolo $\circ$):

$$(0\ \circ\ 4\ \circ\ 3\ \circ\ \alpha\ \circ\ 1\ 2\ \circ\ 6\ 7\ 10).$$

As definições de *strips* são análogas às definições apresentadas para *breakpoints* em permutações. No entanto, existe o seguinte caso adicional na classificação de *strips*: uma *strip* formada apenas de elementos iguais a α é considerada uma *strip* crescente.

Definição 2.4.12. Dado um modelo $\mathcal{M}$, $b_{\mathcal{M}}(A, \iota^n)$ (ou $b_{\mathcal{M}}(\mathcal{I})$) denota o número de *breakpoints* para a instância $\mathcal{I} = (A, \iota^n)$.

Definição 2.4.13. Dado um modelo $\mathcal{M}$ e um rearranjo (ou sequência de rearranjos) β, $\Delta b_{\mathcal{M}}(\mathcal{I}, \beta) = \Delta b_{\mathcal{M}}(A, \iota^n, \beta) = b_{\mathcal{M}}(A, \iota^n) - b_{\mathcal{M}}(A \cdot \beta, \iota^n)$ denota a variação no número de *breakpoints* após β ser aplicado em A.

2.4.3 Breakpoints Intergênicos

O conceito de *breakpoints*, apresentado na Seção 2.4.2, também é válido para uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$. No entanto, os *breakpoints* consideram apenas os elementos das strings A e ι^n, ignorando os tamanhos das regiões intergênicas. Dessa forma, apresentamos os *breakpoints* intergênicos, que consideram tanto os elementos das strings quanto os tamanhos das regiões intergênicas de pares de elementos adjacentes. Para *breakpoints* intergênicos, também consideramos que as strings A e ι^n estão nas suas versões estendidas.

Definição 2.4.14. Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, e um modelo de rearranjo $\mathcal{M}$, dizemos que (A_i, A_{i+1}) é um *breakpoint intergênico*, para $0 \leq i \leq |A|$, se uma dessas duas condições é verdadeira: (A_i, A_{i+1}) é um *breakpoint*; ou (A_i, A_{i+1}) não é um *breakpoint*, $A_i \neq \alpha$, e $\breve{A}_{i+1} \neq \breve{\iota}^n_j$, onde $\breve{\iota}^n_j$ é a região intergênica do genoma de destino $\mathcal{G}_d$ entre os dois elementos de ι^n que correspondem aos elementos A_i e A_{i+1}.

Assim como na Seção 2.4.2, consideramos que as strings A e ι^n estão nas suas versões estendidas. A partir do modelo $\mathcal{M}$, podemos inferir qual a definição de *breakpoints* que será considerada (*breakpoints* de reversões sem sinais ou *breakpoints* de transposições).

Definição 2.4.15. Seja (A_i, A_{i+1}) um *breakpoint* intergênico, tal que (A_i, A_{i+1}) não é um *breakpoint* e $\breve{\iota}^n_j$ é a região intergênica com $j = \max(A_i, A_{i+1})$, ou seja, $\breve{\iota}^n_j$ é a região intergênica do genoma de destino $\mathcal{G}_d$ que está entre os dois elementos de ι^n que correspondem aos elementos A_i e A_{i+1}. Dizemos que o *breakpoint* intergênico (A_i, A_{i+1}) é *sobrecarregado* se $\breve{A}_{i+1} > \breve{\iota}^n_j$. Caso contrário, (A_i, A_{i+1}) é *subcarregado*.

Exemplo 2.4.7. Considere $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, onde $n = 6$, A é a string estendida (0 5 α 3 2 1 6 7), $\breve{A} = (2, 15, 10, 8, 10, 18, 15)$, e $\breve{\iota}^n = (2, 10, 15, 9, 11, 14, 10)$. Os valores entre parênteses representam os tamanhos das regiões intergênicas.

$$\mathcal{G}_o = (0\ (2)\ 5\ (15)\ \alpha\ (10)\ 3\ (8)\ 2\ (10)\ 1\ (18)\ 6\ (15)\ 7)$$
$$\mathcal{G}_d = (0\ (2)\ 1\ (10)\ 2\ (15)\ 3\ (9)\ 4\ (11)\ 5\ (14)\ 6\ (10)\ 7)$$

Considerando *breakpoints* de reversão sem sinais, temos os seguintes *breakpoints* intergênicos: $(0, 5)$, $(5, \alpha)$, $(\alpha, 3)$, $(3, 2)$, $(1, 6)$, e $(6, 7)$. O *breakpoint* intergênico $(3, 2)$ é subcarregado e o *breakpoint* $(6, 7)$ é sobrecarregado.

Definição 2.4.16. Para um modelo $\mathcal{M}$, o número de *breakpoints* intergênicos é denotado por $bi_{\mathcal{M}}(\mathcal{I}^{ig}) = bi_{\mathcal{M}}(\mathcal{G}_o, \mathcal{G}_d)$. Dado um rearranjo (ou sequência de operações) β, denotamos por $\Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \beta) = \Delta bi_{\mathcal{M}}(\mathcal{G}_o, \mathcal{G}_d, \beta) = bi_{\mathcal{M}}(\mathcal{G}_o, \mathcal{G}_d) - bi_{\mathcal{M}}(\mathcal{G}_o \cdot \beta, \mathcal{G}_d)$ a variação no número de *breakpoints* intergênicos causada por β.

As definições de *strips* são análogas às usadas nas seções anteriores.

2.5 Grafo de Ciclos

A estrutura grafo de ciclos, também chamada de grafo de *breakpoints*, foi inicialmente apresentada para o problema da Ordenação de Permutações com Sinais por Reversões [16]. O grafo de ciclos é utilizado para representar uma instância de um problema de Distância de Rearranjos usando uma única estrutura matemática. No entanto, em sua forma original, o grafo de ciclos pode ser usado apenas em permutações. Como parte deste trabalho, nós propomos duas adaptações chamadas de *grafo de ciclos rotulado* e *grafo de ciclos ponderado e rotulado*. O grafo de ciclos rotulado é usado para uma instância clássica de genomas balanceados ou desbalanceados, enquanto o grafo de ciclos ponderado e rotulado é usado para uma instância intergênica de genomas balanceados ou desbalanceados. A seguir, apresentamos esses três tipos de grafos.

2.5.1 Grafo de Ciclos para Permutações

Para as próximas definições, consideramos que permutações estão na sua forma estendida. O *grafo de ciclos* de uma permutação π é o grafo não direcionado $G(\pi) = (V, E_o \cup E_d)$, onde $V = \{+\pi_0, -\pi_1, +\pi_1, -\pi_2, +\pi_2, \ldots, -\pi_n, +\pi_n, -\pi_{n+1}\}$ é o conjunto de vértices, $E_o = \{(-\pi_i, +\pi_{i-1}) \mid 1 \leq i \leq n+1\}$ é o conjunto de *arestas de origem*, e $E_d = \{(-\iota_i^n, +\iota_{i-1}^n) \mid 1 \leq i \leq n+1\}$ é o conjunto de *arestas de destino*.

Arestas de origem conectam vértices correspondentes a elementos adjacentes na permutação π, enquanto arestas de destino conectam vértices correspondentes a elementos adjacentes na permutação ι^n. Essas arestas também são chamadas de arestas pretas (origem) e arestas cinzas (destino) na literatura de rearranjos de genomas [46].

Em um *ciclo alternante*, arestas com uma extremidade em comum (arestas adjacentes) possuem tipos distintos (aresta de origem e aresta de destino). Todo vértice de $G(\pi)$ é extremidade de exatamente uma aresta de origem e uma aresta de destino. Dessa forma, existe uma decomposição única do grafo $G(\pi)$ em ciclos alternantes [16]. É importante ressaltar que essas definições de grafo de ciclos funcionam para todos os modelos estudados considerando permutações com sinais. No entanto, no caso de permutações sem sinais, as propriedades do grafo de ciclos apresentadas nesta seção só valem para os modelos que utilizam *breakpoints* de transposição (Definição 2.4.3). Por exemplo, as propriedades apresentadas nesta seção não são válidas para reversões em permutações sem sinais.

Daqui em diante, dizemos que a aresta de origem $(-\pi_i, +\pi_{i-1})$ tem índice i e é denotada por e_i. Além disso, a aresta de destino $(-\iota_i^n, +\iota_{i-1}^n)$ tem índice i e é denotada por e_i'. Normalmente, nos referimos a uma aresta de origem apenas pelo seu índice.

Utilizamos a seguinte convenção presente na literatura para o desenho de $G(\pi)$: desenhamos os vértices em uma linha horizontal na ordem em que esses elementos aparecem em π, sempre colocando o vértice $-\pi_i$ à esquerda do vértice $+\pi_i$, ou seja, desenhamos os vértices da esquerda para a direita seguindo a ordem da sequência $(+\pi_0, -\pi_1, +\pi_1, -\pi_2, +\pi_2, \ldots, -\pi_n, +\pi_n, -\pi_{n+1})$; desenhamos arestas de origem como linhas horizontais; e desenhamos arestas de destino como arcos. Devido ao posicionamento dos vértices de $G(\pi)$, as arestas de origem são dispostas da esquerda para direita de forma que os seus índices formam a sequência $1, 2, \ldots, n, n+1$.

Um m-ciclo é um ciclo com m arestas de origem e m arestas de destino. Além disso, dizemos que um m-ciclo possui *tamanho* m. Um 1-ciclo também é chamado de ciclo *unitário* ou *trivial*. Uma permutação π é uma *permutação simples* se todos os ciclos em $G(\pi)$ possuem tamanho menor ou igual a 3. Também classificamos um m-ciclo de acordo com a paridade de m, sendo que um m-ciclo é par, quando m é par, ou ímpar, quando m é ímpar.

Representamos um m-ciclo C usando a lista de índices das suas arestas, sendo que essa lista é construída percorrendo as arestas do ciclo C, iniciando pelo vértice mais à direita de C e percorrendo a aresta de origem incidente a esse vértice: $C = (o_1, d_1, o_2, d_2, \ldots, o_m, d_m)$, tal que $o_1 > o_j$, para todo $1 < j \leq m$. Na maioria dos casos, usamos uma representação simplificada de C que possui apenas os índices das arestas de origem, já que as arestas de destino podem ser inferidas a partir dessa informação.

As figuras 2.3 e 2.4 mostram exemplos de grafos de ciclos para uma permutação sem

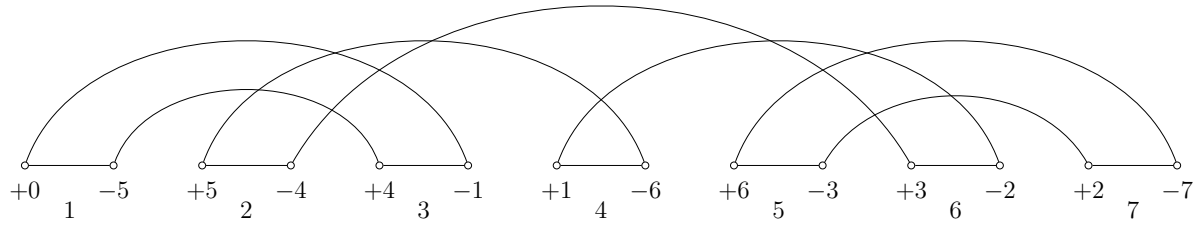


Figura 2.3: Grafo de ciclos $G(\pi)$ da permutação sem sinais $\pi = (5\ 4\ 1\ 6\ 3\ 2)$. Linhas horizontais e arcos representam arestas de origem e arestas de destino, respectivamente. O índice de uma aresta de origem é indicado por um número abaixo dessa aresta. Neste exemplo, temos três ciclos em $G(\pi)$: $C_1 = (3, 1)$, $C_2 = (6, 2, 4)$ e $C_3 = (7, 5)$. O ciclo C_2 é ímpar e os ciclos C_1 e C_3 são pares.

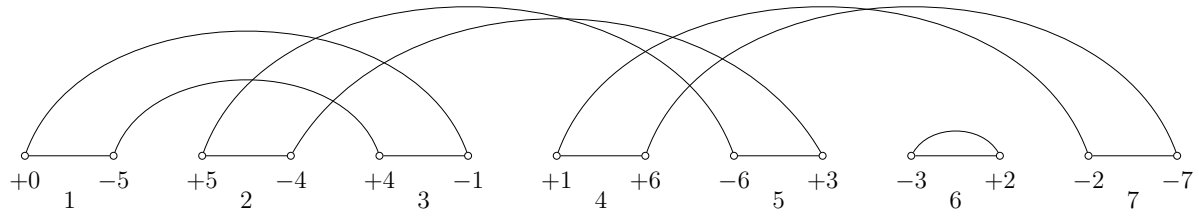


Figura 2.4: Grafo de ciclos $G(\pi)$ da permutação com sinais $\pi = (+5\ +4\ +1\ -6\ -3\ -2)$. Neste exemplo, temos 4 ciclos em $G(\pi)$: $C_1 = (3, 1)$, $C_2 = (5, 2)$, $C_3 = (6)$ e $C_4 = (7, 4)$.

sinais e uma permutação com sinais, respectivamente. Também exemplificamos algumas notações utilizadas nessas figuras.

O número de ciclos e o número de ciclos ímpares em $G(\pi)$ são denotados por $c(\pi)$ e $c_{odd}(\pi)$, respectivamente. Para um rearranjo (ou sequência de rearranjos) β, usamos $\Delta c(\pi, \beta)$ para denotar a variação no número de ciclos causado pelo rearranjo β, ou seja, $\Delta c(\pi, \beta) = c(\pi \cdot \beta) - c(\pi)$. De forma similar, temos $\Delta c_{odd}(\pi, \beta) = c_{odd}(\pi \cdot \beta) - c_{odd}(\pi)$.

Dado um m-ciclo $C = (o_1, o_2, \ldots, o_m)$, classificamos C como *não orientado* se $o_1 > o_2 > \ldots > o_m$. Caso contrário, classificamos C como *orientado*. Na Figura 2.3, o ciclo $C_2 = (6, 2, 4)$ é orientado enquanto os ciclos $C_1 = (3, 1)$ e $C_3 = (7, 5)$ são não orientados.

Dizemos que três arestas de origem com índices o_i, o_j e o_k, com $i < j < k$, que pertencem ao mesmo ciclo C formam uma *tripla orientada* se pelo menos uma das condições seguintes é verdadeira: $o_i > o_k > o_j$; $o_j > o_i > o_k$, ou $o_k > o_j > o_i$. Bafna e Pevzner [17] provaram que todo ciclo orientado possui uma tripla orientada o_i, o_j e o_k, com $i < j < k$, tal que $o_i > o_k > o_j$ e $k = j + 1$. Além disso, eles demonstraram que uma transposição τ tem $\Delta c(\pi, \tau) = 2$ se, e somente se, essa transposição é aplicada em três arestas de origem que formam uma tripla orientada.

Dado um m-ciclo $C = (o_1, o_2, \ldots, o_m)$, uma aresta de origem e_{o_i} é dita *convergente* se ela é percorrida da direita para a esquerda. Caso contrário, e_{o_i} é dita *divergente*. Note que a aresta e_{o_1} é sempre convergente seguindo a convenção de como o ciclo é percorrido quando listando os índices. Um par de arestas de origem (o_i, o_j) é divergente se uma das arestas é convergente e a outra é divergente. Um ciclo C é *divergente* se pelo menos uma aresta de C é divergente. Se todas arestas de C são convergentes, então C é *convergente*.

Um grafo de ciclos $G(\pi)$ possui arestas divergentes se, e somente se, existe algum elemento com sinal "$-$" em π. Portanto, para permutações sem sinais, todos os ciclos de $G(\pi)$ são convergentes.

Agora, apresentamos algumas definições que serão usadas apenas em modelos que contém transposições.

Dizemos que uma transposição τ é uma m-transposição se $\Delta c_{odd}(\pi, \tau) = m$. Por exemplo, uma 2-transposição é uma operação que aumenta o número de ciclos ímpares em 2. Uma $(2, 2)$-sequência é um par de 2-transposições que podem ser aplicadas consecutivamente em π, ou seja, (τ, τ') é uma $(2, 2)$-sequência se $\Delta c_{odd}(\pi, \tau) = \Delta c_{odd}(\pi \cdot \tau, \tau') = 2$.

Dizemos que uma tripla orientada (o_i, o_j, o_k), com $i < j < k$, é *válida* se a transposição τ que age nessas arestas é uma 2-transposição. Ou seja, uma transposição aplicada em uma tripla orientada válida aumenta tanto o número de ciclos quanto o número de ciclos ímpares em 2 unidades.

2.5.2 Grafo de Ciclos Rotulado

Nesta seção, apresentamos uma adaptação do grafo de ciclos que torna possível o seu uso em instâncias clássicas de genomas desbalanceados.

Definição 2.5.1. Dada uma instância clássica $\mathcal{I} = (A, \iota^n)$, definimos as *strings simplificadas* $\pi^A = (\pi_1^A \ \ldots \ \pi_{n'}^A)$ e $\pi^\iota = (\pi_1^\iota \ \ldots \ \pi_{n'}^\iota)$ como cópias de A e ι^n, respectivamente, mas removendo elementos que não pertencem ao conjunto $\Sigma_A \cap \Sigma_{\iota^n}$.

Exemplo 2.5.1. Considerando a instância $\mathcal{I} = (A, \iota^n)$, com $A = (4 \ \alpha \ 2 \ 5 \ \alpha \ 1)$ e $\iota^n = (1 \ 2 \ 3 \ 4 \ 5)$, temos $\pi^A = (4 \ 2 \ 5 \ 1)$ e $\pi^\iota = (1 \ 2 \ 4 \ 5)$, com $n' = 4$.

Para as próximas definições, consideramos uma instância clássica $\mathcal{I} = (A, \iota^n)$ e assumimos que as strings simplificadas π^A e π^ι estão nas suas versões estendidas. Além disso, consideramos que $|\pi^A| = |\pi^\iota| = n'$ é a quantidade de elementos das strings simplificadas desconsiderando os elementos da versão estendida (0 e $n + 1$), ou seja, $\pi^A = (0 \ \pi_1^A \ \pi_2^A \ \ldots \ \pi_{n'}^A \ (n + 1))$ e $\pi^\iota = (0 \ \pi_1^\iota \ \pi_2^\iota \ \ldots \ \pi_{n'}^\iota \ (n + 1))$.

O *grafo de ciclos rotulado* é o grafo não direcionado $G(\mathcal{I}) = G(A, \iota^n) = (V, E_o \cup E_d, \ell)$, onde $V = \{+\pi_0^A, -\pi_1^A, +\pi_1^A, -\pi_2^A, +\pi_2^A, \ldots, -\pi_{n'}^A, +\pi_{n'}^A, -\pi_{n'+1}^A\}$ é o conjunto de vértices, E_o é o conjunto de *arestas de origem*, E_d é o conjunto de *arestas de destino*, e $\ell : (E_o \cup E_d) \to (\Sigma_{\iota^n} \setminus \Sigma_A) \cup \{\alpha, \emptyset\}$ é uma função que atribui um rótulo a cada aresta do grafo.

Arestas de origem conectam vértices que correspondem a elementos adjacentes em π^A, enquanto arestas de destino conectam vértices que correspondem a elementos adjacentes em π^ι. O conjunto de arestas de origem é definido como $E_o = \{e_i = (+\pi_{i-1}^A, -\pi_i^A) : 1 \leq i \leq n' + 1\}$, sendo que a aresta de origem $e_i = (+\pi_{i-1}^A, -\pi_i^A)$ possui índice i. O rótulo $\ell(e_i) = \emptyset$ se π_{i-1}^A e π_i^A são consecutivos em A. Caso contrário, temos que $\ell(e_i) = \alpha$. O conjunto de arestas de destino é definido como $E_d = \{e'_i = (+\pi_{i-1}^\iota, -\pi_i^\iota) : 1 \leq i \leq n' + 1\}$, sendo que a aresta de destino $e'_i = (+\pi_{i-1}^\iota, -\pi_i^\iota)$ possui índice i. O rótulo $\ell(e'_i) = \emptyset$ se π_{i-1}^ι e π_i^ι são consecutivos em ι^n. Caso contrário, temos o rótulo $\ell(e'_i) = \pi_{i-1}^\iota + 1$.

Assim como o grafo de ciclos para permutações, no grafo de ciclos rotulado $G(\mathcal{I})$ cada vértice de $G(\mathcal{I})$ é extremidade de exatamente uma aresta de origem e uma aresta de destino. Portanto, existe uma decomposição única do grafo $G(\mathcal{I})$ em ciclos alternantes.

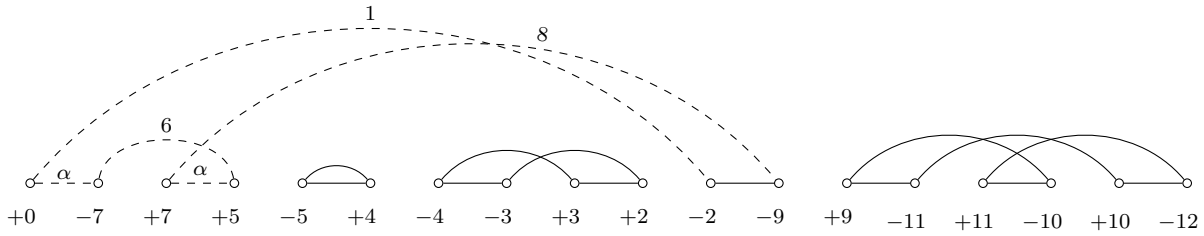


Figura 2.5: Grafo de ciclos rotulado $G(\mathcal{I}) = G(A, \iota^n)$ para as strings ι^n, com $n = 11$, e $A = (\alpha\ {+7}\ \alpha\ {-5}\ {-4}\ {+3}\ {-2}\ {+9}\ {+11}\ {+10})$. Existem quatro ciclos nesse grafo. O ciclo $C_1 = (6, 1, 2)$ é um ciclo rotulado divergente. Todos os outros ciclos são ciclos limpos. O ciclo $C_2 = (3)$ é um ciclo unitário, o ciclo $C_3 = (5, 4)$ é um ciclo divergente, e o ciclo $C_4 = (9, 7, 8)$ é um ciclo orientado.

Todas as notações e definições relacionadas a ciclos para um grafo de ciclos (Seção 2.5.1) se estendem para um grafo de ciclos rotulado.

Para uma aresta de destino ou origem e, dizemos que essa aresta é *limpa* se o seu rótulo é vazio (i.e., $\ell(e) = \emptyset$). Caso contrário, o seu rótulo é não vazio (i.e., $\ell(e) \neq \emptyset$) e dizemos que a aresta e é *rotulada*.

A forma como desenhamos o grafo é similar ao grafo de ciclos para permutações, sendo que representamos arestas rotuladas como linhas tracejadas e colocamos os rótulos correspondentes acima das arestas. Essa representação visual é apresentada na Figura 2.5.

Um ciclo C é *limpo* se todas as suas arestas de origem são limpas, caso contrário, dizemos que C é *rotulado*.

Definição 2.5.2. O número de ciclos limpos em $G(\mathcal{I})$ é denotado por $c_{\text{clean}}(A, \iota^n)$ ou $c_{\text{clean}}(\mathcal{I})$.

Definição 2.5.3. O número de ciclos rotulados em $G(\mathcal{I})$ é denotado por $c_{\text{labeled}}(A, \iota^n)$ ou $c_{\text{labeled}}(\mathcal{I})$.

Note que o grafo $G(A, \iota^n)$ possui $n + 1$ ciclos unitários limpos se, e somente se, $A = \iota^n$. Como o número de arestas de origem pode ser alterado por inserções, precisamos de uma nova definição para $\Delta c(\mathcal{I}, \beta)$, que denota a variação na quantidade de ciclos relativo à quantidade de arestas de origem.

Definição 2.5.4. Dada uma operação (ou uma sequência de operações) β, definimos $\Delta c(\mathcal{I}, \beta) = \Delta c(A, \iota^n, \beta) = (|\pi^A| + 1 - c(A, \iota^n)) - (|\pi^A \cdot \beta| + 1 - c(A \cdot \beta, \iota^n))$.

Definição 2.5.5. Dada uma operação (ou uma sequência de operações) β, definimos $\Delta c_{\text{clean}}(\mathcal{I}, \beta) = \Delta c_{\text{clean}}(A, \iota^n, \beta) = (|\pi^A| + 1 - c_{\text{clean}}(A, \iota^n)) - (|\pi^A \cdot \beta| + 1 - c_{\text{clean}}(A \cdot \beta, \iota^n))$.

2.5.3 Grafo de Ciclos Rotulado e Ponderado

Nesta seção, apresentamos uma adaptação do grafo de ciclos que torna possível o seu uso em instâncias intergênicas de genomas desbalanceados.

Para as próximas definições, consideramos uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, onde $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$. Além disso, assumimos que as strings simplificadas π^A e

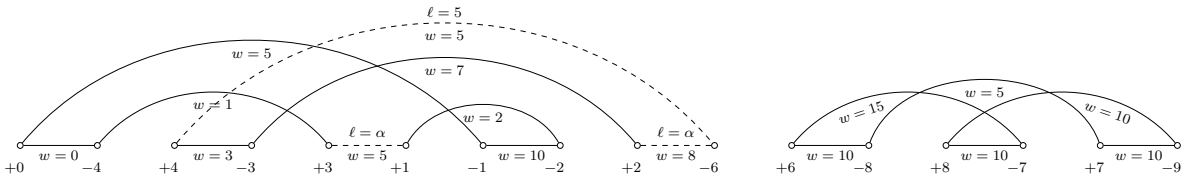


Figura 2.6: Grafo de ciclos rotulado e ponderado para $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, onde $n = 8$, $A = (+0\ +4\ +3\ \alpha\ -1\ +2\ \alpha\ +6\ +8\ +7\ +9)$, $\breve{A} = (0, 3, 2, 3, 10, 2, 6, 10, 10, 10)$, $\iota^n = (+0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9)$ e $\breve{\iota}^n = (5, 2, 7, 1, 2, 3, 15, 5, 10)$. Neste exemplo temos os ciclos $C_1 = (4, 1, 3)$, $C_2 = (5, 2)$, $C_3 = (8, 6, 7)$. O ciclo C_1 é um ciclo divergente, rotulado e desbalanceado (a soma dos custos das arestas de origem é igual a 15, enquanto a soma dos custos das arestas de destino é igual a 8). O ciclo C_2 é um ciclo convergente, rotulado e desbalanceado. O ciclo C_3 é um ciclo convergente, balanceado e limpo.

π^ι estão nas suas versões estendidas. Consideramos que $|\pi^A| = |\pi^\iota| = n'$ é a quantidade de elementos das strings simplificadas desconsiderando os elementos da versão estendida (0 e $n+1$), ou seja, $\pi^A = (0\ \pi_1^A\ \pi_2^A\ \ldots\ \pi_{n'}^A\ (n+1))$ e $\pi^\iota = (0\ \pi_1^\iota\ \pi_2^\iota\ \ldots\ \pi_{n'}^\iota\ (n+1))$.

O *grafo de ciclos rotulado e ponderado* é o grafo não direcionado $G(\mathcal{I}^{ig}) = G(\mathcal{G}_o, \mathcal{G}_d) = (V, E_o \cup E_d, \ell, w)$, onde $V = \{+\pi_0^A, -\pi_1^A, +\pi_1^A, -\pi_2^A, +\pi_2^A, \ldots, -\pi_{n'}^A, +\pi_{n'}^A, -\pi_{n'+1}^A\}$ é o conjunto de vértices, E_o é o conjunto de *arestas de origem*, E_d é o conjunto de *arestas de destino*, $\ell : (E_o \cup E_d) \to (\Sigma_{\iota^n} \setminus \Sigma_A) \cup \{\alpha, \emptyset\}$ é uma função que atribui um rótulo a cada aresta do grafo, e $w : (E_o \cup E_d) \to \mathbb{Z}^*$ é uma função de custo que relaciona arestas a tamanhos de regiões intergênicas.

O conjunto de arestas de origem E_o, o conjunto de arestas de destino E_d e a função ℓ são definidos de forma análoga ao grafo de ciclos rotulado $G(A, \iota^n)$. Além disso, todas as definições usadas para o grafo de ciclos rotulado $G(A, \iota^n)$ são válidas para o grafo de ciclos rotulado e ponderado $G(\mathcal{I}^{ig})$, exceto pelas definições de ciclos limpos e rotulados.

Definição 2.5.6. Um ciclo C é *limpo* se todas as suas arestas de origem e todas as suas arestas de destino são limpas, caso contrário, dizemos que C é *rotulado*.

Definição 2.5.7. O número de ciclos limpos em $G(\mathcal{I}^{ig})$ é denotado por $c_{\text{clean}}(\mathcal{I}^{ig})$ e o número de ciclos rotulados em $G(\mathcal{I}^{ig})$ é denotado por $c_{\text{labeled}}(\mathcal{I}^{ig})$.

A seguir, definimos a função de custo w e as definições adicionais relacionadas ao custo das arestas do grafo. Para $1 \leq i \leq n'+1$, uma aresta de origem $e_i = (+\pi_{i-1}^A, -\pi_i^A)$ possui custo $w(e_i) = \sum_{k=i'+1}^{j'} \breve{A}_k$, onde $A_{i'} = \pi_{i-1}^A$ e $A_{j'} = \pi_i^A$, ou seja, o custo de e_i é igual à soma dos tamanhos das regiões intergênicas entre os elementos π_{i-1}^A e π_i^A em $\mathcal{G}_o$. Para $1 \leq i \leq n'+1$, a aresta de destino $e'_i = (+\pi_{i-1}^\iota, -\pi_i^\iota)$ possui custo $w(e'_i) = \sum_{k=i'+1}^{j'} \breve{\iota}_k^n$, onde $i' = \pi_{i-1}^\iota$ e $j' = \pi_i^\iota$, ou seja, o custo de e'_i é igual à soma dos tamanhos das regiões intergênicas entre os elementos π_{i-1}^ι e π_i^ι em $\mathcal{G}_d$.

Definição 2.5.8. Um ciclo $C = (o_1, d_1, o_2, d_2, \ldots, o_m, d_m)$ é *balanceado* se a soma dos custos das arestas de destino é igual à soma dos custos das arestas de origem, ou seja, $\sum_{i=1}^m w(e'_{d_i}) = \sum_{i=1}^m w(e_{o_i})$. Caso contrário, o ciclo C é *desbalanceado*.

Um ciclo desbalanceado é classificado em *positivo* ou *negativo*. Um ciclo positivo $C = (o_1, d_1, o_2, d_2, \ldots, o_m, d_m)$ é um ciclo tal que a soma dos custos das arestas de destino é maior que a soma dos custos das arestas de origem, ou seja, $\sum_{i=1}^{m} w(e'_{d_i}) > \sum_{i=1}^{m} w(e_{o_i})$. Um ciclo negativo $C = (o_1, d_1, o_2, d_2, \ldots, o_m, d_m)$ é um ciclo tal que a soma dos custos das arestas de destino é menor que a soma dos custos das arestas de origem, ou seja, $\sum_{i=1}^{m} w(e'_{d_i}) < \sum_{i=1}^{m} w(e_{o_i})$. A Figura 2.6 mostra um exemplo de um grafo de ciclos rotulado e ponderado.

Definição 2.5.9. Dizemos que um ciclo C é *bom* se ele é balanceado e limpo. Caso contrário, C é desbalanceado ou rotulado, e dizemos que C é um ciclo *ruim*. O número de ciclos bons em $G(\mathcal{I}^{ig})$ é denotado por $c_g(\mathcal{G}_o, \mathcal{G}_d)$ ou $c_g(\mathcal{I}^{ig})$.

Note que o grafo $G(\mathcal{I}^{ig})$ possui apenas ciclos unitários bons se, e somente se, $\mathcal{G}_o = \mathcal{G}_d$. Em outra palavras, $\mathcal{G}_o = \mathcal{G}_d$ se, e somente se, $|\pi^A| + 1 - c_g(\mathcal{G}_o, \mathcal{G}_d) = 0$.

Definição 2.5.10. Dada uma operação (ou uma sequência de operações) β, a variação na quantidade de arestas de origem menos a quantidade de ciclos bons do grafo é denotada por $\Delta c_g(\mathcal{I}^{ig}, \beta) = \Delta c_g(\mathcal{G}_o, \mathcal{G}_d, \beta) = (|\pi^A| + 1 - c_g(\mathcal{G}_o, \mathcal{G}_d)) - (|\pi^{A'}| + 1 - c_g(\mathcal{G}'_o, \mathcal{G}_d))$, onde $\mathcal{G}_o \cdot \beta = \mathcal{G}'_o = (A', \breve{A}')$.

Capítulo 3

Ordenação de Permutações por Transposições e Outros Rearranjos

Muitos algoritmos de aproximação para problemas de distância de rearranjos foram desenvolvidos mesmo sendo que as complexidades desses problemas ainda estavam em aberto [46].

Em 1995, Kececioglu e Sankoff [54] apresentaram uma 2-aproximação para o problema da Ordenação de Permutações sem Sinais por Reversões. Esses autores também apresentaram algoritmos exatos, porém esses algoritmos não possuem complexidade polinomial. Apenas em 1999, Caprara [36] provou que esse problema é NP-difícil. Até o momento, o melhor algoritmo de aproximação conhecido para esse problema possui fator de aproximação de 1.375 [19].

Em 1996, Bafna e Pevzner [16] também estudaram reversões, mas consideraram que a orientação dos genes era conhecida. Os autores apresentaram uma 1.5-aproximação para a Ordenação de Permutações com Sinais por Reversões. Em 1999, Hannenhalli e Pevzner [51] apresentaram um algoritmo polinomial exato para esse problema, sendo esse um dos trabalhos mais importantes na área de genômica comparativa.

Em 1995, Bafna e Pevzner [15] apresentaram uma 1.5-aproximação para a Ordenação de Permutações (sem Sinais) por Transposições. Após isso, em 2006, Elias e Hartman [43] conseguiram melhorar o fator de aproximação, apresentando um algoritmo com fator de aproximação igual a 1.375. Apenas em 2012, Bulteau e coautores [34] apresentaram uma prova de que o problema é NP-difícil.

A 1.375-aproximação de Elias e Hartman [43] tem complexidade de tempo quadrática e depende de um processo que transforma a permutação de entrada em uma permutação simples. Após essa transformação, o próximo passo do algoritmo é aplicar uma $(2, 2)$-sequência na nova permutação para garantir o fator de aproximação nos passos seguintes. No entanto, um estudo recente [74] mostrou que essa busca deve ser feita antes do processo de simplificação da permutação, pois existem casos em que há uma $(2, 2)$-sequência para a permutação de entrada π, mas não há uma $(2, 2)$-sequência para a permutação simplificada gerada a partir de π pelo algoritmo de Elias e Hartman [43]. Portanto, em alguns casos, o algoritmo de Elias e Hartman [43] falha em garantir a 1.375-aproximação.

Silva e coautores [74] apresentaram um novo algoritmo para a Ordenação de Permutações por Transposições que garante o fator de aproximação de 1.375 em todos os casos

usando uma abordagem algébrica. Esse algoritmo usa uma busca exaustiva para achar a $(2,2)$-sequência inicial e possui complexidade de tempo de $O(n^6)$. O algoritmo de Elias e Hartman [43] é quadrático porque faz a busca da $(2,2)$-sequência inicial, caso alguma exista, na permutação simplificada, que possui características que tornam esse procedimento mais simples. No entanto, para uma permutação qualquer, o melhor resultado conhecido até então para achar uma $(2,2)$-sequência, caso alguma exista, tem complexidade de tempo de $O(n^6)$ [74].

Na Seção 3.1 deste capítulo, apresentamos um algoritmo com complexidade de $O(n^5)$ para achar uma $(2,2)$-sequência, caso alguma exista, em uma permutação qualquer. Esse procedimento é essencial para obter um fator de aproximação igual a 1.375 para a Ordenação de Permutações por Transposições. A partir dessa melhoria, nós propomos uma nova versão do algoritmo de 1.375-aproximação com complexidade de tempo de $O(n^5)$.

Outro problema muito estudado na literatura é o da Ordenação de Permutações (com ou sem Sinais) por Reversões e Transposições. Em 1998, Walter e coautores [80] apresentaram uma 2-aproximação para a versão com sinais do problema. Após isso, Rahman e coautores [71] mostraram outro algoritmo que possui fator de aproximação igual a 2, para permutações com sinais, e mostraram uma $2k$-aproximação, para permutações sem sinais, onde k é o fator de aproximação para o problema da Decomposição Máxima de Ciclos Alternantes em um grafo de *breakpoints*. O melhor fator de aproximação conhecido para o problema da Decomposição Máxima de Ciclos Alternantes é de $17/12 + \epsilon$ [37], para qualquer ϵ positivo.

Em 2019, Oliveira e coautores [64] provaram que a Ordenação de Permutações com ou sem Sinais por Reversões e Transposições é NP-difícil. Além da abordagem não ponderada, Oliveira e coautores [64] também estudaram a versão ponderada do problema onde uma reversão tem custo w_ρ e uma transposição tem custo w_τ. Os autores provaram que a Ordenação de Permutações (com ou sem Sinais) por Reversões e Transposições Ponderadas é NP-difícil para quaisquer valores de w_ρ e w_τ tal que $w_\tau/w_\rho \leq 1.5$.

Além dos modelos mais estudados de reversões, transposições e a combinação de ambas operações, também foram propostos algoritmos de aproximação e exatos para modelos que possuem transposições inversas e revrevs, apesar da complexidade do problema de distância de rearranjos com esses modelos ser desconhecida [46, 53]. Gu e coautores [50] criaram uma 2-aproximação para o problema da Ordenação de Permutações com Sinais por Reversões, Transposições e Transposições Inversas. Lou e Zhu [57] desenvolveram um algoritmo com fator de aproximação de 2.25 para o mesmo modelo de rearranjos, mas considerando permutações sem sinais.

Lin e Xue [56] estudaram o problema da Ordenação de Permutações com Sinais por Reversões, Transposições, Transposições Inversas e Revrevs e desenvolveram uma 1.75-aproximação para esse problema. Já para o problema da Ordenação de Permutações com Sinais por Transposições, Transposições Inversas e Revrevs, o melhor resultado é um algoritmo com fator de aproximação de 1.5 [52].

Para a abordagem ponderada, neste trabalho, consideramos uma função de custo baseada no tipo de rearranjo. O custo de uma reversão é indicado por w_ρ, enquanto os custos de uma transposição, transposição inversa ou revrev são indicados simplesmente por w_τ. Utilizamos o mesmo custo w_τ para essas últimas três operações devido ao fato de

que todas elas afetam três adjacências do genoma, enquanto as reversões afetam apenas duas adjacências.

As reversões são os rearranjos mais observados em muitos cenários evolutivos [22]. Por esse motivo, o peso de uma reversão (w_ρ) tende a ser menor do que o peso de uma transposição ou rearranjos similares (w_τ). Bader e coautores [14] estudaram o problema da Ordenação de Permutações com Sinais por Reversões, Transposições e Transposições Inversas Ponderadas e apresentaram uma 1.5-aproximação para valores de w_ρ e w_τ tal que $1 \leq w_\tau/w_\rho \leq 2$. Eriksen [44] já havia considerado o mesmo problema e apresentado uma 7/6-aproximação, mas esse fator de aproximação só é garantido quando $w_\tau/w_\rho = 2$.

Na Seção 3.2 deste capítulo, considerando que $w_\tau/w_\rho \leq 1.5$, provamos que os problemas de Ordenação de Permutações com ou sem Sinais por Rearranjos são NP-difíceis para modelos que possuem transposições juntamente com um ou mais dos seguintes rearranjos: reversões, transposições inversas e revrevs.

3.1 Uma 1.375-Aproximação Mais Eficiente para Transposições

Nesta seção, apresentamos um novo algoritmo de 1.375-aproximação para a Ordenação de Permutações por Transposições com complexidade de tempo de $O(n^5)$.

A 1.375-aproximação de Elias e Hartman [43] tem complexidade de tempo quadrática, mas possui um erro em um dos passos do algoritmo, a busca de uma $(2,2)$-sequência na permutação inicial, que faz com que o algoritmo não garanta o fator de aproximação de 1.375 para todas as instâncias [74]. Silva e coautores [74] mostraram um novo algoritmo de 1.375-aproximação que corrige esse problema e possui complexidade de $O(n^6)$. O nosso algoritmo é uma modificação do algoritmo de Elias e Hartman [43] que garante uma melhoria na complexidade de tempo em relação ao algoritmo proposto por Silva e coautores [74]. Essa melhoria é alcançada pelo uso de um novo procedimento para a busca de 2-transposições em uma permutação qualquer. Note que um algoritmo de busca exaustiva para a busca de uma 2-transposição, caso exista, tem complexidade de tempo de $O(n^3)$, já que precisamos testar todas as combinações de triplas (i,j,k), com $1 \leq i < j < k \leq n+1$. Nesta seção, apresentamos um algoritmo para a busca de uma 2-transposição, caso exista, com complexidade de tempo de $O(n^2)$.

A seguir, listamos resultados conhecidos na literatura sobre a variação no número de ciclos ímpares causada por uma transposição e, também, sobre um limitante para a distância $d_\tau(\pi)$ relacionado com a quantidade de ciclos ímpares em $G(\pi)$.

Lema 3.1.1 (Bafna e Pevzner [17], Lema 2.3)**.** *Para qualquer permutação π e transposição τ, temos que $\Delta c_{odd}(\pi,\tau) = \{-2,0,2\}$.*

Lema 3.1.2 (Bafna e Pevzner [17], Teorema 2.4)**.** *Para qualquer permutação π, temos que $d_\tau(\pi) \geq \frac{n+1-c_{odd}(\pi)}{2}$.*

Agora, apresentamos propriedades sobre as 2-transposições em ciclos pares e ímpares. O próximo lema caracteriza os tipos de ciclos afetados por uma 2-transposição.

Lema 3.1.3 (Christie [39], Lemas 3.2.5 e 3.3.1)**.** *Se existe uma 2-transposição aplicada nas arestas de origem o_i, o_j, e o_k, então essas arestas pertencem a dois ciclos pares ou todas as três arestas pertencem ao mesmo ciclo orientado.*

Quando existem pelo menos dois ciclos pares no grafo, Christie [39] provou que é possível encontrar uma 2-transposição em tempo linear. Portanto, nos próximos lemas, focamos apenas em 2-transposições que afetam ciclos orientados, sendo que dividimos a análise em três casos para ciclos orientados ímpares.

Lema 3.1.4. *Se existe um ciclo orientado C em $G(\pi)$ que é um ciclo par, então existe uma tripla orientada válida (o_i, o_j, o_k) em $C = (o_1, o_2, \ldots, o_m)$, com $i < j < k$, tal que $k = j + 1$.*

Demonstração. Bafna e Pevzner [17, Lema 2.3] mostraram que todo ciclo orientado C possui tripla orientada (o_i, o_j, o_k), com $i < j < k$, tal que $o_i > o_k > o_j$ e $k = j + 1$. Uma transposição aplicada nessas arestas de origem transforma C em três ciclos C_1, C_2, C_3, tal que pelo menos um deles é um ciclo unitário, sendo que esse ciclo unitário possui a aresta de destino d_j que é adjacente às arestas de origem o_j e o_k. Suponha, sem perda de generalidade, que C_1 é o ciclo unitário que contém a aresta de destino d_j. Como C é um ciclo par, sabemos que dentre os ciclos C_2 e C_3 temos um ciclo par e um ciclo ímpar. Portanto, essa transposição adiciona dois ciclos ímpares no grafo de ciclos e é uma 2-transposição, o que implica que (o_i, o_j, o_k) é uma tripla orientada válida. □

Lema 3.1.5. *Se existe uma tripla orientada válida (o_i, o_j, o_k) em um ciclo ímpar $C = (o_1, o_2, \ldots, o_m)$, tal que $i < j < k$ e $o_i > o_k > o_j$, então existe uma tripla orientada válida $(o_{i'}, o_{j'}, o_{k'})$ em C, com $i' < j' < k'$, tal que $i' \in \{1, 2\}$ ou $k' = j' + 1$.*

Demonstração. Note que uma 2-transposição que afeta apenas um ciclo também aumenta o número de ciclos no grafo em duas unidades. A 2-transposição que age nas arestas (o_i, o_j, o_k) transforma o ciclo C em três ciclos D, D' e D'', tal que D possui as arestas de destino do caminho que vai de o_i até o_j, D' possui as arestas de destino do caminho que vai de o_j até o_k, e D'' possui as arestas de destino do caminho que vai de o_k até o_i. Portanto, o tamanho desses ciclos são $|D| = j - i$, $|D'| = k - j$, $|D''| = |C| + i - k$. Como essa 2-transposição afeta o ciclo ímpar C, temos que $|D|, |D'|$, e $|D''|$ são valores ímpares.

Quando $i \in \{1, 2\}$ ou $k = j + 1$, temos que a própria tripla (o_i, o_j, o_k) já atende as condições descritas no enunciado do lema. Caso contrário, temos que $i \geq 3$ e $k \geq j + 3$. A seguir, dividimos a prova em casos dependendo da paridade dos valores de i ou k. Para cada caso, mostramos que existe uma tripla orientada válida que atende as condições do lema.

Se i é ímpar, então j deve ser par e k deve ser ímpar. Lembre-se que pela nossa definição de como as arestas de origem de um ciclo são listadas, temos que $o_1 > o_i$ para qualquer ciclo. Portanto, (o_1, o_j, o_k) é uma tripla orientada válida, já que $o_1 > o_i$ e os valores $j - 1$, $k - j$, e $|C| + 1 - k$ são todos ímpares.

Se i é par, então j deve ser ímpar e k deve ser par. Se $o_2 > o_k$, então (o_2, o_j, o_k) é uma tripla orientada válida, já que $j - 2$, $k - j$, e $|C| + 2 - k$ são valores ímpares. Se $o_2 < o_k$, ainda dividimos a prova nos seguintes casos.

- Se $o_{k-1} > o_k > o_2$, então (o_1, o_2, o_{k-1}) é uma tripla orientada válida, pois $o_1 > o_{k-1} > o_2$ e, já que k é par, temos que os valores 1, $(k-1)-2$ e $|C|+1-(k-1)$, que correspondem aos tamanhos dos ciclos criados por uma transposição que age nessa tripla, são ímpares.

- Se $o_k > o_{k-1}$, então (o_i, o_{k-1}, o_k) é uma tripla orientada válida, pois $o_i > o_k > o_{k-1}$ e os valores $(k-1)-i$, 1 e $|C|+i-k$, que correspondem aos tamanhos dos ciclos criados por uma transposição que age nessa tripla, são todos ímpares.

□

Lema 3.1.6. *Se existe uma tripla orientada válida (o_i, o_j, o_k) em um ciclo ímpar $C = (o_1, o_2, \ldots, o_m)$, tal que $i < j < k$ e $o_k > o_j > o_i$, então existe uma tripla orientada válida $(o_{i'}, o_{j'}, o_{k'})$ em C, com $i' < j' < k'$, tal que $i' = 1$.*

Demonstração. Se $i = 1$, então a tripla (o_i, o_j, o_k) já atende as condições descritas no enunciado do lema. Caso contrário, temos que $i \geq 2$.

Se j é par, então k é ímpar e (o_1, o_j, o_k) é uma tripla orientada válida, pois $o_1 > o_k > o_j$ e uma transposição aplicada nessa tripla cria ciclos com tamanhos $j-1$, $k-j$ e $|C|+1-k$, que são todos ímpares. Caso contrário, temos que j é ímpar, i é par e (o_1, o_i, o_j) é uma tripla orientada válida, pois $o_1 > o_j > o_i$ e uma transposição aplicada nessa tripla cria ciclos com tamanhos $i-1$, $j-i$, $|C|+1-j$, que são todos ímpares. □

Lema 3.1.7. *Se existe uma tripla orientada válida (o_i, o_j, o_k) em um ciclo ímpar $C = (o_1, o_2, \ldots, o_m)$, tal que $i < j < k$ e $o_j > o_i > o_k$, então existe uma tripla orientada válida $(o_{i'}, o_{j'}, o_{k'})$ em C, com $i' < j' < k'$, tal que pelo menos uma dessas condições é verdadeira: $i' \in \{1, 2\}$; $j' = i' + 1$; ou $k' = j' + 1$.*

Demonstração. Note que se $i \in \{1, 2\}$, ou $j = i + 1$, ou $k = j + 1$, então a tripla (o_i, o_j, o_k) já atende as condições do enunciado do lema. Caso contrário, temos que $i \geq 3$, $j \geq i + 3$ e $k \geq j + 3$.

Se j é ímpar, então i e k são pares. Consequentemente, temos a tripla orientada válida (o_1, o_i, o_j), pois $o_1 > o_j > o_i$ e os valores $i-1$, $j-i$ e $|C|+1-j$ são ímpares.

Como temos muitos casos quando j é par, iremos apenas listar as condições e a tripla orientada válida correspondente que atende as condições do enunciado do lema, sendo fácil conferir que ela é de fato uma tripla orientada válida. De agora em diante, assuma que j é par e ambos os valores de i e k são ímpares.

Primeiramente, consideramos o valor da aresta de origem o_2:

- Se $o_2 > o_j$, então (o_2, o_i, o_j) é uma tripla orientada válida.

- Se $o_2 < o_i$, então (o_1, o_2, o_i) é uma tripla orientada válida.

Caso as condições acima sejam ambas falsas, temos que $o_j > o_2 > o_i$. Agora, considere a aresta de origem o_{i+1}, que é a aresta de origem subsequente a o_i no ciclo C. Lembre-se que $i + 1 < j$ e ambos $i + 1$ e j são pares.

- Se $o_{i+1} < o_k$, então (o_1, o_{i+1}, o_k) é uma tripla orientada válida.

- Se $o_{i+1} > o_2$, então (o_i, o_{i+1}, o_k) é uma tripla orientada válida.
- Se $o_2 > o_{i+1} > o_i$, então (o_2, o_i, o_{i+1}) é uma tripla orientada válida.

Caso as condições anteriores sejam todas falsas, temos que $o_1 > o_j > o_2 > o_i > o_{i+1} > o_k$. Agora, considere a aresta de origem o_{j+1}, a aresta de origem subsequente a o_j no ciclo C. Lembre-se que $j + 1 < k$ e $j + 1$ é ímpar.

- Se $o_{j+1} > o_i$, então (o_1, o_{i+1}, o_{j+1}) é uma tripla orientada válida.
- Caso contrário, temos $o_{j+1} < o_i$ e, portanto, (o_i, o_j, o_{j+1}) é uma tripla orientada válida.

Dessa forma, cobrimos todos os casos possíveis e, para cada um dos casos, mostramos que existe uma tripla orientada válida que atende as condições do enunciado do lema. □

Os lemas 3.1.4 ao 3.1.7 implicam no seguinte resultado.

Corolário 3.1.1. *Se existe uma 2-transposição afetando um ciclo orientado C de $G(\pi)$, então existe uma 2-transposição aplicada em três arestas de origem o_i, o_j e o_k do ciclo C, com $i < j < k$, tal que pelo menos uma destas condições é verdadeira: $i \in \{1, 2\}$; $j = i + 1$; ou $k = j + 1$.*

Dado uma tripla orientada válida (o_i, o_j, o_k), com $i < j < k$, definimos os parâmetros de uma 2-transposição τ que age nessa tripla da seguinte forma:

- Se $o_i > o_k > o_j$, então $\tau = \tau(o_j, o_k, o_i)$;
- Se $o_j > o_i > o_k$, então $\tau = \tau(o_k, o_i, o_j)$;
- Se $o_k > o_j > o_i$, então $\tau = \tau(o_i, o_j, o_k)$.

A partir desses resultados, apresentamos o Algoritmo 1, que recebe uma permutação π de entrada e retorna uma 2-transposição, caso exista, ou indica que não existe 2-transposição para essa permutação.

Lema 3.1.8. *Dada uma permutação π, se existe pelo menos uma 2-transposição que pode ser aplicada em $G(\pi)$, então o Algoritmo 1 retorna uma 2-transposição. Caso contrário, o algoritmo retorna que não existe 2-transposição para $G(\pi)$.*

Demonstração. Pelo Lema 3.1.3, qualquer 2-transposição τ é aplicada em dois ciclos pares ou em um único ciclo orientado.

Se existe um par de ciclos pares em $G(\pi)$, então o algoritmo sempre acha uma 2-transposição de acordo com o Lema 3.2.5 de Christie [39].

Se a 2-transposição é aplicada em um ciclo orientado $C = (o_1, o_2, \ldots, o_m)$, então existe uma 2-transposição aplicada em uma tripla orientada válida $(o_{i'}, o_{j'}, o_{k'})$, com $i' < j' < k'$, tal que $i' \in \{1, 2\}$, ou $j' = i' + 1$, ou $k' = j' + 1$ (Corolário 3.1.1). Como o algoritmo faz uma busca exaustiva verificando todas as triplas (o_i, o_j, o_k), com $i < j < k$, tal que $i \in \{1, 2\}$, ou $j = i + 1$, ou $k = j + 1$, então o algoritmo irá encontrar uma 2-transposição nesse caso.

Se não existe 2-transposição, então o Algoritmo 1 retorna vazio na sua última linha. □

Algoritmo 1: Busca por uma 2-transposição

Entrada: Uma permutação π
Saída: Uma 2-transposição τ, caso exista, ou $\emptyset$

```
1  Construa o grafo G(π)
2  se existem pelo menos dois ciclos pares em G(π) então
3      retorne a 2-transposição do Lema 3.2.5 de Christie [39]
4  senão
5      para todo ciclo orientado C = (o_1, o_2, ..., o_m) em G(π) faça
6          para todo j ∈ {2, ..., m − 1} faça
7              para todo k ∈ {j + 1, ..., m} faça
8                  se (o_1, o_j, o_k) é uma tripla orientada válida então
9                      retorne a 2-transposição que age em (o_1, o_j, o_k)
10                 senão se (o_2, o_j, o_k) é uma tripla orientada válida então
11                     retorne a 2-transposição que age em (o_2, o_j, o_k)
12     para todo ciclo orientado C = (o_1, o_2, ..., o_m) em G(π) faça
13         para todo i ∈ {3, ..., m − 2} faça
14             para todo j ∈ {i + 1, ..., m − 1} faça
15                 se (o_i, o_j, o_{j+1}) é uma tripla orientada válida então
16                     retorne a 2-transposição que age em (o_i, o_j, o_{j+1})
17             para todo k ∈ {i + 2, ..., m} faça
18                 se (o_i, o_{i+1}, o_k) é uma tripla orientada válida então
19                     retorne a 2-transposição que age em (o_i, o_{i+1}, o_k)
20     retorne ∅                          ▷ não existe 2-transposição para G(π)
```

Para analisar a complexidade do Algoritmo 1, verificamos primeiramente que a construção do grafo de ciclos $G(\pi)$ leva tempo linear. A complexidade de tempo para as linhas 2 e 3 também é linear, como mostrado na prova do Lema 3.2.5 de Christie [39]. A complexidade de tempo da busca nas linhas 5–11 é a seguinte, onde c é uma constante relacionada às operações das linhas 8–11.

$$\sum_{C\in G(\pi)} \sum_{j=2}^{|C|-1} \sum_{k=j+1}^{|C|} c = \sum_{C\in G(\pi)} \sum_{j=2}^{|C|-1} (|C|-j)c < c \sum_{C\in G(\pi)} |C|^2 = O(n^2),$$

já que $\sum_{C\in G(\pi)} |C| \leq n+1$.

De forma similar, podemos provar que a complexidade de tempo das linhas 12–19 também é quadrática. Dessa forma, temos que o Algoritmo 1 possui complexidade de tempo de $O(n^2)$.

Agora, apresentamos como alcançar uma complexidade de tempo de $O(n^5)$ para o algoritmo de 1.375-aproximação proposto por Elias e Hartman [43]. Primeiro, notamos que o Algoritmo 1 não pode ser usado para listar todas as possíveis 2-transposições aplicadas em uma permutação, apesar de que o algoritmo sempre retorna uma 2-transposição se $G(\pi)$ admite pelo menos uma 2-transposição.

O Algoritmo 2 retorna uma $(2,2)$-sequência, caso exista, ou indica que não existe uma $(2,2)$-sequência para $G(\pi)$. Esse algoritmo testa todas as combinações de triplas (i,j,k), com $1 \leq i < j < k \leq n+1$, a fim de encontrar 2-transposições para $G(\pi)$. Para toda 2-transposição $\tau(i,j,k)$ encontrada na linha 6, o Algoritmo 2 utiliza o Algoritmo 1 para verificar se existe uma 2-transposição para o grafo $G(\pi \cdot \tau(i,j,k))$.

Por fim, o Algoritmo 3 usa o Algoritmo 2 como sub-rotina para encontrar uma $(2,2)$-sequência, caso exista, e também usa o algoritmo de Elias e Hartman [43], o qual chamamos de `Algoritmo_EH`, que é aplicado após a busca de uma $(2,2)$-sequência para a permutação π.

Algoritmo 2: Retorna uma $(2,2)$-sequência, caso exista, ou uma sequência vazia

Entrada: Uma permutação π
Saída: Uma $(2,2)$-sequência, caso exista, ou $(\emptyset, \emptyset)$

```
1  Construa G(π)
2  Seja z o número de arestas de origem em G(π)
3  para todo i ∈ {1, ..., z − 2} faça
4      para todo j ∈ {i + 1, ..., z − 1} faça
5          para todo k ∈ {j + 1, ..., z} faça
6              se τ(i, j, k) é uma 2-transposição então
7                  π' ← π · τ(i, j, k)
8                  τ' ← Algoritmo_1(π')
9                  se τ' ≠ ∅ então
10                     retorne (τ, τ')
11 retorne (∅, ∅)
12                                   ▷ Não existe uma (2, 2)-sequência para π
```

Algoritmo 3: Uma 1.375-Aproximação Mais Eficiente para Transposições

Entrada: Uma permutação π
Saída: Uma sequência de transposições $\tau_1, \ldots, \tau_r$ que ordena π

```
1  (τ1, τ2) ← Algoritmo_2(π)
2  se τ1 ≠ ∅ então
3      π' ← π · τ1 · τ2          ▷ existe uma (2, 2)-sequência para π
4      retorne (τ1, τ2) + Algoritmo_EH(π')
5  senão
6      retorne Algoritmo_EH(π)
```

Lembre-se que o principal problema do `Algoritmo_EH` em garantir o fator de aproximação de 1.375 é possivelmente negligenciar uma $(2,2)$-sequência após transformar π em uma permutação simples $\hat{\pi}$.

Primeiramente, iremos analisar o Algoritmo 2 que busca por uma $(2,2)$-sequência para uma permutação π. Para cada transposição τ gerada usando todos os possíveis valores para i, j e k, se τ é uma 2-transposição, então o algoritmo aplica essa operação em π, resultando em $\pi' = \pi \cdot \tau$. Após isso, o Algoritmo 2 usa o Algoritmo 1 com π' como

sub-rotina. Se o Algoritmo 1 retorna uma segunda 2-transposição, então o Algoritmo 2 retorna uma $(2, 2)$-sequência na linha 10. Se não existe uma $(2, 2)$-sequência para π, então o algoritmo retorna uma sequência vazia.

O Algoritmo 2 usa laços aninhados nas linhas 3–5 para a busca da primeira 2-transposição $\tau(i, j, k)$, usando todas as combinações para os índices i, j e k. Para executar as instruções dentro desses laços aninhados (linhas 6–10), é necessário complexidade de tempo de $O(n^2)$. Portanto, o Algoritmo 2 executa em tempo $O(n^5)$.

Após usar o Algoritmo 2 na linha 1, o Algoritmo 3 usa a 1.375-aproximação de Elias e Hartman [43] (`Algoritmo_EH`) na linha 4 ou na linha 6. Como o `Algoritmo_EH` tem complexidade de tempo quadrática, concluímos que o Algoritmo 3 possui complexidade de tempo de $O(n^5)$.

É garantido que o Algoritmo 3 possui fator de aproximação de 1.375 para qualquer permutação, pois ele garante a aplicação de uma $(2, 2)$-sequência na permutação de entrada, caso ela exista, e apenas depois disso usa o `Algoritmo_EH`, que transforma a permutação de entrada em uma permutação simples.

3.1.1 Resultados Experimentais

Nesta seção, apresentamos resultados experimentais para o Algoritmo 3. Comparamos os resultados do nosso algoritmo com os resultados apresentados para os algoritmos propostos por Elias e Hartman [43] e Silva e coautores [74].

Testamos os algoritmos para todas as permutações de tamanho $n \leq 12$ e comparamos os tamanhos das sequências de ordenação retornadas pelos algoritmos com as distâncias exatas disponíveis no sistema GRAAu [47]. O GRAAu é uma ferramenta de auditoria para algoritmos de ordenação de permutações por rearranjos, sendo que essa ferramenta possui um banco de dados com os valores das distâncias exatas para permutações pequenas.

A Tabela 3.1 sumariza os resultados dos algoritmos testados, os quais são identificados por:

- **ALG3**: Algoritmo 3 apresentado nesta seção;
- **EH**: Algoritmo apresentado por Elias e Hartman [43];
- **SKRW**: Algoritmo apresentado por Silva e coautores [74].

As permutações são agrupadas por tamanho e cada linha apresenta resultados para o grupo de permutações de tamanho n, indicado na primeira coluna da tabela. As colunas **APROX MAX** e **APROX MED** representam o valor máximo e o valor médio do fator de aproximação observado, respectivamente. A coluna **DIST MED** representa a média dos tamanhos das sequências de ordenação retornadas pelos algoritmos e a coluna **% SOLUÇÕES ÓTIMAS** indica o percentual de instâncias em que uma solução ótima foi encontrada.

O algoritmo de Elias e Hartman [43] retornou sequências de ordenação com um fator de aproximação acima de 1.375 (comparado com a distância exata) em 2 instâncias de tamanho 8, 20 instâncias de tamanho 9, 110 instâncias de tamanho 10, 440 instâncias de tamanho 11 e, por último, 1448 instâncias de tamanho 12. Por outro lado, o algoritmo

proposto nesta seção e o algoritmo proposto por Silva e coautores [74] não retornaram nenhuma sequência de ordenação com aproximação acima de 1.333.

Na Tabela 3.1, os valores para o fator de aproximação máximo dos algoritmos **ALG3** e **SKRW** foram os mesmos, considerando todos os grupos de permutações. Ao analisar os valores médios dos fatores de aproximação para os algoritmos **ALG3** e **SKRW**, concluímos que nos grupos de tamanho menor ou igual 6, o valor de 1.0 foi mantido, o que indica que, para todas as instâncias nesses grupos, ambos os algoritmos encontraram sequências de ordenação ótimas (a coluna **% SOLUÇÕES ÓTIMAS** também indica a mesma informação). Para grupos de permutações com tamanhos maiores que 6, ao comparar os três algoritmos, concluímos que o algoritmo apresentado nesta seção obteve melhores resultados de aproximação máxima, aproximação média, médias das distâncias, e percentual de solução ótimas encontradas.

É importante destacar que o percentual de instâncias em que o algoritmo **ALG3** encontra uma solução ótima foi maior que 88% para todos os grupos de permutações. O Algoritmo **SKRW** mantém esse comportamento apenas para grupos de permutações com tamanho menor ou igual a 9. Assim, concluímos que o algoritmo proposto nesta seção traz uma melhoria do ponto de vista da qualidade da solução, além da melhoria da complexidade do algoritmo já demonstrada anteriormente.

Optamos por não comparar os tempos de execução das implementações dos algoritmos por não considerarmos que a análise seria justa, dados os seguintes motivos: (i) o algoritmo **SKRW** é implementado em uma linguagem de programação distinta da utilizada na implementação do algoritmo **ALG3**; (ii) o algoritmo **EH** possui complexidade de tempo consideravelmente menor do que os outros algoritmos, apesar de não ser um algoritmo de aproximação válido, como mostrado na Tabela 3.1.

Tabela 3.1: Comparação entre os resultados experimentais do Algoritmo 3 e os resultados dos algoritmos propostos por Elias e Hartman [43] (**EH**) e Silva e coautores [74] (**SKRW**), em todas permutações de tamanho $n \leq 12$, excluindo a permutação identidade ι^n.

	APROX MAX			**APROX MED**			**DIST MED**			**% SOLUÇÕES ÓTIMAS**		
n	**EH**	**SKRW**	**ALG3**	**EH**	**SKRW**	**ALG3**	**EH**	**SKRW**	**ALG3**	**EH**	**SKRW**	**ALG3**
2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	100.00	100.00	100.00
3	1.00	1.00	1.00	1.00	1.00	1.00	1.20	1.20	1.20	100.00	100.00	100.00
4	1.00	1.00	1.00	1.00	1.00	1.00	1.6086	1.6086	1.6086	100.00	100.00	100.00
5	1.00	1.00	1.00	1.00	1.00	1.00	2.0924	2.0924	2.0924	100.00	100.00	100.00
6	1.33	1.00	1.00	1.0004	1.00	1.00	2.6063	2.6050	2.6050	99.86	100.00	100.00
7	1.33	1.25	1.25	1.0129	1.0113	1.0014	3.1762	3.1704	3.1311	94.90	95.47	99.40
8	1.50	1.25	1.25	1.0210	1.0183	1.0042	3.7178	3.7076	3.6512	91.64	92.65	98.29
9	1.50	1.25	1.25	1.0301	1.0256	1.0085	4.2796	4.2603	4.1846	86.62	88.54	96.10
10	1.50	1.25	1.25	1.0341	1.0282	1.0125	4.8051	4.7772	4.7032	83.80	86.53	93.94
11	1.50	1.33	1.33	1.0392	1.0321	1.0170	5.3526	5.3157	5.2367	79.40	82.98	90.88
12	1.50	1.33	1.33	1.0415	1.0336	1.0206	5.8694	5.8248	5.7514	76.67	80.91	88.27

3.2 Complexidade de Problemas com Transposições e Outros Rearranjos

Nesta seção, apresentamos provas de NP-dificuldade para problemas de Ordenação de Permutações por Rearranjos considerando os seguintes modelos de rearranjos:

- $\mathcal{M}_1 = \{\tau, \rho\tau\}$: Transposições e Transposições Inversas;
- $\mathcal{M}_2 = \{\rho, \tau, \rho\tau\}$: Reversões, Transposições e Transposições Inversas;
- $\mathcal{M}_3 = \{\tau, \rho\rho\}$: Transposições e Revrevs;
- $\mathcal{M}_4 = \{\rho, \tau, \rho\rho\}$: Reversões, Transposições e Revrevs;
- $\mathcal{M}_5 = \{\tau, \rho\tau, \rho\rho\}$: Transposições, Transposições Inversas e Revrevs.
- $\mathcal{M}_6 = \{\rho, \tau, \rho\tau, \rho\rho\}$: Reversões, Transposições, Transposições Inversas e Revrevs.

Além disso, denotamos por τ o modelo que possui apenas transposições. A seguir, apresentamos formalmente a versão de decisão dos problemas estudados.

Ordenação de Permutações por Rearranjos (SbR)
Entrada: Uma permutação π e um inteiro k.
Objetivo: Considerando um modelo de rearranjos $\mathcal{M}$, decidir se é possível ordenar π com uma sequência de rearranjos S, tal que $|S| \leq k$ e $\beta \in \mathcal{M}$, para todo $\beta \in S$. Ou seja, determinar se $d_{\mathcal{M}}(\pi) \leq k$.

Ordenação de Permutações por Rearranjos Ponderados (SbWR)
Entrada: Uma permutação π e um inteiro k.
Objetivo: Considerando um modelo de rearranjos $\mathcal{M}$ e pesos $\{w_\rho, w_\tau\}$, decidir se é possível ordenar π com uma sequência de rearranjos S, tal que $w(S) \leq k$ e $\beta \in \mathcal{M}$, para todo $\beta \in S$. Ou seja, determinar se $d^w_{\mathcal{M}}(\pi) \leq k$.

Quando $w_\rho = w_\tau$, esse problema é equivalente à abordagem não ponderada. Se um modelo não contém reversões, então consideramos que $w_\rho = \infty$.

Seja π uma permutação sem sinais e $k = b_\tau(\pi)/3$. Um limitante bastante conhecido para a distância de transposições é $d_t(\pi) \geq b_\tau(\pi)/3$ [34]. As provas de dificuldade apresentadas nesta seção são baseadas em reduções do problema NP-difícil **B3T** [34].

Ordenação de Permutações por Transposições Ótimas (B3T)
Entrada: Uma permutação sem sinais π.
Objetivo: Decidir se é possível ordenar a permutação π usando exatamente $b_\tau(\pi)/3$ transposições, ou seja, determinar se $d_\tau(\pi) = b_\tau(\pi)/3$.

Como a permutação identidade ι^n é a única com zero *breakpoints*, podemos interpretar o processo de ordenar π como o de remover todos os *breakpoints* de π. Agora, mostramos como os rearranjos podem afetar o número de *breakpoints* em uma permutação e também definir limitantes para a distância. Também apresentamos como os rearranjos estudados afetam o número de *breakpoints* em algumas famílias de permutações, que serão úteis nas provas de complexidade.

Lema 3.2.1. *Para qualquer permutação com sinais π, pesos w_ρ e w_τ, e modelo $\mathcal{M} \in \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6\}$, temos que*

$$d^w_{\mathcal{M}}(\pi) \geq \min\left\{\frac{w_\rho}{2}, \frac{w_\tau}{3}\right\} b_{\mathcal{M}}(\pi).$$

Demonstração. Como uma reversão $\rho(i,j)$ quebra apenas as adjacências (π_{i-1}, π_i) e (π_j, π_{j+1}), então no máximo dois *breakpoints* podem ser removidos ou adicionados e, portanto, $-2 \leq \Delta b_{\mathcal{M}}(\pi, \rho) \leq 2$. De forma similar, se uma operação β é uma transposição, transposição inversa ou revrev, então apenas três adjacências de π são quebradas e, portanto, no máximo três *breakpoints* podem ser removidos ou adicionados (i.e., $-3 \leq \Delta b_{\mathcal{M}}(\pi, \beta) \leq 3$).

Sendo assim, o custo mínimo para remover um *breakpoint* é igual a $\min\left\{\frac{w_\rho}{2}, \frac{w_\tau}{3}\right\}$. Como qualquer sequência S que ordena π remove $b_{\mathcal{M}}(\pi)$ *breakpoints*, concluímos que S possui custo maior ou igual a $\min\left\{\frac{w_\rho}{2}, \frac{w_\tau}{3}\right\} b_{\mathcal{M}}(\pi)$. □

Lema 3.2.2. *Para qualquer permutação com sinais π tal que π possui apenas strips positivas, temos que:*

- $\Delta b_\rho(\pi, \rho) \leq 0$*, para qualquer reversão ρ;*
- $\Delta b_{\rho\tau}(\pi, \rho\tau) \leq 1$*, para qualquer transposição inversa $\rho\tau$;*
- $\Delta b_{\rho\rho}(\pi, \rho\rho) \leq 1$*, para qualquer revrev $\rho\rho$.*

Demonstração. Note que π possuir apenas *strips* positivas implica que todos elementos de π possuem sinal "+".

Considere uma reversão ρ e seja $\pi' = \pi \cdot \rho = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{-\pi_j \ \ldots \ -\pi_i} \ \pi_{j+1} \ \ldots \ \pi_n)$, com $1 \leq i \leq j \leq n$. Provaremos, por contradição, que não existe reversão que remove *breakpoints* de π. Suponha que $\Delta b_\rho(\pi, \rho) > 0$, o que implica que $(\pi_{i-1}, -\pi_j)$ não é um *breakpoint* ou $(-\pi_i, \pi_{j+1})$ não é um *breakpoint*. Se $(\pi_{i-1}, -\pi_j)$ não é um *breakpoint*, então π_{i-1} e $-\pi_j$ devem ter o mesmo sinal. De forma similar, se $(-\pi_i, \pi_{j+1})$ não é um *breakpoint*, então $-\pi_i$ e π_{j+1} devem ter o mesmo sinal. Em ambos os casos chegamos a uma contradição, pois π possui apenas elementos com sinal "+" e, portanto, os elementos desses pares possuem sinais distintos. Sendo assim, $\Delta b_\rho(\pi, \rho) \leq 0$.

Considere uma transposição inversa tipo 1 $\rho\tau_1$ e seja $\pi' = \pi \cdot \rho\tau_1 = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \ldots \ \pi_{k-1}} \ \underline{-\pi_{j-1} \ \ldots \ -\pi_i} \ \pi_k \ \ldots \ \pi_n)$, com $1 \leq i < j < k \leq n+1$. Usando um argumento similar ao usado para reversões, temos que os pares $(\pi_{k-1}, -\pi_{j-1})$ e $(-\pi_i, \pi_k)$ devem ser *breakpoints* e apenas o par (π_{i-1}, π_j) pode não ser um *breakpoint*. Portanto, $\Delta b_{\rho\tau}(\pi, \rho\tau_1) \leq 1$. Usamos um argumento análogo para uma transposição inversa tipo 2. Agora, considere uma revrev $\rho\rho$ e seja $\pi' = \pi \cdot \rho\rho = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{-\pi_{j-1} \ \ldots \ -\pi_i}$

$\underline{-\pi_{k-1} \ \ldots \ -\pi_j}\ \pi_k \ \ldots \ \pi_n)$, com $1 \leq i < j < k \leq n+1$. Usando um argumento similar ao usado para reversões, temos que os pares $(\pi_{i-1}, -\pi_{j-1})$ e $(-\pi_j, \pi_k)$ devem ser *breakpoints* e apenas o par $(-\pi_i, -\pi_{k-1})$ pode não ser um *breakpoint*. Portanto, $\Delta b_{\rho\rho}(\pi, \rho\rho) \leq 1$. □

Teorema 3.2.1. *Para os modelos* $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6\}$ *e* $w_\tau/w_\rho \leq 1.5$*, temos que o problema da Ordenação de Permutações por Rearranjos Ponderados* **SbWR** *é NP-difícil para permutações com sinais.*

Demonstração. Considere o modelo de rearranjos $\mathcal{M} = \mathcal{M}_6$ nesta prova. A demonstração é similar para os outros modelos, já que a nossa estratégia é mostrar que em uma instância satisfeita (i.e., uma instância em que é possível ordenar π com custo menor ou igual a k) apenas transposições são usadas para ordenar a permutação π. Portanto, um argumento similar pode ser usado para os outros modelos, pois eles possuem um subconjunto das operações permitidas em $\mathcal{M}_6$.

Agora, apresentamos uma redução do problema **B3T** para o problema **SbWR** considerando permutações com sinais. Dada uma instância $\pi = (\pi_1 \ \ldots \ \pi_n)$ para o problema **B3T**, construímos a instância (π', k) para **SbWR**, onde π' é a permutação com sinais $(+\pi_1 \ +\pi_2 \ \ldots \ +\pi_n)$ e $k = w_\tau b_\tau(\pi)/3$.

Mostramos que a instância π é ordenada usando $b_\tau(\pi)/3$ transposições se, e somente se, $d^w_{\mathcal{M}}(\pi') \leq w_\tau b_\tau(\pi)/3$.

($\rightarrow$) Se a permutação π é ordenada por uma sequência S de tamanho $b_\tau(\pi)/3$, então S também ordena a permutação π', já que π' possui apenas elementos positivos, e $w(S) = w_\tau b_\tau(\pi)/3$. Note que S possui apenas transposições e cada transposição tem custo w_τ.

($\leftarrow$) Se π' é ordenada por uma sequência S de custo igual ou menor a $w_\tau b_\tau(\pi)/3$, então afirmamos que S possui apenas transposições e, portanto, S também ordena π e S tem tamanho $b_\tau(\pi)/3$.

Note que o custo mínimo para remover um *breakpoint* em π' é igual a $\min\{w_\rho/2, w_\tau/3\} = w_\tau/3$, já que $w_\tau/w_\rho \leq 1.5$. Como π' possui apenas elementos positivos, $\pi_{i+1} - \pi_i = 1$ se, e somente se, $\pi'_{i+1} - \pi'_i = 1$ e, portanto, $b_\tau(\pi) = b_{\mathcal{M}}(\pi')$. Sendo assim, o limitante inferior do Lema 3.2.1 se torna $w_\tau b_{\mathcal{M}}(\pi')/3 = w_\tau b_\tau(\pi)/3$.

Temos que $w(S)$ é igual ao limitante inferior $w_\tau b_\tau(\pi)/3$ e, consequentemente, todo rearranjo de S deve remover exatamente $w' \times 3/w_\tau$ *breakpoints*, onde w' é o custo do rearranjo. Note que uma sequência que ordena π' remove $b_\tau(\pi)$ *breakpoints*. Agora, suponha que S tem algum rearranjo que não é uma transposição. Considere que $S = (\beta_1, \beta_2, \ldots, \beta_{|S|})$. Seja β_i o primeiro rearranjo de S a ser aplicado que não é uma transposição, ou seja, todos rearranjos em $(\beta_1, \beta_2, \ldots, \beta_{i-1})$ são transposições, β_i não é uma transposição e o valor de i é mínimo. Antes de β_i ser aplicado, todas *strips* na permutação são positivas, já que apenas transposições foram aplicadas anteriormente e elas não alteram o sinal dos elementos. Pelo Lema 3.2.2, uma reversão não remove *breakpoints* e uma transposição inversa ou revrev removem no máximo um *breakpoint* em permutações que possuem apenas *strips* positivas, o que contradiz o fato de que todo rearranjo de S remove $w' \times 3/w_\tau$ *breakpoints*. Portanto, S possui apenas transposições e S possui tamanho $b_\tau(\pi)/3$. □

Corolário 3.2.1. *Para os modelos* $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6\}$*, temos que o problema da Ordenação de Permutações por Rearranjos* **SbR** *é NP-difícil para permutações com sinais.*

Lema 3.2.3. *Para qualquer permutação sem sinais π, pesos w_ρ e w_τ, e modelo $\mathcal{M} \in \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6\}$, temos que*

$$d^w_{\mathcal{M}}(\pi) \geq \min\left\{\frac{w_\rho}{2}, \frac{w_\tau}{3}\right\} b_{\mathcal{M}}(\pi).$$

Demonstração. Similar à prova do Lema 3.2.1. □

Lema 3.2.4. *Para qualquer permutação sem sinais π tal que π possui apenas strips crescentes, temos que:*

- $\Delta b_\rho(\pi, \rho) \leq 0$*, para qualquer reversão ρ;*
- $\Delta b_{\rho\tau}(\pi, \rho\tau) \leq 1$*, para qualquer transposição inversa $\rho\tau$;*
- $\Delta b_{\rho\rho}(\pi, \rho\rho) \leq 1$*, para qualquer revrev $\rho\rho$.*

Demonstração. Considere uma reversão ρ e seja $\pi' = \pi \cdot \rho(i, j) = (\pi_1 \ \ldots \ \pi_{i-1} \ \overline{\pi_j \ \ldots \ \pi_i} \ \pi_{j+1} \ \ldots \ \pi_n)$. Suponha, por contradição, que $\Delta b_\rho(\pi, \rho) > 0$, o que indica que ou $\overline{(\pi_{i-1}, \pi_j)}$ não é um *breakpoint* ou (π_i, π_{j+1}) não é um *breakpoint*.

Note que todas as *strips* em $(\pi'_i, \ldots, \pi'_j)$ são decrescentes, pois π possui apenas *strips* crescentes. Seja $\sigma = (\pi_{i'}, \ldots, \pi_{i-1})$ a *strip* crescente contendo o elemento π_{i-1} em π. Se (π'_{i-1}, π'_i) não é um *breakpoint*, então a *strip* σ se torna a *strip* $\sigma' = (\pi_{i'}, \ldots, \pi_{i-1}, \pi_j, \ldots, \pi_{j'})$ em π'. Como σ é crescente, temos que $\pi_{i-1} = \pi_0$ ou $i' < i - 1$. Então, a *strip* σ' deve ser uma *strip* crescente, o que contradiz o fato de que as *strips* em $(\pi'_i, \ldots, \pi'_j)$ são todas decrescentes. Alcançamos uma contradição semelhante se (π'_j, π'_{j+1}) não é um *breakpoint*. Portanto, temos que $\Delta b_\rho(\pi, \rho) \leq 0$.

Considere uma transposição inversa tipo 1 $\rho\tau_1$ e seja $\pi' = \pi \cdot \rho\tau_1(i, j, k) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_j \ \ldots \ \pi_{k-1}} \ \underline{\pi_{j-1} \ \ldots \ \pi_i} \ \pi_k \ \ldots \ \pi_n)$. Usando um argumento similar ao apresentado para reversões, temos que os pares (π_{k-1}, π_{j-1}) e (π_i, π_k) são *breakpoints* e apenas o par (π_{i-1}, π_j) pode não ser um *breakpoint*. Portanto, temos que $\Delta b_{\rho\tau}(\pi, \rho\tau) \leq 1$. Usamos um argumento similar para uma transposição inversa tipo 2.

Considere uma revrev $\rho\rho$ e seja $\pi' = \pi \cdot \rho\rho(i, j, k) = (\pi_1 \ \ldots \ \pi_{i-1} \ \underline{\pi_{j-1} \ \ldots \ \pi_i} \ \underline{\pi_{k-1} \ \ldots \ \pi_j} \ \pi_k \ \ldots \ \pi_n)$. Usando um argumento similar ao apresentado para reversões, temos que os pares (π_{i-1}, π_{j-1}) e (π_j, π_k) devem ser *breakpoints* e apenas o par (π_i, π_{k-1}) pode não ser um *breakpoint*. Portanto, temos que $\Delta b_{\rho\rho}(\pi, \rho\rho) \leq 1$. □

Teorema 3.2.2. *Para os modelos $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6\}$ e $w_\tau/w_\rho \leq 1.5$, temos que o problema da Ordenação de Permutações por Rearranjos Ponderados* **SbWR** *é NP-difícil para permutações sem sinais.*

Demonstração. Considere o modelo de rearranjos $\mathcal{M} = \mathcal{M}_6$ nesta prova. A demonstração é similar para os outros modelos, já que a nossa estratégia é mostrar que em uma instância satisfeita (i.e., uma instância em que é possível ordenar π com custo menor ou igual a k) apenas transposições são usadas para ordenar a permutação π. Portanto, um argumento similar pode ser usado para os outros modelos, pois eles possuem um subconjunto das operações permitidas em $\mathcal{M}_6$.

Para esta prova, também apresentamos uma redução do problema **B3T** para o problema **SbWR**, mas considerando permutações sem sinais. Dada uma instância $\pi = (\pi_1 \ \dots \ \pi_n)$ para o problema **B3T**, construímos a instância (π', k) para **SbWR**, onde π' é uma permutação sem sinais com $2n$ elementos tal que $\pi'_{2i-1} = 2\pi_i - 1$ e $\pi'_{2i} = 2\pi_i$, com $1 \leq i \leq n$, e $k = w_\tau b_\tau(\pi)/3$.

Mostramos que a instância π é ordenada por $b_\tau(\pi)/3$ transposições se, e somente se, $d^w_{\mathcal{M}}(\pi') \leq w_\tau b_\tau(\pi)/3$.

($\rightarrow$) Se π é ordenada por uma sequência $S = (\tau_1, \tau_2, \dots, \tau_{|S|})$ com $|S| = b_\tau(\pi)/3$, então construímos uma sequência $S' = (\tau'_1, \tau'_2, \dots, \tau'_{|S|})$ tal que, para toda transposição $\tau_i = \tau(x, y, z)$ em S, temos que $\tau'_i = \tau(2x - 1, 2y - 1, 2z - 1)$. Como todo elemento de π foi mapeado em dois elementos consecutivos em π', a sequência S' ordena π' e $w(S') = w_\tau b_\tau(\pi)/3$.

($\leftarrow$) Se π' é ordenada por uma sequência S' com $w(S') \leq w_\tau b_\tau(\pi)/3$, então afirmamos que existe uma sequência S tal que S possui apenas transposições, S ordena π, e $|S| = b_\tau(\pi)/3$.

Como π'_{2i-1} e π'_{2i} são elementos consecutivos, os pares (π'_{2i-1}, π'_{2i}) não são *breakpoints*, para qualquer $1 \leq i \leq n$. Assim, temos que todas as *strips* de π' são crescentes pois $\pi'_{2i} > \pi'_{2i-1}$. Além disso, para $0 \leq i \leq n$, temos que:

- se $\pi_{i+1} - \pi_i = 1$, então $\pi'_{2i+1} - \pi'_{2i} = 2\pi_{i+1} - 1 - 2\pi_i = 1$;
- se $\pi_{i+1} - \pi_i > 1$, então $\pi'_{2i+1} - \pi'_{2i} = 2\pi_{i+1} - 1 - 2\pi_i = 2(\pi_{i+1} - \pi_i) - 1 > 1$;
- se $\pi_{i+1} - \pi_i < 1$, então $\pi'_{2i+1} - \pi'_{2i} = 2\pi_{i+1} - 1 - 2\pi_i = 2(\pi_{i+1} - \pi_i) - 1 < 1$.

Dessa forma, $b_\tau(\pi) = b_{\mathcal{M}}(\pi')$. Note que o custo mínimo para remover um *breakpoint* em π' é igual a $\min\{w_\rho/2, w_\tau/3\} = w_\tau/3$, pois $w_\tau/w_\rho \leq 1.5$. Então, o limitante inferior do Lema 3.2.3 se torna $w_\tau b_{\mathcal{M}}(\pi')/3 = w_\tau b_\tau(\pi)/3$.

Temos que $w(S')$ é igual ao limitante inferior $w_\tau b_\tau(\pi)/3$ e, consequentemente, todo rearranjo de S' deve remover exatamente $w' \times 3/w_\tau$ *breakpoints*, onde w' é o custo do rearranjo.

Suponha, por contradição, que existe rearranjo em S' que não seja uma transposição. Considere $S' = (\beta_1, \beta_2, \dots, \beta_\ell)$ e seja β_i o primeiro rearranjo de S' a ser aplicado que não é uma transposição, ou seja, todos rearranjos em $(\beta_1, \beta_2, \dots, \beta_{i-1})$ são transposições, β_i não é uma transposição e o valor de i é mínimo. Antes de β_i ser aplicado, todas as *strips* na permutação são crescentes pois transposições que removem três *breakpoints* não criam *strips* decrescentes.

Pelo Lema 3.2.4, β_i não remove $w' \times 3/w_\tau$ *breakpoints*, onde w' é o custo de β_i, o que é uma contradição. Portanto, S' possui apenas transposições.

Como $w(S') = w_\tau b_\tau(\pi)/3$, a sequência S' tem $b_\tau(\pi)/3$ transposições. Note que as transposições de S' não quebram os pares (π'_{2i-1}, π'_{2i}), para $1 \leq i \leq n$, pois cada transposição remove três *breakpoints*.

Considere $S' = (\tau'_1, \tau'_2, \dots, \tau'_{|S'|})$, com $|S'| = b_\tau(\pi)/3$. Agora, construímos uma sequência $S = (\tau_1, \tau_2, \dots, \tau_{|S'|})$ que ordena π, tal que $\tau_i = \tau((x+1)/2, (y+1)/2, (z+1)/2)$ para $\tau'_i = \tau(x, y, z)$, com $1 \leq i \leq |S'|$. □

Corolário 3.2.2. *Para os modelos* $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5, \mathcal{M}_6\}$*, temos que o problema da Ordenação de Permutações por Rearranjos* **SbR** *é NP-difícil para permutações sem sinais.*

3.3 Conclusões

Neste capítulo estudamos problemas de distância de rearranjos utilizando a representação clássica de genomas que envolvem transposições, ou seja, estudamos problemas de Ordenação de Permutações por Transposições e Outros Rearranjos.

Na Seção 3.1, nós apresentamos um novo algoritmo de aproximação com fator de 1.375 para a Ordenação de Permutações por Transposições. Esse algoritmo tem complexidade de tempo de $O(n^5)$, o que representa uma melhoria em relação ao algoritmo recém publicado por Silva e coautores [74] que corrige um problema no algoritmo de Elias e Hartman [43], sendo esse último um dos algoritmos mais conhecidos da área de rearranjos de genomas. Além da melhoria na complexidade de tempo, nossos testes em permutações pequenas mostraram que o nosso algoritmo também apresenta melhoria na qualidade das soluções encontradas, sendo que o nosso algoritmo encontrou soluções ótimas em mais casos do que os outros dois algoritmos, além de apresentar um valor médio do fator de aproximação menor do que os outros dois algoritmos.

Na Seção 3.2, nós demonstramos que os problemas de Ordenação de Permutações (com ou sem Sinais) por Rearranjos Ponderados são NP-difíceis para 12 modelos de rearranjos que incluem transposições junto com a combinação de reversões, transposições inversas e revrevs, considerando que o custo de uma reversão é igual a w_ρ, os custos de uma transposição, de uma transposição inversa ou de uma revrev são os mesmos e iguais a w_τ, e atendendo a restrição de que $w_\tau/w_\rho \leq 1.5$. Note que quando $w_\tau/w_\rho = 1$, a versão ponderada é equivalente a não ponderada. Portanto, a prova de NP-dificuldade também é válida para a Ordenação de Permutações (com ou sem Sinais) por Rearranjos considerando esses 12 modelos.

Quando $w_\tau/w_\rho > 1.5$, a complexidade desses problemas permanece aberta, assim como a complexidade do problema de Ordenação de Permutações por Reversões e Transposições Ponderadas com a mesma restrição de pesos. Uma direção para trabalhos futuros é o estudo da complexidade nesses casos.

Capítulo 4

Distância em Genomas Desbalanceados

Os primeiros trabalhos da área de rearranjo de genomas envolveram apenas permutações. A partir do ano 1999, trabalhos considerando genomas com conjuntos distintos de genes foram introduzidos [73]. Em 2000, El-Mabrouk [42] estudou o problema da Distância de Reversões e Indels em Strings com Sinais, apresentando heurísticas baseadas no algoritmo exato de Hannenhalli e Pevzner [51] para a versão do problema com permutações.

Yancopoulos e coautores [84] estudaram uma operação de rearranjo chamada DCJ (*Double-Cut-and-Join*). Um DCJ consegue simular reversões e outras operações de rearranjos, mas algumas operações de rearranjo, como as transposições, transposições inversas, revrevs e *block interchanges*, não podem ser reproduzidas com uma única operação de DCJ, sendo necessário o uso de duas ou mais operações de DCJ [46]. Tanto a Ordenação de Permutações por DCJs [18] quanto a Distância de DCJs e Indels [23] possuem algoritmos polinomiais exatos.

Usando como base os resultados para a Distância de DCJs e Indels [23], Willing e coautores [83] criaram algoritmos polinomiais exatos para a Distância de Reversões e Indels em Strings com Sinais considerando classes específicas de grafos de *breakpoints*. Apenas recentemente, em 2020, Willing e coautores [82] estenderam o algoritmo anterior e desenvolveram um algoritmo exato polinomial que funciona para qualquer instância do problema da Distância de Reversões e Indels em Strings com Sinais.

Neste capítulo, apresentamos algoritmos de aproximação para a Distância de Rearranjos em Strings com ou sem Sinais para reversões, transposições, a combinação de reversões e transposições, *block interchanges*, e a combinação de reversões e *block interchanges*. Além disso, exceto para os modelos com *block interchanges* e o modelo com reversões para strings com sinais, apresentamos demonstrações que esses problemas são NP-difíceis.

4.1 Complexidade dos Problemas

Nesta seção, apresentamos provas de NP-dificuldade para problemas de Distância de Rearranjos em Genomas Desbalanceados considerando os seguintes modelos de rearranjos:

- $\mathcal{M}_{\rho}^{\phi,\psi} = \{\rho, \psi, \phi\}$: reversões e *indels* em strings sem sinais;
- $\mathcal{M}_{\tau}^{\phi,\psi} = \{\tau, \psi, \phi\}$: transposições e *indels* em strings sem sinais;

- $\mathcal{M}_{\rho,\tau}^{\phi,\psi} = \{\rho, \tau, \psi, \phi\}$: reversões, transposições, e *indels* em strings com ou sem sinais.

Lema 4.1.1. *O problema de Distância de Rearranjos é NP-difícil para os modelos $\mathcal{M}_{\rho}^{\phi,\psi}$ e $\mathcal{M}_{\tau}^{\phi,\psi}$, considerando strings sem sinais, e para o modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$, considerando strings com ou sem sinais.*

Demonstração. Considere o modelo $\mathcal{M}_{\rho}^{\phi,\psi}$. O problema da Ordenação de Permutações sem Sinais por Reversões (**SbR**) já foi provado ser NP-difícil [36]. A versão de decisão desse problema tem como entrada uma permutação π e um inteiro positivo k, consistindo em decidir se a permutação π pode ser ordenada por no máximo k reversões.

De forma similar, a versão de decisão do problema da Distância de Reversões e Indels em Strings sem Sinais (**RID**) recebe como entrada uma instância $\mathcal{I} = (A, \iota^n, k)$, e consiste em decidir se a string A pode ser transformada em ι^n usando no máximo k operações de reversões ou *indels*.

Nesta demonstração, apresentamos uma redução do problema **SbR** para o problema **RID**. Dada uma instância (π, k) para **SbR**, tal que π tem tamanho n, criamos a instância (A, ι^n, k), com $A = \pi$, para o problema **RID**. Note que π pode ser ordenada por uma sequência de reversões S com $|S| \leq k$ se, e somente se, A pode ser transformada em ι^n usando no máximo k operações do modelo $\mathcal{M}_{\rho}^{\phi,\psi}$, já que $\Sigma_A \setminus \Sigma_{\iota^n} = \Sigma_{\iota^n} \setminus \Sigma_A = \emptyset$ e *indels* não podem ser usados nas sequências de rearranjos.

A prova é similar para os outros modelos, já que a Ordenação de Permutações por Transposições [34] e a Ordenação de Permutações com ou sem Sinais por Reversões e Transposições [64] são NP-difíceis. □

4.2 Algoritmos de Aproximação Usando Breakpoints

Nesta seção, apresentamos algoritmos de aproximação para o problema de Distância de Rearranjos em Strings sem Sinais considerando os modelos $\mathcal{M}_{\rho}^{\phi,\psi}$, $\mathcal{M}_{\tau}^{\phi,\psi}$ e $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$. Sempre consideramos que as strings de uma instância $\mathcal{I} = (A, \iota^n)$ estão nas suas versões estendidas.

Usamos o conceito de *breakpoints*, apresentado na Seção 2.4.2, para a definição de limitantes para a distância e a criação dos algoritmos de aproximação. A seguir, apresentamos uma outra definição utilizada nos limitantes para a distância:

Definição 4.2.1. Dada uma operação (ou sequência de rearranjos) β e uma instância $\mathcal{I} = (A, \iota^n)$, definimos $\Delta\Phi(\mathcal{I}, \beta) = \Delta\Phi(A, \iota^n, \beta) = |\Sigma_{\iota^n} \setminus \Sigma_A| - |\Sigma_{\iota^n} \setminus \Sigma_{A'}|$, onde $A' = A \cdot \beta$.

Para uma operação ou sequência β, se $\Delta\Phi(\mathcal{I}, \beta) > 0$, então β diminui a quantidade de elementos que precisam ser adicionados para que as strings se tornem balanceadas.

Assim como nos problemas de Ordenação de Permutações por Rearranjos, usamos o conceito de *breakpoints* de reversões sem sinais para o modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$. Para simplificar a notação, usamos b_ρ e b_τ para indicar *breakpoints* de reversões sem sinais e *breakpoints* de transposições, respectivamente. Os próximos lemas mostram como um rearranjo afeta o valor de $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$.

Lema 4.2.1. *Para qualquer inserção ϕ e $\mathcal{I} = (A, \iota^n)$, temos que:*

$$\Delta\Phi(\mathcal{I}, \phi) + \Delta b_\rho(\mathcal{I}, \phi) \leq 2,$$
$$\Delta\Phi(\mathcal{I}, \phi) + \Delta b_\tau(\mathcal{I}, \phi) \leq 2.$$

Demonstração. Considere a inserção $\phi(i, \sigma)$ da string σ após o i-ésimo elemento de A e seja $A' = A \cdot \phi(i, \sigma)$.

Note que $|\sigma| = \Delta\Phi(\mathcal{I}, \phi)$. Além disso, o único *breakpoint* que pode ser removido é o *breakpoint* entre os elementos A_i e A_{i+1}, caso exista.

Lembramos que, como cada par de elementos adjacentes em σ representam segmentos maximais contíguos do genoma de destino, todo par de elementos distintos do conjunto $\Sigma_{\iota^n} \setminus \Sigma_A$ são não adjacentes em ι^n.

Se $|\sigma| = 1$, então o limitante é válido. Caso contrário, pelo menos $|\sigma| - 1$ *breakpoints* foram adicionados na string, já que existe um *breakpoint* entre cada par (σ_k, σ_{k+1}), com $1 \leq k < |\sigma|$. Portanto, temos que $\Delta\Phi(\mathcal{I}, \phi)+\Delta b_\rho(\mathcal{I}, \phi) \leq 2$ e $\Delta\Phi(\mathcal{I}, \phi)+\Delta b_\tau(\mathcal{I}, \phi) \leq 2$. □

Lema 4.2.2. *Para qualquer deleção ψ e $\mathcal{I} = (A, \iota^n)$, temos que:*

$$\Delta\Phi(\mathcal{I}, \psi) + \Delta b_\rho(\mathcal{I}, \psi) \leq 2,$$
$$\Delta\Phi(\mathcal{I}, \psi) + \Delta b_\tau(\mathcal{I}, \psi) \leq 2.$$

Demonstração. Considere a deleção $\psi(i, j)$ que remove o segmento $(A_i, \ldots, A_j)$. Note que todos os elementos de $(A_i, \ldots, A_j)$ possuem valor α. De acordo com as definições de *breakpoints* apresentadas na Seção 2.4.2, não existe *breakpoint* entre um par de elementos quando ambos são iguais a α. Dessa forma, não existem *breakpoints* entre elementos de $(A_i, \ldots, A_j)$. Assim, os únicos *breakpoints* que podem ser removidos, caso existam, estão nas posições (A_{i-1}, A_i) e (A_j, A_{j+1}). Como uma deleção não afeta o conjunto $|\Sigma_{\iota^n} \setminus \Sigma_A|$, os limitantes são válidos. □

Lema 4.2.3. *Para qualquer reversão ρ e $\mathcal{I} = (A, \iota^n)$, temos que:*

$$\Delta\Phi(\mathcal{I}, \rho) + \Delta b_\rho(\mathcal{I}, \rho) \leq 2.$$

Demonstração. Considere a reversão $\rho(i, j)$ e seja $A' = A \cdot \rho(i, j) = (A_1 \ \ldots \ A_{i-1} \ A_j \ \ldots \ A_i \ A_{j+1} \ \ldots \ A_n)$.

Para $i \leq k < j$, o par (A_k, A_{k+1}) é um *breakpoint* se, e somente se, o par $(A'_{k'}, A'_{k'+1})$ é um *breakpoint*, tal que $A_k = A'_{k'+1}$ e $A_{k+1} = A'_{k'}$. Dessa forma, essa reversão só pode remover *breakpoints* entre os pares de elementos (A_{i-1}, A_i) e (A_j, A_{j+1}), caso existam. Note que $\Delta\Phi(\mathcal{I}, \rho) = 0$, pois uma reversão não adiciona elementos. Portanto, no melhor cenário, dois *breakpoints* são removidos e o limitante é válido. □

Lema 4.2.4. *Para qualquer transposição τ e $\mathcal{I} = (A, \iota^n)$, temos que:*

$$\Delta\Phi(\mathcal{I}, \tau) + \Delta b_\rho(\mathcal{I}, \tau) \leq 3,$$
$$\Delta\Phi(\mathcal{I}, \tau) + \Delta b_\tau(\mathcal{I}, \tau) \leq 3.$$

Demonstração. Similar à prova do Lema 4.2.3, mas devemos considerar que uma transposição afeta três adjacências de A. □

Com esses lemas, podemos apresentar limitantes para a distância de rearranjos considerando os modelos $\mathcal{M}_{\rho}^{\phi,\psi}$, $\mathcal{M}_{\tau}^{\phi,\psi}$ e $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$.

Lema 4.2.5. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$ de strings sem sinais, temos que:*

$$d_{\mathcal{M}_{\rho}^{\phi,\psi}}(A, \iota^n) \geq \frac{b_\rho(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|}{2},$$
$$d_{\mathcal{M}_{\tau}^{\phi,\psi}}(A, \iota^n) \geq \frac{b_\tau(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|}{3},$$
$$d_{\mathcal{M}_{\rho,\tau}^{\phi,\psi}}(A, \iota^n) \geq \frac{b_\rho(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|}{3}.$$

Demonstração. Considere o modelo $\mathcal{M}_{\rho}^{\phi,\psi}$. Uma instância $\mathcal{I}' = (A', \iota^n)$ possui $b_\rho(\mathcal{I}') + |\Sigma_{\iota^n} \setminus \Sigma_{A'}| = 0$ se, e somente se, $A' = \iota^n$. Portanto, toda sequência de rearranjos que transforma A em ι^n deve diminuir o valor de $b_\rho(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ para 0. Pelos lemas 4.2.1, 4.2.2 e 4.2.3, qualquer reversão ou *indel* diminui esse valor em no máximo 2 e, portanto, o limitante para $d_{\mathcal{M}_{\rho}^{\phi,\psi}}(A, \iota^n)$ é válido.

A prova é similar para os modelos que contém transposições, mas considerando o limitante do Lema 4.2.4. □

Os algoritmos apresentados nesta seção são algoritmos gulosos que priorizam os rearranjos com maior valor de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta)$. Os próximos lemas apresentam casos em que sempre é possível achar um *indel* com $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$. Essas operações serão úteis nos três algoritmos de aproximação.

Lema 4.2.6. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que $|\Sigma_A \setminus \Sigma_{\iota^n}| > 0$, existe uma deleção ψ tal que $\Delta\Phi(\mathcal{I}, \psi) + \Delta b_{\mathcal{M}}(\mathcal{I}, \psi) \geq 1$.*

Demonstração. Seja $(A_i, \ldots, A_j)$ uma *strip* em A com $A_k = \alpha$, para $i \leq k \leq j$. Tal *strip* deve existir em A já que $|\Sigma_A \setminus \Sigma_{\iota^n}| > 0$. Por definição, uma *strip* é uma sequência maximal e, portanto, existem *breakpoints* entre ambos os pares (A_{i-1}, A_i) e (A_j, A_{j+1}).

A deleção $\psi(i, j)$ tem o seguinte efeito em A:

$$A = (A_1 \ \ldots \ A_{i-1} \ A_i \ \ldots \ A_j \ A_{j+1} \ \ldots \ A_n),$$
$$A' = A \cdot \psi(i, j) = (A_1 \ \ldots \ A_{i-1} \ A_{j+1} \ \ldots \ A_n).$$

Como o par (A_{i-1}, A_{j+1}) pode formar um *breakpoint*, o número de *breakpoints* diminui em pelo menos 1, enquanto o tamanho de $\Sigma_{\iota^n} \setminus \Sigma_A$ permanece o mesmo. Portanto, temos que $\Delta\Phi(\mathcal{I}, \psi) + \Delta b_{\mathcal{M}}(\mathcal{I}, \psi) \geq 1$. □

Lema 4.2.7. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que A não possui breakpoints e $|\Sigma_{\iota^n} \setminus \Sigma_A| > 0$, existe uma inserção ϕ tal que $\Delta\Phi(\mathcal{I}, \phi) + \Delta b_{\mathcal{M}}(\mathcal{I}, \phi) = 1$.*

Demonstração. Como A não possui *breakpoints*, todos os elementos de A estão em ordem crescente. Seja (A_i, A_{i+1}) um par de elementos tal que $A_i \neq A_{i+1} - 1$. Como $|\Sigma_{\iota^n} \setminus \Sigma_A| > 0$,

deve existir pelo menos um par de elementos no qual essa condição é válida. A inserção $\phi(i, \sigma)$, com $\sigma = (A_i + 1)$, diminui o tamanho de $\Sigma_{\iota^n} \setminus \Sigma_A$ em 1 e não altera o número de *breakpoints*. Portanto, $\Delta\Phi(\mathcal{I}, \phi) + \Delta b_{\mathcal{M}}(\mathcal{I}, \phi) = 1$. □

4.2.1 Algoritmo de 2-Aproximação para Modelo com Reversões e Indels

Apresentamos um algoritmo guloso com fator de aproximação igual a 2 para a Distância de Reversões e Indels em Strings sem Sinais. Nesta seção, consideramos que $\mathcal{M} = \mathcal{M}_{\rho}^{\phi,\psi}$. Os próximos lemas apresentam casos em que sempre é possível achar uma reversão que remove *breakpoints*.

Lema 4.2.8. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$, $|\Sigma_A \setminus \Sigma_{\iota^n}| = 0$, e A possui pelo menos uma strip decrescente, então existe uma reversão que remove pelo menos um breakpoint de A.*

Demonstração. Seja $(A_i, \ldots, A_j)$ uma *strip* decrescente tal que A_j é mínimo. Seja $(A_{i'}, \ldots, A_{j'})$ a *strip* contendo o elemento $\texttt{anterior}(A_j, \mathcal{I})$. Pela nossa escolha de A_j, a *strip* $(A_{i'}, \ldots, A_{j'})$ é crescente e $A_{j'} = \texttt{anterior}(A_j, \mathcal{I})$.

Se $i' < i$, então a reversão $\rho(j' + 1, j)$ remove o *breakpoint* entre as posições j' e $j' + 1$.

$$A = (0\ A_1\ \ldots\ A_{i'}\ \ldots\ A_{j'}\ A_{j'+1}\ \ldots\ A_i\ \ldots\ A_j\ A_{j+1}\ \ldots\ A_m\ n+1),$$
$$A \cdot \rho(j'+1, j) = (0\ A_1\ \ldots\ A_{i'}\ \ldots\ A_{j'}\ \underline{A_j\ \ldots\ A_i\ \ldots\ A_{j'+1}}\ A_{j+1}\ \ldots\ A_m\ n+1).$$

Caso contrário, temos que $i' > i$ e a reversão $\rho(j + 1, j')$ remove o *breakpoint* entre as posições j e $j + 1$.

$$A = (0\ A_1\ \ldots\ A_i\ \ldots\ A_j\ A_{j+1}\ \ldots\ A_{i'}\ \ldots\ A_{j'}\ A_{j'+1}\ \ldots\ A_m\ n+1),$$
$$A \cdot \rho(j+1, j') = (0\ A_1\ \ldots\ A_i\ \ldots\ A_j\ \underline{A_{j'}\ \ldots\ A_{i'}\ \ldots\ A_{j+1}}\ A_{j'+1}\ \ldots\ A_m\ n+1).$$

□

Lema 4.2.9. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$ tal que $b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$, $|\Sigma_A \setminus \Sigma_{\iota^n}| = 0$ e A possui pelo menos uma strip decrescente, se qualquer reversão que remove pelo menos um breakpoint de A resulta em uma string com apenas strips crescentes, então existe apenas uma reversão que remove breakpoints de A e essa reversão remove dois breakpoints.*

Demonstração. Suponha, por contradição, que existem duas reversões distintas $\rho(i, j)$ e $\rho(x, y)$ que removem *breakpoints* de A. Assuma, sem perda de generalidade, que $i < x$. Note que se ρ é uma reversão que deixa A sem *strips* decrescentes, então as *strips* no intervalo afetado por ρ devem ser todas *strips* decrescentes, já que uma *strip* crescente nesse intervalo seria transformada em uma *strip* decrescente em $A \cdot \rho$. As reversões $\rho(i, j)$ e $\rho(x, y)$ são distintas e, portanto, o intervalo $[i, j] - [x, y]$ é não vazio. Além disso, as *strips* contidas em $[i, j] - [x, y]$ são decrescentes, já que essas *strips* são afetadas por $\rho(i, j)$. Portanto, após aplicar a reversão $\rho(x, y)$, as *strips* decrescentes contidas em $[i, j] - [x, y]$ continuam sendo *strips* decrescentes em $A \cdot \rho(x, y)$, o que é uma contradição.

Nesse ponto, considere que $\rho(i,j)$ é a reversão que remove *breakpoints* de A e que $A' = A \cdot \rho(i,j)$. Todas as *strips* fora do intervalo $[i,j]$ são crescentes e, como mencionado, as *strips* em $[i,j]$ são decrescentes. Portanto, existem *breakpoints* nos pares (A_{i-1}, A_i) e (A_{j-1}, A_j).

Suponha, por contradição, que $\rho(i,j)$ remove apenas um único *breakpoint* de A. Sem perda de generalidade, assuma que o *breakpoint* entre as posições $i-1$ e i não é removido por $\rho(i,j)$. Seja $(A_x, \ldots, A_y)$ a *strip* em $[i,j]$ tal que A_y é mínimo, e seja $(A_{x'}, \ldots, A_{y'})$ a *strip* crescente que contém o elemento $\texttt{anterior}(A_y, \mathcal{I})$.

Se $x' > y$, então existe reversão que remove um *breakpoint* e deixa uma *strip* decrescente na string resultante, o que é uma contradição.

$$A = (0\ A_1\ \ldots\ A_x\ \ldots\ A_y\ A_{y+1}\ \ldots\ A_{x'}\ \ldots\ A_{y'}\ A_{y'+1}\ \ldots\ A_m\ n+1),$$
$$A \cdot \rho(y+1, y') = (0\ A_1\ \ldots\ A_x\ \ldots\ A_y\ \underline{A_{y'}\ \ldots\ A_{x'}\ \ldots\ A_{y+1}}\ A_{y'+1}\ \ldots\ A_m\ n+1)$$

Portanto, temos que $x' < x$ e $\rho(y'+1, y)$ é uma reversão que remove um *breakpoint* entre as posições y' e $y'+1$. Já que $\rho(i,j)$ é a única reversão que remove *breakpoints* de A, temos que $\rho(i,j) = \rho(y'+1, y)$ e o *breakpoint* entre as posições $y' = i-1$ e $y'+1 = i$ é removido, o que também é uma contradição. Portanto, concluímos que $\rho(i,j)$ remove dois *breakpoints* e o resultado enunciado neste lema é válido. □

O Algoritmo 4 é um algoritmo guloso que, a cada iteração, escolhe a operação β com valor máximo de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta)$. Se o algoritmo chega em um ponto onde não existem operações com $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$, então a string não possui *strips* decrescentes (Lema 4.2.8). Nesse caso, o algoritmo inverte a *strip* $(A_i, \ldots, A_j)$ tal que $i > 0$ e i é mínimo. Dessa forma, na próxima iteração existirá pelo menos uma *strip* decrescente na instância e, portanto, existe pelo menos uma operação β com $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$. O algoritmo termina quando todos os *breakpoints* forem removidos e o conjunto de rótulos das duas strings forem iguais.

Note que $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| \in O(n)$. Portanto, o loop do algoritmo executa $O(n)$ vezes. Para encontrar a operação com valor máximo de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta)$ temos que testar todas as possíveis combinações de reversões e *indels*, o que leva tempo $O(n^2)$. Portanto, o Algoritmo 4 possui complexidade de tempo de $O(n^3)$. Os próximos lemas e teoremas apresentam um limitante superior no número de operações usadas pelo algoritmo e uma prova para o fator de aproximação.

Lema 4.2.10. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que A possui pelo menos uma strip decrescente, o Algoritmo 4 transforma A em ι^n usando no máximo $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| - 1$ operações.*

Demonstração. Provaremos por indução no valor de $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ que o resultado é válido.

Note que não existe instância com $b_{\mathcal{M}}(\mathcal{I}) = 1$ de acordo com a definição de *breakpoints*. Além disso, como existe *strip* decrescente em A, então temos $b_{\mathcal{M}}(\mathcal{I}) > 0$. Portanto, como caso base, considere que $b_{\mathcal{M}}(\mathcal{I}) = 2$ e $|\Sigma_{\iota^n} \setminus \Sigma_A| \geq 0$. Sejam (A_i, A_{i+1}) e (A_j, A_{j+1}) os dois *breakpoints* em A. Essa string possui exatamente três *strips*: $(A_0, \ldots, A_i)$; $(A_{i+1}, \ldots, A_j)$;

Algoritmo 4: 2-Aproximação para a Distância de Reversões e Indels em Strings sem Sinais

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

```
Seja S ← ∅
enquanto b_M(I) + |Σ_{ι^n} \ Σ_A| > 0 faça
    Seja β a operação em M_ρ^{φ,ψ} com valor máximo de ΔΦ(I, β) + Δb_M(I, β)
    se ΔΦ(I, β) + Δb_M(I, β) > 0 então
        A ← A · β
        Adicione β na sequência S
    senão
        Seja (A_i, ..., A_j) uma strip tal que i > 0 e i é mínimo
        A ← A · ρ(i, j)
        Adicione ρ(i, j) na sequência S
retorne a sequência S
```

$(A_{j+1}, \ldots, A_{m+1})$, onde $|A| = m$. Como A possui uma *strip* decrescente, temos que a *strip* $(A_{i+1}, \ldots, A_j)$ é decrescente e a reversão $\rho(i+1, j)$ remove os dois *breakpoints* de A, já que após essa reversão todas as *strips* são crescentes (Lema 4.2.9). Quando $b_{\mathcal{M}}(\mathcal{I}) = 0$, pelo Lema 4.2.7, sempre existe uma inserção ϕ com $\Delta\Phi(\mathcal{I}, \beta) = 1$. Portanto, para transformar A em ι^n, o algoritmo usa $1 + |\Sigma_{\iota^n} \setminus \Sigma_A| = b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| - 1$ operações.

Suponha que o resultado é válido para qualquer instância $\mathcal{I} = (A, \iota^n)$ tal que A contém uma *strip* decrescente e $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| \leq k - 1$.

Para uma instância $\mathcal{I} = (A, \iota^n)$ tal que A contém uma *strip* decrescente e $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| = k$, temos os seguintes casos dependendo da operação β escolhida pelo algoritmo.

Caso 1: Suponha que a operação escolhida β é um *indel*. Uma operação de *indel* não torna *strips* decrescentes em crescentes e só é escolhida pelo algoritmo se $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$. Seja $\mathcal{I}' = (A', \iota^n)$ com $A' = A \cdot \beta$. Essa instância possui $b_{\mathcal{M}}(\mathcal{I}') + |\Sigma_{\iota^n} \setminus \Sigma_{A'}| = k' \leq k - 1$ e, pela hipótese de indução, o algoritmo transforma a string A' em ι^n usando no máximo $k' - 1$ operações. Portanto, o algoritmo usa no máximo $1 + (k' - 1) = k' \leq k - 1 = b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| - 1$ operações para transformar A em ι^n.

Caso 2: Suponha que β é uma reversão: Como A possui uma *strip* decrescente, pelo Lema 4.2.8, temos que $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta) > 0$. Seja $\mathcal{I}' = (A', \iota^n)$ com $A' = A \cdot \beta$. Essa instância possui $b_{\mathcal{M}}(\mathcal{I}') + |\Sigma_{\iota^n} \setminus \Sigma_{A'}| = k' \leq k - 1$.

Se A' possui pelo menos uma *strip* decrescente, então o algoritmo transforma A' em ι^n usando no máximo $k' - 1$ operações (hipótese de indução). Portanto, o algoritmo usa no máximo $1 + (k' - 1) = k' \leq k - 1 = b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| - 1$ operações para transformar A em ι^n.

Se A' não possui *strips* decrescentes, então $b_{\mathcal{M}}(\mathcal{I}') + |\Sigma_{\iota^n} \setminus \Sigma_{A'}| = b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| - 2 = k - 2 = k'$, já que dois *breakpoints* foram removidos por β (Lema 4.2.9). Na próxima iteração, o algoritmo escolhe uma reversão β' que inverte uma *strip* de A', gerando a string $A'' = A' \cdot \beta'$ que possui uma *strip* decrescente. Seja $\mathcal{I}'' = (A'', \iota^n)$ e note que $b_{\mathcal{M}}(\mathcal{I}') + |\Sigma_{\iota^n} \setminus \Sigma_{A'}| = b_{\mathcal{M}}(\mathcal{I}'') + |\Sigma_{\iota^n} \setminus \Sigma_{A''}| = k'$. Pela hipótese de indução, o algoritmo transforma A'' em ι^n usando no máximo $k' - 1$ operações. Portanto, o algoritmo transforma

A em ι^n usando no máximo $2 + k' - 1 = 1 + (k-2) = k - 1$ operações. □

Lema 4.2.11. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, o Algoritmo 4 transforma A em ι^n usando no máximo $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ operações.*

Demonstração. Se A possui uma *strip* decrescente, então o resultado desse lema segue diretamente do Lema 4.2.10. Caso contrário, usando um argumento similar ao usado no Caso 2 do Lema 4.2.10, temos que o algoritmo eventualmente usa uma reversão para tornar uma *strip* crescente em uma *strip* decrescente e, além disso, a string resultante A' é transformada em ι^n usando no máximo $b_{\mathcal{M}}(\mathcal{I}') + |\Sigma_{\iota^n} \setminus \Sigma_{A'}| - 1$ operações (Lema 4.2.10). Como todas as operações usadas antes dessa reversão satisfazem a condição $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta) \geq 1$, concluímos que o algoritmo usa no máximo $b_{\mathcal{M}}(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ operações para transformar A em ι^n. □

Teorema 4.2.1. *O Algoritmo 4 é uma 2-aproximação para o problema da Distância de Reversões e Indels em Strings sem Sinais.*

Demonstração. Segue diretamente dos lemas 4.2.5 e 4.2.11. □

4.2.2 Algoritmos de 3-Aproximação para Modelos com Transposições

Nesta seção, consideramos o problema da Distância de Transposições em Strings sem Sinais e o problema da Distância de Reversões e Transposições em Strings sem Sinais. Os próximos lemas apresentam casos em que sempre existe uma transposição que remove *breakpoints*.

Lema 4.2.12. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que $b_\tau(\mathcal{I}) > 0$ e $|\Sigma_A \setminus \Sigma_{\iota^n}| = 0$, existe uma transposição que remove pelo menos um breakpoint de A.*

Demonstração. Seja $(A_0, \ldots, A_i)$ a primeira *strip* de A. Seja $(A_j, \ldots, A_{k-1})$ a *strip* que contém o elemento $\texttt{posterior}(A_i, \mathcal{I})$. Note que como existem apenas *strips* crescentes ao considerar *breakpoints* de transposição, temos que $A_j = \texttt{posterior}(A_i, \mathcal{I})$. Portanto, a transposição $\tau(i+1, j, k)$ remove o *breakpoint* que existe entre A_i e A_{i+1}.

$$A = (A_1 \ \ldots \ A_i \ A_{i+1} \ \ldots \ A_{j-1} \ A_j \ \ldots \ A_{k-1} \ A_k \ \ldots \ A_m),$$
$$A \cdot \tau(i+1, j, k) = (A_1 \ \ldots \ A_i \ \underline{A_j \ \ldots \ A_{k-1}} \ \underline{A_{i+1} \ \ldots \ A_{j-1}} \ A_k \ \ldots \ A_m).$$

□

O Algoritmo 5 também é um algoritmo guloso que escolhe a operação β com valor máximo de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_\tau(\mathcal{I}, \beta)$. Para transposições e *indels*, os lemas 4.2.6, 4.2.7 e 4.2.12 garantem que sempre existe uma operação que satisfaz a condição $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_\tau(\mathcal{I}, \beta) > 0$. A complexidade de tempo do Algoritmo 5 é de $O(n^4)$, já que testar todas as possíveis transposições possui complexidade de tempo de $O(n^3)$ e o loop da linha 2 executa no máximo $O(n)$ vezes.

Algoritmo 5: 3-Aproximação para a Distância de Transposições e Indels em Strings sem Sinais

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

Seja $S \leftarrow \emptyset$
enquanto $b_\tau(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| > 0$ **faça**
    Seja β a operação em $\mathcal{M}_\tau^{\phi,\psi}$ com valor máximo de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_\tau(\mathcal{I}, \beta)$
    $A \leftarrow A \cdot \beta$
    Adicione β na sequência S
retorne a sequência S

Lema 4.2.13. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, o Algoritmo 5 transforma A em ι^n usando no máximo $b_\tau(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ operações.*

Demonstração. Segue diretamente do fato de que sempre existe uma operação que satisfaz a condição $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_\tau(\mathcal{I}, \beta) > 0$ (lemas 4.2.6, 4.2.7 e 4.2.12) e do fato de que $b_\tau(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| = 0$ se, e somente se, $A = \iota^n$. □

Teorema 4.2.2. *O Algoritmo 5 é uma 3-aproximação para o problema da Distância de Transposições e Indels em Strings sem Sinais.*

Demonstração. Segue diretamente dos lemas 4.2.5 e 4.2.13. □

Agora, consideramos o modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$. No próximo lema, mostramos que sempre é possível achar uma reversão ou transposição que remove *breakpoints* quando A não possui nenhum elemento com rótulo igual a α.

Lema 4.2.14. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que $b_\rho(\mathcal{I}) > 0$ e $|\Sigma_A \setminus \Sigma_{\iota^n}| = 0$, existe uma reversão ou transposição que remove pelo menos um breakpoint de A.*

Demonstração. Se A possui uma *strip* decrescente, então existe uma reversão que remove pelo menos um *breakpoint* em A (Lema 4.2.8). Caso contrário, a string A possui apenas *strips* crescentes e, pelo Lema 4.2.12, existe uma transposição que remove pelo menos um *breakpoint* em A. □

Algoritmo 6: 3-Aproximação para a Distância de Reversões, Transposições e Indels em Strings sem Sinais

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

Seja $S \leftarrow \emptyset$
enquanto $b_\rho(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A| > 0$ **faça**
    Seja β a operação em $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$ com valor máximo de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_\rho(\mathcal{I}, \beta)$
    $A \leftarrow A \cdot \beta$
    Adicione β na sequência S
retorne a sequência S

O Algoritmo 6 é similar ao Algoritmo 5, exceto pelo fato de que consideramos o modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$ e a definição de *breakpoints* de reversões sem sinais. A complexidade desse algoritmo também é $O(n^4)$.

Lema 4.2.15. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, o Algoritmo 6 transforma A em ι^n usando no máximo $b_\rho(\mathcal{I}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ operações.*

Demonstração. Similar à prova do Lema 4.2.13. □

Teorema 4.2.3. *O Algoritmo 6 é uma 3-aproximação para o problema da Distância de Reversões, Transposições e Indels em Strings sem Sinais.*

Demonstração. Segue diretamente dos lemas 4.2.5 e 4.2.15. □

4.3 Algoritmos de Aproximação Usando Grafo de Ciclos Rotulado

Nesta seção, apresentamos algoritmos de 2-aproximação para os seguintes modelos:

- $\mathcal{M}_{\tau}^{\phi,\psi} = \{\tau, \psi, \phi\}$: transposições e *indels* em strings sem sinais;
- $\mathcal{M}_{\mathcal{BI}}^{\phi,\psi} = \{\mathcal{BI}, \psi, \phi\}$: *block interchanges* e *indels* em strings sem sinais;
- $\mathcal{M}_{\rho,\tau}^{\phi,\psi} = \{\rho, \tau, \psi, \phi\}$: reversões, transposições, e *indels* em strings com sinais;
- $\mathcal{M}_{\rho,\mathcal{BI}}^{\phi,\psi} = \{\rho, \mathcal{BI}, \psi, \phi\}$: reversões, *block interchanges*, e *indels* em strings com sinais.

Esses algoritmos e os limitantes inferiores apresentados nesta seção utilizam os conceitos relacionados ao grafo de ciclos rotulado (Seção 2.5.2). Sempre consideramos que as strings de uma instância $\mathcal{I} = (A, \iota^n)$ e suas formas simplificadas (π^A e π^ι) estão nas suas versões estendidas.

Além das definições de ciclos limpos e rotulados, que são conceitos específicos de grafo de ciclos rotulados, a seguir introduzimos o conceito de *runs*. Os *runs* são úteis na definição de limitantes inferiores e na criação de algoritmos.

Um *run de inserção* é um caminho maximal que inicia e termina com arestas de destino rotuladas e toda aresta de origem nesse caminho é uma aresta limpa. De forma similar, um *run de deleção* é um caminho maximal que inicia e termina com arestas de origem rotuladas e toda aresta de destino nesse caminho é uma aresta limpa. O número de *runs* (inserção ou deleção) de um ciclo C é denotado por $\Lambda(C)$.

Lema 4.3.1. *Para toda instância $\mathcal{I} = (A, \iota^n)$ e grafo $G(\mathcal{I})$, qualquer ciclo $C \in G(\mathcal{I})$ possui zero, um, ou um número par de runs.*

Demonstração. Suponha, por contradição, que um ciclo C em $G(\mathcal{I})$ possui x *runs*, tal que $x > 1$ e x é ímpar. Sejam $r_1, r_2, \ldots, r_x$ os *runs* de C na ordem que eles são percorridos no ciclo, considerando que começamos da aresta de origem mais à direita, ou seja, da aresta de origem com maior valor de índice. Pela definição de um *run*, para qualquer $1 \leq i \leq x$, os *runs* r_i e r_j devem ser de tipos diferentes, onde $j = (i \mod x) + 1$. No entanto, como x é um número ímpar maior que 1 e existem apenas dois tipos de *runs*, r_1 e r_x devem ser do mesmo tipo, o que é uma contradição. □

Dizemos que uma inserção ϕ remove um *run* de inserção r se todas as arestas de destino rotuladas de r são transformadas em arestas limpas após a aplicação da inserção ϕ. De forma similar, dizemos que uma deleção ψ remove um *run* de deleção r se todas as arestas de origem rotuladas de r se tornam arestas limpas após a aplicação da deleção ψ.

Note que para cada elemento inserido na string A, temos que um par de vértices, uma aresta de origem e uma aresta de destino são inseridos no grafo de ciclos rotulado. Portanto, uma inserção $\phi(i, \sigma)$ adiciona $2|\sigma|$ vértices, $|\sigma|$ arestas de origem e $|\sigma|$ arestas de destino no grafo. No entanto, uma deleção afeta apenas o rótulo de uma única aresta de origem, já que uma deleção só pode ser aplicada em uma sequência contígua de elementos iguais a α. Dessa forma, um *run* de deleção pode ser removido somente se esse *run* corresponde a uma única aresta de origem rotulada.

O *potencial de indels* de um ciclo C relaciona o número de *runs* em um ciclo e o número de *indels* necessários para remover todos os *runs* desse ciclo.

Definição 4.3.1. Dado um ciclo C, o *potencial de indels* de C é igual a:

$$\lambda(C) = \begin{cases} \left\lceil \frac{\Lambda(C)+1}{2} \right\rceil, & \text{se } \Lambda(C) > 0 \\ 0, & \text{caso contrário.} \end{cases}$$

Definimos $\lambda(\mathcal{I}) = \lambda(A, \iota^n) = \sum_{C \in G(A,\iota^n)} \lambda(C)$. Dado um rearranjo (ou sequência de rearranjos) β, denotamos $\Delta\lambda(\mathcal{I}, \beta) = \Delta\lambda(A, \iota^n, \beta) = \lambda(A, \iota^n) - \lambda(A \cdot \beta, \iota^n)$.

Para um ciclo C com *runs* $r_1, r_2, \ldots, r_x$, com $x > 2$, ao remover um *run* r_i de C, os seus *runs* adjacentes r_j e r_k formam um único *run* no novo ciclo, onde $k = (i \mod x) + 1$ e $j = i - 1$, se $i > 1$, ou $j = x$, se $i = 1$. Dessa forma, uma operação β que remove um *run* de um ciclo tem $\Delta\lambda(\mathcal{I}, \beta) = 1$.

Definição 4.3.2. O *potencial de inserções* $\lambda_\phi(C)$ de um ciclo C é denotado por:

$$\lambda_\phi(C) = \begin{cases} \lambda(C) - 1, & \text{se } \Lambda(C) > 1 \\ 1, & \text{se } C \text{ possui apenas um } \textit{run} \text{ de inserção} \\ 0, & \text{se } C \text{ não possui } \textit{runs} \text{ de inserção.} \end{cases}$$

Além disso, definimos $\lambda_\phi(\mathcal{I}) = \lambda_\phi(A, \iota^n) = \sum_{C \in G(A,\iota^n)} \lambda_\phi(C)$. Dado um rearranjo (ou sequência de rearranjos) β, denotamos $\Delta\lambda_\phi(\mathcal{I}, \beta) = \Delta\lambda_\phi(A, \iota^n, \beta) = \lambda_\phi(A, \iota^n) - \lambda_\phi(A \cdot \beta, \iota^n)$.

Lembre que, para um grafo de ciclos rotulado $G(\mathcal{I})$, as definições de ciclos limpos e ciclos rotulados consideram apenas as arestas de origem. Dessa forma, chegamos aos resultados das observações 4.3.1 e 4.3.2.

Observação 4.3.1. Para todo ciclo limpo C, temos que $\lambda_\phi(C) = \lambda(C)$ e, para qualquer ciclo rotulado C', temos que $\lambda_\phi(C') = \lambda(C') - 1$.

Observação 4.3.2. Sejam H e H' os conjuntos de ciclos limpos e de ciclos rotulados de

$G(\mathcal{I})$, respectivamente. Temos que:

$$\begin{aligned}\lambda(\mathcal{I}) &= \sum_{C\in H}\lambda(C) + \sum_{C\in H'}\lambda(C)\\ &= \sum_{C\in H}\lambda_\phi(C) + \sum_{C\in H'}(\lambda_\phi(C)+1)\\ &= \left(\sum_{C\in H}\lambda_\phi(C) + \sum_{C\in H'}\lambda_\phi(C)\right) + c_{\text{labeled}}(\mathcal{I})\\ &= \lambda_\phi(\mathcal{I}) + c_{\text{labeled}}(\mathcal{I}).\end{aligned}$$

Dessa forma,

$$\begin{aligned}&|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I}) =\\ &|\pi^A| + 1 - (c_{\text{clean}}(\mathcal{I}) + c_{\text{labeled}}(\mathcal{I})) + (\lambda_\phi(\mathcal{I}) + c_{\text{labeled}}(\mathcal{I})) =\\ &|\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_\phi(\mathcal{I}).\end{aligned}$$

O grafo $G(\mathcal{I})$ só possui ciclos unitários e potencial de *indel* igual a zero se, e somente se, $A = \iota^n$. Em outras palavras, $A = \iota^n$ se, e somente se, $n + 1 - c(\mathcal{I}) + \lambda(\mathcal{I}) = n + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_\phi(\mathcal{I}) = 0$. Dessa forma, transformar A em ι^n pode ser interpretado como tornar $|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I}) = |\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_\phi(\mathcal{I}) = 0$. Observe que quando $A = \iota^n$, temos que $|\pi^A| = n$.

Agora, apresentamos limitantes para o valor de $\Delta c(\mathcal{I}, \beta) + \Delta\lambda(\mathcal{I}, \beta)$ dependendo do tipo do rearranjo β.

Lema 4.3.2. *Para qualquer deleção ψ e instância $\mathcal{I} = (A, \iota^n)$, temos que $\Delta c(\mathcal{I}, \psi) + \Delta\lambda(\mathcal{I}, \psi) \leq 1$.*

Demonstração. Uma deleção pode apenas remover o rótulo de uma aresta de origem, já que deve ser aplicada a uma sequência contígua de elementos α. No melhor cenário, essa operação remove um *run* de deleção que é formado por apenas uma aresta de origem rotulada e diminui o potencial de *indel* em uma unidade, ou seja, $\Delta c(\mathcal{I}, \psi) = 1$. Os ciclos e vértices de $G(\mathcal{I})$ não são afetados e, portanto, $\Delta c(\mathcal{I}, \psi) = 0$. □

Lema 4.3.3. *Para qualquer inserção ϕ e instância $\mathcal{I} = (A, \iota^n)$, temos que $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) \leq 1$.*

Demonstração. Considere uma inserção que remove o rótulo de arestas de destino de um mesmo ciclo C. Se as arestas afetadas pertencem ao mesmo *run* de inserção, no melhor cenário, esse *run* é removido e $\Delta\lambda(\mathcal{I}, \phi) = 1$. No melhor cenário, um novo ciclo é criado para cada aresta de origem inserida no grafo e $\Delta c(\mathcal{I}, \phi) = 0$.

Se as arestas afetadas pertencem a x diferentes *runs* de um mesmo ciclo C, no melhor cenário, x *runs* de inserção são removidos, mas essa inserção cria pelo menos x ciclos que possuem *runs* de deleção, como mostrado na Figura 4.1. Sejam $D_1, D_2, \ldots, D_y$ os ciclos criados pela inserção que possuem exatamente um *run*, e sejam $D'_1, D'_2, \ldots, D'_z$ os ciclos criados pela inserção que possuem mais que um *run*. Pelo Lema 4.3.1, $\Lambda(D'_i)$ é par para

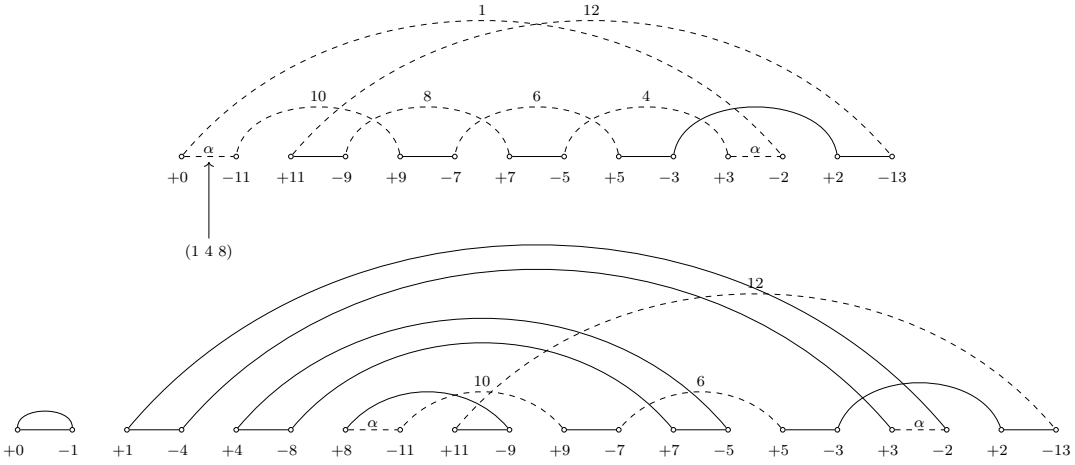


Figura 4.1: Exemplo de uma inserção que remove o rótulo de arestas de destino de diferentes *runs* de um mesmo ciclo. Neste exemplo, temos $A = (0\ \alpha\ 11\ 9\ 7\ 5\ 3\ \alpha\ 2\ 13)$ e ι^n com $n = 12$. A operação aplicada em A é a inserção $\phi(0, (1\ 4\ 8))$.

todo $1 \leq i \leq z$. Note que $y + z \geq x$ e que $\Lambda(C) = x + \sum_{1 \leq i \leq y} \Lambda(D_i) + \sum_{1 \leq i \leq z} \Lambda(D'_i) = x + y + \sum_{1 \leq i \leq z} \Lambda(D'_i)$. A soma do potencial de *indel* dos novos ciclos é:

$$\begin{aligned}
&\sum_{1 \leq i \leq y} \left\lceil \frac{\Lambda(D_i) + 1}{2} \right\rceil + \sum_{1 \leq i \leq z} \left\lceil \frac{\Lambda(D'_i) + 1}{2} \right\rceil \\
&= y + \sum_{1 \leq i \leq z} \left(\frac{\Lambda(D'_i)}{2} + 1 \right) \ = \ y + z + \frac{\sum_{1 \leq i \leq z} \Lambda(D'_i)}{2} \\
&= \frac{2y + 2z + \sum_{1 \leq i \leq z} \Lambda(D'_i)}{2} \geq \ \frac{y + z + x + \sum_{1 \leq i \leq z} \Lambda(D'_i)}{2} \\
&= \frac{\Lambda(C) + z}{2} \geq \ \frac{\Lambda(C)}{2} \geq \lambda(C) - 1.
\end{aligned}$$

Portanto, temos que $\Delta\lambda(\mathcal{I}, \phi) \leq 1$. Como anteriormente, no melhor cenário, um novo ciclo é criado para cada elemento inserido no grafo e $\Delta c(\mathcal{I}, \phi) = 0$.

Agora, considere uma inserção que remove o rótulo de arestas de destino de k ciclos distintos. Se um *run* de inserção é removido de cada ciclo, então $\Delta\lambda(\mathcal{I}, \phi) = k$. No entanto, esses ciclos são unidos pelos novos elementos adicionados no grafo e, consequentemente, $\Delta c(\mathcal{I}, \phi) \leq -(k - 1)$. Portanto, temos que $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) \leq 1$. A Figura 4.2 mostra um exemplo de tal operação. Quando mais de um *run* é removido de cada ciclo, um argumento similar ao usado anteriormente pode ser usado. □

Lema 4.3.4. *Para qualquer block interchange $\mathcal{BI}$ e instância $\mathcal{I} = (A, \iota^n)$, temos que $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) \leq 2$.*

Demonstração. Dividimos essa demonstração de acordo com o número de ciclos afetados por $\mathcal{BI}$ [38].

Se $\mathcal{BI}$ afeta quatro ciclos C_1, C_2, C_3 e C_4, então essa operação transforma esses quatro ciclos em dois novos ciclos C'_1 e C'_2. Assim, temos que $\Delta c(\mathcal{I}, \mathcal{BI}) = -2$. No melhor cenário,

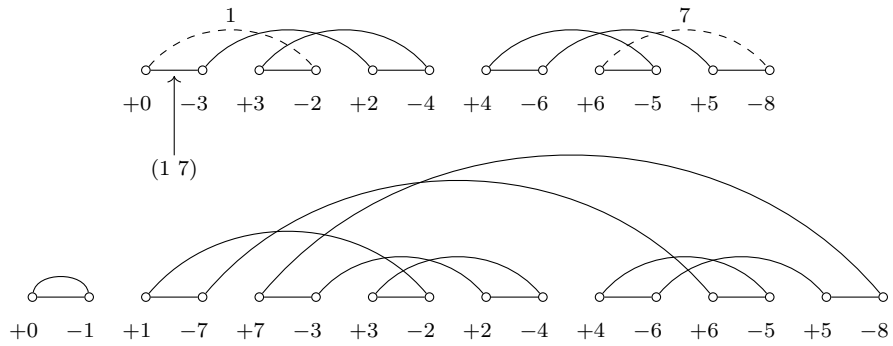


Figura 4.2: Exemplo de uma inserção que remove o rótulo de arestas de destino de diferentes ciclos. Neste exemplos, temos $A = (0\ 3\ 2\ 4\ 6\ 5\ 8)$ e ι^n com $n = 7$. A operação aplicada em A é a inserção $\phi(0, (1\ 7))$.

dois *runs* de deleção e dois *runs* de inserção de C_1 e C_3 são unidos em C'_1. Analogamente, dois *runs* de deleção e dois *runs* de inserção de C_2 e C_4 são unidos em C'_2. Neste caso, $\Lambda(C'_1) = \Lambda(C_1) + \Lambda(C_3) - 2$ e $\Lambda(C'_2) = \Lambda(C_2) + \Lambda(C_4) - 2$. Portanto, $\Delta\lambda(A, \iota^n, \mathcal{BI}) = 4$ e $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) = 2$. Um exemplo é apresentado na Figura 4.3.

Se $\mathcal{BI}$ afeta três ciclos C_1, C_2 e C_3, então essa operação une esses três ciclos em um novo ciclo C'. Assim, temos que $\Delta c(\mathcal{I}, \mathcal{BI}) = -2$. De forma similar ao caso anterior, no melhor cenário, o número de *runs* diminui em quatro e $\Lambda(C') = \Lambda(C_1) + \Lambda(C_2) + \Lambda(C_3) - 4$. Portanto, $\Delta\lambda(A, \iota^n, \mathcal{BI}) = 4$ e $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) = 2$.

Se $\mathcal{BI}$ afeta dois ciclos C_1 e C_2, então essa operação transforma esses dois ciclos em dois novos ciclos ou em quatro novos ciclos. Se $\mathcal{BI}$ transforma C_1 e C_2 em dois novos ciclos C'_1 e C'_2, então, no melhor cenário, o número de *runs* diminui em quatro com $\Lambda(C'_1) = \Lambda(C_1) - 2$ e $\Lambda(C'_2) = \Lambda(C_1) - 2$. Dessa forma, o potencial de *indel* diminui em uma unidade para cada ciclo e $\Delta\lambda(\mathcal{I}, \mathcal{BI}) = 2$. Como $\Delta c(\mathcal{I}, \mathcal{BI}) = 0$, temos que $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) = 2$.

Se $\mathcal{BI}$ afeta dois ciclos C_1 e C_2, transformando esses ciclos em quatro novos ciclos C'_1, C'_2, C'_3 e C'_4, então, no melhor cenário, dois pares de *runs* de deleção são unidos, porém note que cada ciclo possui pelo menos um *run* de inserção, como mostrado na Figura 4.4. Portanto, $\Lambda(C'_1) = X$, tal que $1 \leq X < \Lambda(C_1)$, $\Lambda(C'_2) = \min(\Lambda(C_1) - X - 2, 1)$, $\Lambda(C'_3) = Y$, tal que $1 \leq Y < \Lambda(C_2)$, e $\Lambda(C'_4) = \min(\Lambda(C_2) - Y - 2, 1)$. Portanto, o potencial de *indel* do grafo permanece o mesmo ($\Delta\lambda(\mathcal{I}, \mathcal{BI}) = 0$) e $\Delta c(\mathcal{I}, \mathcal{BI}) = 2$.

Se $\mathcal{BI}$ afeta um único ciclo C_1, então essa operação transforma esse ciclo em um novo ciclo ou em três novos ciclos. Se $\mathcal{BI}$ não aumenta o número de ciclos do grafo, então, no melhor cenário, essa operação pode diminuir o número de *runs* do ciclo em quatro e $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) = 2$. Se $\mathcal{BI}$ transforma C_1 em três novos ciclos C'_1, C'_2 e C'_3, então, no melhor cenário, dois pares de *runs* de deleção são unidos, mas note que cada ciclo possui pelo menos um *run* de inserção. De forma similar ao caso anterior, o potencial de *indel* do grafo permanece o mesmo e $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) = 2$. □

Lema 4.3.5. *Para qualquer transposição τ e instância $\mathcal{I} = (A, \iota^n)$, temos que $\Delta c(\mathcal{I}, \tau) + \Delta\lambda(\mathcal{I}, \tau) \leq 2$.*

Demonstração. Segue diretamente do Lema 4.3.4 e do fato de que uma transposição é

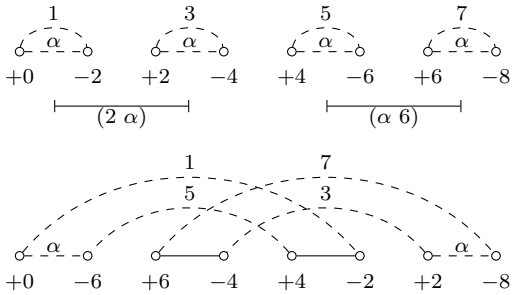


Figura 4.3: Exemplo de uma operação de *block interchange* que age em quatro ciclos. Neste exemplo, temos $A = (0\ \alpha\ 2\ \alpha\ 4\ \alpha\ 6\ \alpha\ 8)$ e ι^n com $n = 7$. O potencial de *indel* do grafo original é igual a $4 \times \lceil (2+1)/2 \rceil = 8$ e o potencial de *indel* do novo grafo é igual a $\lceil (2+1)/2 \rceil + \lceil (2+1)/2 \rceil = 4$.

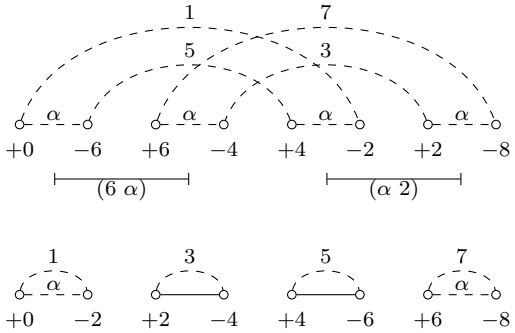


Figura 4.4: Exemplo de uma operação de *block interchange* que age em dois ciclos e cria quatro novos ciclos. Neste exemplo, temos $A = (0\ \alpha\ 6\ \alpha\ 4\ \alpha\ 2\ \alpha\ 8)$ e ι^n com $n = 7$. O potencial de *indel* do grafo original é igual a $\lceil (4+1)/2 \rceil + \lceil (4+1)/2 \rceil = 6$ e o potencial de indel no novo grafo é igual a $\lceil (2+1)/2 \rceil + \lceil (1+1)/2 \rceil + \lceil (1+1)/2 \rceil + \lceil (2+1)/2 \rceil = 6$.

um caso particular de *block interchange* que troca a posição relativa de dois segmentos adjacentes. □

Lema 4.3.6. *Para qualquer reversão ρ e instância $\mathcal{I} = (A, \iota^n)$, temos que $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) \leq 1$.*

Demonstração. Bafna e Pevzner [16] mostraram que uma reversão ρ modifica um grafo de ciclos das seguintes formas:

- Se a reversão ρ afeta duas arestas de origem de dois ciclos C_1 e C_2, então ρ une esses dois ciclos em um ciclo C' e, portanto, $\Delta c(\mathcal{I}, \rho) = -1$.
- Se a reversão ρ afeta duas arestas de um mesmo ciclo C, então ou (i) ρ cria um novo ciclo C' com os mesmos vértices que C ou (ii) ρ transforma C em dois ciclos C'_1 e C'_2. No subcaso (i) temos $\Delta c(\mathcal{I}, \rho) = 0$ e no subcaso (ii) temos $\Delta c(\mathcal{I}, \rho) = 1$.

Suponha que ρ afeta arestas de origem de dois ciclos C_1 e C_2 unindo esses dois ciclos em um ciclo C'. No melhor cenário, dois *runs* de deleção e dois *runs* de inserção de C_1 e C_2 são unidos em C'. Assim, $\Lambda(C') = \Lambda(C_1) + \Lambda(C_2) - 2$. Como nesse cenário existem

pelo menos um *run* de inserção e um *run* de deleção em cada um dos ciclos C_1, C_2, e C', temos que:

$$\begin{aligned}
\lambda(C') &= \left\lceil \frac{\Lambda(C_1) + \Lambda(C_2) - 2 + 1}{2} \right\rceil \\
&= \left\lceil \frac{\Lambda(C_1) + \Lambda(C_2) - 1}{2} \right\rceil \\
&= \frac{\Lambda(C_1) + \Lambda(C_2)}{2} \\
&= \left\lceil \frac{\Lambda(C_1) + 1}{2} \right\rceil - 1 + \left\lceil \frac{\Lambda(C_2) + 1}{2} \right\rceil - 1 \\
&= \lambda(C_1) + \lambda(C_2) - 2.
\end{aligned}$$

Lembre-se que $\Lambda(C)$ é par sempre que $\Lambda(C) \geq 2$ (Lema 4.3.1). Sendo assim, temos $\Delta\lambda(\mathcal{I}, \rho) = 2$ e $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) = 1$.

Suponha que ρ afeta duas arestas de um mesmo ciclo C e transforma esse ciclo em um ciclo C'. No melhor cenário, dois *runs* de deleção e dois *runs* de inserção são unidos e, portanto, $\Lambda(C') = \Lambda(C) - 2$ e $\lambda(C') = \lambda(C) - 1$. Sendo assim, $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) = 1$.

Por último, suponha que ρ afeta duas arestas de um mesmo ciclo C e transforma esse ciclo em dois ciclos C'_1 e C'_2. No melhor cenário, dois *runs* de deleção são unidos, mas note que cada ciclo C'_1 e C'_2 possui pelo menos um *run* de inserção. Suponha, sem perda de generalidade, que os *runs* de deleção unidos estão em C'_1. Sendo assim, $\Lambda(C'_1) = X$, tal que $2 \leq X < \Lambda(C)$, e $\Lambda(C'_2) = \min(\Lambda(C) - X - 2, 1)$.

Note que X é par. Suponha que $\min(\Lambda(C) - X - 2, 1) = \Lambda(C) - X - 2$. Como nesse cenário C possui pelo menos dois *runs* de inserção e dois *runs* de deleção, temos que:

$$\begin{aligned}
\lambda(C'_1) + \lambda(C'_2) &= \left\lceil \frac{\Lambda(C'_1) + 1}{2} \right\rceil + \left\lceil \frac{\Lambda(C'_2) + 1}{2} \right\rceil \\
&= \left\lceil \frac{X + 1}{2} \right\rceil + \left\lceil \frac{\Lambda(C) - X - 2 + 1}{2} \right\rceil \\
&= \frac{X + 2}{2} + \left\lceil \frac{\Lambda(C) - X - 1}{2} \right\rceil \\
&= \left\lceil \frac{X + 2 + \Lambda(C) - X - 1}{2} \right\rceil \\
&= \left\lceil \frac{X + 2 + \Lambda(C) - X - 1}{2} \right\rceil \\
&= \left\lceil \frac{\Lambda(C) + 1}{2} \right\rceil \\
&= \lambda(C).
\end{aligned}$$

Se $\min(\Lambda(C) - X - 2, 1) = 1$, então $X = \Lambda(C) - 2$ e temos que:

$$\begin{aligned}
\lambda(C'_1) + \lambda(C'_2) &= \left\lceil \frac{\Lambda(C'_1) + 1}{2} \right\rceil + \left\lceil \frac{\Lambda(C'_2) + 1}{2} \right\rceil \\
&= \frac{X + 2}{2} + \left\lceil \frac{1 + 1}{2} \right\rceil \\
&= \frac{(\Lambda(C) - 2) + 2}{2} + 1 \\
&= \frac{\Lambda(C)}{2} + 1 \\
&= \lambda(C).
\end{aligned}$$

Portanto, temos que $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) = 1$. □

Com esses resultados, podemos definir limitantes inferiores para o valor de $d_{\mathcal{M}}(\mathcal{I})$, considerando $\mathcal{M} = \{\mathcal{M}^{\phi,\psi}_{\tau}, \mathcal{M}^{\phi,\psi}_{\mathcal{BI}}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}, \mathcal{M}^{\phi,\psi}_{\rho,\mathcal{BI}}\}$.

Lema 4.3.7. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, temos que*

$$\begin{aligned}
d_{\mathcal{M}^{\phi,\psi}_{\mathcal{BI}}}(\mathcal{I}) &\geq \left\lceil \frac{|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I})}{2} \right\rceil, \\
d_{\mathcal{M}^{\phi,\psi}_{\tau}}(\mathcal{I}) &\geq \left\lceil \frac{|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I})}{2} \right\rceil, \\
d_{\mathcal{M}^{\phi,\psi}_{\rho,\mathcal{BI}}}(\mathcal{I}) &\geq \left\lceil \frac{|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I})}{2} \right\rceil, \\
d_{\mathcal{M}^{\phi,\psi}_{\rho,\tau}}(\mathcal{I}) &\geq \left\lceil \frac{|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I})}{2} \right\rceil.
\end{aligned}$$

Demonstração. Considere o modelo $\mathcal{M}^{\phi,\psi}_{\mathcal{BI}}$. Como $|\pi^{A'}|+1-c(A', \iota^n)+\lambda(A', \iota^n) = 0$ somente se $A' = \iota^n$, qualquer sequência de rearranjos S que transforma A em ι^n deve tornar o valor $|\pi^A|+1-c(A, \iota^n)+\lambda(A, \iota^n)$ igual a zero. Pelos lemas 4.3.2, 4.3.3 e 4.3.4, qualquer rearranjo β em $\mathcal{M}^{\phi,\psi}_{\mathcal{BI}}$ satisfaz $\Delta c(\mathcal{I}, \beta) + \Delta\lambda(\mathcal{I}, \beta) \leq 2$ e, portanto, $|S| \geq \left\lceil \frac{|\pi^A|+1-c(\mathcal{I})+\lambda(\mathcal{I})}{2} \right\rceil$.

A demonstração é similar para os outros modelos e também considera os resultados dos lemas 4.3.5 e 4.3.6. □

Lema 4.3.8. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, temos que*

$$\begin{aligned}
d_{\mathcal{M}^{\phi,\psi}_{\tau}}(\mathcal{I}) &\geq \left\lceil \frac{|\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_{\phi}(\mathcal{I})}{2} \right\rceil, \\
d_{\mathcal{M}^{\phi,\psi}_{\rho,\tau}}(\mathcal{I}) &\geq \left\lceil \frac{|\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_{\phi}(\mathcal{I})}{2} \right\rceil.
\end{aligned}$$

Demonstração. Segue diretamente do Lema 4.3.7 e das observações 4.3.1 e 4.3.2. □

4.3.1 Algoritmos de 2-Aproximação para Modelos com Block Interchanges

Nesta seção, apresentamos algoritmos de 2-aproximação para o problema de Distância de Rearranjos considerando os modelos $\mathcal{M}_{\mathcal{BI}}^{\phi,\psi}$ e $\mathcal{M}_{\rho,\mathcal{BI}}^{\phi,\psi}$. Os próximos lemas apresentam operações que diminuem o valor de $\lambda(\mathcal{I})$ ou que aumentam o número de ciclos, dependendo de características do grafo de ciclos rotulado $G(\mathcal{I})$.

Lema 4.3.9. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $G(\mathcal{I})$ possui pelo menos um run de inserção, então existe uma inserção ϕ com $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) = 1$.*

Demonstração. Considere o *run* de inserção $(v_1, v_2, \ldots, v_j)$ de um ciclo C, tal que v_1 possui o mesmo sinal que o seu elemento correspondente em A, ou seja, existe um elemento A_i em A tal que $v_1 = A_i$. Note que (v_1, v_2) é uma aresta de destino rotulada pela definição de *runs*. Sejam $o_1, o_2, \ldots, o_k$ valores inteiros tal que (v_{o_i}, v_{o_i+1}) é a i-ésima aresta de destino rotulada desse *run*.

Construímos $\sigma = (x_1, x_2, \ldots, x_k)$ da seguinte forma: para cada $1 \leq i \leq k$, se v_{o_i+1} possui sinal "$-$", então $x_i = \ell((v_{o_i}, v_{o_i+1}))$; caso contrário, $x_i = -\ell((v_{o_i}, v_{o_i+1}))$. A inserção de σ após o elemento de A correspondente ao vértice v_1 remove o *run* e adiciona k ciclos no grafo.

Um ciclo unitário é criado com os vértices $(v_1, -x_1)$. Para cada elemento x_i, com $1 \leq i < k$, existe um ciclo $(+x_i, v_{o_i+1}, v_{o_i+2}, \ldots, v_{o_{i+1}}, -x_{i+1}, +x_i)$. O último vértice $+x_k$ pertence ao que sobrou do ciclo C ou $+x_k$ pertence a um ciclo unitário, no caso em que todas as arestas de destino de C pertencem ao *run* que foi removido pela inserção. Um exemplo dessa operação é apresentado na Figura 4.5.

Se $\Lambda(C) \leq 2$, então a remoção de um *run* de C reduz tanto o número de *runs* quanto o potencial de *indel* em um. Caso contrário, ao remover o *run* de inserção, dois *runs* de deleção são unidos. Nesse caso, o número de *runs* de C diminui em dois e o potencial de *indel* diminui em um. Como a inserção adiciona k elementos em A e k ciclos limpos no grafo, temos que $\Delta c(\mathcal{I}, \phi) = 0$. Portanto, $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) = 1$. □

Lema 4.3.10. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui arestas de destino limpas e $G(\mathcal{I})$ não possui ciclos divergentes, então existe block interchange $\mathcal{BI}$ tal que $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq 1$.*

Demonstração. Considere que $G(\mathcal{I})$ possui um ciclo orientado $C = (o_1, \ldots, o_\ell)$. Nesse caso, sempre existe tripla orientada o_i, o_j e o_k, com $i < j < k$, tal que $o_i > o_k > o_j$ e $k = j + 1$ [17]. Uma operação de *block interchange* aplicada nessas três arestas de origem cria três ciclos C', C'' e C''' [17]. Sejam σ_1 e σ_2 os dois segmentos afetados por esse *block interchange*. Se todas as arestas de origem afetadas são rotuladas, podemos mover os elementos α de forma que esses rótulos fiquem todos no mesmo ciclo. Para fazer isso, incluímos em σ_1 qualquer elemento α correspondente ao rótulo da aresta de origem mais à esquerda (aresta de origem com índice o_j), e incluímos em σ_2 qualquer elemento α correspondente ao rótulo da aresta de origem mais à direita (aresta de origem com índice o_i). Dessa forma, os elementos α das três arestas de origem afetadas são acumulados em

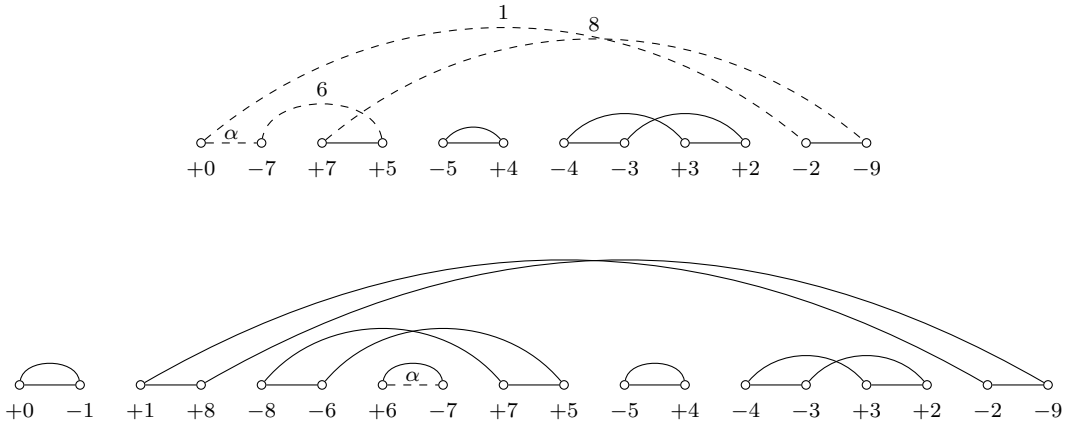


Figura 4.5: Exemplo de uma inserção que remove um *run* de um ciclo. Neste exemplo, temos o *run* de inserção $(+0, -2, -9, +7, +5, -7)$. A inserção de $\sigma = (+1\ -8\ +6)$ no início da string do genoma origem remove esse *run* e cria três novos ciclos.

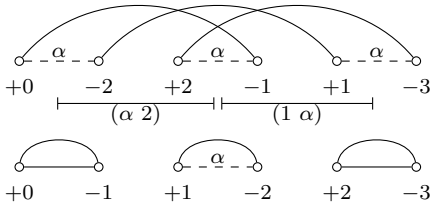


Figura 4.6: Exemplo de uma operação de *block interchange* que afeta uma tripla orientada de um ciclo orientado e cria três novos ciclos. Os elementos α são movidos de forma que apenas uma das arestas de origem afetadas permanece rotulada. Neste exemplo, temos $A = (0\ \alpha\ 2\ \alpha\ 1\ \alpha\ 3)$ e ι^n com $n = 2$.

uma só aresta de origem do novo grafo, como mostrado na Figura 4.6. Uma operação análoga é usada se apenas duas das três arestas de origem afetadas são rotuladas.

Como $k = j + 1$, podemos garantir que o ciclo mais à esquerda é um ciclo unitário limpo. Esse ciclo possui a aresta de destino que é adjacente às arestas de origem e_{o_j} e e_{o_k}. No entanto, não podemos garantir que o ciclo mais à direita é sempre unitário. Note que não existe aresta de destino rotulada e, portanto, $\lambda(C') + \lambda(C'') + \lambda(C''') \leq \lambda(C) + 1$, pois um dos ciclos é unitário e limpo. Dessa forma, $\Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq -1$, $\Delta c(\mathcal{I}, \mathcal{BI}) = 2$ e $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq 1$.

Agora, considere que $G(\mathcal{I})$ possui apenas ciclos não orientados. Seja $C = (o_1, \ldots, o_\ell)$ um ciclo de $G(\mathcal{I})$. Bafna e Pevzner [17] mostraram que para todo par de arestas de origem (e_{o_i}, e_{o_j}) de C, com $o_i > o_j$, existe um ciclo $D = (o'_1, \ldots, o'_{\ell'})$ com arestas de origem $e_{o'_x}$ e $e_{o'_y}$ tal que ou $o_i > o'_x > o_j > o'_y$ ou $o'_x > o_i > o'_y > o_j$. Assuma, sem perda de generalidade, que $o_i > o'_x > o_j > o'_y$, $o_i = o_1$ e $o_j = o_\ell$. Uma operação de *block interchange* que age nessas quatro arestas de origem cria quatro novos ciclos C', C'', D' e D'': C' é formado pelo caminho que vai de e_{o_i} até e_{o_j} com uma aresta de origem incidente ao primeiro vértice e ao último vértice desse caminho; C'' é formado pelo caminho que vai de e_{o_j} até e_{o_i} com uma aresta de origem incidente ao primeiro vértice e ao último vértice desse caminho; D' e D'' seguem um padrão similar aos dos ciclos C' e C''. Como $o_i = o_1$ e $o_j = o_\ell$, existe

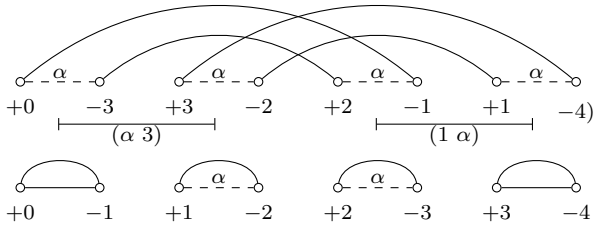


Figura 4.7: Exemplo de uma operação de *block interchange* que age em dois ciclos não orientados e cria quatro novos ciclos. Neste exemplo, temos $A = (0\ \alpha\ 3\ \alpha\ 2\ \alpha\ 1\ \alpha\ 4)$ e ι^n com $n = 3$. Os elementos α são movidos de forma que apenas duas das quatro arestas de origem afetadas permanecem rotuladas.

apenas uma aresta de destino no caminho de e_{o_j} até e_{o_i} e portanto, C'' é um ciclo unitário.

O primeiro segmento afetado pela operação começa na aresta de origem o'_y, incluindo qualquer elemento α correspondente ao rótulo da aresta $e_{o'_y}$, e termina na aresta de origem o_j sem incluir qualquer elemento α. O segundo segmento afetado pela operação começa com a aresta de origem o'_x, sem incluir qualquer elemento α, e termina com a aresta de origem o_i, incluindo qualquer elemento α correspondente ao rótulo da aresta e_{o_i}. Um exemplo é mostrado na Figura 4.7. Dessa forma, garantimos que D'' é um ciclo unitário limpo. Note que não existe aresta de destino rotulada e, portanto, $\lambda(C') + \lambda(C'') + \lambda(D') + \lambda(D'') \leq \lambda(C) + \lambda(D) + 1$. Sendo assim, temos $\Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq -1$, $\Delta c(\mathcal{I}, \mathcal{BI}) = 2$ e $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq 1$. □

Lembramos que em uma instância de strings sem sinais, o grafo de ciclos rotulado $G(\mathcal{I})$ possui apenas ciclos convergentes. Já em uma instância de strings com sinais, o grafo de ciclos rotulado $G(\mathcal{I})$ contém ciclos divergentes se, e somente se, existe pelo menos um elemento com sinal "−". No próximo lema, apresentamos como usar reversões para lidar com ciclos divergentes.

Lema 4.3.11. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui arestas de destino limpas e $G(\mathcal{I})$ possui um ciclo divergente, então existe reversão ρ com $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) = 1$.*

Demonstração. Seja $C = (o_1, o_2, \ldots, o_k)$ um ciclo divergente em $G(\mathcal{I})$ e seja $(e_{o_x}, e_{o_{x+1}})$ um par de arestas de origem divergentes com x mínimo. Uma reversão aplicada nessas arestas de origem transforma o ciclo C em um ciclo unitário C' e um outro ciclo C'' [16]. Essa reversão pode ser aplicada de uma forma que qualquer elemento α é movido para o ciclo C'', o que faz $\lambda(C') = 0$ e $\lambda(C'') = \lambda(C)$, já que $G(\mathcal{I})$ só possui arestas de destino limpas. Dessa forma, temos $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) = 1$. Um exemplo dessa operação é apresentado na Figura 4.8. □

Lema 4.3.12. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $|\pi^A| + 1 - c(\mathcal{I}) = 0$ e $G(\mathcal{I})$ possui pelo menos um run de deleção, então existe deleção ψ com $\Delta c(\mathcal{I}, \psi) + \Delta\lambda(\mathcal{I}, \psi) = 1$.*

Demonstração. Já que $|\pi^A| + 1 - c(\mathcal{I}) = 0$, todo ciclo do grafo $G(\mathcal{I})$ é um ciclo unitário. Cada ciclo possui no máximo um *run* de inserção e um *run* de deleção. Seja C um ciclo

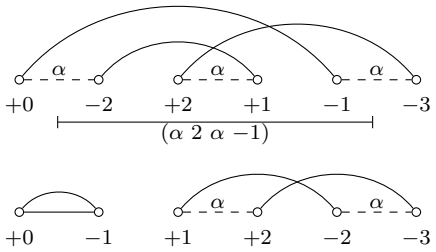


Figura 4.8: Exemplo de uma reversão que age em um ciclo divergente e cria dois novos ciclos. Neste exemplo, temos $A = (0\ \alpha\ 2\ \alpha\ {-1}\ \alpha\ 3)$ e ι^n com $n = 2$. Essa reversão move qualquer elemento α de forma que o ciclo unitário criado possui apenas arestas limpas.

unitário de $G(\mathcal{I})$ com uma aresta de origem rotulada. Note que esse ciclo possui um *run* de deleção que é formado apenas por essa aresta de origem. Seja ψ uma operação que remove o segmento correspondente ao rótulo dessa aresta de origem. Dessa forma, o *run* de deleção é removido e $\Delta c(\mathcal{I}, \psi) + \Delta\lambda(\mathcal{I}, \psi) = 1$. □

Com esses lemas, podemos apresentar os algoritmos 7 e 8. Esses algoritmos possuem complexidade de tempo de $O(n^2)$, já que o grafo de ciclos rotulado $G(\mathcal{I})$ pode ser criado em tempo linear, todos os laços de repetição executam $O(n)$ vezes, e qualquer operação dentro dos laços pode ser realizada em tempo linear.

No Teorema 4.3.1 demonstramos que esses algoritmos possuem fator de aproximação igual a 2 para os problemas de Distância de Rearranjos considerando os modelos $\mathcal{M}^{\phi,\psi}_{\mathcal{BI}}$ e $\mathcal{M}^{\phi,\psi}_{\rho,\mathcal{BI}}$.

Teorema 4.3.1. *Os algoritmos 7 e 8 são algoritmos de 2-aproximação para os problemas da Distância de Block Interchanges e Indels e da Distância de Block Interchanges, Reversões e Indels, respectivamente.*

Demonstração. Considere o Algoritmo 7. Qualquer operação β aplicada pelo algoritmo satisfaz $\Delta c(\mathcal{I}, \beta) + \Delta\lambda(\mathcal{I}, \beta) \geq 1$ e, portanto, a sequência retornada pelo algoritmo possui tamanho de no máximo $|\pi^A| + 1 - c(\mathcal{I}) + \lambda(\mathcal{I})$. Pelo Lema 4.3.7, esse algoritmo é uma 2-aproximação para a Distância de Block Interchanges e Indels.

A prova é similar para o Algoritmo 8. □

4.3.2 Algoritmos de 2-Aproximação para Modelos com Transposições

Nesta seção, apresentamos algoritmos de 2-aproximação para o problema de Distância de Rearranjos considerando os modelos $\mathcal{M}^{\phi,\psi}_{\tau}$ e $\mathcal{M}^{\phi,\psi}_{\rho,\tau}$.

Temos os seguintes corolários que seguem diretamente dos lemas 4.3.9, 4.3.11 e 4.3.12.

Corolário 4.3.1. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $G(\mathcal{I})$ possui pelo menos um run de inserção, então existe uma inserção ϕ com $\Delta c_{\text{clean}}(\mathcal{I}, \phi) + \Delta\lambda_{\phi}(\mathcal{I}, \phi) = 1$.*

Algoritmo 7: Algoritmo de 2-Aproximação para a Distância de Block Interchanges e Indels

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

Seja $S \leftarrow \emptyset$
enquanto $G(\mathcal{I})$ *possui runs de inserção* **faça**
  Seja ϕ uma inserção com $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) = 1$ (Lema 4.3.9)
  $A \leftarrow A \cdot \phi$
  Adicione ϕ na sequência S
enquanto $|\pi^A| + 1 - c(\mathcal{I}) > 0$ **faça**
  Seja $\mathcal{BI}$ uma operação com $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq 1$ (Lema 4.3.10)
  $A \leftarrow A \cdot \mathcal{BI}$
  Adicione $\mathcal{BI}$ na sequência S
enquanto $G(\mathcal{I})$ *possui runs de deleção* **faça**
  Seja ψ uma deleção que remove um *run* de deleção (Lema 4.3.12)
  $A \leftarrow A \cdot \psi$
  Adicione ψ na sequência S
retorne a sequência S

Corolário 4.3.2. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui arestas de destino limpas e $G(\mathcal{I})$ possui um ciclo divergente, então existe uma reversão ρ com $\Delta c_{\text{clean}}(\mathcal{I}, \rho) + \Delta\lambda_\phi(\mathcal{I}, \rho) = 1$.*

Corolário 4.3.3. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $|\pi^A| + 1 - c(\mathcal{I}) = 0$ e $G(\mathcal{I})$ possui pelo menos um run de deleção, então existe uma deleção ψ com $\Delta c_{\text{clean}}(\mathcal{I}, \psi) + \Delta\lambda_\phi(\mathcal{I}, \psi) = 1$.*

Se $G(\mathcal{I})$ só possui arestas de destino limpas, então qualquer transposição τ possui $\lambda_\phi(\mathcal{I}, \tau) = 0$, já que uma transposição não altera o conjunto $\Sigma_A \cap \Sigma_{\iota^n}$. Os próximos quatro lemas mostram como podemos usar transposições para criar novos ciclos limpos quando $G(\mathcal{I})$ não possui arestas de destino rotuladas. O primeiro deles é usado quando existem ciclos orientados em $G(\mathcal{I})$. Os outros lemas podem ser usados quando o grafo $G(\mathcal{I})$ satisfaz uma dessas condições: (i) existem dois ciclos rotulados e um deles é não unitário; (ii) existe um único ciclo não unitário rotulado; (iii) todos os ciclos não unitários são limpos.

Lema 4.3.13. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui arestas de destino limpas e $G(\mathcal{I})$ não possui ciclos divergentes, se existe um ciclo orientado $C \in G(\mathcal{I})$, então existe uma transposição τ com $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$.*

Demonstração. Como existe um ciclo orientado $C = (o_1, \ldots, o_\ell)$ em $G(\mathcal{I})$, sempre existe tripla orientada o_i, o_j e o_k, com $i < j < k$, tal que $o_i > o_k > o_j$ e $k = j + 1$ [17]. Uma transposição aplicada nessas arestas de origem transforma C em três ciclos C', C'' e C''' tal que pelo menos um deles é unitário. Dessa forma, temos $\Delta c(\mathcal{I}, \tau) = 2$.

Se C é um ciclo limpo, então os três novos ciclos também são limpos e, portanto, $\Delta c_{\text{clean}}(\mathcal{I}, \tau) = 2$. Caso contrário, C é rotulado e, portanto, pelo menos um dos três ciclos também é rotulado. No entanto, podemos garantir que o ciclo unitário é sempre limpo ao

Algoritmo 8: Algoritmo de 2-Aproximação para a Distância de Block Interchanges, Reversões e Indels

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

Seja $S \leftarrow \emptyset$
enquanto $G(\mathcal{I})$ *possui runs de inserção* **faça**
    Seja ϕ uma inserção com $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) = 1$ (Lema 4.3.9)
    $A \leftarrow A \cdot \phi$
    Adicione ϕ na sequência S
enquanto $|\pi^A| + 1 - c(\mathcal{I}) > 0$ **faça**
    se $G(\mathcal{I})$ *possui ciclos divergentes* **então**
        Seja ρ uma operação com $\Delta c(\mathcal{I}, \rho) + \Delta\lambda(\mathcal{I}, \rho) = 1$ (Lema 4.3.11)
        $A \leftarrow A \cdot \rho$
        Adicione ρ na sequência S
    senão
        Seja $\mathcal{BI}$ uma operação com $\Delta c(\mathcal{I}, \mathcal{BI}) + \Delta\lambda(\mathcal{I}, \mathcal{BI}) \geq 1$ (Lema 4.3.10)
        $A \leftarrow A \cdot \mathcal{BI}$
        Adicione $\mathcal{BI}$ na sequência S
enquanto $G(\mathcal{I})$ *possui runs de deleção* **faça**
    Seja ψ uma deleção que remove um *run* de deleção (Lema 4.3.12)
    $A \leftarrow A \cdot \psi$
    Adicione ψ na sequência S
retorne a sequência S

mover qualquer elemento α para um dos outros dois ciclos, como mostrado no exemplo da Figura 4.9. Portanto, temos que $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$. □

Lema 4.3.14. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui arestas de destino limpas e $G(\mathcal{I})$ não possui ciclos divergentes ou ciclos orientados, se existem pelo menos dois ciclos C e D que são não unitários e rotulados em $G(\mathcal{I})$, então existe uma transposição τ com $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$.*

Demonstração. Sejam $C = (o_1, \ldots, o_\ell)$ e $D = (o'_1, \ldots, o'_k)$, tal que $\ell \geq 2$ e $k \geq 2$, dois ciclos rotulados em $G(\mathcal{I})$. Suponha, sem perda de generalidade, que $o_\ell > o'_k$. Uma transposição τ aplicada nas arestas de origem e_{o_1}, e_{o_ℓ} e $e_{o'_k}$ cria dois ciclos C' e D', tal que C' é unitário e D' é um $(\ell + k - 1)$-ciclo. Podemos escolher a transposição de forma que C' seja um ciclo limpo, para isso basta mover qualquer α das arestas e_{o_1} e e_{o_ℓ} para o ciclo D', como mostrado na Figura 4.10. Portanto, temos $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$. □

Lema 4.3.15. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, tal que $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui arestas de destino limpas e $G(\mathcal{I})$ não possui ciclos divergentes ou ciclos orientados, se existe apenas um único ciclo rotulado C em $G(\mathcal{I})$ e C é não unitário, então existe uma sequência S_τ de k transposições tal que $\Delta c_{\text{clean}}(\mathcal{I}, S_\tau) + \Delta\lambda_\phi(\mathcal{I}, S_\tau) = k$, para $k \in \{2, 3\}$.*

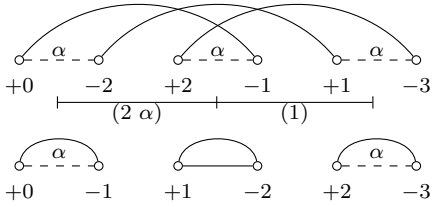


Figura 4.9: Exemplo de uma transposição que age em uma tripla orientada e cria três novos ciclos. A transposição move os elementos α de forma que o ciclo unitário criado é sempre limpo. Neste exemplo, temos $A = (0\ \alpha\ 2\ \alpha\ 1\ \alpha\ 3)$ e ι^n com $n = 2$.

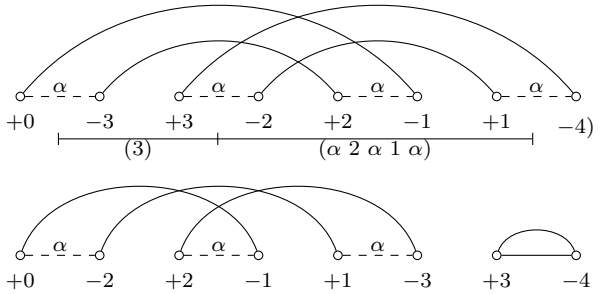


Figura 4.10: Exemplo de uma transposição que age em dois ciclos rotulados não orientados e cria um novo ciclo limpo. Neste exemplo, temos $A = (0\ \alpha\ 3\ \alpha\ 2\ \alpha\ 1\ \alpha\ 4)$ e ι^n com $n = 3$.

Demonstração. Dividimos a prova de acordo com o número de arestas de origem do ciclo rotulado C. Suponha que $C = (o_1, o_2)$ é o único ciclo rotulado de $G(\mathcal{I})$. Bafna e Pevzner [17, Lema 4.6] mostraram que existe outro ciclo não orientado $D = (o'_1, \ldots, o'_k)$, com $k \geq 2$, tal que ou $o_1 > o'_x > o_2 > o'_{x+1}$ ou $o'_x > o_1 > o'_{x+1} > o_2$, para algum $1 \leq x \leq k-1$. Como C é o único ciclo rotulado, D deve ser um ciclo limpo. Suponha, sem perda de generalidade, que $o_1 > o'_x > o_2 > o'_{x+1}$. Podemos aplicar duas transposições nesses ciclos que criam dois novos ciclos limpos no grafo. Primeiramente, aplicamos uma transposição τ_1 nas arestas de origem e_{o_1}, e_{o_2} e $e_{o'_{x+1}}$ que gera um ciclo unitário rotulado C' e um ciclo limpo D' que deve ser orientado [17]. Para que D' seja limpo, basta que a transposição mova qualquer α para a aresta do ciclo unitário C'. Agora, usando o Lema 4.3.13, aplicamos uma transposição τ_2 no ciclo limpo D' que o transforma em três ciclos limpos. Portanto, temos $\Delta c_{\text{clean}}(\mathcal{I}, S_\tau) + \Delta\lambda_\phi(\mathcal{I}, S_\tau) = 2$ com $S = (\tau_1, \tau_2)$. A Figura 4.11 mostra um exemplo dessa sequência de transposições.

Suponha que $C = (o_1, \ldots, o_\ell)$, com $\ell \geq 3$, é o único ciclo rotulado de $G(\mathcal{I})$. De forma similar ao caso anterior, aplicamos três transposições que geram três novos ciclos limpos. Primeiramente, aplicamos uma transposição τ_1 em um ou dois ciclos não unitários limpos D e E, que devem sempre existir [17, Teorema 4.7], que torna C em um ciclo orientado rotulado C', tal que $\Delta c(\mathcal{I}, \tau_1) = \Delta c_{\text{clean}}(\mathcal{I}, \tau_1) = 0$. Agora, usando o Lema 4.3.13, aplicamos uma transposição τ_2 em C' que o transforma em três ciclos tal que pelo menos um dele é um ciclo unitário limpo. De acordo com o Teorema 4.7 de Bafna e Pevzner [17], a sequência (τ_1, τ_2) transforma outro ciclo do grafo em um ciclo orientado, sendo que esse ciclo possui arestas dos ciclos D e E que foram afetados pela primeira transposição. Esse

ciclo deve ser limpo uma vez que D e E são ciclos limpos. Usando novamente o Lema 4.3.13, aplicamos uma transposição τ_3 nesse ciclo orientado limpo que o transforma em três novos ciclos limpos. Portanto, temos $\Delta c_{\text{clean}}(\mathcal{I}, S_\tau) + \Delta\lambda_\phi(\mathcal{I}, S_\tau) = 3$ com $S = (\tau_1, \tau_2, \tau_3)$. As figuras 4.12 e 4.13 mostram exemplos desse caso. □

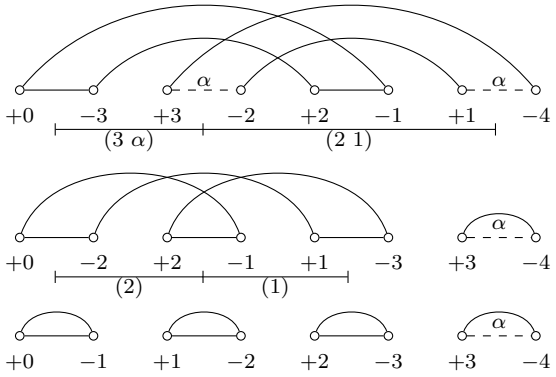


Figura 4.11: Exemplo de uma sequência de transposições agindo em dois ciclos não orientados $C = (4, 2)$ e $D = (3, 1)$, tal que C é rotulado e D é limpo. Essas operações criam dois novos ciclos limpos. Neste exemplo, temos $A = (0\ 3\ \alpha\ 2\ 1\ \alpha\ 4)$ e ι^n com $n = 3$.

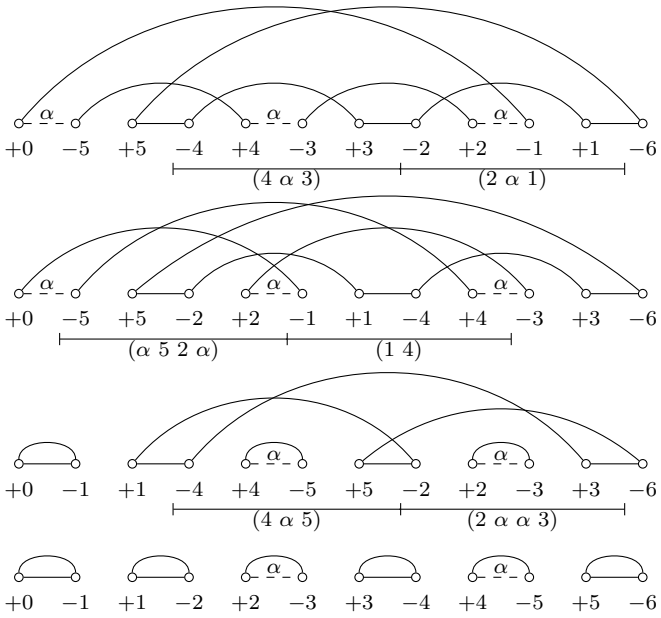


Figura 4.12: Exemplo de uma sequência de transposições que agem em dois ciclos não orientados $C = (5, 3, 1)$ e $D = (6, 4, 2)$, tal que C é rotulado e D é limpo. Essas operações criam três novos ciclos limpos. Neste exemplo, temos $A = (0\ \alpha\ 5\ 4\ \alpha\ 3\ 2\ \alpha\ 1\ 6)$ e ι^n com $n = 5$.

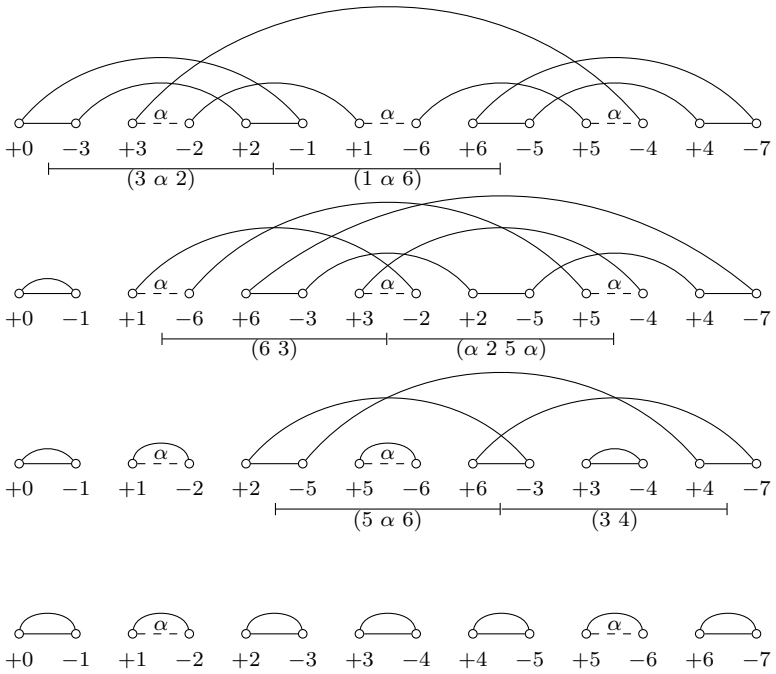


Figura 4.13: Exemplo de uma sequência de transposições que agem em três ciclos $C = (6, 4, 2)$, $D = (3, 1)$ e $E = (7, 5)$, tal que C é o único ciclo rotulado. Essas operações criam três novos ciclos limpos. Neste exemplo, temos $A = (0\ 3\ \alpha\ 2\ 1\ \alpha\ 6\ 5\ \alpha\ 4\ 7)$ e ι^n com $n = 6$.

Lema 4.3.16. *Para qualquer instância $\mathcal{I} = (A, \iota^n)$, se $|\pi^A| + 1 - c(\mathcal{I}) > 0$, $G(\mathcal{I})$ só possui ciclos limpos e $G(\mathcal{I})$ não possui ciclos divergentes ou ciclos orientados, então existe sequência $S_\tau = (\tau_1, \tau_2)$ tal que $\Delta c_{\text{clean}}(\mathcal{I}, S_\tau) + \Delta\lambda_\phi(\mathcal{I}, S_\tau) = 2$.*

Demonstração. Nesse caso, como todos os ciclos de $G(\mathcal{I})$ são limpos e não existem ciclos divergentes, a string A é uma permutação e podemos usar os resultados da Ordenação de Permutações por Transposições para grafo de ciclos. Considerando as condições do enunciado deste lema, Bafna e Pevzner [17] mostraram que sempre existe uma sequência $S_\tau = (\tau_1, \tau_2)$ que aumenta o número de ciclos em dois. Como transposições não inserem ou removem elementos, isso implica que $\Delta c_{\text{clean}}(\mathcal{I}, S_\tau) + \Delta\lambda_\phi(\mathcal{I}, S_\tau) = 2$. □

Com esses lemas, podemos apresentar os algoritmos 9 e 10. Assim como os algoritmos com *block interchanges*, esses algoritmos também possuem complexidade de tempo de $O(n^2)$, já que o grafo de ciclos rotulado $G(\mathcal{I})$ pode ser criado em tempo linear, todos os laços de repetição executam $O(n)$ vezes, e toda operação dentro dos laços pode ser realizada em tempo linear.

No Teorema 4.3.2 demonstramos que esses algoritmos possuem fator de aproximação igual a 2 para os problemas de Distância de Rearranjos considerando os modelos $\mathcal{M}_\tau^{\phi,\psi}$ e $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$.

Teorema 4.3.2. *Os algoritmos 9 e 10 são algoritmos de 2-aproximação para os problemas da Distância de Transposições e Indels e da Distância de Transposições, Reversões e Indels, respectivamente.*

Demonstração. Considere o Algoritmo 9. A cada iteração, qualquer sequência S' aplicada pelo algoritmo possui k operações e satisfaz $\Delta c_{\text{clean}}(\mathcal{I}, S') + \Delta\lambda_\phi(\mathcal{I}, S') \geq k$. Portanto, a sequência retornada ao fim do algoritmo possui tamanho de no máximo $|\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_\phi(\mathcal{I})$. Pelo Lema 4.3.8, esse algoritmo é uma 2-aproximação para o problema da Distância de Transposições e Indels.

A prova é similar para o Algoritmo 10. □

Algoritmo 9: Algoritmo de 2-Aproximação para a Distância de Transposições e Indels

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

Seja $S \leftarrow \emptyset$
enquanto $|\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_\phi(\mathcal{I}) > 0$ **faça**
  se $G(\mathcal{I})$ *possui runs de inserção* **então**
    Seja $S' = (\phi)$, tal que ϕ é uma inserção com $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) = 1$ (Corolário 4.3.1)
  senão se $G(\mathcal{I})$ *possui algum ciclo unitário rotulado* **então**
    Seja $S' = (\psi)$, tal que ψ é uma deleção que remove um *run* de deleção de um ciclo unitário (Corolário 4.3.3)
  senão se $G(\mathcal{I})$ *possui algum ciclo orientado* **então**
    Seja $S' = (\tau)$, tal que τ é uma transposição com $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$ (Lema 4.3.13)
  senão se $G(\mathcal{I})$ *possui dois ou mais ciclos rotulados* **então**
    Seja $S' = (\tau)$, tal que τ é uma transposição com $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$ (Lema 4.3.14)
  senão se $G(\mathcal{I})$ *possui um único ciclo rotulado* **então**
    Seja S' uma sequência de k transposições tal que $\Delta c_{\text{clean}}(\mathcal{I}, S') + \Delta\lambda_\phi(\mathcal{I}, S') = k$ (Lema 4.3.15)
  senão
    Seja $S' = (\tau_1, \tau_2)$, tal que $\Delta c_{\text{clean}}(\mathcal{I}, S') + \Delta\lambda_\phi(\mathcal{I}, S') = 2$ (Lema 4.3.16)
  $A \leftarrow A \cdot S'$
  Adicione as operações de S' na sequência S
retorne a sequência S

4.4 Conclusões

Neste capítulo, estudamos problemas de Distância de Rearranjos em Genomas Desbalanceados utilizando a representação clássica de genomas. Para strings com e sem sinais, estudamos modelos que envolvem a combinação de *indels* com reversões, transposições e *block interchanges*.

Na Seção 4.1, demonstramos que os seguintes problemas são NP-difíceis: a Distância de Reversões e Indels em Strings sem Sinais; a Distância de Transposições e Indels em Strings sem Sinais; e a Distância de Transposições, Reversões e Indels em Strings com ou sem Sinais.

Algoritmo 10: Algoritmo de 2-Aproximação para a Distância de Transposições, Reversões e Indels

Entrada: Uma instância $\mathcal{I} = (A, \iota^n)$
Saída: Uma sequência de rearranjos que transforma A em ι^n

Seja $S \leftarrow \emptyset$
enquanto $|\pi^A| + 1 - c_{\text{clean}}(\mathcal{I}) + \lambda_\phi(\mathcal{I}) > 0$ **faça**
  se $G(\mathcal{I})$ *possui runs de inserção* **então**
    Seja $S' = (\phi)$, tal que ϕ é uma inserção com $\Delta c(\mathcal{I}, \phi) + \Delta\lambda(\mathcal{I}, \phi) = 1$ (Corolário 4.3.1)
  senão se $G(\mathcal{I})$ *possui ciclos divergentes* **então**
    Seja $S' = (\rho)$, tal que ρ é uma reversão com $\Delta c_{\text{clean}}(\mathcal{I}, \rho) + \Delta\lambda_\phi(\mathcal{I}, \rho) = 1$ (Corolário 4.3.2)
  senão se $G(\mathcal{I})$ *possui algum ciclo unitário rotulado* **então**
    Seja $S' = (\psi)$, tal que ψ é uma deleção que remove um *run* de deleção de um ciclo unitário (Corolário 4.3.3)
  senão se $G(\mathcal{I})$ *possui algum ciclo orientado* **então**
    Seja $S' = (\tau)$, tal que τ é uma transposição com $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$ (Lema 4.3.13)
  senão se $G(\mathcal{I})$ *possui dois ou mais ciclos rotulados* **então**
    Seja $S' = (\tau)$, tal que τ é uma transposição com $\Delta c_{\text{clean}}(\mathcal{I}, \tau) + \Delta\lambda_\phi(\mathcal{I}, \tau) \geq 1$ (Lema 4.3.14)
  senão se $G(\mathcal{I})$ *possui um único ciclo rotulado* **então**
    Seja S' uma sequência de k transposições tal que $\Delta c_{\text{clean}}(\mathcal{I}, S') + \Delta\lambda_\phi(\mathcal{I}, S') = k$ (Lema 4.3.15)
  senão
    Seja $S' = (\tau_1, \tau_2)$, tal que $\Delta c_{\text{clean}}(\mathcal{I}, S') + \Delta\lambda_\phi(\mathcal{I}, S') = 2$ (Lema 4.3.16)
  $A \leftarrow A \cdot S'$
  Adicione as operações de S' na sequência S
retorne a sequência S

Na Seção 4.2, apresentamos algoritmos de aproximação que usam uma adaptação do conceito de *breakpoints* para genomas desbalanceados. Já na Seção 4.3, apresentamos algoritmos de 2-aproximação usando o grafo de ciclos rotulado, que foi introduzido na Seção 2.5.2 do Capítulo 2. A Tabela 4.1 resume o fator de aproximação alcançado para cada problema estudado neste capítulo.

Tabela 4.1: Resumo dos algoritmos apresentados neste capítulo para os problemas de Distância de Rearranjos em Genomas Desbalanceados.

Modelo	Seção 4.2	Seção 4.3
Reversões e Indels (sem sinais)	2-aproximação	-
Transposições e Indels (sem sinais)	3-aproximação	2-aproximação
Transposições, Reversões e Indels (sem sinais)	3-aproximação	-
Transposições, Reversões e Indels (com sinais)	-	2-aproximação
Block Interchanges e Indels (sem sinais)	-	2-aproximação
Block Interchanges, Reversões e Indels (com sinais)	-	2-aproximação

Capítulo 5

Distância em Genomas Desbalanceados com Regiões Intergênicas

Estudos que incorporam regiões intergênicas são relativamente recentes. Esses estudos assumem que não há genes repetidos nos genomas e que inserções e deleções afetam apenas regiões intergênicas. Dessa forma, os genomas possuem o mesmo conjunto de genes e podem ser modelados usando permutações e uma lista de valores numéricos representando os tamanhos das regiões intergênicas. Para permutações com sinais, Fertin e coautores [45] mostraram que o problema de Distância de Rearranjos Intergênicos para o modelo que contém apenas DCJs é NP-difícil e apresentaram uma 4/3-aproximação, um esquema de aproximação de tempo polinomial, e uma formulação de programação linear inteira. Quando inserções e deleções de nucleotídeos (i.e., *indels* que afetam apenas o tamanho de regiões intergênicas, mas não inserem ou removem genes) são incorporadas ao modelo com DCJs para genomas balanceados, a distância pode ser calculada em tempo polinomial [35]. Note que ao considerar permutações em problemas com representação intergênica, qualquer *indel* intergênico deve ser um *indel* de nucleotídeos já que o uso de permutações implica que os genomas são balanceados.

Oliveira e coautores [65] apresentaram uma 2-aproximação para reversões intergênicas em permutações com sinais, além da prova de NP-dificuldade para esse problema. Eles também apresentaram uma 2-aproximação para uma versão do problema com reversões e *indels* intergênicos em permutações com sinais, que ainda possui sua complexidade em aberto. Já para o modelo com reversões e transposições intergênicas em permutações com sinais, Oliveira e coautores [67] apresentaram uma 3-aproximação e uma prova de NP-dificuldade.

Considerando permutações sem sinais, Brito e coautores [29] demonstraram que o problema de Distância de Rearranjos Intergênicos é NP-difícil para os seguintes modelos: reversões intergênicas; reversões e *indels* intergênicos; reversões e transposições intergênicas; e reversões, transposições e *indels* intergênicos. Os autores apresentaram uma 4-aproximação para os modelos com reversões intergênicas e reversões e *indels* intergênicos, e apresentaram uma 4.5-aproximação para os outros dois modelos. Para transposições intergênicas em permutações sem sinais, Oliveira e coautores [66] desenvolveram uma 3.5-aproximação e provaram que o problema é NP-difícil.

Neste capítulo, estudamos problemas de Distância de Rearranjos Intergênicos em

genomas desbalanceados, considerando os seguintes modelos:

- $\mathcal{M}_{\rho}^{\phi,\psi} = \{\rho, \psi, \phi\}$: reversões e *indels* em strings com ou sem sinais;
- $\mathcal{M}_{\tau}^{\phi,\psi} = \{\tau, \psi, \phi\}$: transposições e *indels* em strings sem sinais;
- $\mathcal{M}_{\rho,\tau}^{\phi,\psi} = \{\rho, \tau, \psi, \phi\}$: reversões, transposições, e *indels* em strings com ou sem sinais.

5.1 Complexidade dos Problemas

Nesta seção, apresentamos provas de NP-dificuldade para problemas de Distância de Rearranjos Intergênicos em Genomas Desbalanceados considerando os modelos de rearranjos $\mathcal{M}_{\rho}^{\phi,\psi}$, $\mathcal{M}_{\tau}^{\phi,\psi}$ e $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$.

Lema 5.1.1. *O problema de Distância de Rearranjos Intergênicos é NP-difícil para os modelos $\mathcal{M}_{\rho}^{\phi,\psi}$ e $\mathcal{M}_{\tau}^{\phi,\psi}$, considerando strings sem sinais, e para o modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$, considerando strings com ou sem sinais.*

Demonstração. Considere o modelo $\mathcal{M}_{\rho}^{\phi,\psi}$. A versão de decisão do problema da Distância de Reversões e Indels Intergênicos em Strings sem Sinais (**IRID**) recebe como entrada uma instância $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d, k)$, onde $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, e consiste em decidir se $\mathcal{G}_o$ pode ser transformado em $\mathcal{G}_d$ usando no máximo k operações de reversões ou *indels*.

Nesta demonstração, apresentamos uma redução do problema de Ordenação de Permutações sem Sinais por Reversões (**SbR**), que é um problema NP-difícil, para o problema **IRID**. Dada uma instância (π, k) para **SbR**, tal que π tem tamanho n, criamos a instância $(\mathcal{G}_o, \mathcal{G}_d, k)$ para o problema **IRID**, onde $\breve{\gamma} = (0, \ldots, 0)$, $|\breve{\gamma}| = n + 1$, $\mathcal{G}_o = (\pi, \breve{\gamma})$, e $\mathcal{G}_d = (\iota^n, \breve{\gamma})$. Mostramos a seguir que a instância (π, k) do problema **SbR** é satisfeita se, e somente se, a instância $(\mathcal{G}_o, \mathcal{G}_d, k)$ para o problema **IRID** é satisfeita.

Se existe uma sequência de S reversões de tamanho no máximo k que transforma π em ι^n, então uma sequência similar de mesmo tamanho pode ser usada para transformar $\mathcal{G}_o$ em $\mathcal{G}_d$, onde uma reversão em S que inverte o segmento $(\pi_i, \ldots, \pi_j)$ é mapeada em $\rho_{(x,y)}^{(i,j)}$, com $x = y = 0$.

Como $\Sigma_{\iota^n} \setminus \Sigma_{\pi} = \emptyset$ e $\breve{\gamma} = (0, \ldots, 0)$, uma sequência de rearranjos de tamanho mínimo que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$ contém apenas reversões. Portanto, se existe um sequência de reversões e *indels* de tamanho no máximo k que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$, então existe uma sequência de reversões de tamanho no máximo k que ordena π.

A prova é similar para os outros modelos, já que a Ordenação de Permutações por Transposições [34] e a Ordenação de Permutações com ou sem Sinais por Reversões e Transposições [64] são NP-difíceis. □

5.2 Algoritmos de Aproximação usando Breakpoints

Nesta seção, apresentamos algoritmos de aproximação para o problema de Distância de Rearranjos Intergênicos em Strings sem Sinais considerando os modelos $\mathcal{M}_{\rho}^{\phi,\psi}$, $\mathcal{M}_{\tau}^{\phi,\psi}$ e $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$. Sempre consideramos que as strings de uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$ estão nas suas versões estendidas.

Usamos o conceito de *breakpoints* intergênicos, apresentado na Seção 2.4.3, para a definição de limitantes para a distância e a criação dos algoritmos de aproximação. A seguir, apresentamos uma outra definição utilizada nos limitantes para a distância, sendo que essa definição é similar à Definição 4.2.1, apresentada no Capítulo 4.

Definição 5.2.1. Dada uma operação (ou sequência de rearranjos) β e uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, definimos $\Delta\Phi(\mathcal{I}^{ig}, \beta) = \Delta\Phi(A, \iota^n, \beta) = |\Sigma_{\iota^n} \setminus \Sigma_A| - |\Sigma_{\iota^n} \setminus \Sigma_{A'}|$, onde $A' = A \cdot \beta$.

Assim como no problema da Distância de Reversões, Transposições e Indels em Strings sem Sinais, usamos o conceito de *breakpoints* de reversões sem sinais para o modelo $\mathcal{M}^{\phi,\psi}_{\rho,\tau}$. Os próximos lemas mostram como um rearranjo afeta o valor de $bi_{\mathcal{M}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$.

Lema 5.2.1. *Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que $\Delta\Phi(\mathcal{I}^{ig}, \beta) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \beta) \leq 2$, para qualquer indel β e modelo $\mathcal{M} \in \{\mathcal{M}^{\phi,\psi}_{\rho}, \mathcal{M}^{\phi,\psi}_{\tau}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}\}$.*

Demonstração. Uma inserção ϕ é aplicada entre dois elementos de A e, portanto, pode remover apenas o *breakpoint* intergênico formado por esses dois elementos, caso exista. Note que modelamos o genoma de forma que qualquer par de elementos em $\Sigma_{\iota^n} \setminus \Sigma_A$ forma um *breakpoint*. Seja σ a string a ser adicionada por ϕ. Para $1 \leq i < |\sigma|$, o par (σ_i, σ_{i+1}) é um *breakpoint* intergênico. Portanto, temos que $\Delta\Phi(\mathcal{I}^{ig}, \phi) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \phi) \leq |\sigma| + (1 - (|\sigma| - 1)) = 2$, para $\mathcal{M} \in \{\mathcal{M}^{\phi,\psi}_{\rho}, \mathcal{M}^{\phi,\psi}_{\tau}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}\}$.

Uma deleção ψ só pode ser aplicada em uma sequência contígua de elementos com valor α. Como, por definição, não existe *breakpoint* intergênico entre um par de elementos com valor α, apenas *breakpoints* intergênicos presentes nas duas extremidades da sequência afetada podem ser removidos. Além disso, para deleções, sempre temos $\Delta\Phi(\mathcal{I}^{ig}, \psi) = 0$, já que apenas inserções afetam o conjunto de elementos a serem adicionados. □

Lema 5.2.2. *Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que $\Delta\Phi(\mathcal{I}^{ig}, \rho) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \rho) \leq 2$, para qualquer reversão ρ e modelo $\mathcal{M} \in \{\mathcal{M}^{\phi,\psi}_{\rho}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}\}$.*

Demonstração. Considere uma reversão $\rho^{(i,j)}_{(x,y)}$. Note que essa reversão só pode remover *breakpoints* entre os pares de elementos (A_{i-1}, A_i) e (A_j, A_{j+1}), caso existam. Assim como as deleções, sempre temos que $\Delta\Phi(\mathcal{I}^{ig}, \rho) = 0$. Portanto, temos que $\Delta\Phi(\mathcal{I}^{ig}, \rho) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \rho) \leq 2$, para $\mathcal{M} \in \{\mathcal{M}^{\phi,\psi}_{\rho}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}\}$. □

Lema 5.2.3. *Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que $\Delta\Phi(\mathcal{I}^{ig}, \tau) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \tau) \leq 3$, para qualquer transposição τ e modelo $\mathcal{M} \in \{\mathcal{M}^{\phi,\psi}_{\tau}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}\}$.*

Demonstração. Similar à prova do Lema 5.2.2, mas devemos considerar que uma transposição afeta três adjacências de A. □

O próximo lema segue diretamente dos lemas 5.2.1, 5.2.2, 5.2.3 e do fato de que $bi_{\mathcal{M}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A| = 0$ se, e somente se, $\mathcal{G}_o = \mathcal{G}_d$, para qualquer $\mathcal{M} \in \{\mathcal{M}^{\phi,\psi}_{\rho}, \mathcal{M}^{\phi,\psi}_{\tau}, \mathcal{M}^{\phi,\psi}_{\rho,\tau}\}$.

Lema 5.2.4. *Para qualquer instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que:*

$$d_{\mathcal{M}_\rho^{\phi,\psi}}(\mathcal{I}^{ig}) \geq \frac{bi_{\mathcal{M}_\rho^{\phi,\psi}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|}{2},$$
$$d_{\mathcal{M}_\tau^{\phi,\psi}}(\mathcal{I}^{ig}) \geq \frac{bi_{\mathcal{M}_\tau^{\phi,\psi}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|}{3},$$
$$d_{\mathcal{M}_{\rho,\tau}^{\phi,\psi}}(\mathcal{I}^{ig}) \geq \frac{bi_{\mathcal{M}_{\rho,\tau}^{\phi,\psi}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|}{3}.$$

Os próximos lemas apresentam casos em que sempre é possível achar um *indel* com $\Delta\Phi(\mathcal{I}^{ig}, \beta) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \beta) > 0$. Essas operações serão úteis em todos os algoritmos de aproximação apresentados nesta seção.

Lema 5.2.5. *Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, se existe algum breakpoint intergênico em $\mathcal{I}^{ig}$ que é sobrecarregado ou subcarregado, então existe um indel β com $\Delta\Phi(\mathcal{I}^{ig}, \beta) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \beta) = 1$, para qualquer $\mathcal{M} \in \{\mathcal{M}_\rho^{\phi,\psi}, \mathcal{M}_\tau^{\phi,\psi}, \mathcal{M}_{\rho,\tau}^{\phi,\psi}\}$.*

Demonstração. Seja (A_i, A_{i+1}) um *breakpoint* intergênico sobrecarregado ou subcarregado. A região intergênica $\breve{\iota}_x^n$, tal que $x = \max(A_i, A_{i+1})$, é a região intergênica do genoma de destino $\mathcal{G}_d$ que está entre os dois elementos de ι^n correspondentes aos elementos A_i e A_{i+1}. Se esse *breakpoint* é sobrecarregado, então a deleção $\psi_{(0,y)}^{(i+1,i+1)}$, onde $y = \breve{A}_{i+1} - \breve{\iota}_x^n$, remove esse *breakpoint*. Se esse *breakpoint* é subcarregado, então a inserção $\phi_{(0)}^{(i,\sigma,\breve{\sigma})}$, onde $\sigma = \emptyset$ e $\breve{\sigma} = (\breve{\iota}_x^n - \breve{A}_{i+1})$ remove esse *breakpoint*. Em ambos os casos, o *indel* descrito torna $\breve{A}_{i+1} = \breve{\iota}_x^n$ e remove o *breakpoint* intergênico entre (A_i, A_{i+1}). Note que $\Delta\Phi(\mathcal{I}^{ig}, \beta) = 0$ em ambos os casos. □

Lema 5.2.6. *Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, se $|\Sigma_A \setminus \Sigma_{\iota^n}| > 0$ ou $|\Sigma_{\iota^n} \setminus \Sigma_A| > 0$, então existe um indel β com $\Delta\Phi(\mathcal{I}^{ig}, \beta) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \beta) \geq 1$, para qualquer $\mathcal{M} \in \{\mathcal{M}_\rho^{\phi,\psi}, \mathcal{M}_\tau^{\phi,\psi}, \mathcal{M}_{\rho,\tau}^{\phi,\psi}\}$.*

Demonstração. Considere que $|\Sigma_A \setminus \Sigma_{\iota^n}| > 0$. Seja $(A_i \ \ldots \ A_j)$ uma sequência maximal de elementos tal que $A_k = \alpha$, para todo $i \leq k \leq j$. Existem *breakpoints* intergênicos entre (A_{i-1}, A_i) e (A_j, A_{j+1}). Portanto, uma deleção ψ que age no segmento $(A_i \ \ldots \ A_j)$ remove esses dois *breakpoints* intergênicos e torna (A_{i-1}, A_{j+1}) adjacentes na nova string. Como (A_{i-1}, A_{j+1}) pode formar um *breakpoint* intergênico, temos que $\Delta\Phi(\mathcal{I}^{ig}, \beta) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, \beta) \geq 1$.

Considere que $|\Sigma_{\iota^n} \setminus \Sigma_A| > 0$. Seja x um elemento que pertence ao conjunto $\Sigma_{\iota^n} \setminus \Sigma_A$. Seja $(A_i \ \ldots \ A_j)$ a *strip* contendo o elemento $x-1$. Note que $\breve{\iota}_x^n$ é a região intergênica entre $(x-1, x)$ em $\mathcal{G}_d$. Se essa *strip* é decrescente, então $A_i = x - 1$ e a inserção $\phi_{(\breve{A}_i)}^{(i-1,(x),(0,\breve{\iota}_x^n))}$ não aumenta o número de *breakpoints* intergênicos e possui $\Delta\Phi(\mathcal{I}^{ig}, \beta) = 1$. Se a *strip* é crescente, então $A_j = x - 1$ e a inserção $\phi_{(0)}^{(j,(x),(\breve{\iota}_x^n,0))}$ não aumenta o número de *breakpoints* intergênicos e possui $\Delta\Phi(\mathcal{I}^{ig}, \beta) = 1$. □

5.2.1 Algoritmo de Aproximação para Modelos com Reversões

Apresentamos um algoritmo de 4-aproximação para o problema da Distância de Reversões e Indels Intergênicos. Além disso, provamos que esse algoritmo também é uma 6-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos

A seguir, mostramos casos em que sempre é possível encontrar uma sequência com no máximo duas operações (reversões ou *indels*) que remove *breakpoints* intergênicos.

Lema 5.2.7. *Para qualquer instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, tal que $\Sigma_A = \Sigma_{\iota^n}$ e não existem breakpoints sobrecarregados ou subcarregados em $\mathcal{G}_o$, se a string A possui pelo menos uma strip decrescente, então existe uma sequência de reversões e indels S tal que $|S| \leq 2$ e $\Delta\Phi(\mathcal{I}^{ig}, S) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, S) \geq 1$.*

Demonstração. Note que se $\Sigma_A = \Sigma_{\iota^n}$, então $\Delta\Phi(\mathcal{I}^{ig}, \beta) = 0$ para qualquer reversão ou inserção β. Seja $(A_i \ \ldots \ A_j)$ a *strip* decrescente tal que A_j é mínimo. Pela nossa escolha de A_j, a *strip* $(A_{i'} \ \ldots \ A_{j'})$ que contém o elemento $A_j - 1$ é crescente e, consequentemente, $A_{j'} = A_j - 1$.

Se $i' < i$, então a reversão $\rho^{(j'+1,j)}_{(x,y)}$ transforma $(A, \breve{A})$ em $(A', \breve{A}')$, tal que os elementos A_j e $A_{j'} = A_j - 1$ são adjacentes em A'. Se $\breve{A}_{j'+1} + \breve{A}_{j+1} \geq \breve{\iota}^n_{A_j}$, então podemos escolher x e y de forma que $\breve{A}'_{j'+1} = \breve{\iota}^n_{A_j}$, fazendo com que um *breakpoint* intergênico seja removido. Caso contrário, após aplicar a reversão, o par $(A_{j'}, A_j)$ em A' forma um *breakpoint* intergênico subcarregado. Nesse caso, existe uma inserção (Lema 5.2.5) que remove esse *breakpoint* intergênico.

Se $i' > i$, então a reversão $\rho^{(j+1,j')}_{(x,y)}$ transforma $(A, \breve{A})$ em $(A', \breve{A}')$, tal que os elementos A_j e $A_{j'} = A_j - 1$ são adjacentes em A'. De forma análoga ao caso anterior, talvez seja necessário usar uma inserção para remover um *breakpoint* intergênico subcarregado.

Portanto, em ambos os casos, existe uma sequência de reversões e *indels* S tal que $|S| \leq 2$ e $\Delta\Phi(\mathcal{I}^{ig}, S) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, S) \geq 1$. □

Lema 5.2.8. *Para qualquer instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, tal que $\Sigma_A = \Sigma_{\iota^n}$ e não existem breakpoints sobrecarregados ou subcarregados em $\mathcal{G}_o$, se a string A possui pelo menos uma strip decrescente e toda reversão ρ que remove breakpoints (não necessariamente intergênicos) resulta na string $A \cdot \rho$ que não possui strips decrescentes, então existe uma sequência de reversões e indels S tal que $|S| \leq 2$ e $\Delta\Phi(\mathcal{I}^{ig}, S) + \Delta bi_{\mathcal{M}}(\mathcal{I}^{ig}, S) = 2$.*

Demonstração. Como $\Sigma_A = \Sigma_{\iota^n}$, temos que A é uma permutação. Se qualquer reversão que remove *breakpoints* (não necessariamente *breakpoints* intergênicos) resulta na string $A \cdot \rho$ que não possui *strips* decrescentes, então existe apenas uma reversão que remove *breakpoints* de A (Lema 4.2.9) e essa reversão remove dois *breakpoints* (não necessariamente intergênicos). Seja $(A_i, \ldots, A_j)$ o intervalo afetado por essa reversão que remove dois *breakpoints* de A.

Seja $\rho^{(i,j)}_{(x,y)}$ uma reversão com valores arbitrários de x e y. Essa reversão inverte o segmento $(A_i, \ldots, A_j)$ e remove dois *breakpoints*. Note que os *breakpoints* removidos são (A_{i-1}, A_i) e (A_j, A_{j+1}). Essa reversão torna os pares (A_{i-1}, A_j) e (A_i, A_{j+1}) adjacentes na

string $A' = A \cdot \rho^{(i,j)}_{(x,y)}$ e, como não existem *strips* decrescentes em A', temos que $A_{i-1}+1 = A_j$ e $A_i + 1 = A_{j+1}$. Sejam $\ell = \breve{A}_i + \breve{A}_{j+1}$ e $m = \breve{\iota}^n_{A_j} + \breve{\iota}^n_{A_{j+1}}$.

Se $\ell \geq m$, então aplicamos a reversão $\rho^{(i,j)}_{(x,y)}$ em $(A, \breve{A})$, com $x = \min(\breve{A}_i, \breve{\iota}^n_{A_j})$ e $y = \breve{\iota}^n_{A_j} - x$, que resulta em $(A', \breve{A}')$, de forma que o par $(A'_{i-1}, A'_i) = (A_{i-1}, A_j)$ não é um *breakpoint* intergênico. No entanto, o par $(A'_j, A'_{j+1}) = (A_i, A_{j+1})$ pode ser um *breakpoint* intergênico, que pode ser removido usando um *indel* de acordo com o Lema 5.2.5.

Considere que $\ell < m$. Nesse caso, primeiro aplicamos a inserção $\phi_0^{(i-1,\sigma,\breve{\sigma})}$, com $\sigma = \emptyset$ e $\breve{\sigma} = (m - \ell)$, que torna $\breve{A}'_i + \breve{A}'_{j+1} = \breve{\iota}^n_{A_j} + \breve{\iota}^n_{A_{j+1}}$, onde $(A', \breve{A}') = (A, \breve{A}) \cdot \phi_0^{(i-1,S,\breve{S})}$. Note que $A' = A$ e nenhum *breakpoint* intergênico é removido por essa inserção. Agora, aplicamos a reversão $\rho^{(i,j)}_{(x,y)}$ em $(A', \breve{A}')$, com $x = \min(\breve{A}'_i, \breve{\iota}^n_{A_j})$ e $y = \breve{\iota}^n_{A_j} - x$, que remove dois *breakpoints* intergênicos.

Em ambos os casos, no máximo duas operações são aplicadas para remover dois *breakpoints* intergênicos. □

Algoritmo 11: Algoritmo para o problema de Distância de Rearranjos Intergênicos considerando os modelos $\mathcal{M}^{\phi,\psi}_{\rho}$ e $\mathcal{M}^{\phi,\psi}_{\rho,\tau}$

Entrada: Uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$

Saída: Uma sequência de rearranjos S que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$

```
Seja S ← ∅
enquanto |Σ_A \ Σ_{ι^n}| > 0 ou |Σ_{ι^n} \ Σ_A| > 0 faça
    Seja β um indel de acordo com o Lema 5.2.6
    G_o ← G_o · β
    Adicione β na sequência S
enquanto bi_M(I^ig) > 0 faça
    se existe breakpoint intergênico sobrecarregado ou subcarregado então
        Seja S' = (β), onde β é um indel de acordo com o Lema 5.2.5
    senão se existe strip decrescente então
        Seja S' uma sequência de reversões e indels de acordo com os lemas 5.2.7 e
         5.2.8.
    senão
        Seja (A_i, ..., A_j) uma strip tal que i > 0 e i é mínimo
        Seja S' = (ρ(i, j))
    G_o ← G_o · S'
    Adicione as operações de S' na sequência S
retorne a sequência S
```

O Algoritmo 11 usa os lemas 5.2.5, 5.2.6, 5.2.7 e 5.2.8 e um passo adicional, quando nenhum dos lemas pode ser aplicado, que torna uma das *strips* do genoma de origem em uma *strip* decrescente. O Lema 5.2.9 apresenta um limitante superior no número de rearranjos usados pelo algoritmo.

Lema 5.2.9. *Para qualquer instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, o Algoritmo 11 transforma $\mathcal{G}_o$ em $\mathcal{G}_d$ usando no máximo $2(bi_{\mathcal{M}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|)$ rearranjos, considerando $\mathcal{M} \in \{\mathcal{M}_{\rho}^{\phi,\psi}, \mathcal{M}_{\rho,\tau}^{\phi,\psi}\}$.*

Demonstração. Essa demonstração é similar às provas dos lemas 4.2.10 e 4.2.11, que foram apresentadas no Capítulo 4. □

Teorema 5.2.1. *O Algoritmo 11 é uma 4-aproximação para o problema da Distância de Reversões e Indels Intergênicos. Além disso, esse algoritmo é uma 6-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos.*

Demonstração. Diretamente dos lemas 5.2.4 e 5.2.9. □

5.2.2 Algoritmo de 4.5-Aproximação para Transposições e Indels

Nesta seção, apresentamos um algoritmo com fator de aproximação igual a 4.5 para o problema da Distância de Transposições e Indels Intergênicos. Assim como o algoritmo apresentado anteriormente, esse algoritmo usa os lemas 5.2.5 e 5.2.6 para tornar $\Sigma_A = \Sigma_{\iota^n}$ e remover *breakpoints* intergênicos sobrecarregados e subcarregados. Para remover os outros tipos de *breakpoints* intergênicos usando transposições e *indels*, esse algoritmo usa o resultado do Lema 5.2.10, que é apresentado a seguir. Para simplificar a notação, definimos $bi_{\tau}(\mathcal{I}^{ig}) = bi_{\mathcal{M}_{\tau}^{\phi,\psi}}(\mathcal{I}^{ig})$.

Lema 5.2.10. *Dada uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, se $\Sigma_A = \Sigma_{\iota^n}$ e não existem breakpoints intergênicos sobrecarregados ou subcarregados, então existe uma transposição τ que remove um breakpoint intergênico ou existe uma sequência S, com $|S| \leq 3$, que remove dois breakpoints intergênicos.*

Demonstração. Note que se $A \neq \iota^n$, então $bi_{\tau}(\mathcal{I}^{ig}) \geq 2$, já que para cada *breakpoint* (A_i, A_{i+1}) existe outro *breakpoint* com o elemento $A_i + 1$.

Sejam (A_{i-1}, A_i) e (A_j, A_{j+1}) dois *breakpoints* intergênicos tal que $i < j$ e j é mínimo. Lembre-se que para o problema da Distância de Transposições e Indels Intergênicos consideramos a definição de *breakpoints* de transposição (Definição 2.4.11). Portanto, existem apenas *strips* crescentes em A.

Considere a *strip* $(A_i, \ldots, A_j)$ e sejam $A_x = A_j + 1$ e $A_y = A_{j+1} - 1$. Pela nossa escolha de A_j, temos que $x > j$.

Se $\breve{A}_{j+1} + \breve{A}_x \geq \breve{\iota}^n_{A_x}$, então a transposição que troca a posição relativa dos segmentos adjacentes $(A_i \ \ldots \ A_j)$ e $(A_{j+1} \ \ldots \ A_{x-1})$ torna os elementos A_j e $A_x = A_j + 1$ adjacentes. Como $\breve{A}_{j+1} + \breve{A}_x \geq \breve{\iota}^n_{A_x}$, podemos mover o número de nucleotídeos necessários de $\breve{A}_{j+1}$ para tornar $\breve{A}'_x = \breve{\iota}^n_{A_x}$ (ou mover o número de nucleotídeos excedentes de $\breve{A}_x$), onde $(A', \breve{A}')$ é o genoma resultante após aplicar a transposição. Dessa forma, essa transposição remove um *breakpoint* intergênico.

Se $\breve{A}_{j+1} + \breve{A}_x < \breve{\iota}^n_{A_x}$, então a seguinte sequência remove dois *breakpoints* intergênicos:

1. Uma inserção de $\breve{\iota}^n_{A_x} - (\breve{A}_{j+1} + \breve{A}_x) + max(0, \breve{\iota}^n_{A_{j+1}} - \breve{A}_{y+1})$ nucleotídeos em $\breve{A}_{j+1}$;

2. Uma transposição que troca a posição relativa dos segmentos adjacentes $(A_i \ \ldots \ A_j)$ e $(A_{j+1} \ \ldots \ A_{x-1})$ torna os elementos A_j e $A_x = A_j + 1$ adjacentes. Como descrito no caso anterior, o número de nucleotídeos necessários (ou excedentes) podem ser movidos de $\breve{A}_{j+1}$ (ou movidos de $\breve{A}_x$), removendo um *breakpoint* intergênico. Além disso, essa transposição torna A_{i-1} e A_{j+1} adjacentes;

3. Uma transposição que torna $A_y = A_{j+1} - 1$ e A_{j+1} adjacentes e move nucleotídeos necessários ou excedentes para remover o *breakpoint* intergênico entre esses elementos.

Note que, após as duas primeiras operações dessa sequência serem aplicadas, o tamanho da região intergênica entre (A_{i-1}, A_{j+1}) somado ao tamanho da região intergênica entre (A_y, A_{y+1}) é maior ou igual ao valor de $\breve{\iota}^n_{A_{j+1}}$. □

Algoritmo 12: Algoritmo para o problema da Distância de Transposições e Indels Intergênicos

Entrada: Uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$
Saída: Uma sequência de rearranjos S que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$

```
Seja S ← ∅
enquanto bi_τ(I^ig) + |Σ_{ι^n} \ Σ_A| > 0 faça
    se |Σ_A \ Σ_{ι^n}| > 0 ou |Σ_{ι^n} \ Σ_A| > 0 então
        Seja S' = (β), onde β é um indel de acordo com o Lema 5.2.6
    senão se existe breakpoint intergênico sobrecarregado ou subcarregado então
        Seja S' = (β), onde β é um indel de acordo com o Lema 5.2.5
    senão
        Seja S' uma sequência de operações de acordo com o Lema 5.2.10
    G_o ← G_o · S'
    Adicione as operações de S' na sequência S
retorne a sequência S
```

O Algoritmo 12 usa os lemas 5.2.5, 5.2.6 e 5.2.10 para construir uma sequência de operações que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$, considerando o modelo $\mathcal{M}^{\phi,\psi}_\tau$. O Lema 5.2.11 apresenta um limitante superior no número de rearranjos usados pelo algoritmo.

Lema 5.2.11. *Para qualquer instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, o Algoritmo 12 transforma $\mathcal{G}_o$ em $\mathcal{G}_d$ usando no máximo $\frac{3}{2}(bi_\tau(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|)$ rearranjos.*

Demonstração. A cada iteração, o algoritmo usa uma operação β com $\Delta\Phi(\mathcal{I}^{ig}, \beta) + \Delta bi_\tau(\mathcal{I}^{ig}, \beta) = 1$ ou uma sequência S' com três operações e $\Delta\Phi(\mathcal{I}^{ig}, S') + \Delta bi_\tau(\mathcal{I}^{ig}, S') = 2$. Dessa forma, no pior caso, o algoritmo usa em média $\frac{3}{2}$ operações para diminuir o valor de $bi_\tau(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ em uma unidade. Consequentemente, a sequência encontrada pelo algoritmo transforma $\mathcal{G}_o$ em $\mathcal{G}_d$ usando no máximo $\frac{3}{2}(bi_\tau(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|)$ operações. □

Teorema 5.2.2. *O Algoritmo 12 é uma* 4.5*-aproximação para o problema da Distância de Transposições e Indels Intergênicos.*

Demonstração. Diretamente dos lemas 5.2.4 e 5.2.11. □

Note que para o problema da Distância de Reversões, Transposições e Indels Intergênicos, podemos melhorar o Algoritmo 11 ao usar as operações descritas no Lema 5.2.10 quando existem apenas *strips* crescentes na string do genoma de origem. No entanto, no pior caso, o fator de aproximação continua o mesmo para essa nova versão do algoritmo.

Os algoritmos 11 e 12 possuem complexidade de tempo de $O(n^2)$. Note que o algoritmo executa por no máximo $O(n)$ iterações, já que $bi_{\mathcal{M}}(\mathcal{I}^{ig}) + |\Sigma_{\iota^n} \setminus \Sigma_A|$ é $O(n)$, e cada operação pode ser encontrada e aplicada em tempo linear.

5.3 Algoritmos de Aproximação usando Grafo de Ciclos

Nesta seção, apresentamos os seguintes algoritmos de aproximação:

- Um algoritmo de 2.5-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais (modelo $\mathcal{M}_{\rho}^{\phi,\psi}$);
- Um algoritmo de 4-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais (modelo $\mathcal{M}_{\tau}^{\phi,\psi}$);
- Um algoritmo de 4-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais (modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$).

Esses algoritmos e os limitantes inferiores apresentados nesta seção utilizam os conceitos relacionados ao grafo de ciclos rotulado e ponderado (Seção 2.5.3). Sempre consideramos que as strings (A e ι^n) de uma instância $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$ e suas formas simplificadas (π^A e π^ι) estão nas suas versões estendidas.

Agora, apresentamos limitantes para o valor de $\Delta c_g(\mathcal{I}^{ig}, \beta)$ dependendo do tipo do rearranjo β.

Lema 5.3.1. *Para qualquer indel β e instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que $\Delta c_g(\mathcal{I}^{ig}, \beta) \leq 1$.*

Demonstração. Uma deleção afeta uma única aresta de origem do grafo, sendo que apenas o custo e o rótulo dessa aresta podem ser alterados. Portanto, apenas um ciclo é afetado por uma deleção. Considere uma deleção ψ e seja C o ciclo afetado por ψ. Quando C é rotulado ou desbalanceado, no melhor cenário, a deleção ψ transforma C em um ciclo bom. Quando C é um ciclo bom, a deleção não aumenta a quantidade de ciclos bons no grafo. Portanto, $\Delta c_g(\mathcal{I}^{ig}, \psi) \leq 1$.

Considere uma inserção ϕ que insere k elementos em A. Essa inserção adiciona $2k$ vértices no grafo e substitui uma aresta de origem de um ciclo C por $k+1$ arestas de origem, também adicionando k arestas de destino no grafo. Como o novo grafo possui $k+1$ arestas de origem distintas em relação ao grafo $G(\mathcal{I}^{ig})$ e pelo menos uma das novas arestas pertence a C, no melhor cenário, k ciclos são criados no grafo. Portanto,

$$
\begin{aligned}
&(|\pi^A| + 1 - c(\mathcal{G}_o, \mathcal{G}_d)) - (|\pi^{A\cdot\phi}| + 1 - c(\mathcal{G}_o \cdot \phi, \mathcal{G}_d)) \\
=&(|\pi^A| - |\pi^{A\cdot\phi}|) - (c(\mathcal{G}_o, \mathcal{G}_d) - c(\mathcal{G}_o \cdot \phi, \mathcal{G}_d)) \\
=&- k - (-k) = 0.
\end{aligned}
$$

No melhor cenário, o ciclo C se torna um ciclo bom e todos os k ciclos adicionados também são ciclos bons, o que resulta em $\Delta c_g(\mathcal{I}^{ig}, \phi) = 1$. Note que se qualquer outro ciclo C' de $G(\mathcal{I}^{ig})$ se torna balanceado e limpo após aplicar ϕ, então pelo menos uma das novas arestas de origem foi adicionada em C' e a inserção não pode ter adicionado k novos ciclos no grafo. De forma similar, se x ciclos distintos de C se tornam ciclos bons, então no máximo $k - x$ novos ciclos foram adicionados no grafo por ϕ, o que também resulta em $\Delta c_g(\mathcal{I}^{ig}, \phi) \leq 1$. □

Lema 5.3.2. *Para qualquer reversão ρ e instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que $\Delta c_g(\mathcal{I}^{ig}, \rho) \leq 1$.*

Demonstração. Bafna e Pevzner [16] demonstraram que $\Delta c(\mathcal{I}^{ig}, \rho) \in \{-1, 0, 1\}$, para qualquer reversão ρ. Além disso, como uma reversão não adiciona elementos, sabemos que a quantidade de arestas no grafo permanece a mesma.

Se $\Delta c(\mathcal{I}^{ig}, \rho) = -1$, então ρ afeta dois ciclos C e D, juntando esses dois ciclos em um único ciclo C'. No melhor cenário, tanto C quanto D são desbalanceados e limpos, mas C' é um ciclo balanceado e limpo, fazendo com que $\Delta c_g(\mathcal{I}^{ig}, \rho) = 1$. Se C ou D é rotulado, então C' também é rotulado e nenhum ciclo bom é adicionado ao grafo.

Se $\Delta c(\mathcal{I}^{ig}, \rho) = 0$, então ρ afeta um único ciclo C, transformando-o em um ciclo C' com os mesmos vértices e arestas de destino que o ciclo C. Note que se C é desbalanceado ou rotulado, então C' também é desbalanceado ou rotulado, já que nenhum rótulo é removido e a soma dos custos das arestas de origem permanece a mesma. Portanto, $\Delta c_g(\mathcal{I}^{ig}, \rho) = 0$.

Se $\Delta c(\mathcal{I}^{ig}, \rho) = 1$, então ρ afeta um único ciclo C, transformando-o em dois ciclos C' e D'. Se C é um ciclo bom, então temos $\Delta c_g(\mathcal{I}^{ig}, \rho) = 1$, mesmo que os ciclos C' e D' sejam bons. Se C é um ciclo desbalanceado ou rotulado, então pelo menos um dos ciclos C' e D' também deve ser desbalanceado ou rotulado. Portanto, no melhor cenário, $\Delta c_g(\mathcal{I}^{ig}, \rho) = 1$. □

Lema 5.3.3. *Para qualquer transposição τ e instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que $\Delta c_g(\mathcal{I}^{ig}, \tau) \leq 2$.*

Demonstração. Uma transposição não remove rótulos de arestas de destino. Além disso, uma transposição só pode remover um rótulo de uma arestas de origem ao transferir um elemento α para outra aresta de origem. Dividimos essa prova de acordo com a quantidade de ciclos afetados por uma transposição τ [17].

Se τ afeta três ciclos C_1, C_2, e C_3, então os vértices desses ciclos são unidos em um único ciclo C'. No melhor cenário, C_1, C_2, e C_3 são ciclos desbalanceados e limpos, e C' é um ciclo bom. Portanto, $\Delta c_g(\mathcal{I}^{ig}, \tau) = 1$.

Se τ afeta dois ciclos C_1 e C_2, então os vértices desses ciclos são rearranjados em dois ciclos C_1' e C_2'. No melhor cenário, C_1 e C_2 são ciclos desbalanceados e limpos, e C_1' e C_2' são ciclos bons. Portanto, $\Delta c_g(\mathcal{I}^{ig}, \tau) = 2$.

Se τ afeta um único ciclo C, então os vértices de C são rearranjados em um ciclo C' ($\Delta c(\mathcal{I}^{ig}, \tau) = 0$) ou C é transformado em três novos ciclos ($\Delta c(\mathcal{I}^{ig}, \tau) = 2$). Se $\Delta c(\mathcal{I}^{ig}, \tau) = 0$, então o número de ciclos bons não é alterado. Se $\Delta c(\mathcal{I}^{ig}, \tau) = 2$, então dividimos a prova em dois casos. Se C é um ciclo bom, então o melhor cenário ocorre quando todos os três novos ciclos também são bons e, consequentemente, $\Delta c_g(\mathcal{I}^{ig}, \tau) = 2$. Se C é desbalanceado ou rotulado, então pelo menos um dos novos ciclos também é um ciclo desbalanceado ou rotulado. Consequentemente, no máximo dois desses novos ciclos são bons e, consequentemente, temos $\Delta c_g(\mathcal{I}^{ig}, \tau) = 2$, no melhor cenário. □

Lema 5.3.4. *Para qualquer instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$, temos que:*

$$d_{\mathcal{M}_{\rho}^{\phi,\psi}}(\mathcal{I}^{ig}) \geq |\pi^A| + 1 - c_g(\mathcal{I}^{ig}),$$
$$d_{\mathcal{M}_{\tau}^{\phi,\psi}}(\mathcal{I}^{ig}) \geq \frac{|\pi^A| + 1 - c_g(\mathcal{I}^{ig})}{2},$$
$$d_{\mathcal{M}_{\rho,\tau}^{\phi,\psi}}(\mathcal{I}^{ig}) \geq \frac{|\pi^A| + 1 - c_g(\mathcal{I}^{ig})}{2}.$$

Demonstração. Considere o modelo $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$. Como $|\pi^A| + 1 - c_g(\mathcal{I}^{ig}) = 0$ se, e somente se, $\mathcal{G}_o = \mathcal{G}_d$, qualquer sequência de rearranjos S que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$ deve tornar o valor de $|\pi^A| + 1 - c_g(\mathcal{I}^{ig})$ igual a zero. Pelos lemas 5.3.1, 5.3.2 e 5.3.3, qualquer rearranjo β em $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$ satisfaz $\Delta c_g(\mathcal{I}^{ig}, \beta) \leq 2$ e, portanto, $|S| \geq \frac{|\pi^A|+1-c_g(\mathcal{I}^{ig})}{2}$.

A prova é similar para os outros modelos. Note que qualquer rearranjo de $\mathcal{M}_{\rho}^{\phi,\psi}$ satisfaz $\Delta c_g(\mathcal{I}^{ig}, \beta) \leq 1$ (lemas 5.3.1 e 5.3.2). □

5.3.1 Uma 2.5-Aproximação para Reversões e Indels

A ideia principal da 2.5-aproximação é criar novos ciclos bons a cada iteração, até que o grafo contenha apenas ciclos unitários bons. Os próximos lemas mostram como aumentar o número de ciclos bons no grafo usando reversões e/ou indels.

Lema 5.3.5. *Para qualquer grafo $G(\mathcal{I}^{ig})$ que contém ciclo unitário $C = (o_1, d_1)$, se (i) C é um ciclo limpo ou (ii) C é um ciclo não negativo e $\ell(e_{o_1}) = \emptyset$, então existe um indel β com $\Delta c_g(\mathcal{I}^{ig}, \beta) = 1$.*

Demonstração. Como $\ell(e_{o_1}) = \emptyset$, as arestas e_{o_1} e e'_{d_1} são incidentes a vértices que correspondem a elementos adjacentes em A. Sejam $+A_{i-1}$ e $-A_i$ os vértices do ciclo unitário C. Dividimos nossa análise em dois casos.

Considere que C é um ciclo limpo. Se $w(e'_{d_1}) > w(e_{o_1})$, então uma inserção de $w(e'_{d_1}) - w(e_{o_1})$ nucleotídeos na região intergênica $\breve{A}_i$ torna o ciclo C em um ciclo balanceado e limpo. Se $w(e'_{d_1}) < w(e_{o_1})$, então uma deleção de $w(e_{o_1}) - w(e'_{d_1})$ nucleotídeos na região intergênica $\breve{A}_i$ torna o ciclo C em um ciclo balanceado e limpo. Como nenhum elemento é adicionado na string e C é transformado em um ciclo bom, então existe *indel* β com $\Delta c_g(\mathcal{I}^{ig}, \beta) = 1$.

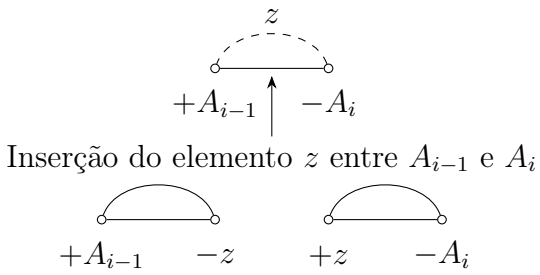


Figura 5.1: Exemplo de inserção que transforma um ciclo unitário não negativo, que possui aresta de origem limpa e aresta de destino rotulada, em dois ciclos bons. Assumimos que A_{i-1} tem sinal "+".

Agora, considere que C é um ciclo não negativo e $\ell(e_{o_1}) = \emptyset$, mas $\ell(e_{d_1}) \neq \emptyset$. Note que se $\ell(e_{d_1}) = \emptyset$, então C é um ciclo limpo e podemos usar um *indel* como descrito no caso anterior. Pela nossa representação de genomas e pela definição do grafo $G(\mathcal{I}^{ig})$, o rótulo de e'_{d_1} é igual a $z = |A_{i-1} + 1|$. Seja x a região intergênica entre A_{i-1} e z em $\mathcal{G}_d$, e seja y a região intergênica entre A_i e z em $\mathcal{G}_d$. Note que A_{i-1} e A_i possuem o mesmo sinal, já que eles formam um ciclo unitário.

A inserção $\phi^{(i-1,\sigma,\breve{\sigma})}_{\min(x,w(e_{o_1}))}$, tal que $\sigma = (A_{i-1} + 1)$, $\breve{\sigma} = (x', y')$, $x' = x - \min(x, w(e_{o_1}))$ e $y' = y - (w(e_{o_1}) - \min(x, w(e_{o_1})))$, transforma C em dois ciclos unitários C' e C'' que são balanceados e limpos, como mostrado na Figura 5.1. Já que um elemento é adicionado na string e dois ciclos bons são criados, essa inserção ϕ satisfaz $\Delta c_g(\mathcal{I}^{ig}, \phi) = 1$. □

Lema 5.3.6. *Para qualquer grafo $G(\mathcal{I}^{ig})$ que contém um ciclo unitário $C = (o_1, d_1)$, se C é desbalanceado ou rotulado, então existe uma sequência S, tal que $|S| \leq 2$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$.*

Demonstração. Se a aresta de origem e_{o_1} é rotulada ou se C é um ciclo negativo, então uma deleção ψ pode remover o rótulo dessa aresta e tornar C em um ciclo não negativo. Essa deleção torna C em um ciclo unitário bom ou em um ciclo unitário não negativo com $\ell(e_{o_1}) = \emptyset$. No primeiro caso, temos que $S = (\psi)$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$. No segundo caso, o novo ciclo unitário satisfaz as condições do Lema 5.3.5 e, portanto, existe *indel* β com $\Delta c_g(\mathcal{I}^{ig}, \beta) = 1$. Nesse caso, temos que $S = (\psi, \beta)$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$.

Caso contrário, o ciclo unitário C é não negativo e $\ell(e_{o_1}) = \emptyset$. Pelo Lema 5.3.5, existe *indel* β com $\Delta c_g(\mathcal{I}^{ig}, \beta) = 1$ e, portanto, temos que $S = (\beta)$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$. □

Lema 5.3.7. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se existe um ciclo rotulado C divergente, então existe uma sequência S, tal que $|S| \leq 2$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$.*

Demonstração. Seja $C = (o_1, d_1, \ldots, o_k, d_k)$ um ciclo divergente rotulado. Seja $(e_{o_x}, e_{o_{x+1}})$ um par divergente de C tal que x é mínimo. Uma reversão ρ aplicada nessas duas arestas de origem $(e_{o_x}, e_{o_{x+1}})$ transforma C em um ciclo unitário C' e um $(k-1)$-ciclo C'', como mostrado na Figura 5.2. Assuma, sem perda de generalidade, que o ciclo unitário C' possui a aresta de destino com índice o_{x+1} no novo grafo. Se a aresta de origem $e_{o_{x+1}}$ é rotulada, então a reversão ρ move qualquer elemento α entre os vértices que são extremidades de $e_{o_{x+1}}$, de forma que os elementos α são acumulados na aresta com índice o_x no novo grafo.

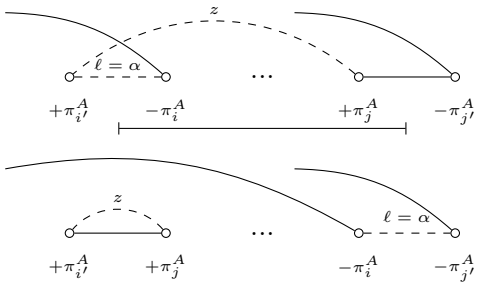


Figura 5.2: Reversão aplicada em um par divergente de um ciclo C que transforma esse ciclo em um ciclo unitário C' e um $(k-1)$-ciclo C''.

Além disso, se $w(e'_{d_x}) < w(e_{o_{x+1}})$, a reversão move o custo excedente de $e_{o_{x+1}}$ para e_{o_x}, fazendo com que o ciclo unitário C' seja um ciclo balanceado. Caso contrário, a reversão não move nucleotídeos das arestas e_{o_x} e $e_{o_{x+1}}$, fazendo com que o ciclo unitário C' seja não negativo. Se C' é balanceado e limpo, então temos $S = (\rho)$ com $\Delta c_g(\mathcal{I}^{ig}, S) = 1$. Caso contrário, C' satisfaz as condições do Lema 5.3.5 e, portanto, existe *indel* β com $\Delta c_g(\mathcal{I}^{ig}, \beta) = 1$. Nesse caso, temos que $S = (\rho, \beta)$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$. □

Lema 5.3.8. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se existe um ciclo limpo C divergente, então existe rearranjo β com $\Delta c_g(\mathcal{I}^{ig}, \beta) = 1$.*

Demonstração. Se $C = (o_1, d_1, \ldots, o_k, d_k)$ é um ciclo positivo, então uma inserção ϕ que aumenta o custo de qualquer aresta de origem de C em $\sum_{i=1}^{k} w(e'_{d_i}) - \sum_{i=1}^{k} w(e_{o_i})$ unidades (i.e., a inserção de $\sum_{i=1}^{k} w(e'_{d_i}) - \sum_{i=1}^{k} w(e_{o_i})$ nucleotídeos na região intergênica que corresponde a $w(e_{o_1})$) transforma C em um ciclo balanceado. Dessa forma, temos que $\Delta c_g(\mathcal{I}^{ig}, \phi) = 1$.

Caso contrário, o ciclo divergente C é negativo ou balanceado. Oliveira e coautores [65] demonstraram que existe reversão ρ que transforma C em dois ciclos C' e C'', aumentando o número de ciclos balanceados no grafo. Se C é negativo, então ou C' ou C'' é balanceado. Caso contrário, ambos C' e C'' são balanceados. Como C é um ciclo limpo e a reversão é aplicada em duas arestas de origem de C, temos que os ciclos C' e C'' são ciclos limpos. Portanto, temos que $\Delta c_g(\mathcal{I}^{ig}, \rho) = 1$. □

Até agora, apresentamos lemas que tratam de ciclos unitários ou ciclos divergentes. O próximo lema mostra como encontrar uma sequência de operações que aumentam o número de ciclos bons quando existem apenas ciclos convergentes no grafo.

Lema 5.3.9. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui apenas ciclos convergentes, então existe uma sequência S, tal que $|S| \leq 5$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 2$.*

Demonstração. Seja $C = (o_1, d_1, \ldots, o_k, d_k)$ um ciclo convergente de $G(\mathcal{I}^{ig})$. Oliveira e coautores [65] mostraram que um dos dois casos sempre ocorre:

- Se C é um ciclo orientado, então existe reversão ρ_1 que age em duas arestas de origem de C, transformando o ciclo C em um ciclo divergente C'.

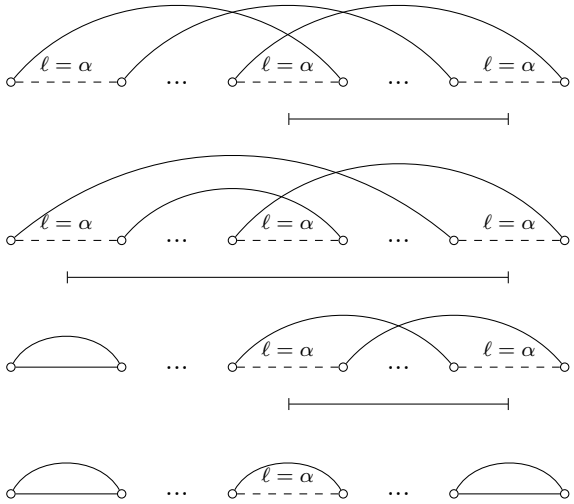


Figura 5.3: Exemplo das operações aplicadas pelo Lema 5.3.9 quando o ciclo C é orientado.

- Se C é não orientado, então existe reversão ρ_1 que age em duas arestas de origem de um outro ciclo convergente D, transformando o ciclo C em um ciclo divergente C'.

Além disso, Oliveira e coautores [65] mostraram que o número de ciclos do grafo permanece o mesmo após a aplicação da reversão ρ_1.

Pelos lemas 5.3.7 e 5.3.8, existe uma sequência S_1 que age no ciclo C', tal que $|S_1| \leq 2$ e $\Delta c_g(\mathcal{G}_o \cdot \rho, \mathcal{G}_d, S_1) = 1$. Note que a sequência S_1 possui apenas uma reversão e, caso $|S_1| = 2$, o segundo rearranjo de S_1 é um indel. Seja ρ_2 a reversão de S_1. Seja $\mathcal{G}_o^{'} = (A', A) = (\mathcal{G}_o \cdot \rho_1) \cdot \rho_2$, ou seja, o genoma resultante após a aplicação da reversão ρ_1, que cria um ciclo divergente no grafo, e da primeira operação da sequência S_1.

Note que o grafo $G(\mathcal{I}^{ig})$ possui apenas ciclos convergentes se, e somente se, não existe elemento A_i em A com sinal "$-$". Suponha, por contradição, que o grafo $G(\mathcal{G}_o^{'}, \mathcal{G}_d)$ possui apenas ciclos convergentes. Isso implica que após aplicar as reversões ρ_1 e ρ_2, não existe elemento A_i' em A' com sinal "$-$". Isso implica que tanto ρ_1 quanto ρ_2 agem no mesmo intervalo e, portanto, temos que $A = A'$. Isso contradiz o fato de que a reversão ρ_1 não altera o número de ciclos do grafo e a reversão ρ_2 transforma C' em dois ciclos. Portanto, o grafo $G(\mathcal{G}_o^{'}, \mathcal{G}_d)$ possui um ciclo divergente.

Como um *indel* não consegue transformar um ciclo divergente em convergente, o grafo $G(\mathcal{G}_o^{''}, \mathcal{G}_d)$, onde $\mathcal{G}_o^{''} = (\mathcal{G}_o \cdot \rho_1) \cdot S_1$, também possui um ciclo divergente D'. Pelos lemas 5.3.7 e 5.3.8, existe uma sequência S_2 que age no ciclo D', tal que $|S_2| \leq 2$ e $\Delta c_g(\mathcal{G}_o^{''}, \mathcal{G}_d, S_2) = 1$. Portanto, a sequência S, que é formada a partir de ρ_1, S_1 e S_2, satisfaz as condições do enunciado deste lema. As figuras 5.3 e 5.4 apresentam exemplos das operações de S. □

Enquanto $\mathcal{G}_o \neq \mathcal{G}_d$, o Algoritmo 13 usa uma sequência S' de acordo com os lemas 5.3.5, 5.3.6, 5.3.7, 5.3.8, e 5.3.9. Como as operações desses lemas garantem que $\Delta c_g(\mathcal{I}^{ig}, S') \geq 1$, eventualmente chegamos em um grafo de ciclos rotulado e ponderado que possui apenas ciclos unitários bons e, consequentemente, $\mathcal{G}_o = \mathcal{G}_d$. O Algoritmo 13 é uma 2.5-aproximação como mostrado no Teorema 5.3.1.

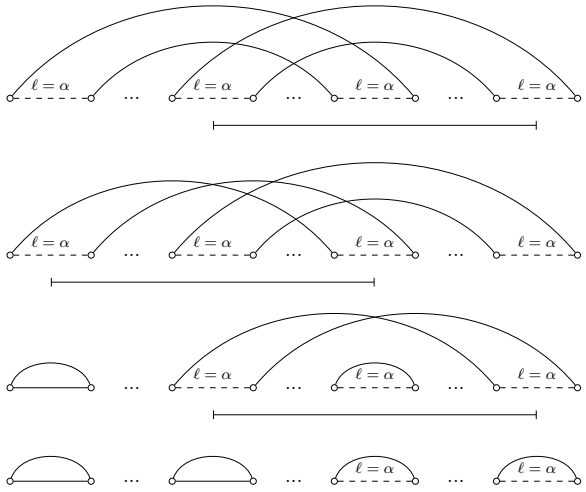


Figura 5.4: Exemplo das operações aplicadas pelo Lema 5.3.9 quando o ciclo C é não orientado.

Teorema 5.3.1. *O Algoritmo 13 é uma* 2.5*-aproximação para o o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais.*

Demonstração. A cada iteração, o algoritmo utiliza uma sequência S' de acordo com um dos lemas 5.3.5, 5.3.6, 5.3.7, 5.3.8, ou 5.3.9. No pior caso, a sequência S' satisfaz $|S'|/\Delta c_g(\mathcal{I}^{ig}, S') = 5/2$. Portanto, para transformar $\mathcal{G}_o$ em $\mathcal{G}_d$, o que é equivalente a tornar $|\pi^A| + 1 - c_g(\mathcal{I}^{ig})$ igual a zero, o algoritmo usa no máximo $5/2(|\pi^A| + 1 - c_g(\mathcal{I}^{ig}))$ operações. Pelo Lema 5.3.4, esse algoritmo é uma 2.5-aproximação. □

Para qualquer grafo de ciclos rotulado e ponderado $G(\mathcal{I}^{ig})$, tanto o número de vértices quanto o número de arestas são $O(n)$ e, portanto, temos que $|\pi^A| + 1 - c_g(\mathcal{I}^{ig}) \in O(n)$. Consequentemente, o número de iterações do laço principal do algoritmo é limitado por $O(n)$. Além disso, cada iteração possui complexidade de tempo linear, já que podemos listar todos os ciclos do grafo, achar o caso apropriado, e aplicar a sequência S', que possui tamanho de no máximo cinco operações, em tempo linear. Dessa forma, concluímos que o algoritmo possui complexidade de tempo de $O(n^2)$.

5.3.2 Uma 4-Aproximação para Modelos com Transposições

Nesta seção, apresentamos algoritmos com fator de aproximação igual a 4 para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais e o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais. Assim como na 2.5-aproximação apresentada na seção anterior, a ideia principal desses algoritmos é criar novos ciclos bons a cada iteração até que o grafo contenha apenas ciclos unitários bons.

Na seção anterior, quando só existem ciclos convergentes no grafo, demonstramos como encontrar uma sequência S tal que $|S|/\Delta c_g(\mathcal{I}^{ig}, S) \leq 5/2$ usando reversões e *indels*. Nos próximos lemas, mostramos como tratar ciclos convergentes usando uma sequência S, que possui apenas transposições e indels, tal que $|S|/\Delta c_g(\mathcal{I}^{ig}, S) \leq 2$. Lembre-se que

Algoritmo 13: Algoritmo de 2.5-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais

Entrada: Uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$
Saída: Uma sequência de rearranjos S que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$

Seja $S \leftarrow \emptyset$
enquanto $\mathcal{G}_o \neq \mathcal{G}_d$ **faça**
  se $G(\mathcal{I}^{ig})$ *possui ciclo unitário que é rotulado ou desbalanceado* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.5 ou 5.3.6
  senão se $G(\mathcal{I}^{ig})$ *possui ciclo divergente* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.7 ou 5.3.8
  senão
    Seja S' uma sequência de acordo com o Lema 5.3.9
  $\mathcal{G}_o \leftarrow \mathcal{G}_o \cdot S'$
  Adicione as operações de S' na sequência S
retorne a sequência S

para strings sem sinais, o grafo de ciclos rotulado e ponderado $G(\mathcal{I}^{ig})$ só possui ciclos convergentes.

Quando consideramos apenas operações que agem em arestas de origem de ciclos bons, os resultados de problemas intergênicos que não consideram *indels* podem ser diretamente aplicados. A seguir, listamos alguns desses resultados.

Lema 5.3.10 (Oliveira e coautores [67], Lema 4.6)**.** *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui um ciclo orientado bom C, então existe uma sequência S tal que $|S| \leq 3$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 2$.*

Lema 5.3.11 (Oliveira e coautores [67], Lema 4.7)**.** *Para qualquer grafo $G(\mathcal{I}^{ig})$, se existe k-ciclo não orientado C em $G(\mathcal{I}^{ig})$, com $k > 2$, não existem ciclos orientados em $G(\mathcal{I}^{ig})$, e todos os ciclos em $G(\mathcal{I}^{ig})$ são bons, então existe uma sequência S tal que $|S| \leq 7$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 4$.*

Lema 5.3.12 (Oliveira e coautores [67], Lema 4.8)**.** *Para qualquer grafo $G(\mathcal{I}^{ig})$, se todo k-ciclo C em $G(\mathcal{I}^{ig})$ é um ciclo bom e $k \leq 2$, então existe uma sequência S tal que $|S| \leq 2$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 2$.*

Os próximos lemas apresentam sequências de transposições ou *indels* que agem em ciclos convergentes rotulados ou desbalanceados.

Lema 5.3.13. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui um ciclo orientado C rotulado ou desbalanceado, então existe transposição τ que transforma C em três novos ciclos, tal que um deles é um ciclo unitário não negativo que possui aresta de origem limpa.*

Demonstração. Como C é um ciclo orientado, sempre existe tripla orientada o_i, o_j e o_k, com $i < j < k$, tal que $o_i > o_k > o_j$ e $k = j + 1$ [17]. Uma transposição aplicada nessas três arestas de origem cria três novos ciclos, de forma que a aresta de origem com índice o_j

no novo grafo pertence a um ciclo unitário, já que $k = j + 1$. Sempre podemos escolher a transposição de forma que o ciclo unitário é não negativo e possui aresta de origem limpa. Para isso, a transposição precisa mover qualquer elemento α e nucleotídeos excedentes, caso existam, da aresta de origem com índice o_j para um dos outros dois ciclos. □

Lema 5.3.14. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui um ciclo orientado C rotulado ou desbalanceado, então existe uma sequência S tal que $|S| \leq 2$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$.*

Demonstração. Diretamente dos lemas 5.3.13 e 5.3.5. Note que, após aplicar a transposição do Lema 5.3.13, existe um ciclo unitário C' que é um ciclo bom ou que satisfaz as condições do Lema 5.3.5. □

Lema 5.3.15. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui pelo menos dois ciclos rotulados ou desbalanceados C e D, tal que C e D são não orientados e não unitários, então existe uma transposição τ que transforma C e D nos ciclos C' e D', tal que ou C' ou D' é um ciclo unitário não negativo que possui aresta de origem limpa.*

Demonstração. Sejam $C = (o_1, d_1, o_2, d_2 \ldots, o_x, d_x)$ e $D = (o'_1, d'_1, o'_2, d'_2, \ldots, o'_y, d'_y)$. Suponha, sem perda de generalidade, que $o_1 < o'_1$. Uma transposição aplicada nas arestas de origem o_1, o_x e o'_1 transforma esses ciclos em dois ciclos C' e D', tal que C' é um ciclo unitário que contém a aresta de destino d_x e D' é um $(y + x - 1)$-ciclo. Podemos mover qualquer elemento α e nucleotídeos excedentes da aresta de origem o_x, onde o ciclo unitário é formado, para o ciclo D'. Dessa forma, podemos garantir que C' é um ciclo unitário não negativo que possui aresta de origem limpa. □

Lema 5.3.16. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui pelo menos dois ciclos rotulados ou desbalanceados C e D, tal que C e D são não orientados e não unitários, então existe uma sequência S tal que $|S| \leq 2$ e $\Delta c_g(\mathcal{I}^{ig}, S) = 1$.*

Demonstração. Diretamente dos lemas 5.3.15 e 5.3.5. Note que, após aplicar a transposição do Lema 5.3.15, existe um ciclo unitário C' que é um ciclo bom ou satisfaz as condições do Lema 5.3.5. □

A Figura 5.5 mostra exemplos das transposições aplicadas de acordo com os lemas 5.3.13 e 5.3.15.

Lema 5.3.17. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui apenas um ciclo rotulado ou desbalanceado C, tal que C é um 2-ciclo não orientado, e todos os outros ciclos de $G(\mathcal{I}^{ig})$ são bons e não orientados, então existe uma sequência S tal que $|S|/\Delta c_g(\mathcal{I}^{ig}, S) \leq 2$.*

Demonstração. Primeiramente, precisamos aplicar *indels* de acordo com a Figura 5.6 para transformar C em ciclos bons. No pior caso, todas as arestas de C são rotuladas. Precisamos aplicar uma inserção de acordo com a Figura 5.6b para remover os rótulos das duas arestas de origem de C. Note que o ciclo unitário mais à esquerda sempre pode ser um ciclo bom, já que podemos realizar a inserção antes de qualquer elemento α e atribuir qualquer custo à aresta de origem desse ciclo unitário. Nesse caso, a inserção cria ciclos C' (unitário), C'' (2-ciclo) e C''' (unitário), tal que apenas C' é um ciclo bom, mas todos eles são não positivos, já que essa inserção pode adicionar nucleotídeos na aresta de origem

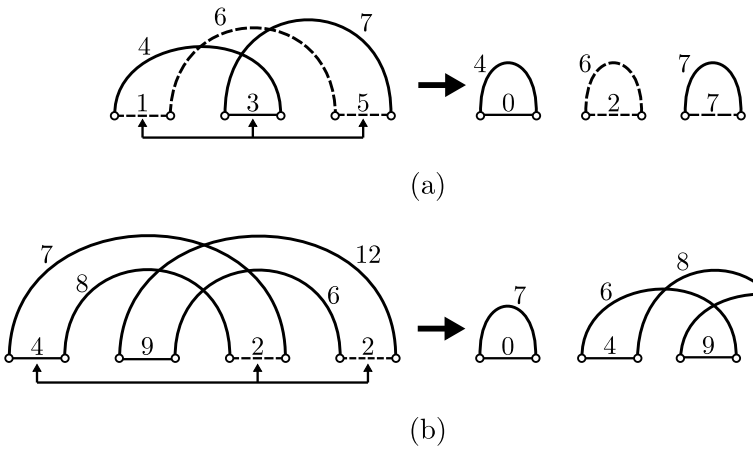


Figura 5.5: Exemplos de transposições aplicadas pelos lemas 5.3.13 e 5.3.15. **(a)** Nesse caso, existe um ciclo orientado C rotulado e desbalanceado. Existe uma transposição que transforma esse ciclo em três novos ciclos, tal que um deles é um ciclo unitário não negativo que possui aresta de origem limpa. **(b)** Nesse caso, existem dois ciclos não unitários C e D rotulados e desbalanceados. Existe uma transposição aplicada em três arestas de origem desses dois ciclos que transforma C e D nos ciclos C' e D', tal que um desses novos ciclos é um ciclo unitário não negativo que possui aresta de origem limpa.

mais à esquerda de C'' e na única aresta de origem de C'''. Note que a aresta mais à esquerda de C'' foi criada pela inserção e, consequentemente, essa aresta de origem é limpa e podemos atribuir qualquer custo a ela. Agora, apenas precisamos aplicar uma deleção para cada um das duas arestas de origem rotuladas, tornando C'' e C''' em ciclos bons.

Até agora, no pior caso, temos uma sequência S_1 com três *indels* e $\Delta c_g(\mathcal{I}^{ig}, S_1) = 1$. Podemos aplicar os lemas 5.3.11 ou 5.3.12 no novo grafo $G(\mathcal{G}_o^{'}, \mathcal{G}_d)$, onde $\mathcal{G}_o^{'} = \mathcal{G}_o \cdot S_1$. Usando a sequência S_1 e o Lema 5.3.11, podemos construir uma sequência S, com no máximo dez operações, tal que $\Delta c_g(\mathcal{I}^{ig}, S) = 5$. Já ao usar a sequência S_1 e o Lema 5.3.12, podemos construir uma sequência S, com no máximo cinco operações, tal que $\Delta c_g(\mathcal{I}^{ig}, S) = 3$. □

Lema 5.3.18. *Para qualquer grafo $G(\mathcal{I}^{ig})$, se $G(\mathcal{I}^{ig})$ possui apenas um ciclo rotulado ou desbalanceado C, tal que C é um x-ciclo não orientado com $x > 2$, e todos os outros ciclos de $G(\mathcal{I}^{ig})$ são bons e não orientados, então existe uma sequência S tal que $|S|/\Delta c_g(\mathcal{I}^{ig}, S) \leq 2$.*

Demonstração. Seja $C = (o_1, d_1, o_2, d_2 \ldots, o_x, d_x)$. Suponha que existe outro ciclo não orientado $D = (o'_1, d'_1, o'_2, d'_2, \ldots, o'_y, d'_y)$, que é um ciclo bom de acordo com o enunciado deste lema, tal que existem triplas (o_i, o_j, o_k) e (o'_r, o'_s, o'_t) que satisfazem uma das seguintes condições: $o_i < o'_r < o_j < o'_s < o_k < o'_t$ ou $o_i > o'_r > o_j > o'_s > o_k > o'_t$. Bafna e Pevzner [17] mostraram que uma transposição τ aplicada nas arestas de origem (o'_r, o'_s, o'_t) cria novos ciclos C' e D' com os mesmos conjuntos de vértices de C e D, respectivamente, tal que C' é um ciclo orientado. Além disso, Bafna e Pevzner [17] mostraram que uma transposição aplicada na tripla orientada de C' torna D' orientado no novo grafo. Pelo Lema 5.3.14, existe uma sequência S_1, tal que $|S_1| \leq 2$ e $\Delta c_g(\mathcal{G}_o \cdot \tau, \mathcal{G}_d, S_1) = 1$, aplicada na tripla orientada de C' que transforma D' em um ciclo orientado D''. Como D'' possui os mesmos vértices de D, sabemos que D'' é um ciclo bom. Dessa forma, usando o Lema 5.3.10, existe uma sequência S_2 que é aplicada no ciclo D'', tal que S_2 possui no

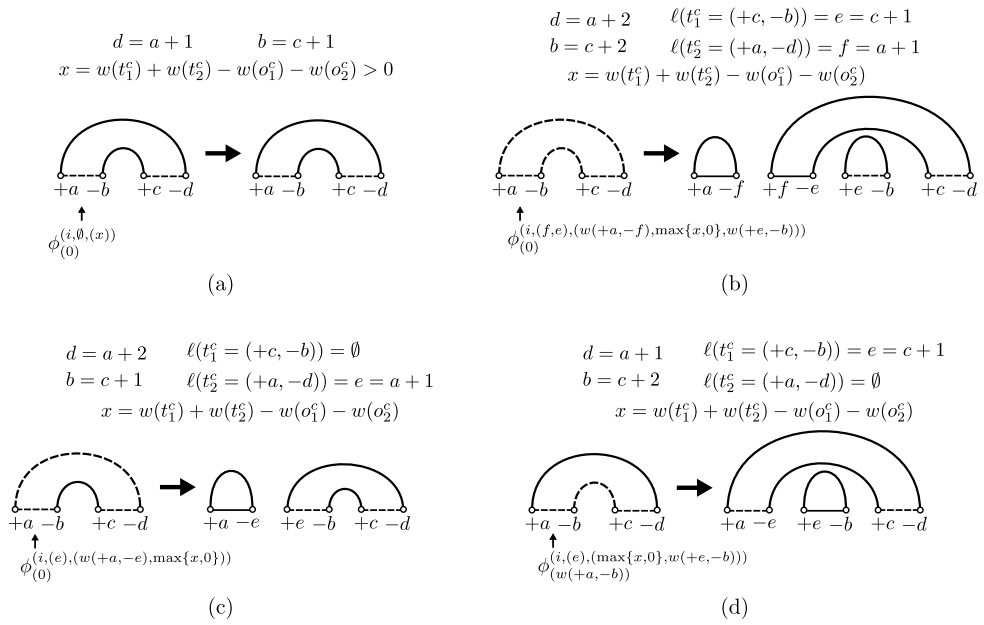


Figura 5.6: Quatro possíveis casos de um inserção aplicada em um 2-ciclo C. **(a)** Nesse caso, C é positivo e ambas arestas de destino de C são limpas, então usamos um *indel* que adiciona a quantidade necessária de nucleotídeos para tornar C balanceado. **(b)** Nesse caso, ambas arestas de destino de C são rotuladas, então usamos um *indel* que adiciona dois elementos, gerando um ciclo unitário bom e um ciclo unitário não positivo, além de adicionar nucleotídeos suficientes para tornar o 2-ciclo em um ciclo não positivo. **(c-d)** Nesses casos, apenas uma das arestas de destino de C é rotulada, então usamos um *indel* que adiciona um elemento, gerando um ciclo unitário bom, além de adicionar nucleotídeos suficientes para tornar o 2-ciclo em um ciclo não positivo.

máximo três operações e adiciona dois ciclos bons no grafo. Portanto, podemos construir uma sequência S tal que $|S| \leq 6$ e $\Delta c_g(\mathcal{I}^{ig}, S) \leq 3$.

Se a condição anterior não é verdadeira, Bafna e Pevzner [17] demonstraram que existem ciclos D e E, que são ciclos bons de acordo com o enunciado deste lema, tal que existe uma transposição τ aplicada nas arestas de D e E que cria dois novos ciclos D' e E', além de transformar C em um ciclo orientado C'. Oliveira e coautores [67] demonstraram como escolher a transposição de forma que os novos ciclos D' e E' sejam ciclos balanceados. Além disso, D' e E' são ciclos limpos, pois são formados pelos mesmos vértices dos ciclos bons D e E. Bafna e Pevzner [17] também demostraram que, após aplicar uma transposição na tripla orientada de C', existe um ciclo orientado F' no grafo, tal que F' tem o mesmo conjunto de vértices de D' ou E'.

De forma similar ao caso anterior, podemos usar uma sequência S_1 (Lema 5.3.10) na tripla orientada de C', tal que $|S_1| \leq 2$ e $\Delta c_g(\mathcal{G}_o \cdot \tau, \mathcal{G}_d, S_1) = 1$, e uma sequência S_2 (Lema 5.3.10) no ciclo F', tal que S_2 possui no máximo três operações e adiciona dois ciclos bons no grafo. □

As transposições descritas na prova do Lema 5.3.18, que criam novos ciclos orientados, são similares às transposições dos exemplos das figuras 4.12 e 4.13, que foram apresentadas no Capítulo 4. Com esses lemas, podemos apresentar os algoritmos 14 e 15. Assim como o Algoritmo 13, esses algoritmos possuem complexidade de tempo de $O(n^2)$.

Algoritmo 14: Algoritmo de 4-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais

Entrada: Uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$
Saída: Uma sequência de rearranjos S que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$

Seja $S \leftarrow \emptyset$
enquanto $\mathcal{G}_o \neq \mathcal{G}_d$ **faça**
  se $G(\mathcal{I}^{ig})$ *possui ciclo unitário que é rotulado ou desbalanceado* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.5 ou 5.3.6
  senão se $G(\mathcal{I}^{ig})$ *possui ciclo orientado* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.10 ou 5.3.14
  senão
    Seja S' uma sequência de acordo com os lemas 5.3.11, 5.3.12, 5.3.16, 5.3.17, ou 5.3.18
  $\mathcal{G}_o \leftarrow \mathcal{G}_o \cdot S'$
  Adicione as operações de S' na sequência S
retorne a sequência S

No Teorema 5.3.2, demonstramos que esses algoritmos possuem fator de aproximação igual a 4 para os problemas de Distância de Rearranjos Intergênicos considerando os modelos $\mathcal{M}_{\tau}^{\phi,\psi}$ e $\mathcal{M}_{\rho,\tau}^{\phi,\psi}$.

Teorema 5.3.2. *Os algoritmos 14 e 15 são algoritmos de* 4*-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais e o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais, respectivamente.*

Demonstração. Similar à prova do Teorema 5.3.1, mas usando o fato de que a cada iteração o algoritmo usa uma sequência S' tal que $|S'|/\Delta c_g(\mathcal{I}^{ig}, S') \leq 2$. □

5.4 Conclusões

Neste capítulo, estudamos os problemas de Distância de Rearranjos Intergênicos em Genomas Desbalanceados. Consideramos modelos que contém a combinação de *indels* com reversões e/ou transposições, para strings com ou sem sinais.

Na Seção 5.1, demonstramos que os seguintes problemas são NP-difíceis: a Distância de Reversões e Indels Intergênicos em Strings sem Sinais; a Distância de Transposições e Indels Intergênicos em Strings sem Sinais; e a Distância de Transposições, Reversões e Indels Intergênicos em Strings com ou sem Sinais.

Na Seção 5.2, apresentamos algoritmos de aproximação que usam o conceito de *breakpoints* intergênicos, considerando strings sem sinais. Já na Seção 5.3, apresentamos algoritmos de aproximação usando o grafo de ciclos rotulado e ponderado, uma nova estrutura que representa uma instância intergênica de genomas desbalanceados. A Tabela 5.1 resume o fator de aproximação alcançado para cada problema estudado neste capítulo.

Algoritmo 15: Algoritmo de 4-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais

Entrada: Uma instância intergênica $\mathcal{I}^{ig} = (\mathcal{G}_o, \mathcal{G}_d)$, com $\mathcal{G}_o = (A, \breve{A})$ e $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$

Saída: Uma sequência de rearranjos S que transforma $\mathcal{G}_o$ em $\mathcal{G}_d$

Seja $S \leftarrow \emptyset$
enquanto $\mathcal{G}_o \neq \mathcal{G}_d$ **faça**
  se $G(\mathcal{I}^{ig})$ *possui ciclo unitário que é rotulado ou desbalanceado* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.5 ou 5.3.6
  senão se $G(\mathcal{I}^{ig})$ *possui ciclo divergente* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.7 ou 5.3.8
  senão se $G(\mathcal{I}^{ig})$ *possui ciclo orientado* **então**
    Seja S' uma sequência de acordo com os lemas 5.3.10 ou 5.3.14
  senão
    Seja S' uma sequência de acordo com os lemas 5.3.11, 5.3.12, 5.3.16, 5.3.17, ou 5.3.18
  $\mathcal{G}_o \leftarrow \mathcal{G}_o \cdot S'$
  Adicione as operações de S' na sequência S
retorne a sequência S

Tabela 5.1: Resumo dos algoritmos apresentados neste capítulo para os problemas de Distância de Rearranjos Intergênicos em Genomas Desbalanceados.

Modelo	Seção 5.2	Seção 5.3
Reversões e Indels (com sinais)	-	2.5-aproximação
Reversões e Indels (sem sinais)	4-aproximação	-
Transposições e Indels (sem sinais)	4.5-aproximação	4-aproximação
Transposições, Reversões e Indels (com sinais)	-	4-aproximação
Transposições, Reversões e Indels (sem sinais)	6-aproximação	-

Capítulo 6

Resultados Experimentais

Neste capítulo, apresentamos experimentos computacionais com genomas sintéticos e genomas reais, considerando os algoritmos desenvolvidos nos capítulos 4 e 5. As implementações de todos os algoritmos deste capítulo estão disponíveis em um repositório público[1]. Este capítulo está dividido da seguinte forma. Na Seção 6.1, descrevemos o procedimento de criação das bases de dados de genomas sintéticos. Na Seção 6.2, apresentamos os resultados dos experimentos considerando genomas sintéticos e os algoritmos do Capítulo 4. Já na Seção 6.3, apresentamos os resultados dos experimentos considerando genomas sintéticos e os algoritmos do Capítulo 5. Por fim, na Seção 6.4, apresentamos experimentos usando genomas reais de cianobactérias da base de dados Cyanorak 2.1 [48].

6.1 Criação de Instâncias Sintéticas

Dados os parâmetros n, k, $\mathcal{M}$, e t, onde n é a quantidade de genes no genoma de destino, k é o número de rearranjos conservativos (reversões ou transposições) e o número de rearranjos não conservativos a serem aplicados (inserções ou deleções), $\mathcal{M}$ é um modelo de rearranjo, e t indica o tipo das strings (`signed` ou `unsigned`), criamos uma instância intergênica sintética da seguinte forma:

1. Criamos uma lista $\breve{\iota}^n$ de $n+1$ números inteiros escolhidos de forma aleatória, considerando uma distribuição uniforme dos valores no intervalo $[0, 100]$;
2. Criamos a string identidade ι^n, tal que ι^n é uma string com sinais, se $t =$ `signed`, ou ι^n é uma string sem sinais, se $t =$ `unsigned`;
3. Criamos o genoma de destino $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$;
4. Inicializamos o genoma de origem $\mathcal{G}_o$ como uma cópia de $\mathcal{G}_d$;
5. Aplicamos k rearranjos conservativos em $\mathcal{G}_o$. As operações são definidas de acordo com as operações de $\mathcal{M}$:
   - k reversões, se $\mathcal{M}$ contém reversões e não contém transposições (i.e., $\mathcal{M} = \mathcal{M}_\rho^{\phi,\psi}$);

[1] https://github.com/compbiogroup

- k transposições, se $\mathcal{M}$ contém transposições e não contém reversões (i.e., $\mathcal{M} = \mathcal{M}_{\tau}^{\phi,\psi}$);
- k reversões ou transposições, se $\mathcal{M}$ contém reversões e transposições (i.e., $\mathcal{M} = \mathcal{M}_{\rho,\tau}^{\phi,\psi}$). Nesse caso, cada operação possui 50% de chance de ser uma reversão ou uma transposição.

6. Aplicamos $k/2$ deleções em $\mathcal{G}_o$;

7. Por fim, aplicamos $k/2$ inserções em $\mathcal{G}_o$.

O nosso procedimento aplica deleções antes das inserções a fim de evitar que um elemento seja adicionado e, posteriormente, removido no processo de criação da instância. Cada rearranjo usado possui parâmetros escolhidos de maneira aleatória, considerando uma distribuição uniforme do conjunto de todos os valores válidos para os parâmetros daquela operação. Esse procedimento usa inserções que adicionam, numa posição escolhida aleatoriamente, um segmento que contém um único caractere, que ainda não pertence a string, e duas regiões intergênicas, sendo que os tamanhos dessas duas regiões intergênicas são escolhidos de forma aleatória, considerando uma distribuição uniforme dos valores no intervalo $[0, 100]$. Na criação de uma instância clássica sintética, seguimos o mesmo processo e desconsideramos as regiões intergênicas.

Nossa base de dados é dividida em conjuntos agrupados pelos parâmetros n, k, $\mathcal{M}$, e t. Para $n \in \{50, 100, \ldots, 500\}$, $k \in \{n/2, n\}$, $\mathcal{M} \in \{\mathcal{M}_{\rho}^{\phi,\psi}, \mathcal{M}_{\tau}^{\phi,\psi}, \mathcal{M}_{\rho,\tau}^{\phi,\psi}\}$ e $t \in \{\texttt{signed}, \texttt{unsigned}\}$, cada conjunto $DS_{n,k,t}^{\mathcal{M}}$ possui 1000 instâncias clássicas sintéticas criadas usando os parâmetros n, k, $\mathcal{M}$ e t. Note que quando $\mathcal{M} = \mathcal{M}_{\tau}^{\phi,\psi}$, consideramos que $t = \texttt{unsigned}$. Além disso, usamos as mesmas combinações de parâmetros para a criação de cada conjunto $DSI_{n,k,t}^{\mathcal{M}}$ com 1000 instâncias intergênicas sintéticas.

6.2 Experimentos com Instâncias Clássicas

Os algoritmos gulosos baseados em *breakpoints* do Capítulo 4 possuem complexidade de tempo de $O(n^3)$ ou $O(n^4)$. Por isso, decidimos implementar uma versão desses algoritmos que possui complexidade de tempo de $O(n^2)$. Nessa implementação, ao invés de encontrar a operação β com melhor valor de $\Delta\Phi(\mathcal{I}, \beta) + \Delta b_{\mathcal{M}}(\mathcal{I}, \beta)$, apenas encontramos uma operação que garante o fator de aproximação, a cada iteração. Essas operações já são descritas nas provas dos lemas do Capítulo 4. Dessa forma, podemos compará-los de forma mais justa com os algoritmos baseados em grafos de ciclos, que também possuem complexidade de tempo de $O(n^2)$.

Cada tabela desta seção apresenta resultados para um algoritmo do Capítulo 4 e um valor de $k \in \{n/2, n\}$. Cada linha dessas tabelas possui dados referentes a execução do algoritmo usando todas as instâncias do conjunto $DS_{n,k,t}^{\mathcal{M}}$, sendo que o valor de n é descrito na primeira coluna da tabela, o valor de k é descrito no cabeçalho da tabela, e os valores de $\mathcal{M}$ e t são inferidos a partir do algoritmo considerado. A segunda, terceira e quarta colunas apresentam os valores de mínimo, média e máximo, respectivamente, para o tamanho da sequência de rearranjos das soluções encontradas pelo algoritmo. A quinta, sexta e sétima

colunas apresentam os valores de mínimo, média e máximo, respectivamente, para o fator de aproximação prático das soluções encontradas pelo algoritmo. O fator de aproximação prático para uma instância é calculado usando o tamanho da solução encontrada pelo algoritmo para essa instância e o seu limitante inferior, sendo que usamos o limitante inferior que corresponde ao fator de aproximação teórico do algoritmo considerado. Note que cada problema tem um limitante inferior específico. Como esperado, em todas as tabelas, os valores referentes à quantidade de operações usadas na solução aumentam à medida que o valor de n aumenta.

As tabelas 6.1 e 6.2 apresentam os resultados do algoritmo de 2-aproximação para a Distância de Reversões e Indels em Strings sem Sinais (Algoritmo 4). Esse algoritmo usa o conceito de *breakpoints*. Apesar do fato de que as instâncias sintéticas da Tabela 6.2 foram criadas usando o dobro de operações em relação às instâncias da Tabela 6.1, a razão entre o tamanho das soluções do experimento da Tabela 6.2 e o tamanho das soluções do experimento da Tabela 6.1 é menor que 1.5 para todas as medidas usadas. Para a Tabela 6.1, a média do fator de aproximação prático está entre 1.68 e 1.81. Já para a Tabela 6.2, os valores para a média do fator de aproximação prático estão entre 1.81 e 1.94. Em ambas as tabelas, os valores dessa coluna formam uma sequência não decrescente. Na Tabela 6.1, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.40 e 1.92, respectivamente, e ocorreram em instâncias com $n = 50$. Na Tabela 6.2, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.64 e 2.00, respectivamente, e ocorreram em instâncias com $n = 50$.

As tabelas 6.3 e 6.4 apresentam os resultados do algoritmo de 3-aproximação para a Distância de Transposições e Indels em Strings sem Sinais (Algoritmo 5), que usa o conceito de *breakpoints*. Já as tabelas 6.5 e 6.6 apresentam os resultados do algoritmo de 2-aproximação (Algoritmo 9), que usa o conceito de grafo de ciclos rotulado, para o mesmo problema. Como esperado, o algoritmo de 2-aproximação possui resultados práticos muito melhores do que o algoritmo de 3-aproximação. Agora, discutiremos com mais detalhes os resultados para cada algoritmo.

A razão entre o tamanho das soluções do experimento da Tabela 6.4 e o tamanho das soluções do experimento da Tabela 6.3 é menor que 1.4 para todas as medidas usadas. Para a Tabela 6.3, a média do fator de aproximação prático foi menor que 2.8 para todos os valores de n. Já na Tabela 6.4, a média do fator de aproximação prático foi menor que 2.8 apenas para $n \in \{50, 100\}$. Os valores dessa coluna, em ambas as tabelas, formam uma sequência não decrescente, sendo que os maiores valores foram de 2.79 e 2.91 nas tabelas 6.3 e 6.4, respectivamente. Para a Tabela 6.4, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.24 e 2.96, respectivamente. Já na Tabela 6.3, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.00 e 2.89, respectivamente.

A razão entre o tamanho das soluções do experimento da Tabela 6.6 e o tamanho das soluções do experimento da Tabela 6.5 é menor que 1.28 para todas as medidas usadas. Em ambas as tabelas, os valores de mínimo, média e máximo para o fator de aproximação prático são consideravelmente menores do que o fator de aproximação teórico. Todos os valores dessas colunas, em ambas as tabelas, estão entre 1.18 e 1.33, o que mostra uma boa estabilidade do algoritmo. Em quase todos os casos, o algoritmo de 3-aproximação

encontra soluções com tamanho maior que o dobro do tamanho das soluções encontradas pelo algoritmo de 2-aproximação.

As tabelas 6.7 e 6.8 apresentam os resultados do algoritmo de 3-aproximação para a Distância de Reversões, Transposições e Indels em Strings sem Sinais (Algoritmo 6), que usa o conceito de *breakpoints*. A razão entre o tamanho das soluções do experimento da Tabela 6.8 e o tamanho das soluções do experimento da Tabela 6.7 é menor que 1.42 para todas as medidas usadas. Para a Tabela 6.7, os valores da sexta coluna (média do fator de aproximação prático) estão entre 2.53 e 2.80. Já para a Tabela 6.8, os valores para a média do fator de aproximação prático estão entre 2.70 e 2.93. Em ambas as tabelas, os valores dessa coluna formam uma sequência não decrescente. Na Tabela 6.7, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.17 e 2.94, respectivamente. Já na Tabela 6.8, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.32 e 2.98, respectivamente.

Por fim, as tabelas 6.9 e 6.10 apresentam os resultados do algoritmo de 2-aproximação para a Distância de Transposições, Reversões e Indels em Strings com Sinais (Algoritmo 10), que usa o conceito de grafo de ciclos rotulado. Assim como os resultados do algoritmo de 2-aproximação para a Distância de Transposições e Indels, essas tabelas mostram que o fator de aproximação prático é consideravelmente menor que o teórico. A razão entre o tamanho das soluções do experimento da Tabela 6.10 e o tamanho das soluções do experimento da Tabela 6.9 é menor que 1.43 para todas as medidas usadas. Para a Tabela 6.9, os valores para a média do fator de aproximação prático estão entre 1.18 e 1.22. Para a Tabela 6.10, os valores para a média do fator de aproximação prático estão entre 1.17 e 1.22. Na Tabela 6.9, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.16 e 1.31, respectivamente. Já na Tabela 6.10, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.15 e 1.24, respectivamente. Esses valores mostram que o algoritmo possui boa estabilidade ao considerar o fator de aproximação prático.

Tabela 6.1: Resultados experimentais do algoritmo de 2-aproximação para a Distância de Reversões e Indels em Strings sem Sinais (Algoritmo 4), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	34	42.62	51	1.40	1.68	1.92
100	77	88.84	102	1.50	1.74	1.90
150	122	134.69	147	1.60	1.76	1.89
200	166	181.83	197	1.68	1.78	1.91
250	207	227.88	249	1.69	1.79	1.88
300	253	275.46	294	1.69	1.80	1.88
350	296	321.41	342	1.72	1.80	1.88
400	344	368.55	392	1.71	1.81	1.88
450	388	415.28	451	1.74	1.81	1.88
500	433	462.67	488	1.74	1.81	1.88

Tabela 6.2: Resultados experimentais do algoritmo de 2-aproximação para a Distância de Reversões e Indels em Strings sem Sinais (Algoritmo 4), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	50	58.74	66	1.64	1.81	2.00
100	107	120.18	130	1.75	1.88	1.97
150	169	182.69	197	1.80	1.90	1.97
200	229	244.79	260	1.83	1.91	1.98
250	290	307.00	326	1.86	1.92	1.97
300	350	369.37	389	1.88	1.93	1.97
350	410	431.73	454	1.87	1.93	1.98
400	472	494.47	517	1.89	1.94	1.98
450	537	557.16	579	1.89	1.94	1.98
500	597	620.05	642	1.90	1.94	1.97

Tabela 6.3: Resultados experimentais do algoritmo de 3-aproximação para a Distância de Transposições e Indels em Strings sem Sinais (Algoritmo 5), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	34	43.08	53	2.00	2.39	2.79
100	80	93.13	107	2.31	2.57	2.81
150	126	143.17	158	2.44	2.66	2.85
200	175	194.60	213	2.52	2.70	2.84
250	225	245.86	266	2.56	2.73	2.87
300	277	297.38	322	2.63	2.75	2.85
350	322	348.58	373	2.64	2.76	2.89
400	379	400.88	425	2.67	2.77	2.89
450	424	451.99	480	2.69	2.79	2.87
500	472	504.65	532	2.70	2.79	2.89

Tabela 6.4: Resultados experimentais do algoritmo de 3-aproximação para a Distância de Transposições e Indels em Strings sem Sinais (Algoritmo 5), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	47	56.96	67	2.24	2.58	2.86
100	108	119.61	130	2.50	2.75	2.91
150	171	182.95	197	2.63	2.81	2.93
200	232	246.11	262	2.68	2.84	2.93
250	289	309.89	328	2.75	2.86	2.94
300	354	373.76	389	2.78	2.88	2.95
350	418	437.47	456	2.79	2.89	2.95
400	481	501.26	520	2.83	2.90	2.96
450	545	565.42	585	2.84	2.91	2.96
500	609	629.45	649	2.86	2.91	2.96

Tabela 6.5: Resultados experimentais do algoritmo de 2-aproximação para a Distância de Transposições e Indels em Strings sem Sinais (Algoritmo 9), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	22	23.66	25	1.19	1.23	1.28
100	47	49.19	52	1.24	1.26	1.27
150	69	71.53	75	1.21	1.22	1.23
200	87	90.08	94	1.19	1.21	1.23
250	116	119.30	124	1.19	1.21	1.22
300	139	143.85	149	1.21	1.23	1.24
350	163	167.17	171	1.20	1.21	1.21
400	196	202.08	207	1.22	1.24	1.25
450	207	212.93	217	1.18	1.18	1.19
500	233	239.49	246	1.21	1.21	1.22

Tabela 6.6: Resultados experimentais do algoritmo de 2-aproximação para a Distância de Transposições e Indels em Strings sem Sinais (Algoritmo 9), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	28	29.97	32	1.28	1.31	1.33
100	53	56.87	61	1.25	1.28	1.32
150	78	81.45	85	1.25	1.27	1.30
200	104	109.31	114	1.26	1.27	1.29
250	123	128.52	133	1.21	1.22	1.24
300	161	166.38	172	1.26	1.27	1.29
350	178	185.29	192	1.22	1.23	1.24
400	214	219.87	226	1.23	1.24	1.25
450	234	241.21	248	1.22	1.23	1.24
500	267	273.73	282	1.24	1.25	1.26

Tabela 6.7: Resultados experimentais do algoritmo de 3-aproximação para a Distância de Reversões, Transposições e Indels em Strings sem Sinais (Algoritmo 6), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	36	44.63	55	2.17	2.53	2.94
100	82	93.74	106	2.40	2.66	2.89
150	127	142.47	158	2.51	2.71	2.90
200	172	192.76	212	2.55	2.74	2.92
250	218	241.93	262	2.61	2.76	2.88
300	271	293.10	315	2.61	2.77	2.88
350	317	342.29	363	2.67	2.78	2.89
400	365	393.17	419	2.67	2.79	2.89
450	418	441.94	466	2.68	2.79	2.89
500	466	492.81	519	2.71	2.80	2.90

Tabela 6.8: Resultados experimentais do algoritmo de 3-aproximação para a Distância de Reversões, Transposições e Indels em Strings sem Sinais (Algoritmo 6), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	51	58.96	68	2.32	2.70	2.95
100	109	121.74	134	2.65	2.81	2.95
150	169	185.06	198	2.73	2.86	2.96
200	229	248.19	260	2.76	2.88	2.97
250	295	311.71	326	2.80	2.90	2.97
300	354	375.19	389	2.82	2.91	2.96
350	418	438.20	458	2.84	2.91	2.98
400	484	502.15	521	2.86	2.92	2.97
450	536	565.53	588	2.86	2.92	2.97
500	606	629.18	648	2.88	2.93	2.98

Tabela 6.9: Resultados experimentais do algoritmo de 2-aproximação para a Distância de Transposições, Reversões e Indels em Strings com Sinais (Algoritmo 10), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	17	19.33	22	1.17	1.22	1.31
100	45	48.37	52	1.16	1.18	1.22
150	65	69.34	74	1.20	1.22	1.25
200	90	95.24	100	1.19	1.21	1.23
250	99	105.86	112	1.17	1.18	1.19
300	128	134.78	140	1.19	1.20	1.21
350	161	168.54	176	1.21	1.22	1.23
400	183	191.17	199	1.21	1.22	1.24
450	194	202.28	211	1.18	1.19	1.20
500	230	238.14	246	1.19	1.20	1.20

Tabela 6.10: Resultados experimentais do algoritmo de 2-aproximação para a Distância de Transposições, Reversões e Indels em Strings com Sinais (Algoritmo 10), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	23	27.59	30	1.15	1.17	1.21
100	51	55.41	60	1.16	1.18	1.21
150	82	88.08	94	1.21	1.22	1.24
200	107	112.87	120	1.17	1.18	1.19
250	132	139.17	146	1.18	1.19	1.20
300	159	165.79	173	1.18	1.20	1.21
350	188	195.62	205	1.18	1.19	1.21
400	210	219.26	227	1.18	1.19	1.20
450	236	245.37	255	1.18	1.19	1.20
500	260	269.07	279	1.16	1.17	1.18

6.3 Experimentos com Instâncias Intergênicas

As tabelas apresentadas nesta seção seguem um padrão similar às tabelas da seção anterior, mas apresentam resultados para os algoritmos do Capítulo 5 e utilizam os conjuntos de instâncias intergênicas $DSI^{\mathcal{M}}_{n,k,t}$. Como esperado, em todas as tabelas, os valores referentes à quantidade de operações usadas na solução aumentam à medida que o valor de n aumenta.

As tabelas 6.11 e 6.12 apresentam os resultados do algoritmo de 4-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings sem Sinais (Algoritmo 11), que usa o conceito de *breakpoints* intergênicos. Apesar do fato de que as instâncias sintéticas da Tabela 6.12 foram criadas usando o dobro de operações em relação às instâncias da Tabela 6.11, a razão entre o tamanho das soluções do experimento da Tabela 6.12 e o tamanho das soluções do experimento da Tabela 6.11 é menor que 1.36 para todas as medidas usadas. Para a Tabela 6.11, a média do fator de aproximação prático está entre 2.00 e 2.03. Já para a Tabela 6.12, os valores dessa coluna estão entre 2.01 e 2.02. Na Tabela 6.11, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.85 e 2.20, respectivamente. Já na Tabela 6.12, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.94 e 2.13, respectivamente. Essas colunas mostram que o fator de aproximação prático é um pouco maior que 2 no pior cenário, enquanto o fator de aproximação teórico é 4.

As tabelas 6.13 e 6.14 apresentam os resultados do algoritmo de 6-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 11), que usa o conceito de *breakpoints* intergênicos. Como o Algoritmo 11 usa apenas reversões e *indels*, implementamos uma modificação desse algoritmo que aplica, sempre que possível, uma transposição de acordo com o Lema 5.2.10, mantendo o fator de aproximação igual a 6. Apesar do fator de aproximação permanecer o mesmo, essa modificação garante que o algoritmo use menos operações quando o genoma de origem possui apenas *strips* crescentes.

A razão entre o tamanho das soluções do experimento da Tabela 6.14 e o tamanho das soluções do experimento da Tabela 6.13 é menor que 1.35 para todas as medidas usadas. Para a Tabela 6.13, a média do fator de aproximação prático está entre 2.99 e 3.06. Já para a Tabela 6.14, os valores dessa coluna estão entre 2.98 e 3.03. Na Tabela 6.13, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.81 e 3.25, respectivamente. Já na Tabela 6.14, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.82 e 3.19, respectivamente. Essas colunas mostram que o fator de aproximação prático é um pouco maior que 3 no pior cenário, enquanto o fator de aproximação teórico é 6.

As tabelas 6.15 e 6.16 apresentam os resultados do algoritmo de 4.5-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 12), que usa *breakpoints* intergênicos, enquanto as tabelas 6.17 e 6.18 apresentam os resultados do algoritmo de 4-aproximação para o mesmo problema (Algoritmo 14), que usa o conceito de grafo de ciclos rotulado e ponderado. Essas tabelas mostram que o algoritmo de 4-aproximação, que usa grafo de ciclos rotulado e ponderado, possui resultados práticos melhores que o algoritmo de 4.5-aproximação, que usa *breakpoints* intergênicos. Agora, discutiremos com mais detalhes os resultados para cada algoritmo.

A razão entre o tamanho das soluções do experimento da Tabela 6.16 e o tamanho das soluções do experimento da Tabela 6.15 é menor que 1.29 para todas as medidas usadas. Para a Tabela 6.15, a média do fator de aproximação prático está entre 3.18 e 3.35. Já para a Tabela 6.16, os valores dessa coluna estão entre 3.13 e 3.25. Na Tabela 6.15, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.88 e 3.50, respectivamente. Já na Tabela 6.16, os valores de mínimo e máximo para o fator de

aproximação prático foram de 2.91 e 3.38, respectivamente. Essas colunas mostram que o fator de aproximação prático foi sempre menor que 3.4, enquanto o fator de aproximação teórico é 4.5.

A razão entre o tamanho das soluções do experimento da Tabela 6.18 e o tamanho das soluções do experimento da Tabela 6.17 é menor que 1.22 para todas as medidas usadas. Para a Tabela 6.17, a média do fator de aproximação prático está entre 2.31 e 2.38. Já para a Tabela 6.18, os valores dessa coluna estão entre 2.66 e 2.72. Na Tabela 6.17, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.00 e 2.69, respectivamente. Já na Tabela 6.18, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.26 e 3.09, respectivamente. Essas colunas mostram que o fator de aproximação prático foi sempre menor que 3.1, enquanto o fator de aproximação teórico é 4.

As tabelas 6.19 e 6.20 apresentam os resultados do algoritmo de 2.5-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais (Algoritmo 13), que usa o conceito de grafo de ciclos rotulado e ponderado. A razão entre o tamanho das soluções do experimento da Tabela 6.20 e o tamanho das soluções do experimento da Tabela 6.19 é menor que 1.46 para todas as medidas usadas. Para a Tabela 6.19, a média do fator de aproximação prático está entre 1.28 e 1.30. Já para a Tabela 6.20, os valores dessa coluna estão entre 1.42 e 1.45. Na Tabela 6.19, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.03 e 1.60, respectivamente. Já na Tabela 6.20, os valores de mínimo e máximo para o fator de aproximação prático foram de 1.19 e 1.71, respectivamente. Essas colunas mostram que o fator de aproximação prático é sempre menor que 1.8, enquanto o fator de aproximação teórico é 2.5.

Por fim, as tabelas 6.21 e 6.22 apresentam os resultados do algoritmo de 4-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais (Algoritmo 15), que usa o conceito de grafo de ciclos ponderado e rotulado. A razão entre o tamanho das soluções do experimento da Tabela 6.22 e o tamanho das soluções do experimento da Tabela 6.21 é menor que 1.38 para todas as medidas usadas. Para a Tabela 6.21, a média do fator de aproximação prático está entre 2.63 e 2.73. Já para a Tabela 6.22, os valores dessa coluna estão entre 2.84 e 2.92. Na Tabela 6.21, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.11 e 3.27, respectivamente. Já na Tabela 6.22, os valores de mínimo e máximo para o fator de aproximação prático foram de 2.27 e 3.35, respectivamente. Essas colunas mostram que o fator de aproximação prático é sempre menor ou igual a 3.4, enquanto o fator de aproximação teórico é 4.

Com esses experimentos, podemos concluir que os algoritmos que usam grafo de ciclos são melhores tanto no fator de aproximação teórico quanto no fator de aproximação prático.

Tabela 6.11: Resultados experimentais do algoritmo de 4-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings sem Sinais (Algoritmo 11), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	42	51.41	61	1.85	2.00	2.20
100	91	103.47	116	1.89	2.02	2.14
150	141	154.62	168	1.93	2.02	2.11
200	191	206.93	225	1.95	2.02	2.10
250	238	257.85	274	1.96	2.02	2.10
300	290	310.60	334	1.97	2.02	2.08
350	329	361.58	386	1.97	2.02	2.09
400	388	413.69	435	1.98	2.02	2.09
450	441	465.22	492	1.97	2.02	2.08
500	488	517.61	545	1.98	2.03	2.08

Tabela 6.12: Resultados experimentais do algoritmo de 4-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings sem Sinais (Algoritmo 11), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	57	65.45	72	1.94	2.01	2.13
100	118	129.47	138	1.97	2.01	2.08
150	183	194.14	206	1.98	2.02	2.09
200	245	258.71	273	1.98	2.02	2.07
250	306	322.95	343	1.99	2.02	2.06
300	367	387.27	403	1.99	2.02	2.06
350	435	451.84	473	1.99	2.02	2.05
400	496	516.31	534	2.00	2.02	2.06
450	561	580.83	601	2.00	2.02	2.05
500	622	645.82	666	2.00	2.02	2.05

Tabela 6.13: Resultados experimentais do algoritmo de 6-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 11), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	43	53.32	63	2.81	2.99	3.25
100	96	107.25	121	2.86	3.03	3.18
150	148	160.38	176	2.90	3.04	3.16
200	194	215.15	235	2.94	3.04	3.17
250	246	268.05	291	2.97	3.05	3.14
300	300	323.33	343	2.98	3.05	3.16
350	345	376.30	399	2.98	3.05	3.13
400	408	431.22	455	2.99	3.05	3.12
450	459	483.79	509	2.99	3.05	3.12
500	507	538.87	570	2.99	3.06	3.13

Tabela 6.14: Resultados experimentais do algoritmo de 6-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 11), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	58	65.72	71	2.82	2.98	3.19
100	120	130.73	140	2.95	3.01	3.14
150	184	195.89	208	2.97	3.02	3.11
200	247	260.96	273	2.97	3.02	3.13
250	312	325.98	341	2.98	3.02	3.13
300	372	391.13	406	2.98	3.03	3.08
350	439	455.98	475	2.99	3.03	3.08
400	501	521.34	539	2.99	3.03	3.10
450	561	586.37	603	3.00	3.03	3.08
500	627	651.29	672	2.99	3.03	3.07

Tabela 6.15: Resultados experimentais do algoritmo de 4.5-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 12), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	46	57.95	67	2.88	3.18	3.50
100	102	118.29	133	3.06	3.26	3.47
150	160	178.31	196	3.11	3.30	3.46
200	220	239.26	259	3.17	3.31	3.44
250	280	300.18	326	3.20	3.33	3.48
300	332	361.69	391	3.23	3.33	3.46
350	395	422.07	446	3.22	3.34	3.45
400	456	483.99	507	3.26	3.35	3.43
450	516	544.27	569	3.26	3.35	3.46
500	570	606.44	637	3.26	3.35	3.45

Tabela 6.16: Resultados experimentais do algoritmo de 4.5-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 12), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	59	69.30	77	2.91	3.13	3.38
100	129	139.10	151	3.00	3.19	3.37
150	196	209.36	220	3.07	3.21	3.36
200	266	279.44	293	3.10	3.22	3.38
250	333	349.88	367	3.11	3.23	3.33
300	401	420.03	436	3.14	3.23	3.32
350	470	490.50	509	3.15	3.24	3.32
400	543	560.49	581	3.17	3.24	3.32
450	607	631.33	655	3.16	3.24	3.33
500	680	701.51	728	3.17	3.25	3.32

Tabela 6.17: Resultados experimentais do algoritmo de 4-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 14), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	32	40.23	48	2.00	2.31	2.69
100	70	81.44	94	2.11	2.36	2.69
150	107	122.32	140	2.17	2.36	2.59
200	148	163.50	180	2.16	2.37	2.55
250	187	205.35	226	2.22	2.37	2.55
300	226	246.71	269	2.24	2.37	2.55
350	264	287.71	318	2.23	2.37	2.55
400	305	330.16	352	2.26	2.38	2.51
450	347	370.77	395	2.23	2.38	2.51
500	386	412.84	440	2.24	2.38	2.51

Tabela 6.18: Resultados experimentais do algoritmo de 4-aproximação para o problema da Distância de Transposições e Indels Intergênicos em Strings sem Sinais (Algoritmo 14), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	39	46.98	54	2.26	2.66	3.09
100	84	93.74	102	2.41	2.70	2.98
150	130	140.80	155	2.46	2.71	2.99
200	176	187.63	199	2.52	2.72	2.94
250	217	234.31	247	2.53	2.72	2.93
300	266	281.43	300	2.54	2.72	2.91
350	312	328.17	347	2.53	2.72	2.89
400	357	374.88	399	2.57	2.72	2.90
450	399	421.64	445	2.57	2.72	2.89
500	450	468.51	486	2.60	2.72	2.89

Tabela 6.19: Resultados experimentais do algoritmo de 2.5-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais (Algoritmo 13), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	31	40.52	51	1.03	1.28	1.60
100	69	81.77	98	1.11	1.29	1.50
150	105	122.56	141	1.14	1.30	1.47
200	145	164.21	185	1.17	1.30	1.43
250	180	205.22	227	1.17	1.30	1.43
300	224	247.02	271	1.20	1.30	1.42
350	254	288.14	311	1.22	1.30	1.44
400	305	329.81	360	1.20	1.30	1.43
450	340	370.39	401	1.22	1.30	1.38
500	385	411.94	446	1.21	1.30	1.38

Tabela 6.20: Resultados experimentais do algoritmo de 2.5-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais (Algoritmo 13), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	45	55.57	67	1.19	1.42	1.71
100	95	110.99	126	1.26	1.43	1.67
150	150	167.06	184	1.29	1.44	1.60
200	199	222.75	244	1.30	1.44	1.59
250	255	278.39	299	1.33	1.44	1.58
300	311	334.73	359	1.33	1.45	1.56
350	361	390.63	415	1.36	1.45	1.55
400	418	446.35	484	1.36	1.45	1.55
450	475	502.24	537	1.38	1.45	1.55
500	529	558.20	588	1.36	1.45	1.53

Tabela 6.21: Resultados experimentais do algoritmo de 4-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais (Algoritmo 15), considerando instâncias criadas com $k = \frac{n}{2}$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	35	46.05	58	2.11	2.63	3.27
100	77	93.57	113	2.33	2.67	3.02
150	121	141.41	166	2.41	2.70	3.01
200	165	189.05	213	2.43	2.71	3.03
250	212	236.93	263	2.47	2.71	2.94
300	259	285.49	313	2.51	2.72	2.93
350	302	333.40	365	2.55	2.72	2.92
400	349	381.91	425	2.52	2.73	2.93
450	393	431.13	465	2.54	2.73	2.91
500	437	478.93	520	2.58	2.73	2.92

Tabela 6.22: Resultados experimentais do algoritmo de 4-aproximação para o problema da Distância de Reversões, Transposições e Indels Intergênicos em Strings com Sinais (Algoritmo 15), considerando instâncias criadas com $k = n$.

	Operações			Aproximação		
Tamanho	Mínimo	Média	Máximo	Mínimo	Média	Máximo
50	48	57.01	69	2.27	2.84	3.35
100	99	114.47	127	2.55	2.88	3.29
150	156	172.17	191	2.59	2.90	3.21
200	212	230.07	250	2.66	2.90	3.18
250	268	288.32	309	2.67	2.91	3.09
300	320	346.40	371	2.69	2.91	3.12
350	381	404.21	431	2.64	2.91	3.10
400	435	462.18	494	2.76	2.92	3.15
450	495	520.33	554	2.77	2.92	3.08
500	542	579.01	618	2.76	2.92	3.12

6.4 Experimentos com Genomas Reais

Nesta seção, evidenciamos a aplicabilidade de um dos nossos algoritmos usando genomas reais de cianobactérias da base de dados Cyanorak 2.1 [48]. Como as reversões são os eventos mais comuns em cianobactérias [40, 55] e a base de dados utilizada possui informações sobre a orientação dos genes, utilizamos o algoritmo de 2.5-aproximação para o problema da Distância de Reversões e Indels Intergênicos em Strings com Sinais (Algoritmo 13) e o

algoritmo polinomial para a Ordenação de Permutações por Reversões [51] (Algoritmo HP).

A base de dados Cyanorak 2.1 possui 94 genomas. O número mínimo e o número máximo de genes nesses genomas são 1834 e 4391, respectivamente. Em média, cada genoma possui um pouco mais de 95% de genes únicos. Para ajustar os dados à entrada do Algoritmo 13, realizamos o seguinte pré-processamento em cada par de genomas para construir uma instância intergênica:

1. Mantemos apenas a primeira ocorrência de genes repetidos em cada genoma, marcando as outras cópias como genes a serem inseridos ou removidos;

2. Mapeamos o genoma de destino em $\mathcal{G}_d = (\iota^n, \breve{\iota}^n)$: neste passo, cada sequência contígua de genes que está presente apenas no genoma de destino é representada por um único elemento na string ι^n. Após isso, para as extremidades e para cada par de elementos consecutivos em ι^n, computamos o tamanho das regiões intergênicas e construímos $\breve{\iota}^n$;

3. Mapeamos o genoma de origem em $\mathcal{G}_o = (A, \breve{A})$: neste passo, cada sequência contígua de genes que está presente apenas no genoma de origem é representada por um único elemento α na string A. Após isso, para as extremidades e para cada par de elementos consecutivos em A, computamos o tamanho das regiões intergênicas e construímos $\breve{A}$.

Como o Algoritmo HP [51] não considera regiões intergênicas e só pode ser aplicado em genomas balanceados, para cada par de genoma, aplicamos o seguinte pré-processamento para construir uma instância:

1. Mantemos apenas a primeira ocorrência de genes repetidos em cada genoma e marcamos as outras cópias como genes a serem inseridos ou removidos;

2. Marcamos como α qualquer sequência contígua de genes presente apenas no genoma de origem ou no genoma de destino;

3. Removemos cada elemento α nos genomas de origem e destino, simulando *indels*;

4. Mapeamos a sequência de genes do genoma de destino em ι^n e a sequência de genes do genoma de origem em π.

Para cada par de genomas, o tamanho da solução final é igual à soma do número de *indels* simulados no passo 3 e da distância $d_\rho(\pi)$, encontrada pelo Algoritmo HP [51].

Para cada um desses dois algoritmos, construímos uma matriz de distâncias usando todos os pares de genomas da base de dados Cyanorak 2.1 [48]. Cada uma dessas matrizes é usada para a criação de árvores filogenéticas. Após isso, comparamos cada uma das árvores criadas com a árvore filogenética apresentada por Laurence e coautores [48]. Usamos como métrica o número de folhas da subárvore de concordância máxima (MAST) [41], que é construída com um par de árvores. Nessa métrica, quanto maior o número de folhas da

Tabela 6.23: Resultados da comparação entre a árvore filogenética apresentada por Laurence e coautores [48] e as árvores filogenéticas construídas com as matrizes de distâncias obtidas pelo Algoritmo 13 e pelo Algoritmo `HP`. Foram usados três diferentes métodos de reconstrução para a criação das árvores filogenéticas. A métrica usada é a quantidade de folhas na subárvore de concordância máxima (MAST).

Método de Reconstrução	Algoritmo	MAST
Neighbor Joining [72]	Algoritmo `HP`	53
	Algoritmo 13	56
Unweighted Neighbor Joining [49]	Algoritmo `HP`	52
	Algoritmo 13	56
Circular Order Reconstruction [58]	Algoritmo `HP`	55
	Algoritmo 13	57

MAST, maior o nível de congruência topológica entre as duas árvores comparadas. A Tabela 6.23 mostra os resultados obtidos ao comparar essas árvores filogenéticas.

Com os resultados da Tabela 6.23, concluímos que, para todos os métodos de reconstrução usados, as árvores filogenéticas criadas com os resultados do Algoritmo 13 obtiveram maior nível de congruência topológica com a árvore filogenética apresentada por Laurence e coautores [48]. A Figura 6.1 mostra a representação visual da árvore filogenética criada a partir dos resultados do Algoritmo 13 e do método de reconstrução *Circular Order Reconstruction* [58], que foi a configuração que obteve maior valor de MAST. Podemos observar que a árvore filogenética da Figura 6.1 apresenta boa separação das espécies e dos grupos de organismos.

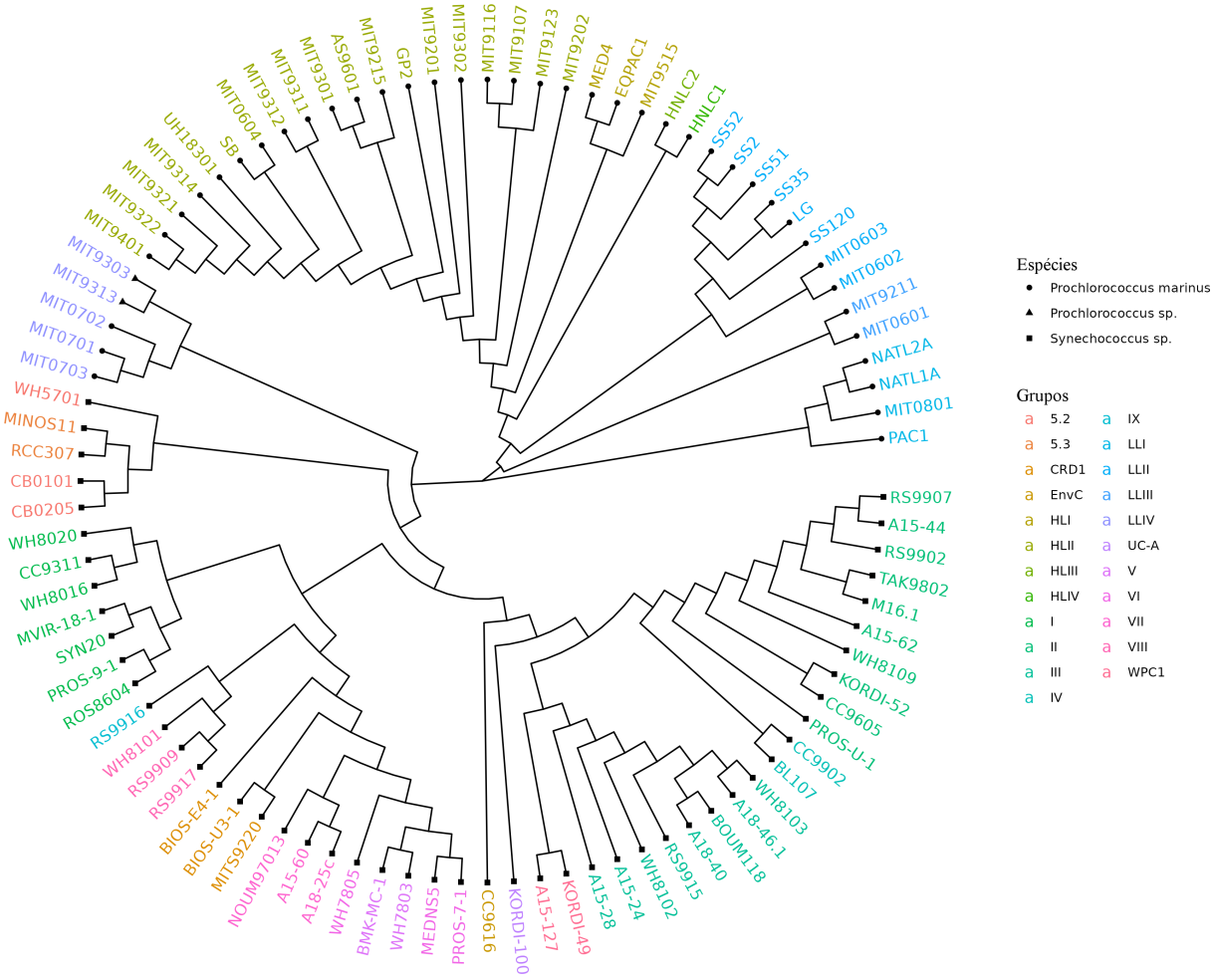


Figura 6.1: Árvore filogenética baseada em rearranjos de genomas criada a partir do Algoritmo 13 e do método de reconstrução *Circular Order Reconstruction* [58] usando genomas da base de dados Cyanorak 2.1 [48]. Utilizamos o pacote `treeio` da linguagem R [81] para a construção desta imagem.

Capítulo 7

Considerações Finais

Esta tese apresentou os resultados mais importantes obtidos durante o período de doutorado. Para todos os problemas considerados nesta tese, foi realizado um estudo extensivo com foco na prova da complexidade desses problemas e no desenvolvimento de algoritmos de aproximação.

Iniciamos esta tese tratando de problemas de Distância de Rearranjos em genomas balanceados, mais conhecidos como problemas de Ordenação de Permutações por Rearranjos. Apresentamos uma nova versão de um dos algoritmos mais conhecidos na literatura da área, a 1.375-aproximação de Elias e Hartman [43], que corrige um problema do algoritmo original com uma complexidade de tempo menor que a de outros algoritmos propostos para correção desse mesmo problema. Além disso, provamos que a Ordenação de Permutações por Rearranjos é NP-difícil para seis modelos de rearranjos que incluem transposições junto com a combinação de reversões, transposições inversas e revrevs, sendo que alguns desses modelos são estudados desde o século passado, mas não possuíam complexidade conhecida.

A cada capítulo, incorporamos mais características genômicas aos problemas com o objetivo de torná-los mais relevantes do ponto de vista biológico. Estudamos a Distância de Rearranjos e a Distância de Rearranjos Intergênicos em genomas desbalanceados, adaptando e criando novos conceitos e estruturas que possibilitaram a criação de algoritmos de aproximação. Também provamos que a maioria dos problemas investigados são NP-Difíceis.

Por fim, apresentamos experimentos em genomas sintéticos e em genomas reais. Os experimentos em genomas sintéticos mostram que o fator de aproximação prático de alguns dos algoritmos foi muito menor que o fator de aproximação teórico. Já no experimento com dados reais, usamos genomas de cianobactérias da base de dados Cyanorak 2.1 [48]. Nesse experimento, criamos árvores filogenéticas usando o nosso algoritmo para a Distância de Reversões e Indels Intergênicos e o algoritmo polinomial exato para a Ordenação de Reversões, criado por Hannenhalli e Pevzner [51]. Quando comparadas com a árvore filogenética criada pelos autores que publicaram a base de dados Cyanorak 2.1, a árvore filogenética criada com o nosso algoritmo obteve um nível de congruência topológica maior do que a árvore filogenética criada com o algoritmo de Hannenhalli e Pevzner [51].

Todos os resultados desta tese foram publicados em congressos e revistas internacionais, nos seguintes artigos:

- "A 1.375-Approximation Algorithm for Sorting by Transpositions with Faster Running Time", apresentado na conferência *Brazilian Symposium on Bioinformatics* (BSB) em 2022 [2] (Capítulo 3, Seção 3.1);
- "On the Complexity of Some Variations of Sorting by Transpositions", publicado na revista *Journal of Universal Computer Science* em 2020 [7] (Capítulo 3, Seção 3.2);
- "Genome Rearrangement Distance with Reversals, Transpositions, and Indels", publicado na revista *Journal of Computational Biology* em 2021 [8] (Capítulo 4, Seção 4.2);
- "Labeled Cycle Graph for Transposition and Indel Distance", publicado na revista *Journal of Computational Biology* em 2022 [10] (Capítulo 4, Seção 4.3);
- "Block Interchange and Reversal Distance on Unbalanced Genomes", apresentado na conferência *Brazilian Symposium on Bioinformatics* (BSB) em 2023 [13] (Capítulo 4, Seção 4.3);
- "Incorporating Intergenic Regions into Reversal and Transposition Distances with Indels", apresentado na conferência RECOMB *Comparative Genomics* em 2021. Uma versão estendida foi publicada na revista *Journal of Bioinformatics and Computational Biology* em 2021 [9] (Capítulo 5, Seção 5.2);
- "Reversal Distance on Genomes with Different Gene Content and Intergenic Regions Information", apresentado na conferência *Algorithms for Computational Biology* (AlCoB) em 2021 [1] (Capítulo 5, Seção 5.3.1);
- "Reversal and Indel Distance with Intergenic Region Information", publicado na revista *IEEE/ACM Transactions on Computational Biology and Bioinformatics* em 2023 [3] (Capítulo 5, Seção 5.3.1);
- "Transposition Distance Considering Intergenic Regions for Unbalanced Genomes", apresentado na conferência *International Symposium on Bioinformatics Research and Applications* (ISBRA) em 2022 [11] (Capítulo 5, Seção 5.3.2);
- "Reversal and Transposition Distance on Unbalanced Genomes Using Intergenic Information", publicado na revista *Journal of Computational Biology* em 2023 [12] (Capítulo 5, Seção 5.3.2).
- "Rearrangement Distance Problems: An updated survey", aceito para publicação na revista *ACM Computing Surveys* em 2024 [63] (Revisão bibliográfica).

Além dos artigos acima, que são diretamente relacionados a resultados específicos desta tese, outras contribuições na área de rearranjos de genomas foram publicadas durante o período de doutorado nos seguintes artigos:

- "Approximation Algorithms for Sorting Permutations by Length-Weighted Short Rearrangements", apresentado na conferência *Latin and American Algorithms, Graphs and Optimization Symposium* (LAGOS) em 2019 [5];

- "Sorting Permutations by Fragmentation-Weighted Operations", publicado na revista *Journal of Bioinformatics and Computational Biology* em 2020 [4];
- "Length-Weighted λ-Rearrangement Distance", publicado na revista *Journal of Combinatorial Optimization* em 2020 [6];
- "Heuristics for Breakpoint Graph Decomposition with Applications in Genome Rearrangement Problems", apresentado na conferência *Brazilian Symposium on Bioinformatics* (BSB) em 2020 [69];
- "Sorting by Reversals and Transpositions with Proportion Restriction", apresentado na conferência *Brazilian Symposium on Bioinformatics* (BSB) em 2020 [24];
- "Reversals Distance Considering Flexible Intergenic Regions Sizes", apresentado na conferência *Algorithms for Computational Biology* (AlCoB) em 2021 [26];
- "Approximation Algorithms for Sorting λ-Permutations by λ-Operations", publicado na revista *Algorithms* em 2021 [60];
- "Reversals and Transpositions Distance with Proportion Restriction", publicado na revista *Journal of Bioinformatics and Computational Biology* em 2021 [25];
- "Approximation Algorithm for Rearrangement Distances Considering Repeated Genes and Intergenic Regions", publicado na revista *Algorithms for Molecular Biology* em 2021 [77];
- "Algorithms for the Maximum Eulerian Cycle Decomposition Problem", apresentado na conferência Simpósio Brasileiro de Pesquisa Operacional (SBPO) em 2021 [70];
- "Heuristics for Cycle Packing of Adjacency Graphs for Genomes with Repeated Genes", apresentado na conferência *Brazilian Symposium on Bioinformatics* (BSB) em 2021 [78];
- "Reversal and Transposition Distance of Genomes Considering Flexible Intergenic Regions", apresentado na conferência *Latin and American Algorithms, Graphs and Optimization Symposium* (LAGOS) em 2021 [31];
- "An Improved Approximation Algorithm for the Reversal and Transposition Distance Considering Gene Order and Intergenic Sizes", publicado na revista *Algorithms for Molecular Biology* em 2021 [30];
- "Signed Rearrangement Distances Considering Repeated Genes and Intergenic Regions", apresentado na conferência *Bioinformatics and Computational Biology* (BICoB) em 2022 [75];
- "A New Approach for the Reversal Distance with Indels and Moves in Intergenic Regions", apresentado na conferência RECOMB *Comparative Genomics* em 2022 [32];
- "Sorting by k-Cuts on Signed Permutations", apresentado na conferência RECOMB *Comparative Genomics* em 2022 [61];

- "Genome Rearrangement Distance with a Flexible Intergenic Regions Aspect", publicado na revista *IEEE/ACM Transactions on Computational Biology and Bioinformatics* em 2023 [27];
- "Approximation Algorithms for Sorting by k-Cuts on Signed Permutations", publicado na revista *Journal of Combinatorial Optimization* em 2023 [62];
- "Rearrangement Distance with Reversals, Indels, and Moves in Intergenic Regions on Signed and Unsigned Permutations", publicado na revista *Journal of Bioinformatics and Computational Biology* em 2023 [33];
- "Signed Rearrangement Distances Considering Repeated Genes, Intergenic Regions, and Indels", publicado na revista *Journal of Combinatorial Optimization* em 2023 [76];
- "Approximating Rearrangement Distances with Replicas and Flexible Intergenic Regions", apresentado na conferência *International Symposium on Bioinformatics Research and Applications* (ISBRA) em 2023 [79];
- "Maximum Alternating Balanced Cycle Decomposition and Applications in Sorting by Intergenic Operations Problems", aceito para apresentação na conferência RECOMB *Comparative Genomics* em 2024 [28].

O primeiro trabalho futuro importante deixado por esta tese é a determinação da complexidade dos problemas de Ordenação de Permutações por Transposições e Outros Rearranjos quando $w_\tau/w_\rho > 1.5$, onde w_ρ é o custo de reversões e w_τ é o custo de transposições e rearranjos similares.

Para os problemas de Distância de Rearranjos e Distância de Rearranjos Intergênicos em genomas desbalanceados, consideramos apenas a abordagem não ponderada nesta tese. Assim, um trabalho futuro interessante é o estudo desses problemas considerando uma abordagem ponderada. Além disso, os seguintes problemas continuam com complexidade em aberto: a Distância de Block Interchanges e Indels em Strings sem Sinais; a Distância de Block Interchanges, Reversões e Indels em Strings com Sinais; e a Distância de Reversões e Indels Intergênicos em Strings com Sinais.

Referências Bibliográficas

[1] Alexsandro Oliveira Alexandrino, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversal Distance on Genomes with Different Gene Content and Intergenic Regions Information. In *Proceedings of the 8th International Conference on Algorithms for Computational Biology (AlCoB'2021)*, volume 12715, pages 121–133. Springer International Publishing, 2021.

[2] Alexsandro Oliveira Alexandrino, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. A 1.375-Approximation Algorithm for Sorting by Transpositions with Faster Running Time. In *Proceedings of the 15th Brazilian Symposium on Bioinformatics (BSB'2022)*, volume 13523 of *Lecture Notes in Computer Science*, pages 147–157. Springer Nature Switzerland, 2022.

[3] Alexsandro Oliveira Alexandrino, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversal and Indel Distance with Intergenic Region Information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(3):1628–1640, 2023.

[4] Alexsandro Oliveira Alexandrino, Carla Negri Lintzmayer, and Zanoni Dias. Sorting Permutations by Fragmentation-Weighted Operations. *Journal of Bioinformatics and Computational Biology*, 18(2):2050006.1–2050006.31, 2020.

[5] Alexsandro Oliveira Alexandrino, Guilherme Henrique Santos Miranda, Carla Negri Lintzmayer, and Zanoni Dias. Approximation Algorithms for Sorting Permutations by Length-Weighted Short Rearrangements. *Electronic Notes in Theoretical Computer Science*, 346:29–40, 2019.

[6] Alexsandro Oliveira Alexandrino, Guilherme Henrique Santos Miranda, Carla Negri Lintzmayer, and Zanoni Dias. Length-weighted λ-rearrangement Distance. *Journal of Combinatorial Optimization*, pages 1–24, 2020.

[7] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. On the Complexity of Some Variations of Sorting by Transpositions. *Journal of Universal Computer Science*, 26(9):1076–1094, 2020.

[8] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Genome Rearrangement Distance with Reversals, Transpositions, and Indels. *Journal of Computational Biology*, 28(3):235–247, 2021.

[9] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Incorporating Intergenic Regions into Reversal and Transposition Distances with Indels. *Journal of Bioinformatics and Computational Biology*, 19(06):2140011, 2021.

[10] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Labeled Cycle Graph for Transposition and Indel Distance. *Journal of Computational Biology*, 29(03):243–256, 2022.

[11] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Transposition Distance Considering Intergenic Regions for Unbalanced Genomes. In *Proceedings of the 18th International Symposium on Bioinformatics Research and Applications (ISBRA'2022)*, volume 13760, pages 100–113. Springer Nature Switzerland, 2022.

[12] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Reversal and Transposition Distance on Unbalanced Genomes Using Intergenic Information. *Journal of Computational Biology*, 30(8):861–876, 2023.

[13] Alexsandro Oliveira Alexandrino, Gabriel Siqueira, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Block Interchange and Reversal Distance on Unbalanced Genomes. In *Proceedings of the 16th Brazilian Symposium on Bioinformatics (BSB'2023)*, volume 13954 of *Lecture Notes in Computer Science*, pages 1–13. Springer Nature Switzerland, 2023.

[14] Martin Bader, Mohamed I. Abouelhoda, and Enno Ohlebusch. A Fast Algorithm for the Multiple Genome Rearrangement Problem with Weighted Reversals and Transpositions. *BMC Bioinformatics*, 9(1):1–13, 2008.

[15] Vineet Bafna and Pavel A. Pevzner. Sorting Permutations by Transpositions. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'1995)*, pages 614–623, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

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

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

[18] Anne Bergeron, Julia Mixtacki, and Jens Stoye. A Unifying View of Genome Rearrangements. In *Proceedings of the 6th Workshop on Algorithms in Bioinformatics (WABI'2006)*, pages 163–173, Heidelberg, Germany, 2006. Springer International Publishing.

[19] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In *Proceedings of the 10th Annual European*

Symposium on Algorithms (ESA'2002), volume 2461 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany, 2002.

[20] Priscila Biller, Laurent Guéguen, Carole Knibbe, and Eric Tannier. Breaking Good: Accounting for Fragility of Genomic Regions in Rearrangement Distance Estimation. *Genome Biology and Evolution*, 8(5):1427–1439, 2016.

[21] Priscila Biller, Carole Knibbe, Guillaume Beslon, and Eric Tannier. Comparative Genomics on Artificial Life. In *Pursuit of the Universal*, pages 35–44. Springer International Publishing, 2016.

[22] Mathieu Blanchette, Takashi Kunisawa, and David Sankoff. Parametric Genome Rearrangement. *Gene*, 172(1):GC11–GC17, 1996.

[23] Marília D. V. Braga, Eyla Willing, and Jens Stoye. Double Cut and Join with Insertions and Deletions. *Journal of Computational Biology*, 18(9):1167–1184, 2011.

[24] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Sorting by Reversals and Transpositions with Proportion Restriction. In *Proceedings of the 13th Brazilian Symposium on Bioinformatics (BSB'2020)*, pages 117–128. Springer International Publishing, 2020.

[25] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversals and Transpositions Distance with Proportion Restriction. *Journal of Bioinformatics and Computational Biology*, 19(04):2150013, 2021.

[26] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversals Distance Considering Flexible Intergenic Regions Sizes. In *Proceedings of the 8th International Conference on Algorithms for Computational Biology (AlCoB'2021)*, pages 134–145. Springer International Publishing, 2021.

[27] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Genome Rearrangement Distance with a Flexible Intergenic Regions Aspect. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(03):1641–1653, 2023.

[28] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Gabriel Siqueira, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Maximum Alternating Balanced Cycle Decomposition and Applications in Sorting by Intergenic Operations Problems. In *Proceedings of the 21st Annual Satellite Conference of RECOMB on Comparative Genomics (RECOMB-CG 2024)*, volume 14616, pages 153–172. Springer International Publishing, 2024.

[29] Klairton Lima Brito, Géraldine Jean, Guillaume Fertin, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Sorting by Genome Rearrangements on both Gene Order and Intergenic Sizes. *Journal of Computational Biology*, 27(2):156–174, 2020.

[30] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. An Improved Approximation Algorithm for the Reversal and Transposition Distance Considering Gene Order and Intergenic Sizes. *Algorithms for Molecular Biology*, 16(1):1–21, 2021.

[31] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. Reversal and Transposition Distance of Genomes Considering Flexible Intergenic Regions. In *Proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS'2021)*, pages 21–29. Procedia Computer Science, Elsevier, 2021.

[32] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. A New Approach for the Reversal Distance with Indels and Moves in Intergenic Regions. In *Proceedings of 19th Annual Satellite Conference of RECOMB on Comparative Genomics (RECOMB-CG 2022)*, volume 13234, pages 205–220. Springer International Publishing, 2022.

[33] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. Rearrangement Distance with Reversals, Indels, and Moves in Intergenic Regions on Signed and Unsigned Permutations. *Journal of Bioinformatics and Computational Biology*, 21(02):2350009, 2023.

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

[35] Laurent Bulteau, Guillaume Fertin, and Eric Tannier. Genome Rearrangements with Indels in Intergenes Restrict the Scenario Space. *BMC Bioinformatics*, 17(14):426, 2016.

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

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

[38] David A. Christie. Sorting Permutations by Block-Interchanges. *Information Processing Letters*, 60(4):165–169, 1996.

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

[40] Daniel A. Dalevi, Niklas Eriksen, Kimmo Eriksson, and Siv G. E. Andersson. Measuring Genome Divergence in Bacteria: A Case Study Using Chlamydian Data. *Journal of Molecular Evolution*, 55(1):24–36, 2002.

[41] Damien M de Vienne, Tatiana Giraud, and Olivier C Martin. A Congruence Index for Testing Topological Similarity Between Trees. *Bioinformatics*, 23(23):3119–3124, 2007.

[42] Nadia El-Mabrouk. Genome Rearrangement by Reversals and Insertions/Deletions of Contiguous Segments. In *Annual Symposium on Combinatorial Pattern Matching (CPM'2000)*, pages 222–234. Springer, 2000.

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

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

[45] Guillaume Fertin, Géraldine Jean, and Eric Tannier. Algorithms for Computing the Double Cut and Join Distance on both Gene Order and Intergenic Sizes. *Algorithms for Molecular Biology*, 12(1):16, 2017.

[46] Guillaume Fertin, Anthony Labarre, Irena Rusu, Éric Tannier, and Stéphane Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. The MIT Press, London, England, 2009.

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

[48] Laurence Garczarek, Ulysse Guyet, Hugo Doré, Gregory K Farrant, Mark Hoebeke, Loraine Brillet-Guéguen, Antoine Bisch, Mathilde Ferrieux, Jukka Siltanen, Erwan Corre, Gildas Le Corguillé, Morgane Ratin, Frances D Pitt, Martin Ostrowski, Maël Conan, Anne Siegel, Karine Labadie, Jean-Marc Aury, Patrick Wincker, David J Scanlan, and Frédéric Partensky. Cyanorak v2. 1: a scalable information system dedicated to the visualization and expert curation of marine and brackish picocyanobacteria genomes. *Nucleic Acids Research*, 49(D1):D667–D676, 2021.

[49] Olivier Gascuel. Concerning the NJ Algorithm and its Unweighted Version, UNJ. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science in Mathematical Hierarchies and Biology*, pages 149–171. Amer, 1997.

[50] Qian-Ping Gu, Shietung Peng, and Ivan H. Sudborough. A 2-Approximation Algorithm for Genome Rearrangements by Reversals and Transpositions. *Theoretical Computer Science*, 210(2):327–339, 1999.

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

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

[53] Tom Hartmann, Nicolas Wieseke, Roded Sharan, Martin Middendorf, and Matthias Bernt. Genome rearrangement with ILP. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(5):1585–1593, 2017.

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

[55] Jean-François Lefebvre, Nadia El-Mabrouk, Elisabeth R. M. Tillier, and David Sankoff. Detection and validation of single gene inversions. *Bioinformatics*, 19(1):i190–i196, 2003.

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

[57] Xiaowen Lou and Daming Zhu. A 2.25-Approximation Algorithm for Cut-and-Paste Sorting of Unsigned Circular Permutations. In *Computing and Combinatorics*, volume 5092, pages 331–341, Heidelberg, Germany, 2008. Springer International Publishing.

[58] V Makarenkov and B Leclerc. Tree Metrics and Their Circular Orders: Some Uses for the Reconstruction and Fitting of Phylogenetic Trees. *Mathematical Hierarchies and Biology, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 37:183–208, 1997.

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

[60] Guilherme Henrique Santos Miranda, Alexsandro Oliveira Alexandrino, Carla Negri Lintzmayer, and Zanoni Dias. Approximation Algorithms for Sorting λ-Permutations by λ-Operations. *Algorithms*, 14(6):175, 2021.

[61] Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Géraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Sorting by k-Cuts on Signed Permutations. In *Proceedings of 19th Annual Satellite Conference of RECOMB on Comparative Genomics (RECOMB-CG 2022)*, volume 13234, pages 189–204. Springer International Publishing, 2022.

[62] Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Géraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Approximation Algorithms for Sorting by k-Cuts on Signed Permutations. *Journal of Combinatorial Optimization*, 45(1):6, 2023.

[63] Andre Rodrigues Oliveira, Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Gabriel Siqueira, Ulisses Dias, and Zanoni Dias. Rearrangement Distance Problems: An updated survey. *ACM Computing Surveys*, 2024 (in press).

[64] Andre Rodrigues Oliveira, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. On the Complexity of Sorting by Reversals and Transpositions Problems. *Journal of Computational Biology*, 26:1223–1229, 2019.

[65] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Klairton Lima Brito, Laurent Bulteau, Ulisses Dias, and Zanoni Dias. Sorting Signed Permutations by Intergenic Reversals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2870–2876, 2021.

[66] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. A 3.5-Approximation Algorithm for Sorting by Intergenic Transpositions. In *Proceedings of the 7th International Conference on Algorithms for Computational Biology (AlCoB'2020)*, pages 16–28. Springer International Publishing, 2020.

[67] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. Sorting Permutations by Intergenic Operations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2080–2093, 2021.

[68] Andre Rodrigues Oliveira, Geraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Super Short Reversals on Both Gene Order and Intergenic Sizes. In *Proceedings of the 11th Brazilian Symposium on Bioinformatics (BSB'2018)*, pages 14–25. Springer International Publishing, Heidelberg, Germany, 2018.

[69] Pedro Olímpio Pinheiro, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Cid Carvalho de Souza, and Zanoni Dias. Heuristics for Breakpoint Graph Decomposition with Applications in Genome Rearrangement Problems. In *Proceedings of the 13th Brazilian Symposium on Bioinformatics (BSB'2020)*, pages 129–140. Springer International Publishing, 2020.

[70] Pedro Olímpio Pinheiro, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Cid Carvalho de Souza, and Zanoni Dias. Algorithms for the Maximum Eulerian Cycle Decomposition Problem. In *Anais do LIII Simpósio Brasileiro de Pesquisa Operacional (SBPO'2021)*, volume 53, page 139228, 2021.

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

[72] Naruya Saitou and Masatoshi Nei. The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[73] David Sankoff. Genome Rearrangement with Gene Families. *Bioinformatics*, 15(11):909–917, 1999.

[74] Luiz Augusto G. Silva, Luis Antonio B. Kowada, Noraí Romeu Rocco, and Maria E. M. T. Walter. A New 1.375-Approximation Algorithm for Sorting by Transpositions. *Algorithms for Molecular Biology*, 17(1):1–17, 2022.

[75] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, and Zanoni Dias. Signed Rearrangement Distances Considering Repeated Genes and Intergenic Regions. In *Proceedings of 14th International Conference on Bioinformatics and Computational Biology (BICoB'2022)*, volume 83, pages 31–42. EasyChair, 2022.

[76] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, and Zanoni Dias. Signed Rearrangement Distances Considering Repeated Genes, Intergenic Regions, and Indels. *Journal of Combinatorial Optimization*, 46(2):16, 2023.

[77] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, and Zanoni Dias. Approximation Algorithm for Rearrangement Distances Considering Repeated Genes and Intergenic Regions. *Algorithms for Molecular Biology*, 16(1):1–23, 2021.

[78] Gabriel Siqueira, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, and Zanoni Dias. Heuristics for Cycle Packing of Adjacency Graphs for Genomes with Repeated Genes. In *Proceedings of the 14th Brazilian Symposium on Bioinformatics (BSB'2021)*, pages 93–105. Springer International Publishing, 2021.

[79] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, and Zanoni Dias. Approximating Rearrangement Distances with Replicas and Flexible Intergenic Regions. In *Proceedings of the 19th International Symposium on Bioinformatics Research and Applications (ISBRA'2023)*, volume 14248, pages 241–254. Springer, 2023.

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

[81] Li-Gen Wang, Tommy Tsan-Yuk Lam, Shuangbin Xu, Zehan Dai, Lang Zhou, Tingze Feng, Pingfan Guo, Casey W Dunn, Bradley R Jones, Tyler Bradley, et al. treeio: An R Package for Phylogenetic Tree Input and Output with Richly Annotated and Associated Data. *Molecular Biology and Evolution*, 37(2):599–603, 2020.

[82] Eyla Willing, Jens Stoye, and Marília D. V. Braga. Computing the Inversion-Indel Distance. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2314–2326, 2021.

[83] Eyla Willing, Simone Zaccaria, Marília D. V. Braga, and Jens Stoye. On the Inversion-Indel Distance. *BMC Bioinformatics*, 14:S3, 2013.

[84] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient Sorting of Genomic Permutations by Translocation, Inversion and Block Interchange. *Bioinformatics*, 21(16):3340–3346, 2005.