Universidade Estadual de Campinas
Instituto de Computação

André Rodrigues Oliveira

Modelos Restritos e Intergênicos para a Ordenação por Reversões e Transposições

CAMPINAS
2019

André Rodrigues Oliveira

Modelos Restritos e Intergênicos para a Ordenação por Reversões e Transposições

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 André Rodrigues Oliveira e orientada pelo Prof. Dr. Zanoni Dias.

CAMPINAS
2019

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Márcia Pillon D'Aloia - CRB 8/5180

OL4m

Oliveira, Andre Rodrigues, 1990-
Modelos restritos e intergênicos para a ordenação por reversões e transposições / Andre Rodrigues Oliveira. – Campinas, SP : [s.n.], 2019.

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 para Biblioteca Digital

Título em outro idioma: Constrained and intergenic models for the sorting by reversals and transpositions
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]
Carla Negri Lintzmayer
Maria Emília Machado Telles Walter
Orlando Lee
Guilherme Pimentel Telles
Data de defesa: 09-12-2019
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-0568-1859
- Currículo Lattes do autor: http://lattes.cnpq.br/9814400643053681

Universidade Estadual de Campinas
Instituto de Computação

André Rodrigues Oliveira

Modelos Restritos e Intergênicos para a Ordenação por Reversões e Transposições

Banca Examinadora:

- Prof. Dr. Zanoni Dias
  Universidade Estadual de Campinas
- Profa. Dra. Carla Negri Lintzmayer
  Universidade Federal do ABC
- 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, 09 de dezembro de 2019

Agradecimentos

Esta tese completa um capítulo muito longo da minha vida dentro da Universidade Estadual de Campinas, um período de muito aprendizado e autoconhecimento. Agradeço primeiramente aos meus pais, João e Edilamar, pelo privilégio de poder me dedicar inteiramente aos estudos em uma cidade longe de casa. Ainda no contexto familiar, agradeço também à minha irmã Thais e ao Lê pelo amor e apoio incondicional.

Durante os primeiros anos de graduação, tive a sorte de, por um semestre, frequentar as aulas da Chris. Além de sua excelente didática, foi graças a sua confiança em mim depositada que eu comecei a engatinhar neste profundo mundo da pesquisa científica.

Após a graduação, tive então o prazer de ser orientado pelo Zanoni durante meu mestrado e doutorado. Nesta caminhada, acabei encontrando também o Ulisses, que coorientou tanto o meu mestrado, ainda que não oficialmente, quanto esse doutorado. Eu não tenho palavras pra agradecer os dois pela disponibilidade, incentivo e por confiar em mim nessa tarefa tão complexa. Obrigado por todas as dicas, acadêmicas ou não, que trocamos durante todos esses anos. Mais do que uma relação entre orientando e orientador, tenho certeza que construímos uma relação de amizade.

E por falar em amizade, gostaria de agradecer também três pessoas que convivi no IC durante esse período em nossas reuniões semanais, e que me ajudaram diversas vezes na escrita e discussão dos resultados: Alexsandro, Gabriel e Klairton.

Mesmo em uma época de muitos cortes no ensino superior, tive a oportunidade ímpar de, durante 12 meses, viver em Nantes, na França, trabalhando com os professores Guillaume Fertin e Géraldine Jean. O suporte dado por eles foi fundamental em muitos dos resultados obtidos, e gostaria de deixar registrado aqui meu *merci beaucoup* aos dois.

Muito obrigado aos membros que aceitaram participar das minhas bancas, tanto de qualificação quanto de defesa, pela disponibilidade em ler, assistir, e me ajudar a entregar um trabalho ainda mais completo.

Também gostaria de agradecer os funcionários do Instituto de Computação pela disponibilidade e ajuda com toda a papelada que os processos institucionais exigem.

Esta tese foi realizada com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do Conselho Nacional de Desenvolvimento Científico e Tecnológico - (CNPq) - Processo 140466/2018-5, e do acordo CAPES/COFECUB - Projeto 831/15 - Processo 99999.000350/2016-08.

Resumo

Rearranjos de Genomas são eventos que afetam longos trechos de um genoma durante a evolução. Dentre os rearranjos mais estudados, temos a reversão, que inverte a ordem e a orientação de um bloco consecutivo de genes, e a transposição, que troca a ordem relativa de dois blocos adjacentes.

Modelos matemáticos vêm sendo utilizados para estimar a distância evolutiva entre diferentes organismos por rearranjos de genomas. A representação de um genoma se dá, na maioria das vezes, pela atribuição de um número único para cada gene e, ao supor que não existem genes repetidos, essa representação pode ser vista como uma permutação. Supondo que os dois genomas a serem comparados compartilham o mesmo conjunto de genes, calcular a distância evolutiva entre eles se torna o problema de encontrar o menor número de rearranjos necessários que transforma uma permutação em outra.

Nesta tese, apresentamos diversos resultados envolvendo problemas de rearranjos de genomas: (i) provas de NP-dificuldade para quatro problemas cuja complexidade era desconhecida; (ii) um algoritmo exato em tempo polinomial para um problema cuja complexidade era desconhecida; e (iii) algoritmos de aproximação e provas de NP-dificuldade para problemas onde a representação dos genomas não considera apenas a ordem dos genes. Descrevemos estas contribuições com maior profundidade nos parágrafos a seguir.

Dentre os problemas que envolvem rearranjos de genomas, existem quatro versões que permitem o uso de reversões e transposições ao mesmo tempo e que, apesar dos diversos algoritmos propostos nos últimos 20 anos, permaneciam com complexidade desconhecida. A primeira contribuição apresentada é a prova de NP-dificuldade desses quatro problemas.

Uma das variações dos problemas de rearranjos de genomas consideram que cada rearranjo pode afetar apenas um pequeno número de genes, também conhecidos como rearranjos curtos e super curtos. Neste contexto, nossa segunda contribuição é a prova de que o único problema cuja complexidade era desconhecida envolvendo reversões super curtas e transposições super curtas admite um algoritmo exato em tempo polinomial.

A grande maioria das abordagens em problemas de rearranjos existentes na literatura focaram apenas na ordem relativa dos genes de um genoma, desconsiderando outras características importantes existentes no genoma. Recentemente, pesquisadores mostraram que considerar as regiões existentes entre cada par de genes, chamadas de regiões intergênicas, pode resultar em melhores estimadores de distância em dados reais. Desta forma, nossa terceira contribuição investiga a incorporação das regiões intergênicas em modelos já existentes para reversões e transposições, tanto na abordagem sem restrições como na abordagem que considera apenas rearranjos super curtos, onde investigamos diversos algoritmos de aproximação para problemas que são NP-difíceis ou possuem complexidade desconhecida.

Abstract

Genome rearrangements are events that affect large stretches of a genome during evolution. Two of the most studied rearrangements are reversal, which reverses the order and orientation of a consecutive block of genes, and transposition, which exchanges the relative order of two adjacent blocks.

Mathematical models have been used to estimate the evolutionary distance between different organisms by genome rearrangements. The representation of a genome is often made by assigning a unique number to each gene. If we assume we have no repeated genes, this representation can be seen as a permutation. By considering that the two genomes to be compared share the same set of genes, finding the evolutionary distance between them becomes the problem of finding the smallest number of genome rearrangements needed to transform one permutation into the other.

In this thesis, we present several results involving genome rearrangement problems: (i) proofs of NP-hardness for four problems whose complexity was unknown; (ii) an exact polynomial algorithm for a problem whose complexity was unknown; and (iii) approximation algorithms and proofs of NP-hardness for problems where the genome representation carries more information than only the gene order. We describe these contributions in more depth in the following paragraphs.

Among the problems involving genome rearrangements, four versions that allow the use of reversals and transpositions at the same time had unknown complexity despite the various algorithms proposed in the last 20 years. The first contribution presented is then the proofs of NP-hardness for these four problems.

A variant of genome rearrangement problem considers that each rearrangement can affect only a small number of genes, also known as short and super short rearrangements. In this context, our second contribution is a proof that the only problem involving super short reversals and super short transpositions whose complexity was unknown admits an exact polynomial algorithm.

Most of the approaches for genome rearrangement problems in the literature so far have focused only on the relative order of genes in a genome, disregarding other important features presented in it. Recently, researchers have shown that considering the regions between each pair of genes, called intergenic regions, can result in better distance estimators in real data. Thus, our third contribution investigates the incorporation of intergenic regions in existing models for reversals and transpositions, both in the unrestricted and size restricted versions (i.e. super short operations), where we propose several approximation algorithms for problems that are either NP-hard or with unknown complexity.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Mutações são alterações permanentes na sequência de DNA de um genoma. Elas podem afetar desde locais muito específicos de um genoma, inserindo, removendo ou trocando alguns nucleotídeos de um gene, bem como modificar grandes porções do genoma. Mutações locais são também conhecidas como *mutações pontuais*, enquanto que mutações afetando grandes porções de um genoma são também conhecidas como *rearranjos de genomas*.

Na área da Biologia Computacional, uma forma de estimar a distância evolucionária entre dois genomas é utilizando problemas de Distância por Rearranjos, em que estamos interessados em encontrar uma sequência de rearranjos de genomas necessária para transformar um genoma no outro. Assume-se que o termo *distância* se refere ao menor tamanho requerido para uma sequência de rearranjos ser capaz de executar esta tarefa, sendo este, segundo o Princípio da Parcimônia, um bom estimador da distância evolucionária entre os genomas.

Se os dois genomas comparados não possuem elementos repetidos e compartilham um conjunto de genes, então é possível representá-los por permutações, onde cada gene compartilhado recebe um número distinto. Além disso, sem perda de generalidade, podemos considerar que um destes genomas é a *permutação identidade*, ou seja, a permutação onde os elementos aparecem em ordem crescente. Desta forma, o problema de encontrar a distância de rearranjo entre dois genomas pode ser visto como o problema de *Ordenação de Permutações por Rearranjos de Genomas*, em que buscamos o número mínimo de rearranjos (distância) que ordenam uma permutação.

Os algoritmos desenvolvidos para problemas baseados na distância por rearranjos permitem a realização de comparações de genomas inteiros e podem ser usados como ferramentas para inferir relações filogenéticas. O método usual preenche uma matriz de distância entre pares de genomas utilizada mais tarde para gerar árvores filogenéticas [4, 39, 56].

Enquanto a maioria dos genomas eucariotos são lineares, os genomas procariotos costumam ser circulares (como os genomas bacterianos e mitocondriais, por exemplo). Permutações podem ser utilizadas para modelar tanto genomas circulares quanto genomas lineares. Além disso, quando a orientação dos genes de um genoma é conhecida, podemos atribuir um sinal a cada elemento da permutação, indicando sua orientação, e dizemos que a permutação é *com sinais*. Caso a orientação dos genes no genoma não seja conhecida, dizemos que a permutação que representa este genoma é *sem sinais*.

Os primeiros estudos considerando rearranjos de genomas surgiram na década de 1990,

e este tópico deu origem a uma literatura muito ampla desde então [25]. Dentre os diversos rearranjos propostos e estudados, podemos citar *reversões* [35], em que um segmento contínuo de genes é invertido (bem como as orientações dos genes, quando conhecidas), *transposições* [3], em que dois blocos contínuos de genes adjacentes são trocados, *block interchange* [18], em que dois blocos contínuos de genes não necessariamente adjacentes são trocados, *double cut and join* [57], em que duas adjacências entre genes são quebradas com a posterior criação de duas novas adjacências, dentre outros.

Um *modelo* define qual ou quais rearranjos são permitidos para ordenar uma permutação, e selecionamos quais rearranjos permitir dependendo do contexto em que os genomas que estamos comparando estão inseridos (por exemplo, através de evidências de que os genomas a serem comparados sofrem uma alta incidência de um rearranjo específico). Cada modelo juntamente com a representação escolhida para o genoma define um problema. Por exemplo, se o modelo permite o uso de reversões, e estamos representando um genoma circular cuja orientação dos genes é conhecida por meio de uma permutação, temos o problema de Ordenação de Permutações Circulares com Sinais por Reversões. Por padrão, chamaremos qualquer permutação linear de permutação apenas. Assim, caso a permutação represente um genoma linear, temos o problema de Ordenação de Permutações com Sinais por Reversões.

A Ordenação de Permutações com Sinais por Reversões é um problema que admite algoritmo exato em tempo polinomial [30], e o melhor algoritmo conhecido possui complexidade de tempo sub-quadrática [54]. Além disso, existe um algoritmo com complexidade linear se apenas a distância é requerida, sem a necessidade de devolver uma sequência de operações [1]. Por outro lado, a Ordenação de Permutações sem Sinais por Reversões é um problema NP-difícil [16], e o melhor resultado existente na literatura é uma 1.375-aproximação [5].

Transposições trocam apenas a posição de dois blocos adjacentes, sem alterar a orientação dos elementos de cada bloco. Assim, quando um modelo permite apenas transposições, nós consideramos que a permutação não possui sinais. Desta forma, temos a Ordenação de Permutações sem Sinais por Transposições, que é um problema NP-difícil [14], cujo melhor resultado existente é um algoritmo de 1.375-aproximação [22].

A Ordenação de Permutações com Sinais por Reversões e Transposições é um problema NP-difícil (Capítulo 3) e o melhor resultado existente na literatura é uma 2-aproximação [55]. A Ordenação de Permutações sem Sinais por Reversões e Transposições é um problema NP-difícil (Capítulo 3) e o melhor resultado existente é um algoritmo de $2k$-aproximação, onde k é o fator de aproximação de um algoritmo utilizado na decomposição de ciclos do Grafo de Ciclos [16].

Vários modelos na literatura consideram restrições sobre as operações aplicadas. Em casos em que um rearranjo é mais comum que outro [8, 57], atribuímos um *custo* menor ao primeiro, resultando assim em abordagens *ponderadas por tipo* [23, 46]. Desta forma, podemos criar uma função de custo que atribui pesos distintos para rearranjos diferentes. O objetivo em problemas ponderados por tipo é encontrar uma sequência que minimiza o custo total.

Outra abordagem, quando se assume que rearranjos que afetam porções maiores do genoma têm menos chances de ocorrer, é utilizar a *ponderação por tamanho*, onde o custo

de uma operação é dado por uma função diretamente relacionada ao número de elementos que a operação afeta [19, 40, 41]. Da mesma forma que a abordagem ponderada por tipo, o objetivo neste caso é encontrar uma sequência que minimiza o custo total.

Uma terceira abordagem considera que rearranjos nunca afetam um grande número de elementos ao mesmo tempo. Esta abordagem difere da segunda porque, enquanto aquela penaliza rearranjos que afetam um grande número de elementos, esta os proíbe completamente. A motivação para esta abordagem está na existência de trabalhos mostrando a prevalência de reversões relativamente curtas em genomas bacteriais [20] e genomas eucariotos [42, 53]. Assim, temos problemas que permitem λ-*operações* (onde λ é um inteiro que indica o número máximo de elementos que podem ser afetados por um único rearranjo) [43], *operações curtas*, rearranjos que nunca afetam mais do que três elementos [27, 31, 32, 34], e *operações super curtas*, rearranjos que afetam apenas um ou dois elementos [26, 27, 33].

Ao representar os genomas apenas pela ordem em que seus genes aparecem (resultando em uma permutação) temos que muitas informações presentes no genoma, mas não contidas diretamente nos genes, são perdidas. Em particular, as *regiões intergênicas*, que são sequências de DNA entre os genes, não são consideradas desde os primeiros estudos nesta área. Na maioria dos casos, as regiões ditas conservadas são genes cuja destruição poderia impactar na sobrevivência do organismo. As regiões intergênicas geralmente são menos conservadas porque o impacto na sobrevivência é menor [6, 38]. Assim, vamos considerar que os rearranjos de genomas agem apenas nas regiões entre os genes.

Recentemente, pesquisadores argumentaram que considerar o tamanho das regiões intergênicas pode melhorar os estimadores de distância [6, 7]. Resultados considerando regiões intergênicas em problemas de rearranjo já foram apresentados para os modelos que consideram Double Cut and Join (DCJ) [24] e DCJs com inserções e deleções (indels) [15]. Quando o modelo permite apenas DCJs, o problema se torna NP-difícil, e um algoritmo de 4/3-aproximação foi proposto juntamente com outros dois algoritmos: um esquema de aproximação em tempo polinomial e uma formulação de programação linear inteira [24]. Quando DCJs e indels são permitidos, o problema se torna polinomial, e testes experimentais com este modelo mostraram que os cenários inferidos utilizando regiões intergênicas estão mais próximos dos simulados do que os cenários que não as utilizam [15].

Esta tese está dividida da seguinte forma. O Capítulo 2 apresenta os conceitos mais gerais dos problemas nos quais trabalhamos, e que serão utilizados por pelo menos dois capítulos desta tese. O Capítulo 3 prova que quatro problemas de Ordenação por Rearranjos de Genomas são NP-difíceis. O Capítulo 4 mostra um algoritmo exato em tempo polinomial para um problema de Ordenação por Rearranjos de Genomas em que as operações são super curtas. Os capítulos 5 e 6 investigam a incorporação das regiões intergênicas nos problemas em que as operações são super curtas e sem restrições de tamanho, respectivamente. Por fim, o Capítulo 7 apresenta as considerações finais desta tese.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta conceitos fundamentais amplamente utilizados nos capítulos seguintes. Buscamos mostrar conceitos compartilhados por mais de um capítulo e, quando necessário, os conceitos e notações particulares de um capítulo são apresentados localmente como uma seção preliminar. Os conceitos foram divididos entre os *problemas clássicos*, que consideram apenas a ordenação dos genes, e os *problemas intergênicos*, que também levam em consideração as regiões intergênicas de um genoma.

2.1 Problemas Clássicos

Um genoma $\mathcal{G}$ pode ser modelado (matematicamente) em uma representação matemática por meio de uma n-tupla cujos elementos representam seus genes. Neste trabalho, assumimos que $\mathcal{G}$ não possui genes repetidos e, desta forma, a n-tupla é uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, com $|\pi_i| \in \{1, 2, \ldots, n\}$ e $|\pi_i| \neq |\pi_j|$ para qualquer $i \neq j$.

Quando a orientação dos genes é conhecida, cada elemento π_i possui um sinal, $+$ ou $-$, indicando a orientação do gene que ele representa e π é dita *com sinais*. Caso contrário, a permutação é dita *sem sinais*.

Se o genoma representado por π é circular, utilizaremos a mesma notação, mas, neste caso, os elementos π_n e π_1 são considerados adjacentes e dizemos que π é uma permutação *circular* (também conhecida como *n-ciclo* na literatura). Caso contrário, π é uma permutação *linear*.

A *permutação identidade* é a permutação onde todos os elementos aparecem em ordem crescente e é definida por $\iota = \iota_n = (1\ 2\ \ldots\ n)$. Quando trabalhamos com permutações com sinais, assumimos que todo elemento de ι possui sinal positivo.

Dadas duas permutações $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ e $\sigma = (\sigma_1\ \sigma_2\ \ldots\ \sigma_n)$, a *composição* entre π e σ, denotada por $\pi \cdot \sigma$, resulta na permutação $\alpha = (\alpha_1\ \alpha_2\ \ldots\ \alpha_n)$. Se $\sigma_i < 0$, então $\alpha_i = -\pi_{|\sigma_i|}$, senão $\alpha_i = \pi_{\sigma_i}$. A *inversa* de σ, denotada por σ^{-1}, é a permutação tal que $\sigma^{-1} \cdot \sigma = \iota_n$.

Dito isto, podemos reescrever um par de permutações (π, σ) como (α,ι_n), onde $\alpha = \sigma^{-1} \cdot \pi$, tal que a distância entre as permutações π e σ é igual à distância entre as permutações α e ι_n, ou seja, igual à distância de ordenação de α. Por exemplo, se $\pi = (+1\ -3\ +2\ +5\ -4)$ e $\sigma = (+2\ +4\ -5\ -1\ +3)$, temos que $\sigma^{-1} = (-4\ +1\ +5\ +2\ -3)$,

e a distância entre π e σ é igual à distância entre $\alpha = \sigma^{-1} \cdot \pi = (-4\ -5\ +1\ -3\ -2)$ e $\iota_5 = (+1\ +2\ +3\ +4\ +5)$.

Dada uma permutação linear $\pi = (\pi_1\ \ldots\ \pi_n)$, dizemos que a *permutação estendida* de π é a permutação que começa com o elemento $\pi_0 = 0$, continua com os elementos $\pi_1, \ldots, \pi_n$, e termina com o elemento $\pi_{n+1} = +(n+1)$.

Dada uma permutação π, um par de elementos distintos (π_i, π_j) é dito uma *inversão* se $|\pi_i| > |\pi_j|$ e $i < j$, com $\{i, j\} \in [1..n]$. Denotamos por $inv(\pi)$ o número de inversões em uma permutação π.

Uma *reversão* é uma operação $\rho(i, j)$ que quando aplicada a uma permutação sem sinais $\pi = (\pi_1\ \ldots\ \pi_n)$, reverte a ordem dos elementos entre as posições i e j, com $1 \leq i < j \leq n$:

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

Quando aplicada a uma permutação com sinais $\pi = (\pi_1\ \ldots\ \pi_n)$, esta reversão altera a ordem e troca os sinais dos elementos entre as posições i e j, com $1 \leq i \leq j \leq n$:

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

Uma reversão $\rho(i, j)$ é também chamada de ℓ-reversão, se $\ell = j - i + 1$. Uma reversão é dita *super curta* se $\ell \in \{1, 2\}$.

Uma *transposição* é uma operação $\tau(i, j, k)$ que quando aplicada a uma permutação com ou sem sinais $\pi = (\pi_1\ \ldots\ \pi_n)$ troca a posição do bloco de elementos adjacentes das posições i até $j - 1$ com o bloco de elementos adjacentes das posições j até $k - 1$, com $1 \leq i < j < k \leq n + 1$:

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

Perceba que transposições nunca trocam os sinais dos elementos afetados.

Uma transposição $\tau(i, j, k)$ é também chamada de ℓ-transposição, se $\ell = k - i$. Uma transposição é dita *super curta* se $\ell = 2$.

Um *modelo* $\mathcal{M}$ define o conjunto de operações que são permitidas para ordenar uma permutação. A *distância de ordenação* de π em um modelo $\mathcal{M}$, denotada por $d_{\mathcal{M}}(\pi)$, é o número de operações em uma sequência de tamanho mínimo $S_{\mathcal{M}} = (\beta_1, \ldots, \beta_k)$ contendo apenas operações permitidas pelo modelo $\mathcal{M}$, tal que $\pi \cdot S_{\mathcal{M}} = \pi \cdot \beta_1 \cdots \beta_k = \iota$.

Com frequência utilizaremos o termo *operações* quando um modelo permitir tanto reversões quanto transposições.

2.2 Problemas Intergênicos

Nesta versão do problema, um genoma $\mathcal{G}$ é representado por uma sequência de genes $(\pi_1, \pi_2, \ldots, \pi_n)$ alternada com uma sequência de regiões intergênicas $(\breve{\pi}_1, \breve{\pi}_2, \ldots, \breve{\pi}_{n+1})$: $\mathcal{G} = (\pi, \breve{\pi}) = \breve{\pi}_1, \pi_1, \breve{\pi}_2, \pi_2, \ldots, \breve{\pi}_n, \pi_n, \breve{\pi}_{n+1}$. Como nos problemas clássicos, assumimos que os genomas não possuem genes repetidos, e que compartilham um mesmo conjunto de genes. Assim, escolhemos um dos genomas como referência, e assumimos que os rótulos

de seus genes formam a permutação identidade. Na Figura 2.1(a), os genes do genoma $\mathcal{G}_1$ são rotulados por $\{1, 2, \ldots, 8\}$. Os genes do genoma $\mathcal{G}_2$ possuem os mesmos rótulos e, assim, são representados pela permutação $\pi = (3\ 2\ 1\ 7\ 6\ 4\ 8\ 5)$. Como nos problemas clássicos, as permutações aqui também possuem sua versão estendida.

Assumimos que o número de nucleotídeos entre os genes é um bom estimador para o *tamanho* de uma região intergênica. Por exemplo, o genoma $\mathcal{G}_1$ da Figura 2.1 possui 3 nucleotídeos antes do "Gene 1", 5 entre o "Gene 1" e o "Gene 2", e assim por diante. Já o genoma $\mathcal{G}_1$ possui 1 nucleotídeo antes do "Gene 1", 7 entre o "Gene 1" e o "Gene 2", e assim por diante. Desta forma, $\iota = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$ e $\pi = (3\ 2\ 1\ 7\ 6\ 4\ 8\ 5)$ representam a ordem dos genes enquanto que $\breve{\iota} = (3, 5, 1, 5, 6, 3, 7, 2, 2)$ e $\breve{\pi} = (1, 7, 0, 4, 2, 7, 4, 0, 9)$ representam as regiões intergênicas dos dois genomas, como mostra a Figura 2.1(b). Como não pode existir um número negativo de nucleotídeos em uma região intergênica, temos que $\breve{\pi}_i \in \mathbb{N}$ para qualquer $1 \leq i \leq n+1$.

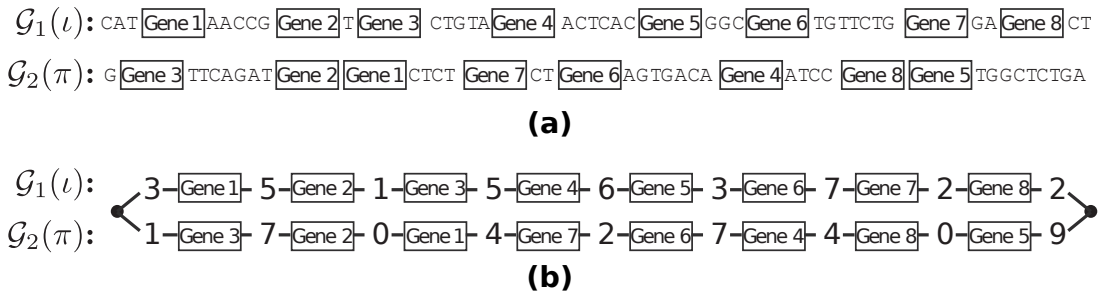


Figura 2.1: Ilustração de um par de genomas fictícios em **(a)** e a representação dos tamanhos das regiões intergênicas em **(b)**.

Uma *reversão intergênica* $\rho^{(i,j)}_{(x,y)}$, com $1 \leq i \leq j \leq n$, $0 \leq x \leq \breve{\pi}_i$, $0 \leq y \leq \breve{\pi}_{j+1}$ e $\{x, y\} \subset \mathbb{N}$, é uma operação que, quando aplicada a $(\pi, \breve{\pi})$, gera $(\pi, \breve{\pi}) \cdot \rho^{(i,j)}_{(x,y)} = (\pi', \breve{\pi}')$ tal que:

- $\pi' = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{-\pi_j\ \ldots\ -\pi_i}\ \pi_{j+1}\ \ldots\ \pi_n)$;
- $\breve{\pi}' = (\breve{\pi}_1, \ldots, \breve{\pi}_{i-1}, \boxed{\breve{\pi}'_i}, \underline{\breve{\pi}_j, \ldots, \breve{\pi}_{i+1}}, \boxed{\breve{\pi}'_{j+1}}, \breve{\pi}_{j+2}, \ldots, \breve{\pi}_{n+1})$, com $\breve{\pi}'_i = x + y$ e $\breve{\pi}'_{j+1} = \breve{\pi}_i - x + \breve{\pi}_{j+1} - y$.

Em outras palavras, $\rho^{(i,j)}_{(x,y)}$ corta $(\pi, \breve{\pi})$ em dois pontos: na região intergênica localizada na posição i de $\breve{\pi}$, após os primeiros x nucleotídeos; e na região intergênica localizada na posição $j+1$ de $\breve{\pi}$, após y nucleotídeos. Este segmento de genes e regiões intergênicas é então invertido bem como os sinais dos elementos de π, e novos valores são atribuídos para as regiões intergênicas nas posições i e $(j+1)$.

Uma *transposição intergênica* é uma operação $\rho^{(i,j,k)}_{(x,y,z)}$, com $1 \leq i < j < k \leq n+1$, $0 \leq x \leq \breve{\pi}_i$, $0 \leq y \leq \breve{\pi}_j$, $0 \leq z \leq \breve{\pi}_k$, e $\{x, y, z\} \subset \mathbb{N}$. Esta transposição intergênica quando aplicada em $(\pi, \breve{\pi})$ gera $(\pi, \breve{\pi}) \cdot \rho^{(i,j,k)}_{(x,y,z)} = (\pi', \breve{\pi}')$, onde:

- $\pi' = (\pi_1\ \pi_2\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \pi_{j+1}\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}}\ \pi_k\ \pi_{k+1}\ \ldots\ \pi_n)$.
- $\breve{\pi}' = (\breve{\pi}_1, \ldots, \breve{\pi}_{i-1}, \boxed{\breve{\pi}'_i}, \breve{\pi}_{j+1}, \ldots, \breve{\pi}_{k-1}, \boxed{\breve{\pi}'_j}, \underline{\breve{\pi}_{i+1}, \ldots, \breve{\pi}_{j-1}}, \boxed{\breve{\pi}'_k}, \breve{\pi}_{k+1}, \ldots, \breve{\pi}_{n+1})$, tal que $\breve{\pi}'_i = x + \breve{\pi}_j - y$, $\breve{\pi}'_j = z + \breve{\pi}_i - x$ e $\breve{\pi}'_k = y + \breve{\pi}_k - z$.

Em outras palavras, ela troca dois segmentos de π e $\breve{\pi}$ de lugar conforme definido acima, agindo em três regiões intergênicas de $\breve{\pi}$: na região intergênica localizada na posição i de $\breve{\pi}$, após x nucleotídeos; na região intergênica localizada na posição j de $\breve{\pi}$, após y nucleotídeos; e na região intergênica localizada na posição k de $\breve{\pi}$, após z nucleotídeos.

Como ι pode ser facilmente recuperada a qualquer momento utilizando o tamanho da permutação π, definimos uma *instância* para problemas que consideram regiões intergênicas como uma tripla $(\pi, \breve{\pi}, \breve{\iota})$. Aqui estamos interessados em encontrar o menor número de rearranjos intergênicos necessários para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

A *distância de ordenação* de uma instância $(\pi, \breve{\pi}, \breve{\iota})$ em um modelo $\mathcal{M}$, denotada por $d_{\mathcal{M}}(\pi, \breve{\pi}, \breve{\iota})$, é o número de operações intergênicas em uma sequência de tamanho mínimo $S_{\mathcal{M}}$, contendo apenas operações permitidas pelo modelo $\mathcal{M}$, tal que $(\pi, \breve{\pi}) \cdot S_{\mathcal{M}} = (\iota, \breve{\iota})$.

Como reversões e transposições são eventos *conservativos*, ou seja, eles não alteram nem os genes nem a quantidade total de nucleotídeos dentro das regiões intergênicas do genoma, nós assumimos aqui que $\sum_{i=1}^{n+1} \breve{\pi}_i = \sum_{i=1}^{n+1} \breve{\iota}_i$, o que garante que o número total de nucleotídeos entre os genomas é conservado. A Figura 2.2 mostra exemplos da aplicação de uma transposição intergênica e de uma reversão intergênica.

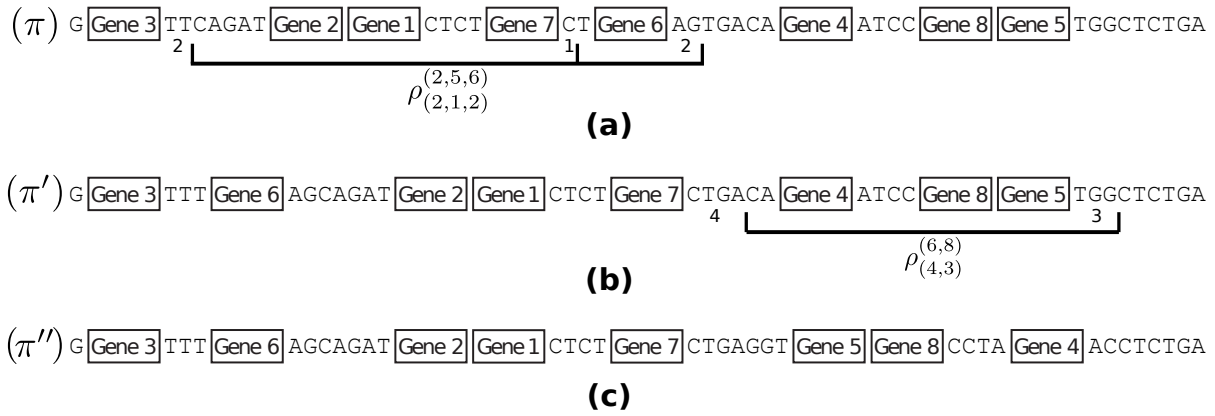


Figura 2.2: Seja um genoma $\mathcal{G}$ representado por $(\pi, \breve{\pi})$ tal que $\pi = (3\ 2\ 1\ 7\ 6\ 4\ 8\ 5)$ e $\breve{\pi} = (1, 7, 0, 4, 2, 7, 4, 0, 9)$. Em (a) aplicamos a transposição intergênica $\rho^{(2,5,6)}_{(2,1,2)}$, gerando o genoma $(\pi', \breve{\pi}')$ em (b) tal que $\pi' = (3\ 6\ 2\ 1\ 7\ 4\ 8\ 5)$ e $\breve{\pi}' = (1, 3, 7, 0, 4, 6, 4, 0, 9)$. Em (b) aplicamos a reversão intergênica $\rho^{(6,8)}_{(4,3)}$ gerando o genoma $(\pi'', \breve{\pi}'')$ em (c) tal que $\pi'' = (3\ 6\ 2\ 1\ 7\ 5\ 8\ 4)$ e $\breve{\pi}'' = (1, 3, 7, 0, 4, 7, 0, 4, 8)$.

2.3 Algoritmos de Aproximação

Quando um problema $\mathcal{Q}$ pertence à classe de problemas NP-difíceis, não é possível encontrar um algoritmo exato em tempo polinomial para $\mathcal{Q}$, a menos que P = NP. Considere agora que $\mathcal{Q}$ é um problema de minimização, e seja $\mathbf{OPT}(\mathcal{I})$ o valor da solução ótima para uma instância $\mathcal{I}$ qualquer do problema $\mathcal{Q}$.

Seja $\mathbf{A}$ um algoritmo polinomial que, para toda instância $\mathcal{I}$ de $\mathcal{Q}$, retorne uma solução viável com valor $\mathcal{S}_A(\mathcal{I})$. Dizemos que $\mathbf{A}$ é uma α-aproximação para $\mathcal{Q}$ se, para qualquer instância $\mathcal{I}$, a inequação $\mathcal{S}_A(\mathcal{I}) \leq \alpha \cdot \mathbf{OPT}(\mathcal{I})$ é satisfeita.

Como nem sempre os valores de $\mathbf{OPT}(\mathcal{I})$ são conhecidos, algoritmos de aproximação para problemas de minimização podem ser obtidos por meio de *limitantes inferiores*. Uma função $\mathcal{L}$ é um *limitante inferior* para um problema $\mathcal{Q}$ se, para qualquer instância $\mathcal{I}$ temos que $\mathbf{OPT}(\mathcal{I}) \geq \mathcal{L}(\mathcal{I})$. Desta forma, $\mathbf{A}$ é uma α-aproximação para $\mathcal{Q}$ se, para qualquer instância $\mathcal{I}$, a inequação $\mathcal{S}_A(\mathcal{I}) \leq \alpha \cdot \mathcal{L}(\mathcal{I})$ é satisfeita (observe que, por definição, $\mathcal{S}_A(\mathcal{I}) \leq \alpha \cdot \mathcal{L}(\mathcal{I})$ implica em $\mathcal{S}_A(\mathcal{I}) \leq \alpha \cdot \mathbf{OPT}(\mathcal{I})$).

Dado um algoritmo $\mathbf{A}$ de α-aproximação para um problema de minimização $\mathcal{Q}$ cujo fator de aproximação é obtido a partir de um limitante inferior $\mathcal{L}$, seja $\mathcal{C} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ um conjunto de instâncias para $\mathcal{Q}$. O *fator de aproximação experimental médio* de $\mathbf{A}$ no conjunto $\mathcal{C}$ é dado por $\mathbf{AE_{avg}} = \frac{\sum_{\mathcal{I} \in \mathcal{C}} \frac{\mathcal{S}_A(\mathcal{I})}{\mathcal{L}(\mathcal{I})}}{n}$. Além disso, o *fator de aproximação experimental máximo* de $\mathbf{A}$ no conjunto $\mathcal{C}$ é dado por $\mathbf{AE_{max}} = \max_{\mathcal{I} \in \mathcal{C}}\{\frac{\mathcal{S}_A(\mathcal{I})}{\mathcal{L}(\mathcal{I})}\}$. Por definição temos que $1 \leq \mathbf{AE_{avg}} \leq \mathbf{AE_{max}} \leq \alpha$.

Nos próximos capítulos apresentaremos vários algoritmos de aproximação para os problemas estudados.

Capítulo 3

Complexidade dos Problemas de Ordenação de Permutações por Reversões e Transposições

Os problemas de Ordenação de Permutações por Rearranjos *não ponderados* não fazem distinção entre as operações, ou seja, dada uma sequência qualquer de operações, cada uma contribui com uma unidade na distância de ordenação.

Existem também problemas de Ordenação de Permutações por Rearranjos em que diferentes rearranjos colaboram com diferentes pesos na distância de ordenação, os chamados problemas *ponderados*. Estes modelos foram propostos de modo a favorecer determinados rearranjos em cenários em que eles são conhecidos por serem mais frequentes que outros e, desta forma, tornaram possível modelar cenários biológicos específicos.

Por exemplo, em um problema onde as operações permitidas pelo modelo são reversões e transposições, podemos assumir que reversões possuem *peso* w_ρ enquanto que o peso das transposições é w_τ.

Assim, dada uma sequência de operações $\mathcal{S}_\mathcal{M}$ permitidas pelo modelo $\mathcal{M}$ e pesos w_β para cada $\beta \in \mathcal{M}$, seja k_β o número de operações do tipo β em $S_\mathcal{M}$. Dizemos que o *custo* de $S_\mathcal{M}$ é igual à $\sum_{\beta\in\mathcal{M}} w_\beta k_\beta$.

Em problemas ponderados, ao invés da distância de ordenação, estamos interessados na *distância ponderada de ordenação*, denotada por $d^w_\mathcal{M}(\pi)$, que nada mais é que o custo mínimo de uma sequência de operações permitidas pelo modelo $\mathcal{M}$, denotada por $S_\mathcal{M}$, tal que $\pi \cdot S_\mathcal{M} = \iota$ (neste capítulo trabalharemos apenas com permutações e não com instâncias intergênicas).

Neste capítulo, provaremos que quatro problemas de Ordenação de Permutações por Rearranjos são NP-difíceis. Dois deles são versões *não ponderadas*: o problema de Ordenação de Permutações por Reversões e Transposições e o problema de Ordenação de Permutações com Sinais por Reversões e Transposições. Os outros dois são versões *ponderadas*: o problema de Ordenação de Permutações por Reversões e Transposições Ponderadas e o problema de Ordenação de Permutações com Sinais por Reversões e Transposições Ponderadas. No caso das versões ponderadas, a prova requer que uma transposição seja no máximo 50% mais custosa que uma reversão.

Este resultado foi publicado em 2019 na revista *Journal of Computational Biology* [44].

3.1 Conceitos e Notações

Nesta seção, apresentamos dois conceitos importantes utilizados neste capítulo: breakpoints e strips.

3.1.1 Breakpoints

De modo geral, um *breakpoint* indica uma vizinhança em π que não existe em ι. Dependendo do modelo e do tipo de permutação, breakpoints podem ser definidos de duas maneiras.

Definição 1. *Dada uma permutação qualquer π, um par de elementos π_i e π_{i+1} de π, com $0 \leq i \leq n$, é um* **breakpoint tipo um** *se $|\pi_{i+1} - \pi_i| \neq 1$.*

Definição 2. *Dada uma permutação qualquer π, um par de elementos π_i e π_{i+1} de π, com $0 \leq i \leq n$, é um* **breakpoint tipo dois** *se $\pi_{i+1} - \pi_i \neq 1$.*

O número de breakpoints das definições 1 e 2 em uma permutação π no modelo $\mathcal{M}$ é denotado por $b_{\mathcal{M}}(\pi)$.

Reversões em permutações sem sinais (r) e reversões e transposições em permutações sem sinais (rt) são modelos que utilizam breakpoints tipo um. Transposições em permutações sem sinais (t), reversões em permutações com sinais ($\bar{r}$), e reversões e transposições em permutações com sinais ($\overline{rt}$) são modelos que utilizam breakpoints tipo dois.

Dada a permutação sem sinais estendida $\pi = (0\ 3\ 2\ 4\ 5\ 1\ 6)$, temos que $b_r(\pi) = 4$, uma vez que os pares $(0, 3)$, $(2, 4)$, $(5, 1)$ e $(1, 6)$ são breakpoints. Além disso, $b_t(\pi) = 5$, uma vez que que os pares $(0, 3)$, $(3, 2)$, $(2, 4)$, $(5, 1)$ e $(1, 6)$ são breakpoints. Dada a permutação com sinais estendida $\pi = (+0\ -3\ -2\ +1\ -4\ +5\ +6)$, temos que $b_{\bar{r}}(\pi) = b_{\overline{rt}}(\pi) = 4$, uma vez que os pares $(+0, -3)$, $(-2, +1)$, $(+1, -4)$ e $(-4, +5)$ são breakpoints.

Denotamos por $\Delta b_{\mathcal{M}}(\pi, \pi \cdot \beta) = b_{\mathcal{M}}(\pi \cdot \beta) - b_{\mathcal{M}}(\pi)$ a variação no número de breakpoints definido pelo modelo $\mathcal{M}$ após aplicar o rearranjo β permitido por $\mathcal{M}$.

Como as transposições cortam uma permutação em três lugares, se o modelo $\mathcal{M}$ permite apenas transposições temos que $\Delta b_{\mathcal{M}}(\pi, \pi \cdot \tau) \in [-3..3]$. Uma reversão corta uma permutação em dois lugares, logo, se o modelo $\mathcal{M}$ permite apenas reversões, $\Delta b_{\mathcal{M}}(\pi, \pi \cdot \rho) \in [-2..2]$. Essas observações resultam nos lemas a seguir.

Lema 1. *No modelo $\overline{rt}$ temos que $d_{\overline{rt}}(\pi) \geq \frac{b_{\overline{rt}}(\pi)}{3}$, e dados w_ρ e w_τ, temos que $d^w_{\overline{rt}}(\pi) \geq \min\{\frac{w_\rho}{2}, \frac{w_\tau}{3}\} b_{\overline{rt}}(\pi)$.*

Lema 2. *No modelo rt temos que $d_{rt}(\pi) \geq \frac{b_{rt}(\pi)}{3}$, e dados w_ρ e w_τ, temos que $d^w_{rt}(\pi) \geq \min\{\frac{w_\rho}{2}, \frac{w_\tau}{3}\} b_{rt}(\pi)$.*

3.1.2 Strips

Strips são sequências maximais de elementos consecutivos sem breakpoints de uma permutação estendida π. A primeira strip começa com o elemento π_0, a última strip termina com o elemento π_{n+1}, e as strips restantes estão sempre localizadas entre dois breakpoints.

Em permutações com sinais, uma strip é dita *positiva* se todos os seus elementos possuem sinais positivos, e é dita *negativa* caso contrário.

Em permutações sem sinais, uma strip com mais de um elemento é dita *crescente* se os seus elementos formam uma sequência crescente e é dita *decrescente* caso contrário. Uma strip com apenas um elemento é chamada de *singleton* e é definida como crescente se o elemento dessa strip pertence ao conjunto $\{0, n{+}1\}$, e decrescente caso contrário.

Por exemplo, a permutação sem sinais estendida $\pi = (0 \cdot 3\ 2 \cdot 4\ 5 \cdot 1 \cdot 6)$ possui cinco strips no modelo rt: dois singletons crescentes $\langle 0 \rangle$ e $\langle 6 \rangle$; um singleton decrescente $\langle 1 \rangle$; uma strip decrescente $\langle 3\ 2 \rangle$; e uma strip crescente $\langle 4\ 5 \rangle$. A permutação com sinais estendida $\pi = (+0 \cdot -3\ -2 \cdot +1 \cdot -4 \cdot +5\ +6)$ possui cinco strips no modelo $\overline{rt}$: três strips positivas $\langle +0 \rangle$, $\langle +1 \rangle$ e $\langle +5\ +6 \rangle$; e duas strips negativas $\langle -3\ -2 \rangle$ e $\langle -4 \rangle$.

Definição 3. *Dizemos que uma reversão $\rho(i, j)$* **corta** *uma strip quando aplicada em π caso (π_{i-1}, π_i) ou (π_j, π_{j+1}) não seja(m) breakpoints, ou seja, pelo menos um dos dois pares é formado por dois elementos consecutivos em uma mesma strip.*

3.2 Provas de NP-dificuldade

Nesta seção, provamos que quatro problemas de decisão envolvendo reversões e transposições são NP-difíceis, sendo dois deles em permutações com sinais e outros dois em permutações sem sinais.

Seja π uma permutação sem sinais e seja $k = b_t(\pi)/3$. Temos que $d_t(\pi) \geq b_t(\pi)/3$ por definição. Denotamos por **B3T** como o problema de decidir se a igualdade $d_t(\pi) = k$ é satisfeita no problema de ordenação por transposições. Bulteau e coautores [14] provaram que **B3T** está em NP-difícil.

Utilizaremos uma redução do problema **B3T** para provar que os quatro problemas de decisão envolvendo reversões e transposições também estão em NP-difícil.

3.2.1 Problemas de Decisão em Permutações com Sinais

Nesta seção, mostramos que os seguintes problemas em permutações com sinais estão em NP-difícil:

- **Ordenação de Permutações com Sinais por Reversões e Transposições (SRT)**: dada uma permutação com sinais π e um inteiro não negativo k, decidir se é possível ordenar π utilizando uma sequência com k reversões ou transposições.
- **Ordenação de Permutações com Sinais por Reversões e Transposições Ponderadas (WSRT)**: dada uma permutação com sinais π, pesos w_ρ e w_τ, e um valor k, decidir se é possível ordenar π com uma sequência de custo menor ou igual a k.

Vamos definir um lema preliminar que será utilizado mais tarde na prova de NP-dificuldade. O lema estabelece uma família de permutações com sinais nas quais nenhuma reversão é capaz de remover breakpoints.

Lema 3. *Se uma permutação com sinais π possui apenas strips crescentes, então para qualquer reversão $\rho(i,j)$ temos que $\Delta b_{\overline{rt}}(\pi, \pi \cdot \rho) \leq 0$.*

Demonstração. Como reversões mudam os sinais dos elementos, uma reversão $\rho(i,j)$ não pode remover breakpoints caso os pares (π_{i-1}, π_j) e (π_i, π_{j+1}) sejam ambos positivos ou ambos negativos. Como π possui apenas strips positivas, o que significa que todos os elementos de π possuem sinais positivos, o lema segue. □

Teorema 1. WSRT *está em* NP*-difícil quando $w_\tau / w_\rho \leq 1.5$.*

Demonstração. Primeiramente note que em **WSRT** as permutações possuem sinais. Assim, dada uma instância $\pi = (\pi_1 \ \ldots \ \pi_n)$ de **B3T**, que não possui sinais, construímos uma instância $(\pi', w_\rho, w_\tau, k')$ para **WSRT** tal que $\pi'_i = +\pi_i$ para $1 \leq i \leq n$, w_ρ e w_τ são tais que $\frac{w_\tau}{w_\rho} \leq 1.5$ e $k' = kw_\tau$ onde $k = \frac{b_t(\pi)}{3}$.

Note que estes modelos utilizam a mesma definição de breakpoints, então $b_{\overline{rt}}(\pi') = k$. Vamos mostrar agora que a instância π de **B3T** é satisfeita se, e somente se, $d^w_{\overline{rt}}(\pi') \leq k'$.

($\Rightarrow$) Seja S_τ uma sequência de k transposições tal que $\pi \cdot S_\tau = \iota$. Note que, para a permutação com sinais π' da instância $(\pi', w_\rho, w_\tau, k')$ também temos que $\pi' \cdot S_\tau = \iota$, então a sequência S_τ com k transposições possui custo kw_τ, e $d^w_{\overline{rt}}(\pi) \leq kw_\tau \leq k'$.

($\Leftarrow$) Seja $S_{\overline{\rho\tau}}$ uma sequência de reversões e transposições que ordena π' e custa exatamente k'. Temos que, na média, cada breakpoint removido por uma reversão implica em um custo de pelo menos $w_\rho/2$ em $S_{\overline{\rho\tau}}$ (uma vez que $\Delta b_{\mathcal{M}}(\pi', \pi' \cdot \rho) \leq 2$), e cada breakpoint removido por uma transposição implica em um custo de pelo menos $w_\tau/3 \leq 1.5w_\rho/3 = w_\rho/2$ em $S_{\overline{\rho\tau}}$ (uma vez que $\Delta b_{\mathcal{M}}(\pi', \pi' \cdot \tau) \leq 3$). Segue então, pelo Lema 1 que $S_{\overline{\rho\tau}}$ deve possuir custo pelo menos $\frac{w_\tau}{3} b_{\overline{rt}}(\pi') = kw_\tau = k'$. Dado que $S_{\overline{\rho\tau}}$ custa exatamente k', temos que toda reversão deve remover dois breakpoints e toda transposição deve remover três breakpoints.

Se $w_\tau < 1.5w_\rho$, então o custo para remover cada breakpoint utilizando reversões deve ser estritamente menor que $w_\rho/2$. Como breakpoints removidos por reversões têm um custo de pelo menos $w_\rho/2$, isto significa que não podem existir reversões em $S_{\overline{\rho\tau}}$.

Se $w_\tau = 1.5w_\rho$, então o custo para remover cada breakpoint utilizando reversões e transposições deve ser $w_\rho/2$, mas note que a permutação com sinais π' possui apenas strips positivas, e enquanto apenas transposições são aplicadas esta permutação permanecerá com strips positivas. Pelo Lema 3, a primeira reversão aplicada não irá diminuir o número de breakpoints, então $S_{\overline{\rho\tau}}$ não pode custar exatamente k' se ela possui uma reversão.

Em ambos os casos, como $S_{\overline{\rho\tau}}$ possui apenas transposições, também temos que $\pi \cdot S_{\overline{\rho\tau}} = \iota$, e $S_{\overline{\rho\tau}}$ possui $k'/w_\tau = b_{\overline{rt}}(\pi')/3 = b_t(\pi)/3 = k$ transposições. □

O corolário a seguir mostra que **SRT** também está em NP-difícil.

Corolário 1. SRT *está em* NP*-difícil.*

Demonstração. Diretamente pelo Teorema 1, dado que **SRT** pode ser modelado como **WSRT** onde $w_\rho = 1$ e $w_\tau = 1$. □

3.2.2 Problemas de Decisão em Permutações sem Sinais

Nesta seção, mostramos que os seguintes problemas em permutações sem sinais estão em NP-difícil:

- **Ordenação de Permutações por Reversões e Transposições (URT)**: dada uma permutação sem sinais π e um inteiro não negativo k, decidir se é possível ordenar π utilizando uma sequência com k reversões ou transposições.
- **Ordenação de Permutações por Reversões e Transposições Ponderadas (WURT)**: dada uma permutação sem sinais π, pesos w_ρ e w_τ e um valor k, decidir se é possivel ordenar π utilizando uma sequência de custo menor ou igual a k.

De modo semelhante às permutações com sinais, vamos definir um lema preliminar que será utilizado durante a prova de NP-dificuldade. O lema a seguir estabelece uma família de permutações sem sinais nas quais nenhuma reversão é capaz de remover breakpoints.

Lema 4. *Se uma permutação sem sinais π possui apenas strips crescentes, então para qualquer reversão $\rho(i,j)$ temos que $\Delta b_{rt}(\pi, \pi \cdot \rho) \leq 0$.*

Demonstração. Note que quando uma reversão $\rho(i,j)$ é aplicada em uma permutação sem sinais, qualquer strip crescente $S = \langle \pi_a \ \ldots \ \pi_b \rangle$ tal que $a \geq i$ e $b \leq j$ é transformada em uma strip decrescente.

Suponha que π possui apenas strips crescentes e que existe uma reversão $\rho(i,j)$ que remove pelo menos um breakpoint quando aplicada a π. Suponha também, sem perda de generalidade, que o breakpoint entre os elementos π_{i-1} e π_i é removido por $\rho(i,j)$, ou seja, $\pi_j = \pi_{i-1} + 1$. Seja S a strip cujo último elemento é π_{i-1} e seja $\pi' = \pi \cdot \rho(i,j)$. Como $\rho(i,j)$ remove o breakpoint (π_{i-1}, π_i), a strip $S = \langle \ldots \pi_{i-1} \rangle$ em π é transformada em $S' = \langle \ldots \ \pi_{i-1} \ \pi_j \ \ldots \rangle$ em π'. Entretanto, note que S' permanece uma strip crescente ($\pi_{i-1} < \pi_j$), o que contradiz o fato de que todas as strips entre π_{i-1} e π_{j+1} são crescentes, e o lema segue. □

Teorema 2. WURT *está em* NP*-difícil se* $w_\tau / w_\rho \leq 1.5$.

Demonstração. Dada uma instância $\pi = (\pi_1 \ \ldots \ \pi_n)$ para **B3T**, construímos uma instância $(\pi', w_\rho, w_\tau, k')$ para **WURT** mapeando cada π_i, com $i \in [1..n]$, em dois valores $\pi'_{2i-1} = 2\pi_i - 1$ e $\pi'_{2i} = 2\pi_i$ em π', tomando w_ρ e w_τ tais que $\frac{w_\tau}{w_\rho} \leq 1.5$, e utilizando $k' = kw_\tau$, onde $k = b_t(\pi)/3$.

Note que π' possui o dobro de elementos que π e, exceto pelo elemento π_0, todo elemento em posições pares possui valor igual ao elemento à sua esquerda acrescido em uma unidade. Isto significa que (i) exceto pela primeira e última strips, qualquer outra strip em π' deve conter pelo menos dois elementos, ou seja, nenhuma delas é um singleton, e (ii) toda strip de π é crescente.

Além disso, $b_{rt}(\pi') = b_t(\pi)$, pois (i) se o par (π_i, π_{i+1}) é um breakpoint em π, então (π'_{2i}, π'_{2i+1}) obrigatoriamente é um breakpoint em π', com $i \in [0..n]$, e (ii) os pares (π'_{2i-1}, π'_{2i}) nunca são breakpoints, para $i \in [1..n]$. Agora mostramos que uma instância para **B3T** é satisfeita se, e somente se, $d^w_{rt}(\pi') \leq k'$.

($\Rightarrow$) Seja S_τ uma sequência com $k = b_t(\pi)/3$ transposições tal que $\pi \cdot S_\tau = \iota$. Vamos definir a sequência S'_τ tal que $\pi' \cdot S'_\tau = \iota$ com base em S_τ. Dado que cada π_i, com $i \in [1..n]$, foi mapeado como dois elementos consecutivos em π', podemos transformar os índices de cada transposição $\tau(i, j, k)$ da sequência S_τ gerando $\tau(2i-1, 2j-1, 2k-1)$ na sequência S'_τ. Segue então que a sequência S'_τ com k transposições possui custo kw_τ, e $d^w_{rt}(\pi') \leq kw_\tau = k'$.

($\Leftarrow$) Seja $S_{\rho\tau}$ uma sequência de reversões e tranposições que ordena π' e custa exatamente k'. De modo similar ao caso com sinal, cada breakpoint removido por reversões implica em um custo de pelo menos $w_\rho/2 \geq 1.5w_\tau/3$ em $S_{\rho\tau}$, e cada breakpoint removido por transposições implica em um custo de pelo menos $w_\tau/3 \leq 1.5w_\rho/3 = w_\rho/2$ em $S_{\rho\tau}$. Segue então pelo Lema 2 que qualquer sequência de reversões e transposições $S_{\rho\tau}$ tal que $\pi' \cdot S_{\rho\tau} = \iota$ possui custo de pelo menos $b_{rt}(\pi')w_\tau/3 = kw_\tau = k'$. Dado que $S_{\rho\tau}$ custa exatamente k', temos que toda reversão de $S_{\rho\tau}$ deve remover dois breakpoints e toda transposição de $S_{\rho\tau}$ deve remover três breakpoints.

Se $w_\tau < 1.5w_\rho$, então o custo para remover cada breakpoint utilizando reversões e transposições de $S_{\rho\tau}$ deve ser estritamente menor que $w_\rho/2$. Como breakpoints removidos por reversões tem um custo de pelo menos $w_\rho/2$, isto significa que não podem existir reversões em $S_{\rho\tau}$.

Se $w_\tau = 1.5w_\rho$, então o custo para remover cada breakpoint utilizando reversões e transposições de $S_{\rho\tau}$ deve ser $w_\rho/2$. Entretanto note que π' possui apenas strips crescentes e nenhuma strip pode ser transformada em um singleton enquanto transposições são aplicadas apenas onde existem breakpoints, então todas as strips permanecem crescentes. Pelo Lema 4, a primeira reversão aplicada em π não irá diminuir o número de breakpoints, logo $S_{\rho\tau}$ deve custar mais que k' caso possua uma reversão que não remove nenhum breakpoint.

Em ambos os casos, como $S_{\rho\tau}$ possui apenas transposições podemos mapear cada transposição $\tau(i, j, k) \in S_{\rho\tau}$ como $\tau(\frac{i+1}{2}, \frac{j+1}{2}, \frac{k+1}{2}) \in S'_\tau$ tal que $\pi \cdot S'_\tau = \iota$, e S'_τ possui $k'/w_\tau = b_{rt}(\pi')/3 = b_t(\pi)/3 = k$ transposições. □

O corolário a seguir mostra que **URT** também está em NP-difícil.

Corolário 2. URT *está em* NP*-difícil.*

Demonstração. Diretamente pelo Teorema 2, dado que **URT** pode ser modelado como **WURT** onde $w_\rho = 1$ e $w_\tau = 1$. □

3.3 Conclusões

Este capítulo apresentou a prova de NP-dificuldade de quatro problemas de ordenação diferentes, a saber: Ordenação de Permutações por Reversões e Transposições, Ordenação de Permutações com Sinais por Reversões e Transposições, Ordenação de Permutações por Reversões e Transposições Ponderadas e Ordenação de Permutações com Sinais por Reversões e Transposições Ponderadas. É importante notar que mostramos que as versões ponderadas estão em NP-difícil se $w_\tau/w_\rho \leq 1.5$, onde w_ρ é o peso de uma reversão e w_τ é o peso de uma transposição. Assim, a complexidade dos problemas ponderados onde $w_\tau/w_\rho > 1.5$ permanece em aberto.

Capítulo 4

Ordenação de Permutações Circulares com Sinais por Operações Super Curtas

Problemas de Ordenação de Permutações por Operações Super Curtas já foram estudados em permutações lineares e circulares, com e sem sinais, quando as operações permitidas são reversões e/ou transposições [26,27,33,36]. Definimos na Seção 4.1 o que chamamos de operações super curtas *cíclicas*, um tipo particular de operação super curta que modifica as permutações lineares de forma cíclica, o que nos auxilia a estender os resultados obtidos para genomas circulares.

Em permutações sem sinais, temos que uma reversão super curta possui o mesmo efeito de uma transposição super curta, o que resulta em apenas duas versões diferentes do problema: (a.1) a Ordenação de Permutações sem Sinais por Operações Super Curtas e (a.2) Ordenação de Permutações sem Sinais por Operações Super Curtas Cíclicas. Além disso, já sabemos que transposições não trocam os sinais dos elementos, logo, um modelo para permutações com sinais não pode permitir apenas o uso de transposições para ordenar permutações — o modelo deve permitir também reversões ou outra operação que troca os sinais dos elementos. Isto resulta em outros quatro problemas diferentes: (b.1) Ordenação de Permutações com Sinais por Reversões Super Curtas, (b.2) Ordenação de Permutações com Sinais por Operações Super Curtas, (b.3) Ordenação de Permutações com Sinais por Reversões Super Curtas Cíclicas e (b.4) Ordenação de Permutações com Sinais por Operações Super Curtas Cíclicas.

A Tabela 4.1 resume os seis problemas e apresenta suas complexidades. Note que os problemas (a.1)-(b.3) já possuem uma prova de que são polinomiais, enquanto que a complexidade do problema (b.4) continuava em aberto.

Neste capítulo, provamos que (b.4) também admite algoritmo exato em tempo polinomial, fechando então esta lacuna na literatura com relação aos problemas que consideram operações super curtas. Este resultado foi publicado em 2018 na revista *Algorithms for Molecular Biology* [48].

Tabela 4.1: Lista dos problemas de Ordenação por Operações Super Curtas e Operações Super Curtas Cíclicas e as soluções existentes na literatura.

Permutação	Operações Permitidas	Solução Polinomial Exata
Sem Sinais	(a.1) Operações Super Curtas	[36]
Sem Sinais	(a.2) Operações Super Curtas Cíclicas	[33]
Com Sinais	(b.1) Reversões Super Curtas	[27]
Com Sinais	(b.2) Operações Super Curtas	[27]
Com Sinais	(b.3) Reversões Super Curtas Cíclicas	[26]
Com Sinais	(b.4) Operações Super Curtas Cíclicas	Neste capítulo

4.1 Conceitos e Notações

Nesta seção, apresentamos notações e conceitos importantes que utilizaremos neste capítulo.

4.1.1 Operações Cíclicas

Dada uma permutação π, uma *reversão cíclica* $\rho^*(i,j)$ é uma extensão de uma reversão: $\rho^*(i,j) = \rho(i,j)$ caso $i \leq j$, enquanto que se $i > j$, a sequência invertida por $\rho^*(i,j)$ quando aplicada a π é $(\pi_i, ..., \pi_n, \pi_1, ..., \pi_j)$. De modo similar à reversão, uma reversão cíclica $\rho^*(i,j)$ é dita uma ℓ-reversão se $\ell = j - i + 1 \pmod{n}$. Dizemos que uma ℓ-reversão é *super curta* se $\ell \in \{1, 2\}$.

Assim como reversões, dada uma permutação π, uma *transposição cíclica* $\tau^*(i,j,k)$ é uma extensão de transposições para os casos (a) $1 \leq k < i < j \leq n$ (em que as sequências $(\pi_i, ..., \pi_{j-1})$ e $(\pi_j, ..., \pi_n, \pi_1, ..., \pi_{k-1})$ são trocadas quando a transposição é aplicada a π), e (b) $1 \leq j < k < i \leq n$ (no qual as sequências $(\pi_i, ..., \pi_n, \pi_1, ..., \pi_{j-1})$ e $(\pi_j, ..., \pi_{k-1})$ são trocadas quando a transposição é aplicada a π). De modo similar à transposição, uma transposição cíclica $\tau^*(i,j,k)$ é dita uma ℓ-transposição se $\ell = x + y$ com $x = j - i \pmod{n}$ e $y = k - j \pmod{n}$, e dizemos que uma ℓ-transposição é *super curta* caso $\ell = 2$.

Daqui em diante, chamaremos de *Operação Super Curta* (ou *OSC*) qualquer 1-reversão, 2-reversão ou 2-transposição (sendo elas cíclicas ou não); além disso, qualquer OSC que não é uma 1-reversão será também chamada de *swap*, denotado por $sw(\pi_i, \pi_j)$ quando ocorre entre os elementos π_i e π_j.

Este capítulo estuda o problema de encontrar a menor sequência de OSCs necessárias para ordenar uma permutação circular, utilizando para isso um algoritmo exato em tempo polinomial desenvolvido para ordenar permutações lineares com sinais por reversões e transposições cíclicas super curtas.

4.1.2 Vetor Deslocamento Válido, Valor Cruzamento e Número de Cruzamentos

A maior parte deste capítulo lida com OSCs *cíclicas* em permutações lineares. Esta seção introduz uma estrutura chamada *Vetor Deslocamento Válido*, que nos permite computar o número mínimo de swaps cíclicos (i.e., 2-reversões ou 2-transposições) que coloca cada elemento em sua posição correta (esse número é chamado de *número de cruzamento*). É importante notar que essa estrutura não leva em consideração os sinais dos elementos por duas razões principais: (i) ela não faz distinção entre 2-reversões e 2-transposições (ambas são swaps) e (ii) ela não leva em consideração 1-reversões, uma vez que não são swaps por definição.

Dada uma sequência qualquer $\mathcal{S} = (s_1, s_2, \ldots, s_k)$ com k OSCs cíclicas para uma permutação linear π, e dado $1 \leq i \leq n$, denotamos por $R_{\mathcal{S}}(\pi_i)$ o número de OSCs cíclicas em $\mathcal{S}$ que movem π_i para a direita, e denotamos por $L_{\mathcal{S}}(\pi_i)$ o número de OSCs cíclicas em $\mathcal{S}$ que movem π_i para a esquerda.

Para qualquer $1 \leq i \leq n$, o *valor deslocamento* de π_i em relação a $\mathcal{S}$ é dado por $v_{\mathcal{S}}(\pi_i) = R_{\mathcal{S}}(\pi_i) - L_{\mathcal{S}}(\pi_i)$, e o *vetor deslocamento* de π associado a $\mathcal{S}$ é $V_{\mathcal{S}}(\pi) = (v_{\mathcal{S}}(\pi_1), v_{\mathcal{S}}(\pi_2), \ldots, v_{\mathcal{S}}(\pi_n))$.

Por exemplo, seja $\pi = (+4\ +2\ +3\ -1\ -5)$ uma permutação e seja $\mathcal{S} = (\rho(3,4), \tau(2,3,4), \tau(1,2,3), \tau(2,3,4), \rho(3,4), \rho(4,4), \rho(5,5))$ uma sequência de OSCs cíclicas (que neste caso ordena π). Temos que $\mathcal{S}$ resulta na seguinte sequência de swaps: $(sw(+3,-1), sw(+2,+1), sw(+4,+1), sw(+4,+2), sw(+4,-3))$. Note que $R_{\mathcal{S}}(-1) = 0$ e $L_{\mathcal{S}}(-1) = 3$, o que resulta em $v_{\mathcal{S}}(-1) = R_{\mathcal{S}}(-1) - L_{\mathcal{S}}(-1) = 0 - 3 = -3$. Computados todos os valores para $v_{\mathcal{S}}(\pi_i)$, temos que $V_{\mathcal{S}}(\pi) = (3, 0, 0, -3, 0)$.

Seja $X = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$ um vetor deslocamento e seja π uma permutação. Dizemos que X é um *Vetor Deslocamento Válido* (VDV) para π caso (i) $\sum_{i=1}^{n} x_i = 0$ e (ii) $|\pi_i| - x_i \equiv i \pmod{n}$ para $i \in [1..n]$. Em outras palavras, a condição (i) implica que para cada elemento de π se movendo uma posição para a direita, outro elemento deve ser movido uma posição para a esquerda e vice-versa, e a condição (ii) implica que todo elemento deve estar em sua posição correta no final. Assim, todo VDV para π está associado a pelo menos uma sequência $\mathcal{S}$ que ordena π, e toda sequência $\mathcal{S}$ que ordena π possui um vetor deslocamento associado que é VDV para π.

Por exemplo, dada a permutação $\pi = (+4\ +2\ +3\ -1\ -5)$, o vetor deslocamento $X = (3, 0, 0, -3, 0)$ é um VDV para π uma vez que $\sum_{i=1}^{n} x_i = 3 + 0 + 0 - 3 + 0 = 0$ e $|\pi_i| - x_i \equiv i \pmod{n}$ para $i \in [1..5]$.

Dado um VDV $X = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$ e dois inteiros distintos $i, j \in [1..n]$, sejam $r = i - j$ e $s = (i + x_i) - (j + x_j)$. Note que r mede o quão distante o elemento π_j está de π_i em π, ou seja, se $r > 0$ (resp. $r < 0$) então π_j está localizado em uma posição à esquerda (resp. direita) de π_i, enquanto s mede o quão distante o elemento π_j estará de π_i em suas posições finais, i.e., posições $(i + x_i)$ para π_i e $(j + x_j)$ para π_j. O *valor cruzamento* entre i e j, $1 \leq i, j \leq n$ e $i \neq j$, com respeito a X é definido da seguinte maneira:

$$c_{ij}(X) = \begin{cases} |\{k \in [r..s] : k \equiv 0 \pmod{n}\}|, \text{caso } r \leq s; \\ -|\{k \in [s..r] : k \equiv 0 \pmod{n}\}|, \text{caso } r > s. \end{cases} \tag{4.1}$$

Em outras palavras, $c_{ij}(X)$ representa o menor número de vezes que π_i e π_j são trocados em um swap (i.e., uma 2-reversão ou 2-transposição) em $\mathcal{S}_X$, uma sequência de ordenação tal que $V_{\mathcal{S}}(\pi) = X$. O sinal de $c_{ij}(X)$ é positivo (resp. negativo) se π_i está à esquerda (resp. direita) de π_j antes do primeiro swap entre estes dois elementos ocorrer. Por esta razão, $c_{ii}(X)$ é indefinido, e $c_{ij}(X) = -c_{ji}(X)$ para qualquer $1 \leq i, j \leq n$ com $i \neq j$. Além disso, $x_i = \sum_{j \neq i} c_{ij}(X)$ para $i \in [1..n]$.

Se X é um VDV associado a alguma sequência de ordenação para π, dizemos que X *induz* um swap α entre dois elementos π_i e π_j se $i \neq j$ e $c_{ij}(X) \neq 0$. Também dizemos que α é *induzido* por X.

Dado um VDV X qualquer para π, existe pelo menos uma sequência de ordenação $\mathcal{S}$ tal que $V_{\mathcal{S}}(\pi) = X$. Por exemplo, podemos utilizar uma sequência de 2-transposições para aplicar os swaps induzidos por X (que colocará todos os elementos em suas posições corretas) seguido de uma sequência de 1-reversões aplicada em cada elemento negativo.

O *número de cruzamentos* de um VDV X é definido como

$$cn(X) = \frac{1}{2} \sum_{i \neq j} |c_{ij}(X)|. \tag{4.2}$$

Informalmente, $cn(X)$ representa o menor número de swaps existentes na sequência de ordenação associada a X.

Considere novamente a permutação $\pi = (+4\ +2\ +3\ -1\ -5)$ e o VDV $X = (3, 0, 0, -3, 0)$. Dados $i = 2$ e $j = 4$, temos que $r = i - j = 2 - 4 = -2$ e $s = (i + x_i) - (j + x_j) = 2 - 1 = 1$, o que resulta no valor cruzamento $c_{24}(X) = |\{k \in [-2..1] : k \equiv 0 \pmod 5\}| = 1$. Para $i = 3$ e $j = 1$, temos que $r = i - j = 3 - 1 = 2$ e $s = (i + x_i) - (j + x_j) = 3 - 4 = -1$, resultando no valor cruzamento $c_{31}(X) = -|\{k \in [-1..2] : k \equiv 0 \pmod 5\}| = -1$. Após computar todos os valores cruzamento, obtemos o número de cruzamento $cn(X) = 5$.

Dado um VDV X para uma permutação π, denotamos por $d(\pi, X)$ o tamanho da menor sequência de OSCs necessárias para ordenar π aplicando swaps induzidos por X. Formalmente, $d(\pi, X) = cn(X) + y$, onde y é o número mínimo de 1-reversões dentre todas as sequências de ordenações associadas ao VDV X.

Dados dois inteiros $1 \leq i, j \leq n$ distintos, definimos a transformação $T_{i,j} : \mathbb{Z}^n \to \mathbb{Z}^n$ como uma transformação que recebe o VDV $X \in \mathbb{Z}^n$ e cria o VDV X' com $x'_k = x_k$, para $k \notin \{i, j\}$, $x'_i = x_i - n$ e $x'_j = x_j + n$.

Quando tal transformação é aplicada, cada valor cruzamento do tipo $c_{ib}(X')$ e $c_{aj}(X')$ é uma unidade menor que $c_{ib}(X)$ e $c_{aj}(X)$, respectivamente, e cada valor cruzamento do tipo $c_{ai}(X')$ e $c_{jb}(X')$ é uma unidade maior que $c_{ai}(X)$ e $c_{jb}(X)$, respectivamente, com $a, b \in [1..n]$, $a \notin \{i, j\}$ e $b \notin \{i, j\}$. Além disso, $c_{ji}(X')$ (resp. $c_{ij}(X')$) é duas unidades maior (resp. menor) que $c_{ji}(X)$ (resp. $c_{ij}(X)$).

A seguinte propriedade foi provada por Jerrum [33]. É importante ressaltar que o autor escreveu $cn(X') = cn(X) + 4(n - x_i + x_j)$ erroneamente, o que foi corrigido mais tarde por Galvão e coautores [26].

Propriedade 1. *Seja $X \in \mathbb{Z}^n$ um VDV para π, e seja $X' = T_{i,j}(X)$. Temos que $cn(X') = cn(X) + 2(n - x_i + x_j)$.*

Uma transformação $T_{i,j}(X) = X'$ é dita uma *contração* (resp. *contração estrita*) se,

e somente se, $x_i - x_j \geq n$ (resp. $x_i - x_j > n$), o que implica, pela Propriedade 1, que $cn(X') \leq cn(X)$ (resp. $cn(X') < cn(X)$). Se um VDV X não admite uma transformação de contração estrita, temos que $x_i - x_j \leq n$ para qualquer $1 \leq i \neq j \leq n$ e, desta forma, para qualquer VDV X', temos que $cn(X') \geq cn(X)$.

Por exemplo, dado o VDV $X = (3, 0, 0, -3, 0)$ para $\pi = (+4\ +2\ +3\ -1\ -5)$, podemos obter o vetor $X' = T_{1,4}(X)$ com $x'_1 = x_1 - n = 3 - 5 = -2$ e $x'_4 = x_4 + n = -3 + 5 = 2$, o que resulta em $X' = (-2, 0, 0, 2, 0)$. Observe que existe uma sequência de ordenação $\mathcal{S}'$ para π tal que $V_{\mathcal{S}'}(\pi) = X'$. Pela Propriedade 1, $cn(X') = cn(X) + 2(n - x_1 + x_4) = 5 - 2 = 3$.

4.1.3 Grafo de Permutação Cíclica

Dado que um VDV não leva em consideração os sinais dos elementos, iremos introduzir uma nova estrutura chamada de *Grafo de Permutação Cíclica*. Este grafo é construído com base em VDVs e nos ajudará a determinar o número mínimo de OSCs que ordena uma permutação (agora levando em consideração os sinais dos elementos e também 1-reversões).

Dado um VDV X para uma permutação π, nós definimos o *Grafo de Permutação Cíclica* de X e π como o grafo não orientado $G^X_\pi = (V, E)$, com $V = \{\pi_1, \pi_2, ..., \pi_n\}$ e $E = \{\{\pi_i, \pi_j\} : c_{ij}(X) > 0\}$. Além disso, associamos pesos às arestas de $E(G^X_\pi)$ da seguinte forma: se $e = \{\pi_i, \pi_j\}$ e $e \in E(G^X_\pi)$, então $w(e) = c_{ij}(X)$.

Perceba que, por construção, temos que $\sum_{e \in E(G^X_\pi)} w(e) = cn(X)$. Se toda aresta $e \in E(G^X_\pi)$ satisfaz $w(e) = 1$, então para qualquer $i \in [1..n]$ o vértice π_i tem grau pelo menos $|x_i|$ arestas (dado que $x_i = \sum_{j=1}^{n} c_{ij}(X)$) e, assim, a componente conexa em que π_i está possui pelo menos $|x_i| + 1$ vértices.

Denotamos por $cc(G^X_\pi)$ o número de *componentes conexas* de G^X_π (ou seja, o número de subgrafos maximais conexos de G^X_π). Além disso, uma componente conexa de G^X_π é dita *ímpar* se ela possui um número ímpar de vértices π_i tal que $\pi_i < 0$, e é dita *par* caso contrário. O número de componentes conexas ímpares de G^X_π é denotado por $cc^-(G^X_\pi)$.

Seja π uma permutação, $X = (x_1, x_2, ..., x_n)$ um VDV para π, e α uma OSC induzida por X (i.e., α é uma 2-reversão ou uma 2-transposição) aplicada sobre dois elementos adjacentes π_i e $\pi_{i \pmod n + 1}$ de π. O VDV resultante X' para $\pi' = \pi \cdot \alpha$ é tal que $x'_k = x_k$ para todo $k \notin \{i, i \pmod n + 1\}$, $x'_i = x_{i \pmod n + 1} + 1$ e $x'_{i \pmod n + 1} = x_i - 1$. Além disso, $cn(X') = cn(X) - 1$, e o grafo de permutação cíclica $G^{X'}_{\pi'}$ pode ser obtido a partir de G^X_π reduzindo o peso da aresta entre os vértices π_i e $\pi_{i \pmod n + 1}$ em uma unidade (ou removendo-a, caso seu peso em G^X_π seja 1).

Por exemplo, utilizando novamente $\pi = (+4\ +2\ +3\ -1\ -5)$, $X = (3, 0, 0, -3, 0)$ e $X' = T_{1,4}(X) = (-2, 0, 0, 2, 0)$, podemos aplicar a 2-reversão $\rho(4, 5)$ induzida por X' para obter a permutação $\pi' = \pi \cdot \rho(4, 5) = (+4\ +2\ +3\ +5\ +1)$ e o VDV $X'' = (-2, 0, 0, 1, 1)$.

Os grafos de permutação cíclica correspondentes a G^X_π, $G^{X'}_\pi$ e $G^{X''}_{\pi'}$ são ilustrados na Figura 4.1. Na Figura 4.1(a), temos $cn(X) = 5$ e $cc(G^X_\pi) = cc^-(G^X_\pi) = 2$; na Figura 4.1(b) temos $cn(X') = 3$, $cc(G^{X'}_\pi) = 3$ e $cc^-(G^{X'}_\pi) = 0$; na Figura 4.1(c) temos $cn(X'') = 2$, $cc(G^{X''}_{\pi'}) = 3$ e $cc^-(G^{X''}_{\pi'}) = 0$.

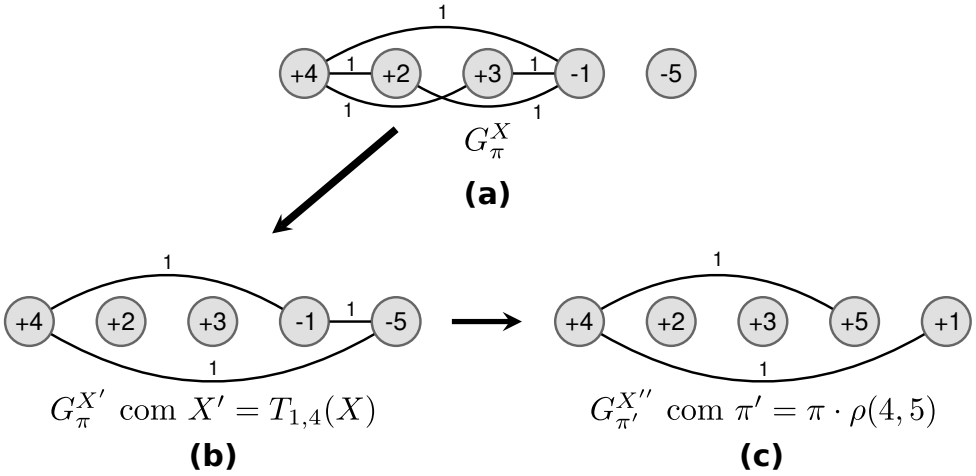


Figura 4.1: Em (a) temos o grafo de permutação cíclica G_π^X para a permutação $\pi = (+4\ +2\ +3\ -1\ -5)$ e o VDV $X = (3, 0, 0, -3, 0)$, onde $cn(X) = 5$, $cc(G_\pi^X) = 2$ e $cc^-(G_\pi^X) = 2$. Em (b) temos o grafo de permutação cíclica $G_\pi^{X'}$ para o VDV $X' = T_{1,4}(X) = (-2, 0, 0, 2, 0)$. Pela Propriedade 1, temos que $cn(X') = cn(X)+2(n-x_1+x_4) = 5 - 2 = 3$ (lembre-se de que $cn(X')$ também pode ser obtido pela soma dos pesos das arestas do grafo $G_\pi^{X'}$), $cc(G_\pi^{X'}) = 3$ e $cc^-(G_\pi^{X'}) = 0$. Olhando para G_π^X e $G_\pi^{X'}$, podemos notar que a 2-reversão $\rho(4, 5)$ é induzida por X', mas não por X; em (c) temos o grafo de permutação cíclica $G_{\pi'}^{X''}$ para a permutação $\pi' = \pi \cdot \rho(4, 5) = (+4\ +2\ +3\ +5\ +1)$ e o VDV $X'' = (-2, 0, 0, 1, 1)$, onde $cn(X'') = cn(X') - 1 = 2$, $cc(G_{\pi'}^{X''}) = 3$ e $cc^-(G_{\pi'}^{X''}) = 0$.

4.2 Resultados Relacionados

Nesta seção, indicamos resultados relacionados em problemas de Ordenação de Permutações por OSCs e OSCs cíclicas.

4.2.1 Ordenação de Permutações por OSCs

Knuth [37, p. 108] mostrou que o problema de Ordenação de Permutações sem Sinais por OSCs admite algoritmo exato em tempo polinomial e que a distância de ordenação é dada por $d(\pi) = inv(\pi)$. Por exemplo, utilizando a permutação sem sinais $\pi = (6\ 4\ 2\ 3\ 1\ 5\ 7)$, temos que $inv(\pi) = |\{(6, 4), (6, 2), (6, 3), (6, 1), (6, 5), (4, 2), (4, 3), (4, 1), (2, 1), (3, 1)\}| = 10$, o que resulta em $d(\pi) = 10$.

O número de inversões pode ser utilizado como um limitante inferior para o problema de Ordenação de Permutações com Sinais por OSCs: para qualquer permutação com sinais π, temos que $d(\pi) \geq inv(\pi)$. Galvão e coautores [27] provaram que o problema de Ordenação de Permutações com Sinais por OSCs admite algoritmo exato em tempo polinomial.

O *Grafo de Inversões* [52] de uma permutação com sinais π, denotado por $IG(\pi)$, é tal que $V(IG(\pi))$ é formado pelos elementos de π, e $E(IG(\pi))$ é formado pelos pares de inversões em π. Uma componente conexa em $IG(\pi)$ é dita *ímpar* se ela possui um número ímpar de elementos negativos (vértices), e é dita *par*, caso contrário. Os autores [27] mostraram que existe uma sequência de ordenação de tamanho mínimo para este problema que utiliza $inv(\pi)$ swaps e k 1-reversões, onde k é o número de componentes ímpares de

$IG(\pi)$.

Por exemplo, dada a permutação com sinal $\pi' = (-6\ +4\ +2\ -3\ +1\ +5\ -7)$, temos que $IG(\pi')$ possui duas componentes: uma componente ímpar com o elemento -7 apenas, e uma componente par com os elementos restantes, o que resulta em $d(\pi') = 10 + 1 = 11$.

4.2.2 Ordenação de Permutações por OSCs Cíclicas

Jerrum [33] mostrou que o problema de Ordenação de Permutações sem Sinais por OSCs Cíclicas admite algoritmo exato em tempo polinomial, sendo que a distância de ordenação de uma permutação sem sinal π nesta versão é dada por $d(\pi) = \min\{cn(X) : X \text{ é um VDV para } \pi\}$.

Dada a permutação $\pi = (6\ 4\ 2\ 3\ 1\ 5\ 7)$, temos que $X = (-2, 2, -1, -1, 3, -1, 0)$ é um VDV para π, e $cn(X) = 6$. Ademais, dado o fato de que X não admite uma transformação de contração estrita, qualquer outro VDV X' para π possui $cn(X') \geq cn(X)$, então temos que $d(\pi) = 6$. Note que a distância de ordenação para π diminui de 10 para 6 quando OSCs cíclicas são permitidas.

Em 2016, Galvão e coautores [26] provaram que o problema de Ordenação de Permutações com Sinais por Reversões Super Curtas Cíclicas também admite algoritmo exato em tempo polinomial. Dada uma permutação com sinais π e um VDV X, os autores definem dois conjuntos: $npar(X, \pi)$ é o conjunto de elementos de π tal que $|x_i|$ é par e $\pi_i < 0$, e $pímpar(X, \pi)$ é o conjunto de elementos de π tal que $|x_i|$ é ímpar e $\pi_i > 0$. De modo similar ao problema de Ordenação de Permutações com Sinais por Reversões Super Curtas, os autores provaram que dado um VDV X para π tal que X possui valor mínimo de número de cruzamentos dentre todos os VDVs para π, a distância de ordenação de π é exatamente $cn(X) + k$, onde k é o número de elementos em $npar(X, \pi) \cup pímpar(X, \pi)$.

Dada novamente a permutação com sinais $\pi' = (-6\ +4\ +2\ -3\ +1\ +5\ -7)$, temos que $X = (-2, 2, -1, -1, 3, -1, 0)$ é um VDV para π', e qualquer VDV X' para π' é tal que $cn(X') \geq cn(X)$. Além disso, $cn(X) = 6$, $npar(X, \pi') = \{-6, -7\}$ e $pímpar(X, \pi') = \{+1, +2, +5\}$, o que resulta em $d(\pi') = 6 + 5 = 11$. Comparado com a versão que não permite operações cíclicas, mas permite o uso de transposições super curtas, a distância da permutação π' diminui de 13 para 11.

Para o problema de Ordenação de Permutações Lineares com Sinais por OSCs cíclicas, um limitante inferior pode ser obtido pela versão sem sinais do problema: $d(\pi) \geq \min\{cn(X) : X \text{ é um VDV para } \pi\}$. Inspirado pelo $IG(\pi)$, definimos na Seção 4.1.3 o grafo de permutação cíclica, no qual as arestas são criadas de acordo com os valores de cruzamento de um VDV X. Apesar destes grafos serem diferentes entre si, a classificação entre componentes pares e ímpares é a mesma e será útil mais tarde neste capítulo.

Note que em todos os problemas anteriores desta seção a distância de ordenação está diretamente associada ou ao número mínimo de inversões ou ao menor valor de número de cruzamentos. O que torna o problema de Ordenação de Permutações Lineares por OSCs Cíclicas não trivial é que, ao contrário dos outros problemas, uma sequência de ordenação de tamanho mínimo não estará necessariamente associada a um VDV cujo número de cruzamentos é mínimo (veja um exemplo na Figura 4.5).

4.3 Ordenação de Permutações Lineares com Sinais por OSCs Cíclicas vs. Ordenação de Permutações Circulares com Sinais por OSCs

Note que, apesar dos problemas de Ordenação de Permutações Lineares com Sinais por OSCs Cíclicas e de Ordenação de Permutações Circulares com Sinais por OSCs serem diferentes, podemos utilizar o primeiro para solucionar o segundo. Como um exemplo, a permutação com sinais $\pi = (+5\ +4\ -2\ -1\ +3)$ possui distância de ordenação 8 considerando o modelo que permite OSCs (existem 8 inversões e seu grafo de inversões não possui componentes ímpares), e possui distância de ordenação 4 considerando o modelo que permite OSCs cíclicas (dado que $cn(X) = 4$ para o VDV $X = (-1, 2, -1, 2, -2)$, X não admite uma transformação de contração estrita e é possível ordenar π utilizando apenas swaps induzidos por X).

Entretanto, se π é uma permutação circular, então $\pi' = (-2\ -1\ +3\ +5\ +4)$ também é uma representação linear de π, dado que respeita todas as adjacências entre os elementos. Esta representação linear π' possui distância de ordenação igual a 2 tanto para o modelo que permite OSCs (existem 2 inversões e seu grafo de inversões não possui componentes ímpares) quanto para o modelo que permite OSCs cíclicas (dado que $cn(X) = 2$ para o VDV $X' = (1, -1, 0, 1, -1)$, X' não admite uma transformação de contração estrita e é possível ordenar π' utilizando apenas swaps induzidos por X').

Além disso, π' é, de fato, a representação linear da permutação circular π com o menor valor de distância de ordenação por OSCs cíclicas, então segue que a distância de ordenação da permutação circular π é 2. Uma explicação mais detalhada de como utilizar o modelo linear para solucionar o modelo circular é dada ao final da Seção 4.4.

4.4 A Ordenação de Permutações com Sinais por OSCs Cíclicas

Nesta seção, provamos que a Ordenação de Permutações Lineares com Sinais por OSCs Cíclicas admite algoritmo exato em tempo polinomial (Teorema 3), e, consequentemente, que a Ordenação de Permutações Circulares com Sinais por OSCs também admite algoritmo exato em tempo polinomial (Teorema 4). Estes resultados são obtidos estudando o problema de Ordenação de Permutações Lineares com Sinais por OSCs Cíclicas, baseando-se no grafo de permutação cíclica introduzido na Seção 4.1.3.

4.4.1 Propriedades dos VDVs

Antes de fornecer uma série de lemas que resultam no algoritmo desenvolvido, vamos mostrar três propriedades que serão fortemente utilizadas nesta seção.

A Propriedade 2 mostra que se um VDV X possui um valor deslocamento x_i cujo valor absoluto é maior ou igual a n, então existe um valor cruzamento $c_{ij}(X)$ maior que um (em valor absoluto). A Propriedade 3 mostra que se um VDV X possui um valor

cruzamento $c_{ij}(X)$ maior que zero, então os elementos π_k para $k \in [i..j]$, devem estar na mesma componente conexa no grafo de permutação cíclica correspondente. Por fim, a Propriedade 4 é uma extensão da Propriedade 3 que mostra que se para um VDV X existe um valor cruzamento $c_{ij}(X)$ maior que um em valor absoluto, então todos os elementos estão em uma única componente conexa no grafo de permutação cíclica correspondente.

Propriedade 2. *Seja $X = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$ um VDV para π. Se existe $1 \leq i \leq n$ tal que $|x_i| \geq n$, então existe j, com $1 \leq j \leq n$ e $j \neq i$, tal que $|c_{ij}(X)| > 1$.*

Demonstração. Seja $X \in \mathbb{Z}^n$ um VDV para π, e suponha que existe $1 \leq i \leq n$ tal que $|x_i| \geq n$. Sabemos, por definição, que $x_i = \sum_{j \neq i} c_{ij}(X)$. Como existem $n-1$ valores cruzamento na forma $c_{ij}(X)$ (um para cada $j \neq i$), isto necessariamente implica que $|c_{ij}(X)| > 1$ para algum $1 \leq j \neq i \leq n$. □

Propriedade 3. *Seja $X = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$ um VDV para π. Se $c_{ij}(X) > 0$ (resp. $c_{ij}(X) < 0$) para algum $1 \leq i \neq j \leq n$, então para qualquer $k \in [i{+}1..j{-}1]$ (resp. $k \in [j{+}1..i{-}1]$) temos $|c_{ik}(X)| + |c_{jk}(X)| \neq 0$ e $\{\{\pi_i, \pi_k\}, \{\pi_j, \pi_k\}\} \cap E(G_\pi^X) \neq \emptyset$.*

Demonstração. Seja X um VDV para π tal que para dois elementos π_i e π_j, $1 \leq i \neq j \leq n$, $c_{ij}(X) \neq 0$. Como $c_{ij}(X) = -c_{ji}(X)$, vamos supor sem perda de generalidade que $c_{ij}(X) = \gamma$ com $\gamma \geq 1$. Seja $r_1 = i - j$ e $s_1 = (i + x_i) - (j + x_j)$. Como $c_{ij}(X)$ é um número positivo, temos, por definição de valor cruzamento, que $r_1 \leq s_1$.

Vamos supor primeiramente que $i < j$. Como $r_1 = i - j < 0$ e $r_1 \leq s_1$, então $s_1 > \gamma - 1$, pois caso contrário teríamos $c_{ij}(X) < \gamma$. Temos assim que $x_i > x_j + j - i + \gamma - 1$. Suponha que exista um elemento π_k com $i < k < j$ tal que $c_{ik}(X) = c_{jk}(X) = 0$. Para $c_{ik}(X)$, temos que $r_2 = i - k < 0$, então $s_2 = (i + x_i) - (k + x_k) < 0$, pois caso contrário teríamos $c_{ik}(X) \neq 0$. Segue então que $x_k > x_i + i - k$, e, como $x_i > x_j + j - i + \gamma - 1$, temos que $x_k > x_j + j - k + \gamma - 1$. Para $c_{jk}(X)$, temos que $r_3 = j - k > 0$, então $s_3 = (j + x_j) - (k + x_k) > 0$, pois caso contrário teríamos que $c_{jk}(X) \neq 0$. Segue então que $x_k < x_j + j - k$, o que é uma contradição com o fato de que $x_k > x_j + j - k + \gamma - 1$ e $\gamma > 0$, e podemos concluir que $|c_{ik}(X)| + |c_{jk}(X)| \neq 0$.

Vamos supor agora que $i > j$. Neste caso, podemos dividir o intervalo que vai de i até j em $[i{+}1..n] \cup [1..j{-}1]$. Como $r_1 = i - j > 0$ e $r_1 \leq s_1$, temos que $s_1 \geq \gamma n$, pois caso contrário teríamos $c_{ij}(X) < \gamma$. Segue então que $x_i \geq x_j + j - i + \gamma n$. Suponha que exista um elemento π_k com $k \in [i{+}1..n] \cup [1..j{-}1]$ tal que $c_{ik}(X) = c_{jk}(X) = 0$. Vamos considerar dois casos: quando $k \in [1..j{-}1]$ (ou seja, $k < j$) e quando $k \in [i{+}1..n]$ (ou seja, $k > i$):

1. $k < \min\{i, j\}$: neste caso temos tanto para $c_{ik}(X)$ quanto para $c_{jk}(X)$ que $r_2 = i - k > 0$ e $r_3 = j - k > 0$, então $0 < s_2 < n$ e $0 < s_3 < n$, ou então teríamos que $|c_{ik}(X)| + |c_{jk}(X)| \neq 0$. Para $c_{ik}(X)$ e $s_2 = (i + x_i) - (k + x_k) < n$, temos que $x_i + i - x_k - k < n$, o que implica em $x_k > x_i + i - k - n$. Como $x_i \geq x_j + j - i + \gamma n$, temos que $x_k > x_j + j - k + (\gamma - 1)n$. Já para $c_{jk}(X)$ e $s_3 = (j + x_j) - (k + x_k) > 0$, temos que $x_j + j - x_k - k > 0$, o que resulta em $x_k < x_j + j - k$, uma contradição com o fato de que $x_k > x_j + j - k + (\gamma - 1)n$ e $(\gamma - 1) \geq 0$.

2. $k > \max\{i, j\}$: neste caso temos tanto para $c_{ik}(X)$ quanto para $c_{jk}(X)$ que $r_2 = i - k < 0$ e $r_3 = j - k < 0$, então $-n < s_2 < 0$ e $-n < s_3 < 0$, pois, caso contrário, teríamos que $|c_{ik}(X)|+|c_{jk}(X)| \neq 0$. Para $c_{ik}(X)$ e $s_2 = (i+x_i)-(k+x_k) < 0$, temos que $x_i + i - x_k - k < 0$, o que resulta em $x_k > x_i + i - k$. Como $x_i \geq x_j + j - i + \gamma n$, temos que $x_k > x_j + j - k + \gamma n$. Para $c_{jk}(X)$ e $s_3 = (j + x_j) - (k + x_k) > -n$, temos que $x_j + j - x_k - k > -n$, o que resulta em $x_k < x_j + j - k + n$, uma contradição com o fato de que $x_k > x_j + j - k + \gamma n$ e $\gamma \geq 1$.

Em todos os casos descritos acima, temos que $|c_{ik}(X)| + |c_{jk}(X)| \neq 0$, o que implica em $\{\{\pi_i, \pi_k\}, \{\pi_j, \pi_k\}\} \cap E(G_\pi^X) \neq \emptyset$. □

Propriedade 4. *Seja $X = (x_1, x_2, ..., x_n) \in \mathbb{Z}^n$ um VDV para π. Se existe um $c_{ij}(X)$ tal que $|c_{ij}(X)| > 1$, então $cc(G_\pi^X) = 1$.*

Demonstração. Seja X um VDV para uma permutação π tal que $|c_{ij}(X)| > 1$ para algum $1 \leq i \neq j \leq n$. Vamos supor, sem perda de generalidade, que $c_{ij}(X) > 1$ (dado que, por definição, $c_{ij}(X) = -c_{ji}(X)$), e, para maior legibilidade, seja $c_{ij}(X) = \gamma$. Temos que $r_1 = i - j$ e $s_1 = (i + x_i) - (j + x_j)$. Como $c_{ij}(X)$ é positivo, temos, por definição de valor cruzamento, que $r_1 \leq s_1$. Vamos analisar então os dois casos possíveis: ou $i < j$ ou $i > j$.

Suponha que $i < j$. Neste caso $r_1 = i - j$ é tal que $-n < r_1 < 0$. Como $r_1 \leq s_1$, temos que $s_1 = (i + x_i) - (j + x_j) \geq (\gamma - 1)n$ (usamos $(\gamma - 1)$ pois $\gamma = |\{k \in [r_1..s_1] : k \equiv 0 \pmod n\}| = |\{k \in [r_1..0] : k \equiv 0 \pmod n\}| + |\{k \in [1..s_1] : k \equiv 0 \pmod n\}| = 1 + |\{k \in [1..s_1] : k \equiv 0 \pmod n\}|$, então $(\gamma - 1) = |\{k \in [1..s_1] : k \equiv 0 \pmod n\}|$ e $s_1 \geq (\gamma - 1)n$), pois caso contrário teríamos $c_{ij}(X) < \gamma$. Segue então que $x_i \geq (\gamma - 1)n + x_j + j - i$. Suponha agora que exista um elemento π_k tal que $c_{ik}(X) = c_{jk}(X) = 0$. Pela Propriedade 3, sabemos que $k \notin (i..j)$. Assim, vamos analisar dois casos:

1. $k < i < j$: neste caso, $r_2 = i - k > 0$ e $r_3 = j - k > 0$, então obrigatoriamente temos que $0 < s_2 < n$ e $0 < s_3 < n$, com $s_2 = (i + x_i) - (k + x_k)$ e $s_3 = (j + x_j) - (k + x_k)$. Para $s_2 < n$, temos que $x_k > x_i + i - k - n$, e como $x_i \geq (\gamma - 1)n + x_j + j - i$ temos que $x_k > (\gamma - 2)n + x_j + j - k$. Para $s_3 > 0$, temos que $x_k < x_j + j - k$, mas isto não é possível dado que $(\gamma - 2)n \geq 0$.

2. $i < j < k$: neste caso, $r_2 = i - k < 0$ e $r_3 = j - k < 0$, então $-n < s_2 < 0$ e $-n < s_3 < 0$, com $s_2 = (i + x_i) - (k + x_k)$ e $s_3 = (j + x_j) - (k + x_k)$. Para $s_2 < 0$, temos que $x_k > x_i + i - k$, e como $x_i \geq (\gamma - 1)n + x_j + j - i$, temos que $x_k > (\gamma - 1)n + x_j + j - k$. Para $s_3 > -n$, temos que $x_k < x_j + j - k + n$, mas isto não seria possível dado que $(\gamma - 1)n \geq n$.

Vamos considerar agora que $i > j$. Neste caso, $r_1 = i - j$ é tal que $0 < r_1 < n$. Como $r_1 \leq s_1$, temos que $s_1 = (i + x_i) - (j + x_j) \geq \gamma n$, pois caso contrário teríamos $c_{ij}(X) < \gamma$. Segue então que $x_i \geq \gamma n + x_j + j - i$. Suponha agora que exista um elemento π_k tal que $c_{ik}(X) = c_{jk}(X) = 0$. Pela Propriedade 3, sabemos que isto não é possível se $k < \min\{i, j\}$ ou $k > \max\{i, j\}$. Assim, a única possibilidade é quando $j < k < i$: neste caso $r_2 = i - k > 0$ e $r_3 = j - k < 0$, então $0 < s_2 < n$ e $-n < s_3 < 0$, com $s_2 = (i+x_i)-(k+x_k)$ e $s_3 = (j+x_j)-(k+x_k)$. Para $s_2 < n$, temos que $x_k > x_i + i - k - n$,

e, como $x_i \geq \gamma n + x_j + j - i$, temos que $x_k > (\gamma - 1)n + x_j + j - k$. Para $s_3 > -n$, temos que $x_k < x_j + j - k + n$, mas isto não é possível dado que $(\gamma - 1)n \geq n$.

Segue como resultado que se $c_{ij}(X) > 1$, então, para qualquer elemento π_k com $k \notin \{i, j\}$, temos que $|c_{ik}(X)| + |c_{jk}(X)| \neq 0$, e desta forma $\{\{\pi_i, \pi_k\}, \{\pi_j, \pi_k\}\} \cap E(G_\pi^X) \neq \emptyset$. Como $c_{ij}(X) > 1$, temos também que $\{\pi_i, \pi_j\} \in G_\pi^X$, e, assim, $cc(G_\pi^X) = 1$. □

4.4.2 OSCs e os Grafos de Permutação Cíclica

Esta seção apresenta cinco lemas que relacionam OSCs com grafos de permutação cíclica. No Lema 5 (resp. Lema 6), analisamos a diferença no número de componentes ímpares de um grafo de permutação cíclica quando aplicamos uma 1-reversão (resp. um swap, i.e., uma 2-reversão ou uma 2-transposição) à permutação π. No Lema 7, mostramos que sempre é possível aplicar um swap induzido por um VDV X sem aumentar o número de componentes ímpares no grafo de permutação cíclica resultante. Vamos denotar por $|\mathcal{S}|$ o tamanho de uma sequência de ordenação $\mathcal{S}$ para π (ou seja, o número de operações em $\mathcal{S}$). No Lema 8 (resp. Lema 9), vamos mostrar que se uma sequência de ordenação $\mathcal{S}$ possui uma OSC que aumenta o número de componentes ímpares (resp. o peso de uma aresta) no grafo de permutação cíclica correspondente, então existe uma outra sequência de ordenação $\mathcal{S}'$ para π com $|\mathcal{S}'| \leq |\mathcal{S}|$ tal que $\mathcal{S}'$ não possui nenhuma OSC com estas características.

Lema 5. *Sejam X e X' dois vetores deslocamento em $\mathbb{Z}^n$ tal que X é um VDV para π e X' é um VDV para $\pi' = \pi \cdot \alpha$, onde α é uma OSC cíclica. Se α é uma 1-reversão, então $X' = X$, $cn(X') = cn(X)$ e $\Delta cc^- = cc^-(G_{\pi'}^{X'}) - cc^-(G_\pi^X) \in \{-1, 1\}$.*

Demonstração. Se α é uma 1-reversão, temos que $|\pi'_i| = |\pi_i|$ para todo $1 \leq i \leq n$, o que implica em $X' = X$, $cn(X') = cn(X)$ e $cc(G_{\pi'}^{X'}) = cc(G_\pi^X)$. Se a componente conexa afetada por α em G_π^X é par (resp. ímpar), então ela será ímpar (resp. par) em $G_{\pi'}^{X'}$, o que implica em $\Delta cc^- = cc^-(G_{\pi'}^{X'}) - cc^-(G_\pi^X) \in \{-1, 1\}$. □

Lema 6. *Sejam X e X' dois vetores deslocamento em $\mathbb{Z}^n$ tal que X é um VDV para π e X' é um VDV para $\pi' = \pi \cdot \alpha$, onde α é uma OSC cíclica. Se α é uma 2-reversão ou uma 2-transposição induzida por X em π, então $cn(X') = cn(X) - 1$, e $cc(G_{\pi'}^{X'}) = cc(G_\pi^X)$ e $\Delta cc^- = cc^-(G_{\pi'}^{X'}) - cc^-(G_\pi^X) = 0$, ou $cc(G_{\pi'}^{X'}) > cc(G_\pi^X)$ e $\Delta cc^- \in \{0, 2\}$.*

Demonstração. Como α é uma OSC cíclica induzida por X, os valores cruzamento entre os elementos π_i e $\pi_{i \pmod n + 1}$ afetados por α são não nulos, o que implica que a aresta $e = \{\pi_i, \pi_{i \pmod n + 1}\}$ existe. Em $G_{\pi'}^{X'}$, por definição, ou a aresta e é removida, ou seu peso é decrescido em uma unidade, logo $cc(G_{\pi'}^{X'}) \geq cc(G_\pi^X)$.

Vamos supor primeiramente que $cc(G_{\pi'}^{X'}) = cc(G_\pi^X)$. Isto significa que a OSC aplicada em π não modifica a componente conexa C_α afetada pela OSC em G_π^X. Se esta OSC é uma 2-reversão (resp. uma 2-transposição), então dois (resp. zero) elementos da componente C_α trocaram de sinais. Em ambos os casos temos então que $\Delta cc^- = cc^-(G_{\pi'}^{X'}) - cc^-(G_\pi^X) = 0$.

Vamos supor agora que $cc(G_{\pi'}^{X'}) > cc(G_\pi^X)$. Isto significa que C_α foi dividida em duas componentes conexas C_1 e C_2, o que resulta em $cc(G_{\pi'}^{X'}) = cc(G_\pi^X) + 1$. Se a OSC é uma

2-transposição, nenhum elemento de π trocou de sinal. Se a OSC é uma 2-reversão, dois elementos de π trocaram de sinais de tal modo que um está em C_1 e o outro está em C_2. Nos dois casos, se C_α é uma componente ímpar, então C_1 e C_2 possuem paridades distintas, e $\Delta cc^- = 0$; se C_α é par então C_1 e C_2 possuem a mesma paridade, e $\Delta cc^- \in \{0, 2\}$. □

Neste ponto, sabemos pelo Lema 5 que uma 1-reversão sempre aumenta ou diminui o número de componentes ímpares em uma unidade, e sabemos pelo Lema 6 que um swap induzido por X pode apenas manter ou aumentar em duas unidades o número de componentes ímpares do grafo de permutação cíclica.

Lema 7. *Seja $X \in \mathbb{Z}^n$ um VDV para π. Se $cn(X) > 0$, então sempre é possível encontrar uma OSC cíclica α induzida por X tal que X' é um VDV para $\pi' = \pi \cdot \alpha$ e $\Delta cc^- = cc^-(G_{\pi'}^{X'}) - cc^-(G_\pi^X) = 0$.*

Demonstração. Seja α um swap induzido por X (note de que α é uma 2-reversão ou uma 2-transposição, por definição), e seja $\pi' = \pi \cdot \alpha$. Dado que α é um swap induzido por X, aplicá-lo em π implica que o peso de uma aresta de G_π^X será obrigatoriamente decrescido em uma unidade em $G_{\pi'}^{X'}$, logo $cn(X') = cn(X) - 1$ e $X' \neq X$.

Se $cc(G_{\pi'}^{X'}) = cc(G_\pi^X)$, então pelo Lema 6, sabemos que $\Delta cc^- = 0$ e o resultado segue. Caso contrário, temos necessariamente que $cc(G_{\pi'}^{X'}) > cc(G_\pi^X)$. Como exibido na prova do Lema 6, se a componente C_α afetada por α em G_π^X for ímpar, sabemos que $\Delta cc^- = 0$ e o resultado segue. Suponha agora que a componente C_α afetada por α em G_π^X seja par.

Vamos analisar as duas componentes obtidas de C_α após aplicar α. Se ambas as componentes forem pares, então trivialmente $\Delta cc^- = 0$ e o resultado segue. Finalmente, caso ambas as componentes forem ímpares, então podemos trocar α por α', onde α' (i) afeta os mesmos elementos de π que α e (ii) é uma 2-transposição (resp. 2-reversão) se α é uma 2-reversão (resp. uma 2-transposição). Note que α' também é induzida por X, e aplicar α' também resulta em duas componentes conexas em $V(C_\alpha)$. Além disso, como 2-reversões alteram sinais e 2-transposições não, as duas componentes obtidas no novo grafo de permutação cíclica após a aplicação de α' em π são pares. Desta forma, $\Delta cc^- = 0$ e α' é a OSC desejada. □

Lema 8. *Seja $\mathcal{S}$ uma sequência de OSCs cíclicas que ordena uma permutação π, e seja $X \in \mathbb{Z}^n$ seu VDV associado. Se $\mathcal{S}$ é uma sequência de tamanho mínimo entre todas as sequências de ordenação associadas a X, então $\mathcal{S}$ não possui OSCs que aumentam o número de componentes ímpares.*

Demonstração. Aqui vamos provar que se uma sequência de ordenação $\mathcal{S}$ para π, associada a X, possui uma OSC cíclica que aumenta o número de componentes ímpares em algum momento em G_π^X, então sempre podemos encontrar uma sequência de ordenação alternativa $\mathcal{S}'$ para π, também associada a X, que não possui nenhuma OSC que aumenta o número de componentes ímpares tal que $|\mathcal{S}'| < |\mathcal{S}|$.

Note que a permutação identidade possui um grafo de permutação cíclica com n componentes conexas pares. Pelos lemas 5 e 6, temos que apenas 1-reversões podem diminuir o número de componentes ímpares. Assim, qualquer sequência $\mathcal{S}$ possui pelo menos $cc^-(G_\pi^X)$ 1-reversões. Suponha que $\mathcal{S}$ é uma sequência de tamanho mínimo que ordena π, e que $\mathcal{S}$ possui uma OSC α que aumenta o número de componentes ímpares.

Se α é uma 1-reversão, então ela é aplicada obrigatoriamente em uma componente conexa par. Assim, o número total de 1-reversões de $\mathcal{S}$ deve ser maior ou igual a $cc^-(G_\pi^X)+2$. Neste caso, seja $\mathcal{S}' = \mathcal{S} - \{\alpha, \alpha'\}$, onde α' é a 1-reversão aplicada ao componente ímpar gerado por α. Note que aplicamos a mesma sequência de swaps em $\mathcal{S}'$ e $\mathcal{S}$, logo ambas as sequências são induzidas por X.

Se α é uma 2-reversão (resp. uma 2-transposição), então, como mostrado na prova do Lemma 6, este swap é aplicado necessariamente em uma componente par C_α, gerando duas componentes ímpares. Assim, o número total de 1-reversões em $\mathcal{S}$ é maior ou igual a $cc^-(G_\pi^X) + 2$. Seja $\mathcal{S}'$ a sequência obtida de $\mathcal{S}$ trocando α por uma 2-transposição (resp. 2-reversão) que afeta os mesmos elementos que α, e removendo as duas 1-reversões aplicadas às componentes ímpares geradas por α. Como α foi "transformada" em uma 2-transposição (resp. uma 2-reversão), ela gera agora duas componentes pares a partir de C_α. Note que aplicamos a mesma sequência de swaps em $\mathcal{S}'$ e $\mathcal{S}$ (neste caso eles diferem apenas em qual tipo de swap mas sempre são aplicados nos mesmos pares de elementos), logo ambas as sequências são induzidas por X.

Nos dois casos acima, temos que a nova sequência $\mathcal{S}'$ também é uma sequência de ordenação para π, e possui tamanho $|\mathcal{S}'| = |\mathcal{S}| - 2$, uma contradição com o fato de que $\mathcal{S}$ é uma sequência de tamanho mínimo. Desta forma, temos que $\mathcal{S}$ não possui OSCs que aumentam o número de componentes ímpares. □

Lema 9. *Dada uma permutação π, seja $\mathcal{S}$ uma sequência de OSCs cíclicas que ordena π, e seja $X \in \mathbb{Z}^n$ seu VDV associado. Existe uma sequência $\mathcal{S}$ mínima que possui apenas OSCs cíclicas que não aumentam os pesos das arestas de G_π^X.*

Demonstração. Vamos provar aqui que se uma sequência de ordenação $\mathcal{S}$ para π, associada ao VDV X, possui uma OSC cíclica que aumenta o peso de uma aresta e de G_π^X em algum ponto, então podemos encontrar uma sequência de ordenação alternativa $\mathcal{S}'$ para π, também associada ao VDV X, que não possui OSC cíclicas que aumentam o peso de uma aresta, e é tal que $|\mathcal{S}'| \leq |\mathcal{S}|$.

Suponha, sem perda de generalidade, que uma OSC cíclica de $\mathcal{S}$ aumenta o peso de uma aresta e no grafo de permutação cíclica, e seja α a primeira OSC cíclica como tal. Note que como 1-reversões não modificam pesos de arestas no grafo de permutação cíclica, α é necessariamente uma 2-reversão ou uma 2-transposição. Note também que como este swap aumenta o peso de uma aresta, ele não é induzido por X. Seja π' a permutação resultante após aplicar todas as operações de $\mathcal{S}$ que precedem α e a operação α, e seja X' o vetor associado à sequência de operações de $\mathcal{S}$ existentes após a operação α.

Se α é aplicada sobre dois elementos de uma mesma componente, então $cc(G_{\pi'}^{X'}) = cc(G_\pi^X)$ e $cc^-(G_{\pi'}^{X'}) = cc^-(G_\pi^X)$. Caso contrário, α irá conectar duas componentes A e B, e $cc(G_{\pi'}^{X'}) = cc(G_\pi^X) - 1$. Se ambas as componentes são ímpares, então a componente gerada será par, logo $cc^-(G_{\pi'}^{X'}) = cc^-(G_\pi^X) - 2$, e caso contrário teremos $cc^-(G_{\pi'}^{X'}) = cc^-(G_\pi^X)$. Como aumentamos o peso da aresta e, temos que $cn(X') = cn(X) + 1$. Como resultado, $cn(X') + cc^-(G_{\pi'}^{X'}) \geq cn(X) + cc^-(G_\pi^X) - 1$.

Seja α' a operação que diminui o peso de e em algum momento da sequência de ordenação tal que $\pi''' = \pi'' \cdot \alpha'$, onde π'' é a permutação com todas as operações que

precedem α' em $\mathcal{S}$. Seja X'' (resp. X''') o vetor associado à sequência de operações de $\mathcal{S}$ existentes a partir da operação α' (resp. após a operação α').

Pelo Lema 7, temos que α' diminui o número de cruzamentos em uma unidade e mantém o número de componentes ímpares, assim, $cn(X''') + cc^-(G_{\pi'''}^{X'''}) = cn(X'') + cc^-(G_{\pi''}^{X''}) - 1$. Segue então que ambas as operações α e α' estão diminuindo a soma de número de cruzamentos e componentes ímpares em uma unidade no máximo.

Seja $\mathcal{S}' = \mathcal{S} - \{\alpha, \alpha'\}$. Se α conecta duas componentes ímpares A e B, adicionamos ao início de $\mathcal{S}'$ duas 1-reversões: uma aplicada a qualquer elemento $\pi_i \in A$, e outra a qualquer elemento $\pi_j \in B$. Como mostrado na prova do Lema 5, cada 1-reversão diminui o número de componentes ímpares em exatamente uma unidade, enquanto que não alteram o número de cruzamentos. Segue então que esta nova sequência construída não é maior que $\mathcal{S}$, ordena π, e usa apenas OSCs cíclicas que nunca aumentam o peso de arestas do grafo de permutação cíclica, logo todos os swaps aplicados são induzidos por X. □

4.4.3 Um Algoritmo Exato em Tempo Polinomial para a Ordenação de Permutações Circulares com Sinais por OSCs

Nesta seção, fornecemos primeiramente uma fórmula fechada para computar o tamanho de uma sequência de ordenação por OSCs cíclicas para permutações lineares com sinais baseado em seu VDV X associado. Em seguida, fornecemos um algoritmo exato em tempo polinomial para ordenar permutações circulares com sinais utilizando OSCs.

Lema 10. *Seja $\mathcal{S}$ uma sequência de OSC cíclicas de tamanho mínimo que ordena uma permutação linear com sinais π, e seja X seu VDV associado. Temos que $d(\pi) = cn(X) + cc^-(G_\pi^X)$.*

Demonstração. Vamos particionar $\mathcal{S}$ em duas sequências $\mathcal{S}_1$ e $\mathcal{S}_2$ tal que $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) contém todas as 1-reversões (resp. swaps) de $\mathcal{S}$. Além disso, como 1-reversões não modificam a ordem dos elementos na permutação, podemos assumir, sem perda de generalidade, que os swaps de $\mathcal{S}_2$ são aplicados primeiro. Vamos mostrar que $|\mathcal{S}_1| = cc^-(G_\pi^X)$. Para ver isto, suponha que aplicamos um swap α de $\mathcal{S}_2$ em π, obtendo a permutação π', e seja $\mathcal{S}' = \mathcal{S} - \{\alpha\}$, e X' seu VDV associado. Pelo Lema 7, sabemos que $cc^-(G_{\pi'}^{X'}) = cc^-(G_\pi^X)$. Além disso, pelo Lema 8, temos que o número de componentes ímpares nunca aumenta pela sequência $\mathcal{S}$ e, pelo Lema 5, o valor de $cc^-(G_\pi^X)$ pode ser reduzido apenas utilizando 1-reversões.

Note que a soma dos pesos das arestas de G_π^X é $cn(X)$, e, pelo Lema 6, aplicar qualquer OSC cíclica $\alpha \in \mathcal{S}_2$ ou aumenta ou diminui este valor em uma unidade, logo $|\mathcal{S}_2| \geq cn(X)$. Pelo Lema 9, podemos assumir que $\mathcal{S}_2$ não possui OSC cíclicas que aumentam os pesos das arestas, então seque que $|\mathcal{S}_2| = cn(X)$. □

O Lema 10 indica que o problema de ordenar uma permutação com sinais π utilizando OSCs cíclicas é equivalente ao seguinte problema de otimização: encontre um VDV $X \in \mathbb{Z}^n$ para π que minimiza o valor $cn(X) + cc^-(G_\pi^X)$.

Vamos provar agora que sempre podemos encontrar tal VDV em tempo polinomial. Primeiro, vamos introduzir o Lema 11, em que mostramos que se um VDV X possui um

grafo de permutação cíclica com apenas um componente, então qualquer VDV X' em que $cn(X') \geq cn(X)$ possui $d(\pi, X') \geq d(\pi, X)$. Depois, no Lema 12, vamos provar que qualquer VDV X^* com $d(\pi, X^*) = d(\pi)$ necessariamente pertence a um dos dois conjuntos que definiremos. Finalmente, mostramos no Teorema 3 que podemos encontrar um VDV X^* tal que $d(\pi, X^*) = d(\pi)$ em tempo polinomial.

Lema 11. *Sejam X e X' dois vetores deslocamento em $\mathbb{Z}^n$ tal que ambos sejam VDVs para uma permutação com sinais π e $cn(X') \geq cn(X)$. Se $cc(G_\pi^X) = 1$, então $d(\pi, X') \geq d(\pi, X)$.*

Demonstração. Sabemos pelo Lema 7 que podemos aplicar $cn(X)$ swaps induzidos por X em π mantendo sempre $cc^-(G_\pi^X)$ componentes ímpares. Utilizando os lemas 8 e 9, temos que $d(\pi, X) = cn(X) + cc^-(G_\pi^X)$. Com o mesmo argumento temos também que $d(\pi, X') = cn(X') + cc^-(G_\pi^{X'})$.

Como $cc(G_\pi^X) = 1$, então todos os elementos de π estão na mesma componente, e $cc^-(G_\pi^X) = 1$ (resp. $cc^-(G_\pi^X) = 0$) se existe um número ímpar (resp. par) de elementos negativos em π, e qualquer VDV X' para π é tal que $cc^-(G_\pi^{X'}) \geq cc^-(G_\pi^X)$ (ou seja, se π possui um número ímpar de elementos negativos então para qualquer VDV X' existe pelo menos um componente ímpar em $G_\pi^{X'}$).

Como $cn(X') \geq cn(X)$, temos então que $d(\pi, X') = cn(X') + cc^-(G_\pi^{X'}) \geq cn(X) + cc^-(G_\pi^X) = d(\pi, X)$. □

Vamos denotar por $cn(\pi) = \min\{cn(X) : \ X \ \text{é um VDV para } \pi\}$, ou seja, $cn(\pi)$ denota o menor valor de número de cruzamentos dentre todos os VDVs para π.

Lema 12. *Seja S o conjunto de todos os VDVs $X \in \mathbb{Z}^n$ para π tais que $cn(X) = cn(\pi)$, e seja S' o conjunto de todos os VDVs $X' \notin S$ para π tais que $X' = T_{i,j}(X)$, para algum $i, j \in [1..n]$, $X \in S$ e $i \neq j$. Existe um VDV $X^* \in S \cup S'$ para π tal que $d(\pi, X) = d(\pi)$.*

Demonstração. Note, primeiramente, que qualquer VDV $X'' \notin S$ admite uma transformação de contração estrita. Considere agora a sequência de transformações de contrações estritas aplicadas em X'' até que alcancemos um VDV $X \in S$. Vamos provar por contradição que sempre existe um VDV X' nesta sequência tal que $X' \in S \cup S'$ e $d(X', \pi) < d(X'', \pi)$, ou seja, todo VDV X'' que não pertence à $S \cup S'$ obrigatoriamente possui $d(\pi, X'') > d(\pi)$.

Se existe um VDV $X \in S$ para π tal que $cc(G_\pi^X) = 1$, então, pelo Lema 11, para qualquer $X' \notin S$ tal que $cn(X') \geq cn(X) + 1$, temos $d(\pi, X') \geq d(\pi, X) + 1$, e segue que o VDV X^* com $d(\pi, X^*) = d(\pi)$ é tal que $X^* \in S$.

Suponha agora que um VDV $X \in S$ para π é tal que $cc(G_\pi^X) \geq 2$. Como $X \in S$, X não admite uma transformação de contração estrita. Assim, para quaisquer dois valores $x_a, x_b \in X$ com a e b distintos, temos que $x_a - x_b \leq n$.

Seja $X' \in S'$ um VDV para π tal que $X' = T_{i,j}(X)$ para algum $1 \leq i \neq j \leq n$, e seja X'' um VDV para π tal que $X'' = T_{k,l}(X')$ para algum $1 \leq k \neq l \leq n$. Como podemos notar na Figura 4.2, se $k = j$ e $l = i$ então $X'' = X$ e $X'' \in S$. Se $k = j$ (resp. $l = i$), então $X'' = T_{i,l}(X)$ (resp. $X'' = T_{k,j}(X)$) e $X'' \in S'$.

Suponha agora que $X \in S$, $X' \in S'$, e $X'' = T_{k,l}(X')$ é tal que $X'' \notin S \cup S'$. Como $X'' \notin S \cup S'$, então $i \notin \{k, l\}$ e $j \notin \{k, l\}$, e temos que $x'_k = x_k$ e $x'_l = x_l$. Isto significa que,

como mostrado na Figura 4.2, X'' pode ser obtido a partir de quatro transformações entre VDVs de S': $T_{k,l}(X')$, $T_{k,j}(Y')$, $T_{i,j}(Z')$ e $T_{i,l}(W')$, tal que $X' = T_{i,j}(X)$, $Y' = T_{i,l}(X)$, $Z' = T_{k,l}(X)$ e $W' = T_{k,j}(X)$. Os VDVs X', Y', Z' e W' estão em S', e dizemos que eles são *adjacentes* a X''.

Como $X \in S$ e $X'' \notin S \cup S'$, temos que $x_i - x_j < n$, $x_i - x_l < n$, $x_k - x_j < n$ e $x_k - x_l < n$, pois, caso contrário, pelo menos um dos VDVs W', X', Y' e Z' seria gerado a partir de uma transformação de contração e estaria no conjunto S, e, como consequência, $X'' \in S'$. Segue então que $cn(X'') > cn(X') > cn(X)$.

Vamos supor que para qualquer VDV $V \in S'$ adjacente a X'' temos que $d(\pi, X'') < d(\pi, V)$. Usando o Lema 11, temos que os valores de $cc(G_\pi^{X'})$, $cc(G_\pi^{Y'})$, $cc(G_\pi^{Z'})$ e $cc(G_\pi^{W'})$ devem ser estritamente maiores que 1, pois, caso contrário, $d(\pi, X'') \geq d(\pi, V)$. Esta observação implica que os valores de $c_{ij}(X)$, $c_{il}(X)$, $c_{kj}(X)$ e $c_{kl}(X)$ são todos iguais a 1 (dado que, por definição de transformação, se $X' = T_{i,j}(X)$, então $c_{ij}(X') = c_{ij}(X) - 2$), pois, caso contrário, em pelo menos um dos quatro VDVs em S' adjacentes a X'' existe um valor cruzamento (em valor absoluto) maior que um e, pela Propriedade 4, o grafo de permutação cíclica correspondente teria apenas uma componente conexa.

Consequentemente, podemos concluir que os quatro elementos π_i, π_j, π_k e π_l estão no mesmo componente em G_π^X. Por esta mesma razão, temos que $x_i, x_k > 0$ e $x_j, x_l < 0$, caso contrário, pelo menos um dos quatro VDVs em S' teria um valor deslocamento envolvido na transformação cujo valor absoluto é maior ou igual a n (lembre que, por definição de transformação, se $X' = T_{i,j}(X)$, então $x'_i = x_i - n$ e $x'_j = x_j + n$), o que implica pela Propriedade 2 que este VDV possui um valor de cruzamento com valor absoluto maior que 1 e, pela Propriedade 4, o grafo de permutação cíclica deste VDV possui apenas uma componente conexa.

Vamos argumentar agora que X'' não pode ser um vetor tal que $d(\pi, X'') = d(\pi)$. Note que como $c_{ij}(X) = c_{il}(X) = c_{kj}(X) = c_{kl}(X) = 1$, sabemos, pela Propriedade 3, que todos os elementos π_a com $a \in [i..j] \cup [i..l] \cup [k..j] \cup [k..l]$ devem estar em uma mesma componente conexa. Suponha, sem perda de generalidade, que $i < k$ e $j < l$. Na Figura 4.3, mostramos todas as possíveis configurações para estes intervalos, dependendo de suas posições relativas. Podemos ver que nós sempre temos: ou um VDV X com apenas uma componente conexa, e pelo Lema 11 temos que $d(\pi, X) \leq cn(X'')$ (Figuras 4.3(a) e 4.3(f)), ou um VDV X' obtido de X com apenas uma componente conexa, e pelo Lema 11 temos que $d(\pi, X') \leq cn(X'')$ (Figuras 4.3(g)-4.3(j)). Segue então que $d(\pi, X'') > d(\pi)$. Desta forma, o VDV X^* com $d(\pi, X^*) = d(\pi)$ satisfaz necessariamente $X^* \in S \cup S'$. □

Teorema 3. *O problema de Ordenação de Permutações com Sinais por OSCs Cíclicas admite algoritmo exato em tempo polinomial.*

Demonstração. O Algoritmo 1 apresenta um algoritmo que encontra um VDV X que minimiza a soma $cn(X) + cc^-(G_\pi^X)$. Dada uma permutação π, computamos primeiro um VDV X (linhas 1-3 do Algoritmo 1), e então aplicamos iterativamente transformações de contrações estritas $T_{i,j}(x)$ (linhas 4-7) até que não seja possível.

Seja S o conjunto com todos os VDVs X tal que $cn(X) = cn(\pi)$. Jerrum [33] provou que (i) quando não é mais possível aplicar uma transformação de contração estrita

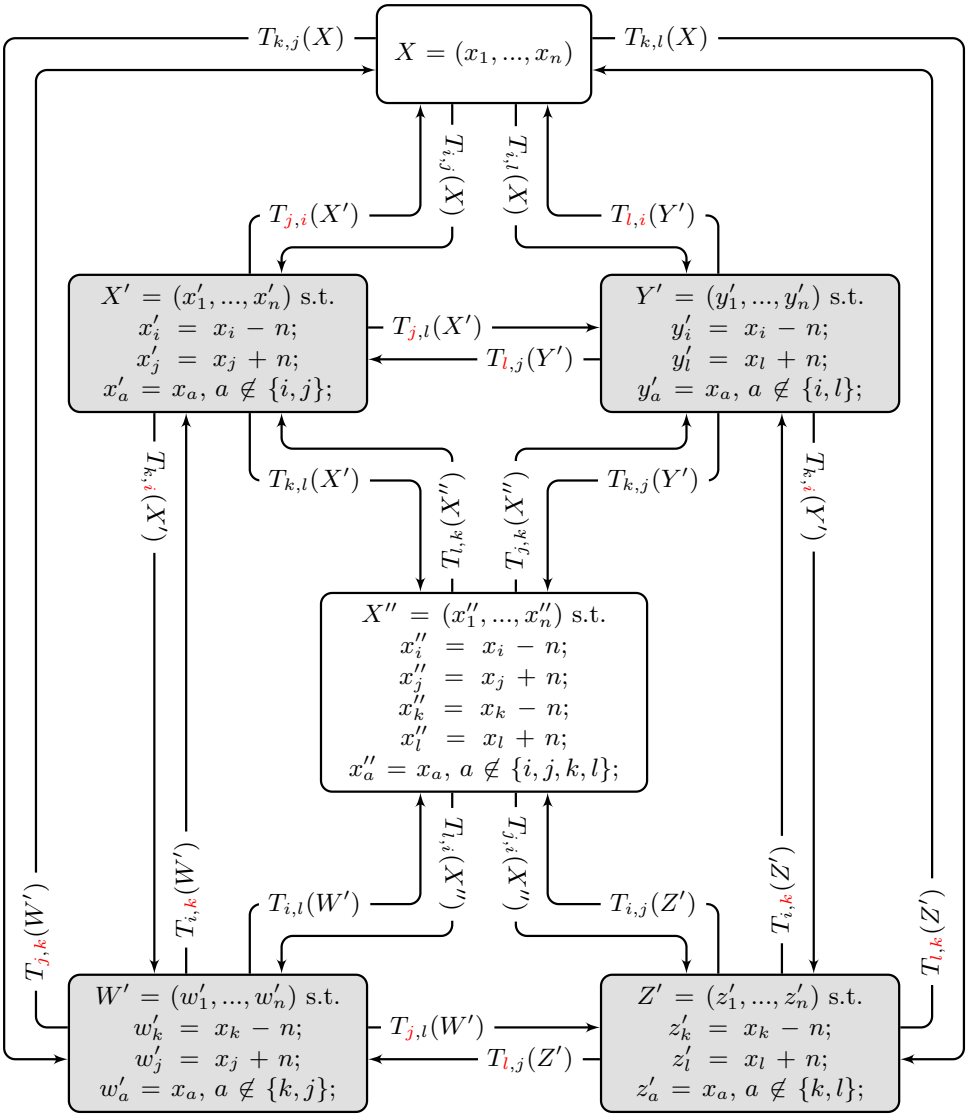


Figura 4.2: Fluxo de transformações a partir de um VDV X. Note que aplicando uma segunda transformação utilizando as mesmas posições da primeira em ordem reversa resulta novamente em X (veja as transformações onde ambos os índices estão em vermelho). Note também que todo VDV obtido a partir de X (em cinza) pode ser transformado em dois outros VDVs diferentes (que também podem ser obtidos diretamente de X) quando utilizamos um dos dois índices da primeira transformação aplicada em X (veja as transformações a partir de VDVs em cinza onde um dos índices está em vermelho). O VDV X'' pode ser obtido a partir dos quatro VDVs em cinza mas não diretamente de X. Supondo que $X \in S$, se todos os VDVs em cinza estão em S' então $X'' \notin S \cup S'$. Se um (ou mais) VDV em cinza também está em S, então temos que $X'' \in S'$.

em um VDV X, temos que $cn(X) = cn(\pi)$; (ii) para quaisquer VDVs X e X' tal que $cn(X) = cn(X') = cn(\pi)$, podemos obter X a partir de X' por uma sequência de transformações de contração, ou seja, não precisamos passar por um VDV que esteja fora de S (conjunto dos VDVs que não admitem transformações de contração estrita). As propriedades acima foram estabelecidas para VDVs de permutações sem sinais, mas como VDVs de permutações com sinais também não levam em conta os sinais dos elementos, temos

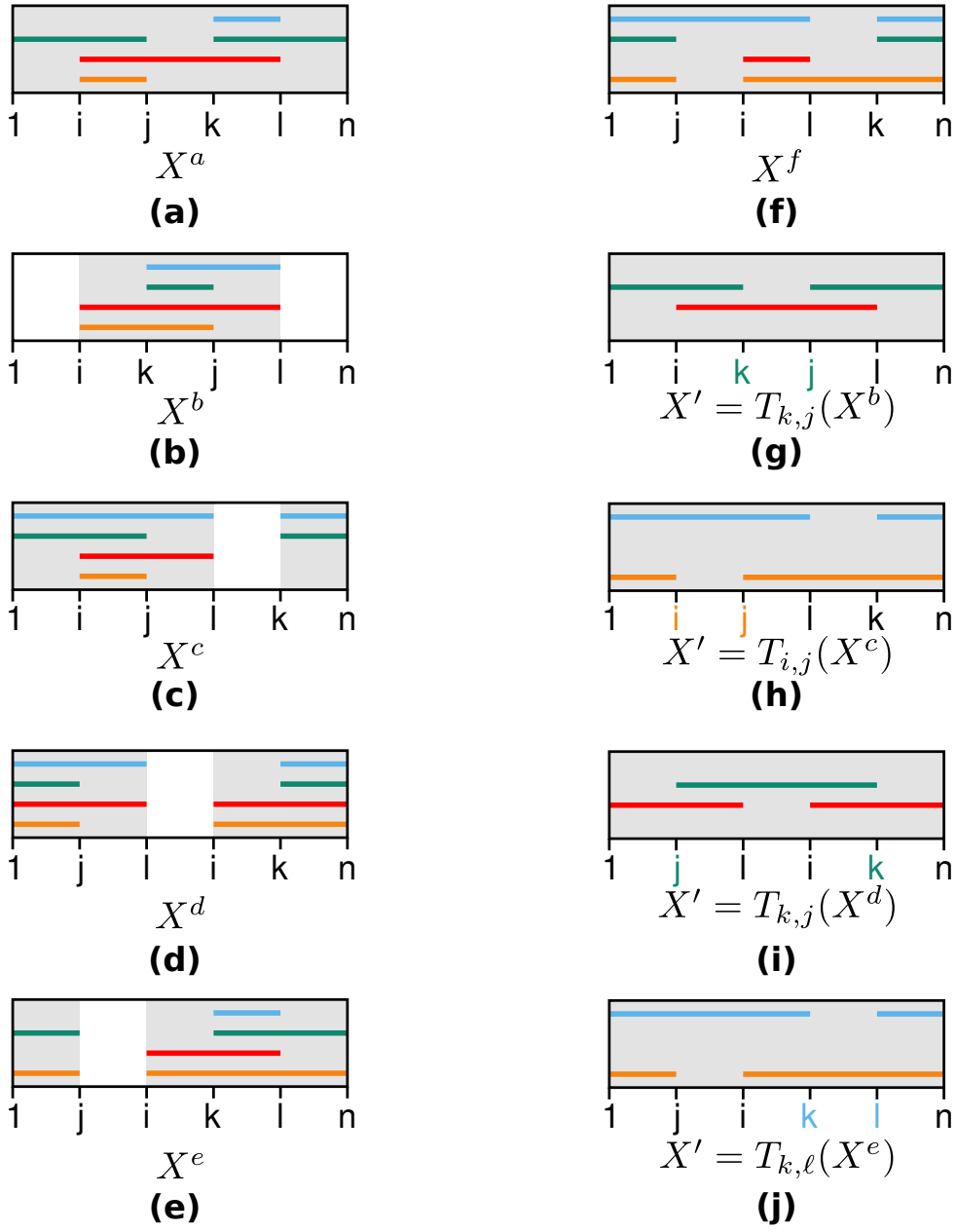


Figura 4.3: As figuras de (a) até (f) representam as seis possíveis configurações para um VDV $X \in S$, as quais chamamos de X^a, X^b, X^c, X^d, X^e e X^f. Para os VDVs X^a e X^f, a união dos intervalos destacados em cinza contém todos os elementos, logo $cc(G_\pi^{X^a}) = cc(G_\pi^{X^f}) = 1$. Para os VDVs de X^b até X^e, a união de intervalos não necessariamente possui todos os elementos (veja por exemplo as regiões brancas), mas para cada uma destas configurações existe um VDV $X' \in S'$ obtido a partir de uma transformação em X, mostrados nas figuras de (g) até (j), nos quais a união de intervalos contém todos os elementos, logo $cc(G_\pi^{X'}) = 1$.

que (i) e (ii) também se aplicam nesse contexto.

Pela propriedade (i), temos que $X \in S$ (linha 9), e pela propriedade (ii) sabemos que podemos usar X para gerar os VDVs remanescentes que também estão em S (linhas 10-18). Se existe um VDV $X \in S$ tal que $cc(G_\pi^X) = 1$ então, pelo Lema 11, temos para qualquer VDV X' que $d(\pi, X') \geq d(\pi, X)$, logo $d(\pi) = d(\pi, X)$ e podemos retornar este

valor como a distância de ordenação (linha 17). Caso contrário, pelo Lema 12, o VDV X^* com $d(\pi, X^*) = d(\pi)$ satisfaz $X^* \in S \cup S'$, onde S' é o conjunto de todos os VDVs obtidos a partir de uma transformação $T_{i,j}(X)$, com $1 \leq i \neq j \leq n$ e $X \in S$ (as linhas 19-24 geram todos os VDVs de S').

O conjunto de instruções das linhas 10-18 do Algoritmo 1 ou geram todos os VDVs de S, ou pelo menos um VDV $X' \in S$ tal que $cc(G_\pi^{X'}) = 1$. Note que se ele gera todos os VDVs de S, apenas precisamos gerar o conjunto S' e encontrar o VDV X^* com $d(\pi, X^*) = d(\pi)$. Caso contrário, se temos um VDV $X' \in S$ tal que $cc(G_\pi^{X'}) = 1$ então, pelo Lema 11, temos que para qualquer outro VDV X^*, $d(\pi, X^*) \geq d(\pi, X')$, logo $d(\pi) = d(\pi, X')$. Mostraremos por contradição que o Algoritmo 1 sempre se encaixa em uma das duas situações.

Suponha que as instruções das linhas 10-18 não geram todos os VDVs de S e que, dentre todos os VDVs X que foram gerados, temos $cc(G_\pi^X) > 1$. Temos os seguintes fatos:

- Como para qualquer $X \in S$ é verdade que $cc(G_\pi^X) > 1$, temos, pela Propriedade 4, que $c_{ij}(X) \in \{-1, 0, +1\}$, com $1 \leq i \neq j \leq n$.

- Para um VDV X' tal que $X' = T_{i,j}(X)$ com $X \in S$, se $X' \in S$, então $cn(X') = cn(X)$, o que implica, pela Propriedade 1, que $x_i - x_j = n$.

- Como para qualquer $X \in S$, $\max_{x \in X}\{x\} - \min_{x \in X}\{x\} \leq n$, se temos $x_i - x_j = n$ e $x_k - x_l = n$ com $i \neq k$ e $j \neq l$, então $x_i = x_k = \max\{X\}$ e $x_j = x_l = \min\{X\}$ (pois, caso contrário, temos que $x_i - x_l > n$ ou $x_k - x_j > n$).

- Note que, por definição de valor deslocamento, $x_k = \sum_{l=1}^{n} c_{kl}(X)$. Como, para qualquer $X \in S$, temos que $c_{kl}(X) \in \{-1, 0, +1\}$, segue, por definição de grafo de permutação cíclica, que π_i (resp. π_j) está localizado em uma componente conexa com pelo menos $|x_i| + 1$ (resp. $|x_j| + 1$) elementos em G_π^X. Se $x_i - x_j = n$ então π_i e π_j devem estar na mesma componente, pois caso contrário G_π^X teria pelo menos $|x_i| + |x_j| + 2 > n$ vértices. Além disso, esta componente conexa possui pelo menos $\max\{|x_i|, |x_j|\} + 1 \geq \frac{n}{2} + 1$ vértices. Pela mesma razão, se temos mais de um par distinto de elementos π_i, π_j tal que $x_i - x_j = n$, então todos estes pares devem estar na mesma componente.

Suponha agora que temos um VDV X'' tal que $cn(X'') = cn(\pi)$ (o que implica que $X'' \in S$), e suponha que X'' não pode ser obtido através de uma transformação de contração em X (ou seja, o Algoritmo 1 não gera X''). Vamos mostrar agora que se X'' requer pelo menos duas transformações de contração, então X admite uma transformação de contração utilizando apenas índices da primeira e segunda transformação tal que o VDV resultante X^* possui apenas uma componente conexa e, pelo Lema 11, X^* é o vetor que possui $d(\pi, X^*) = d(\pi)$.

Vamos assumir então, sem perda de generalidade, que $X'' = T_{k,l}(T_{i,j}(X))$, onde $T_{i,j}$ e $T_{k,l}$ são duas transformações de contração distintas (ou seja, $x_i - x_j = x_k - x_l = n$ com $i \neq l$ e $j \neq k$). De modo similar como explicado na prova do Lema 12, isto significa que o VDV X'' pode ser alcançado por quatro pares distintos de transformações (sejam eles $T_{k,l}(T_{i,j}(X))$, $T_{k,j}(T_{i,l}(X))$, $T_{i,l}(T_{k,j}(X))$ e $T_{i,j}(T_{k,l}(X))$). Entretanto, pela Propriedade 1

e utilizando os fatos apresentados acima, podemos concluir que $x_i = x_k = \max_{x \in X}\{x\}$ e $x_j = x_l = \min_{x \in X}\{x\}$. Logo, estes quatro VDV intermediários entre X e X'' também estão em S, dado que estas transformações são de contração.

Como estes quatro vetores são gerados pelo Algoritmo 1 e estão em S, e como, para qualquer $X \in S$, nós assumimos que $cc(G_\pi^X) > 1$, temos, pela Propriedade 4, que $c_{ij}(X) = c_{il}(X) = c_{kj}(X) = c_{kl}(X) = 1$. Agora podemos utilizar novamente a Figura 4.3 para mostrar que, com as propriedades acima, e não importando a ordem na qual os elementos x_i, x_j, x_k e x_l aparecem, sempre temos pelo menos um VDV $X' \in S$ que pode ser obtido a partir de X com apenas uma transformação de contração tal que $cc(G_\pi^{X'}) = 1$, uma contradição com o fato de que S não possui um VDV X' tal que $cc(G_\pi^{X'}) = 1$.

Segue então que o Algoritmo 1 ou gera todos os VDVs $X \in S$ ou gera pelo menos um VDV $X' \in S$ tal que $cc(G_\pi^{X'}) = 1$.

O Algoritmo 1 apresenta o procedimento mencionado acima, que consiste em encontrar um VDV X que minimiza $cn(X) + cc^-(G_\pi^X)$, sendo este valor a distância de interesse. Vamos fazer uma análise computacional da complexidade do Algoritmo 1.

O laço nas linhas 1-2, a linha 3 e o laço nas linhas 4-7 executam em tempo linear cada. A linha 8 utiliza tempo de execução $O(n^2)$ para computar $cn(X)$ mais $O(|V(G_\pi^X)| + |E(_\pi^X))|) = O(n^2)$ para computar $cc^-(G_\pi^{X'})$, resultando então em um tempo de execução de $O(n^2)$. A linha 9 executa em tempo linear. O laço nas linhas 10-18 utiliza tempo de execução $O(n^4)$: ele itera $O(n^2)$ vezes e a linha 12 utiliza tempo de execução $O(n^2)$. O laço nas linhas 19-24 utiliza $O(n^6)$: ele itera $O(n^4)$ vezes e a linha 22 executa em tempo $O(n^2)$. Assim, a complexidade de tempo do algoritmo é $O(n^6)$.

Um exemplo onde o Algoritmo 1 não gera todos os VDVs em S é dado na Figura 4.4, e um exemplo onde o Algoritmo 1 gera todos os VDVs em $S \cup S'$ é dado na Figura 4.5. □

O Teorema 3 mostra que computar o tamanho de uma sequência de ordenação mínima para permutações lineares com sinais utilizando OSCs cíclicas pode ser realizado em tempo polinomial. A partir deste resultado, é possível derivar um algoritmo de tempo polinomial para ordenar permutações *circulares* com sinais por OSCs de maneira ótima: basta cortar a permutação circular (onde existem n possíveis cortes), e decidir qual extremidade será a esquerda e qual será a direita (dois casos possíveis). Após realizar este procedimento, temos uma permutação linear com sinais, que pode ser ordenada utilizando OSCs cíclicas conforme o Algoritmo 1. A distância de ordenação será o menor valor obtido pelo Algoritmo 1 dentre as $2n$ representações lineares possíveis obtidas. Deste modo, temos o seguinte resultado.

Teorema 4. *O problema de Ordenação de Permutações Circulares com Sinais por OSCs admite algoritmo exato em tempo polinomial.*

Algoritmo 1: Computando a distância de ordenação de permutações lineares com sinais por OSCs cíclicas

Dados: uma permutação linear com sinais π.
Resultado: o tamanho da menor sequência de ordenação para π utilizando OSCs cíclicas.

para $i = 1$ **até** n **faça** ▷ Compute um VDV X para π
  $x_i \leftarrow |\pi_i| - i$
$X \leftarrow (x_1, x_2, ..., x_n)$
enquanto $\max_{p\in[1..n]}(x_p) - \min_{p\in[1..n]}(x_p) > n$ **faça**
  Seja i, j tal que $x_i = \max_{p\in[1..n]}(x_p)$ e $x_j = \min_{p\in[1..n]}(x_p)$
  $x_i \leftarrow x_i - n$
  $x_j \leftarrow x_j + n$
$d \leftarrow cn(X) + cc^-(G_\pi^X)$
$S \leftarrow \{X\}$
para cada *par* i, j **faça** ▷ Gerando os vetores $X \in S$
  $X^* \leftarrow T_{i,j}(X)$
  $d' \leftarrow cn(X^*) + cc^-(G_\pi^{X^*})$
  se $cn(X^*) = cn(X)$ **então**
    se $d' < d$ **então**
      $d \leftarrow d'$
    se $cc(G_\pi^{X^*}) = 1$ **então**
      retorna d
    $S \leftarrow S \cup \{X^*\}$
para cada $X \in S$ **faça** ▷ Gerando os vetores $X' \in S'$
  para cada *par* i, j **faça**
    $X' \leftarrow T_{i,j}(X)$
    $d' \leftarrow cn(X') + cc^-(G_\pi^{X'})$
    se $d' < d$ **então**
      $d \leftarrow d'$
retorna d

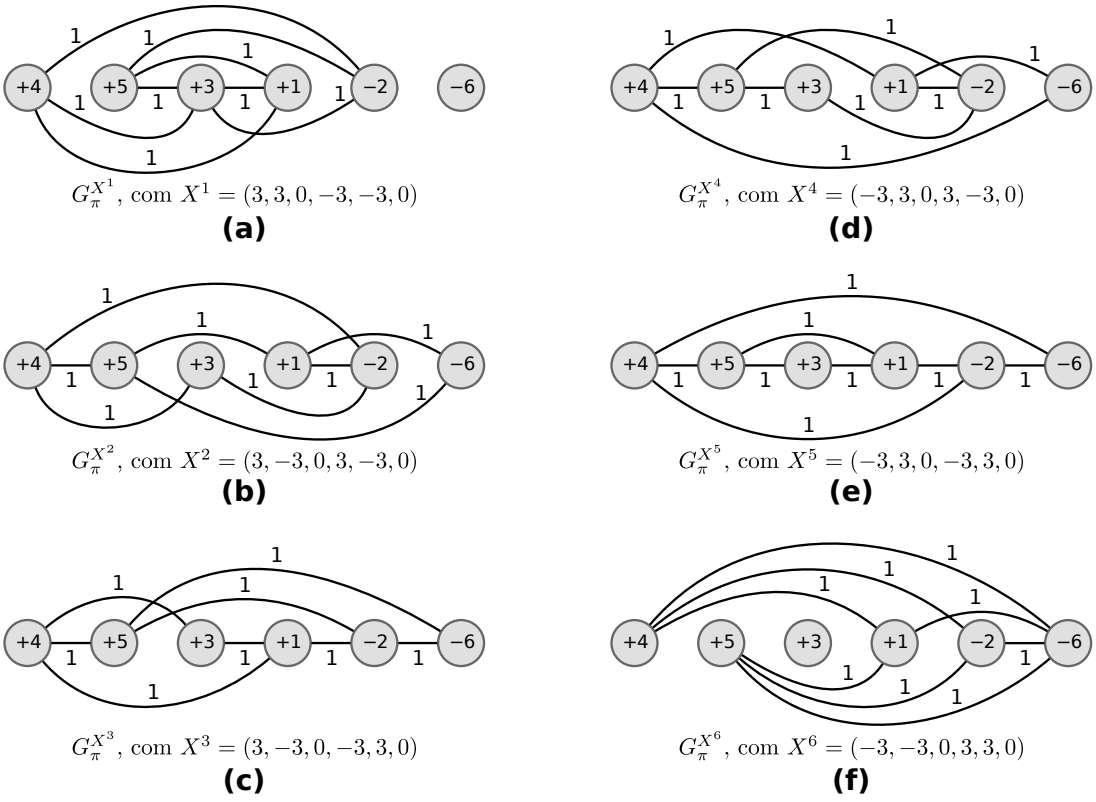
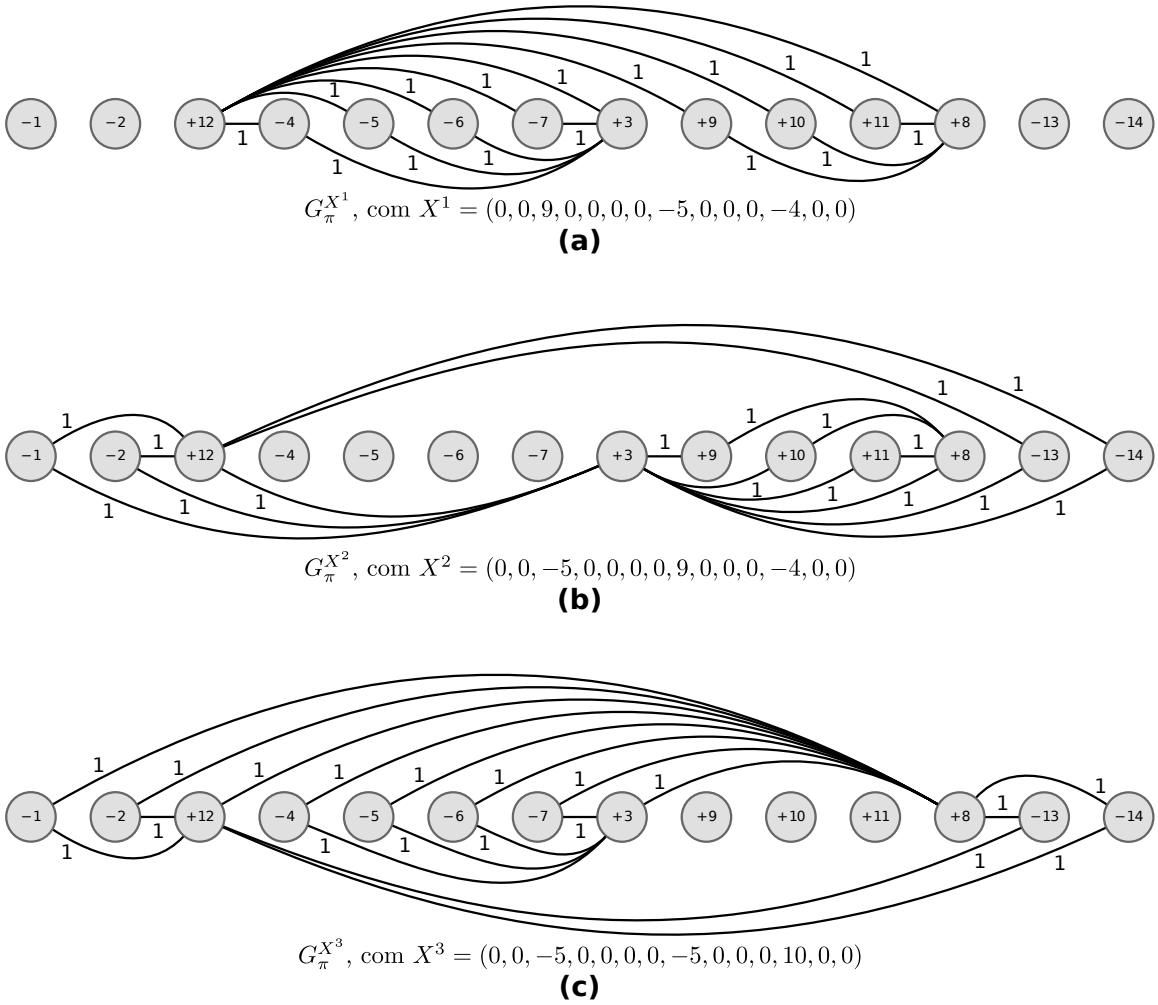


Figura 4.4: As figuras de (a) até (f) mostram os grafos de permutação cíclica para os seis possíveis VDVs para $\pi = (+4\ +5\ +3\ +1\ -2\ -6)$ cujo número de cruzamentos é mínimo (ou seja, $cn(X^i) = cn(\pi) = 8$ com $i \in [1..6]$). Note que, por definição, estes VDVs estão em S, mas o Algoritmo 1 não gera todos eles. Por exemplo, se o algoritmo começa S (na linha 9) com X^1, ele não irá gerar (no laço das linhas 10-18) X^6, dado que este VDV requer duas transformações a partir de X^1 para ser gerado (note que eles diferem em exatamente quatro valores deslocamento). Como provado no Teorema 3, como o Algoritmo 1 não é capaz de gerar todos os VDVs de S, então existe pelo menos um VDV gerado no caminho entre X^1 e X^6 com apenas uma componente conexa (neste exemplo, os quatro VDVs intermediários X^2, X^3, X^4 e X^5 que são gerados pelo Algoritmo 1 satisfazem esta condição).



$G_\pi^{X^1}$, com $X^1 = (0, 0, 9, 0, 0, 0, 0, -5, 0, 0, 0, -4, 0, 0)$

(a)

$G_\pi^{X^2}$, com $X^2 = (0, 0, -5, 0, 0, 0, 0, 9, 0, 0, 0, -4, 0, 0)$

(b)

$G_\pi^{X^3}$, com $X^3 = (0, 0, -5, 0, 0, 0, 0, -5, 0, 0, 0, 10, 0, 0)$

(c)

Figura 4.5: Dada a permutação com sinais $\pi = (-1\ -2\ +12\ -4\ -5\ -6\ -7\ +3\ +9\ +10\ +11\ +8\ -13\ -14)$, as figuras (a) e (b) mostram o grafo de permutação cíclica para os dois VDVs com valor de cruzamentos mínimo (ou seja, $cn(X^1) = cn(X^2) = cn(\pi) = 16$), logo $\{X^1, X^2\} \in S$. Note que X^2 (resp. X^1) pode ser obtido através de X^1 (resp. X^2) aplicando a transformação $T_{3,8}(X^1)$ (resp. $T_{8,3}(X^2)$). Logo, o Algoritmo 1 irá gerar ambos os VDVs, começando tanto por X^1 ou por X^2. Note que $cc^-(G_\pi^{X^1}) = cc^-(G_\pi^{X^1}) = 4$, logo $d(\pi, X^1) = d(\pi, X^2) = 16 + 4 = 20$. Em (c) temos o VDV X^3 com $cn(X^3) = 18 > cn(\pi)$, então $X^3 \notin S$, mas $X^3 \in S'$ dado que $X^3 = T_{3,12}(X^1) = T_{8,12}(X^2)$. Veja que $cc^-(G_\pi^{X^3}) = 0$, logo $d(\pi, X^3) = 18$. Dentre todos os VDVs em $S \cup S'$, X^3 é de fato o VDV que minimiza a soma e segue que $d(\pi) = d(\pi, X^3) = 18$.

4.4.4 Gerando uma Sequência de Ordenação de Tamanho Mínimo

Note que o Algoritmo 1 retorna apenas o tamanho de uma sequência de ordenação de tamanho mínimo, não uma sequência de fato. Entretanto, é possível gerar uma sequência com a ajuda do grafo de permutação cíclica G_π^X, onde π é a permutação linear com sinais dada como entrada e X é o VDV tal que $d(\pi, X) = d(\pi)$. Para isto, nós removemos iterativamente cada aresta de G_π^X aplicando um swap em dois elementos adjacentes que possuem uma aresta entre si, e escolhendo entre aplicar uma 2-reversão ou uma 2-transposição de modo que o número de componentes conexas ímpares não aumente, conforme mostrado no Lema 7. Quando G_π^X não possui mais arestas, basta aplicar $cc^-(G_\pi^X)$ 1-reversões nos elementos negativos restantes.

4.5 Conclusões

Neste capítulo apresentamos um algoritmo exato em tempo polinomial para ordenar permutações circulares com sinais por OSCs. Esta solução fecha uma lacuna existente na literatura com relação ao uso de OSCs para ordenar permutações com e sem sinais, tanto lineares quanto circulares.

Capítulo 5

Ordenação de Permutações por Operações Intergênicas Super Curtas

Neste capítulo vamos apresentar os resultados para a Ordenação de Permutações cujos modelos consideram operações intergênicas super curtas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Começaremos com algumas definições básicas e uma nova estrutura em grafo que será utilizada no decorrer do capítulo e, em seguida, apresentamos cinco algoritmos de aproximação, sendo três deles para instâncias onde a permutação não possui sinais, para os modelos (i) reversões super curtas, (ii) transposições super curtas e (iii) reversões super curtas e transposições super curtas, e os demais para instâncias onde a permutação possui sinais, para os modelos (iv) reversões super curtas e (v) reversões super curtas e transposições super curtas. Os resultados deste capítulo foram publicados na revista *Algorithms for Molecular Biology* [51].

5.1 Conceitos e Notações

Dada uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π possui n elementos, e $\breve{\pi}$ e $\breve{\iota}$ possuem $m = n + 1$ elementos cada, denotamos por $\Delta_i(\breve{\pi}, \breve{\iota}) = \breve{\pi}_i - \breve{\iota}_i$ o *desbalanço* entre as i-ésimas regiões intergênicas de $\breve{\pi}$ e $\breve{\iota}$, com $1 \leq i \leq m$.

Além disso, denotamos por $S_j(\breve{\pi}, \breve{\iota}) = \sum_{i=1}^{j} \Delta_i(\breve{\pi}, \breve{\iota})$ a *soma cumulativa de desbalanços* das regiões intergênicas de $\breve{\pi}$ e $\breve{\iota}$ localizadas entre as posições 1 e j, com $1 \leq j \leq m$. Como temos que $\sum_{i=1}^{m} \breve{\pi}_i = \sum_{i=1}^{m} \breve{\iota}_i$, então $S_m(\breve{\pi}, \breve{\iota}) = 0$.

O *Grafo Intergênico*, denotado por $I(\pi, \breve{\pi}, \breve{\iota}) = (V, E)$, é tal que o conjunto V possui dois tipos de vértices: vértices intergênicos (um para cada $\breve{\pi}_i \in \breve{\pi}$), e vértices de permutação (um para cada π_i da representação estendida de π). O conjunto E é composto por arestas de inversão: uma aresta $e = \{\breve{\pi}_i, \breve{\pi}_{i+2}\} \in E$ se existe um $j \neq i$ tal que (π_i, π_j) ou (π_j, π_{i+1}) é uma inversão, com $1 \leq i \leq n-1$ e $1 \leq j \leq n$.

Dividimos os vértices de um grafo intergênico $I(\pi, \breve{\pi}, \breve{\iota})$ em *blocos*. Um bloco sempre começa e termina com vértices de permutação. Além disso, o primeiro bloco começa com o vértice de permutação π_0, e o último bloco termina com o vértice de permutação π_{n+1}. Blocos consecutivos compartilham exatamente um vértice de permutação, ou seja, o último vértice de permutação π_i de um bloco é o primeiro vértice de permutação do

bloco adjacente à direita. Por fim, a soma das regiões intergênicas de $\breve{\pi}$ em um bloco deve ser igual a soma das regiões intergênicas correspondentes (ou seja, nas mesmas posições) em $\breve{\iota}$.

Se um bloco $\mathcal{b}$ começa com o vértice de permutação π_i e termina com o vértice de permutação π_j, com $i < j$, então $\breve{\pi}_k \in \mathcal{b}$ para $i < k \leq j$ e $\pi_k \in \mathcal{b}$ para $i \leq k \leq j$. Além disso, quaisquer dois vértices intergênicos que possuem uma aresta de inversão entre si devem pertencer ao mesmo bloco. Desta forma, se $\mathcal{b}$ termina com o vértice π_j, então $e = \{\breve{\pi}_j, \breve{\pi}_{j+2}\} \notin E$.

A ideia é que os blocos dividem $I(\pi, \breve{\pi}, \breve{\iota})$ em problemas menores, sendo possível fazer uma redistribuição de elementos e regiões intergênicas apenas dentro de cada bloco (ou seja, sem necessidade de trocar regiões intergênicas entre dois blocos diferentes). Isto requer que qualquer bloco $\mathcal{b}$ começando em π_i e terminando em π_j tenha obrigatoriamente $\sum_{k=i+1}^{j} \breve{\pi}_k - \breve{\iota}_k = 0$.

Formalmente, dado um grafo intergênico $I(\pi, \breve{\pi}, \breve{\iota})$, um *bloco* $\mathcal{b}$ é um conjunto minimal de vértices de V no qual: (i) se $\mathcal{b}$ não é o primeiro bloco e, por consequência começa com o vértice de permutação π_i para um $i > 0$ qualquer, então $S_i(\breve{\pi}, \breve{\iota}) = 0$; (ii) se $e = \{\breve{\pi}_i, \breve{\pi}_{i+2}\} \in E$ e $\breve{\pi}_i \in \mathcal{b}$, então $\breve{\pi}_{i+2} \in \mathcal{b}$ e vice-versa, para qualquer $1 \leq i \leq n-1$; (iii) se $\{\breve{\pi}_i, \breve{\pi}_j\} \in \mathcal{b}$ com $i < j$, então $\{\pi_{i-1}, \pi_j\} \in \mathcal{b}$ e para qualquer $i < k < j$ $\{\breve{\pi}_k, \pi_k\} \in \mathcal{b}$; e (iv) $\sum_{k \mid \breve{\pi}_k \in \mathcal{b}} \breve{\pi}_k - \breve{\iota}_k = 0$.

Um bloco com apenas um vértice intergênico é dito *trivial*, e ele é dito *não trivial* caso contrário. O número de vértices intergênicos em um bloco $\mathcal{b}$ é denotado por $\mathcal{b}_r$. Um bloco $\mathcal{b}$ é dito *ímpar* caso $\mathcal{b}_r$ seja ímpar, e é dito *par* caso contrário. O número de blocos em um grafo intergênico é denotado por $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, e o número de blocos ímpares (resp. pares) é denotado por $\mathcal{B}_i(I(\pi, \breve{\pi}, \breve{\iota}))$ (resp. $\mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$). A Figura 5.1 mostra três exemplos de grafos intergênicos.

Note que a definição de blocos não leva em consideração se a permutação possui sinais ou não. Assim, vamos apresentar lemas que podem ser aplicados em instâncias que possuem ou não sinais. Os próximos dois lemas analisam o impacto no número de blocos de um grafo intergênico ao aplicar operações super curtas.

Lema 13. *Seja $(\pi', \breve{\pi}')$ o genoma resultante após a aplicação de uma 1-reversão em $(\pi, \breve{\pi})$. Temos que $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) \leq \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + 1$.*

Demonstração. Uma 1-reversão $\rho_{(x,y)}^{(i,i)}$ é aplicada sobre as regiões intergênicas $\breve{\pi}_i$ e $\breve{\pi}_{i+1}$, com $1 \leq i \leq n$. Além disso, como 1-reversões não criam nem removem inversões, os grafos intergênicos $I(\pi', \breve{\pi}', \breve{\iota}) = (V', E')$ e $I(\pi, \breve{\pi}, \breve{\iota}) = (V, E)$ possuem o mesmo número de arestas.

Se $\breve{\pi}_i \in \mathcal{b}$ e $\breve{\pi}_{i+1} \notin \mathcal{b}$, então esta 1-reversão é aplicada sobre vértices intergênicos de dois blocos distintos, o que significa que $\breve{\pi}_i$ é o último vértice intergênico de $\mathcal{b}$ e, por definição, $S_i(\breve{\pi}, \breve{\iota}) = 0$. Se $x + y \neq \breve{\pi}_i$, temos que $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) - 1$, como mostrado na Figura 5.2(a).

Considere agora que $\breve{\pi}_i, \breve{\pi}_{i+1} \in \mathcal{b}$. Se $i < n$ e $\{\breve{\pi}'_i, \breve{\pi}'_{i+2}\} \in E'$, ou $i > 1$ e $\{\breve{\pi}'_{i-1}, \breve{\pi}'_{i+1}\} \in E'$, então $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$. Caso contrário, temos dois casos para considerar: $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, se $S_i(\breve{\pi}', \breve{\iota}) \neq 0$ (como mostrado na Figura 5.2(b)); e $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + 1$ se $S_i(\breve{\pi}', \breve{\iota}) = 0$ (como mostrado na Figura 5.2(c)). □

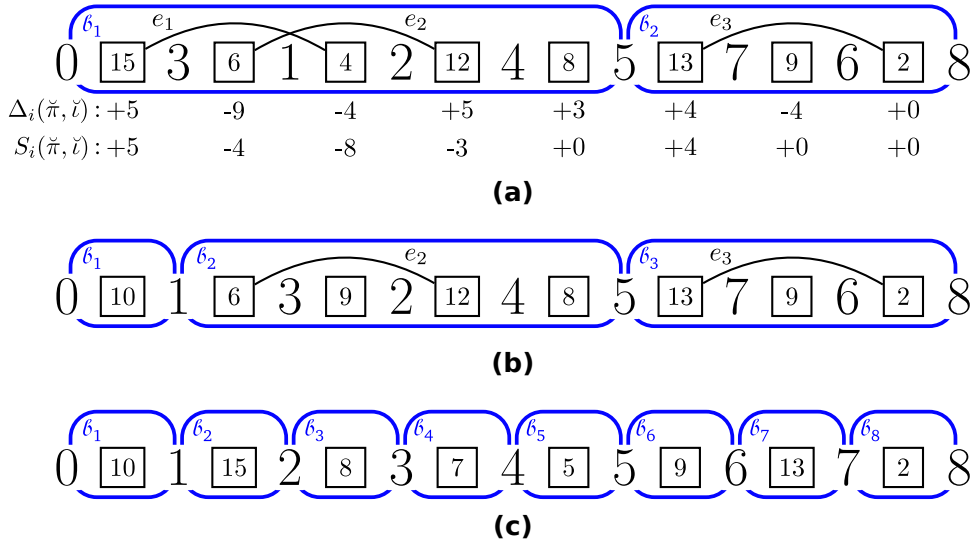


Figura 5.1: Três grafos intergênicos $I(\pi,\breve{\pi},\breve{\iota})$, $I(\pi',\breve{\pi}',\breve{\iota})$ e $I(\iota,\breve{\iota},\breve{\iota})$, com $\pi = (3\ 1\ 2\ 4\ 5\ 7\ 6)$, $\breve{\pi} = (15,6,4,12,8,13,9,2)$, $\pi' = (1\ 3\ 2\ 4\ 5\ 7\ 6)$, $\breve{\pi}' = (10,6,9,12,8,13,9,2)$, $\iota = (1\ 2\ 3\ 4\ 5\ 6\ 7)$ e $\breve{\iota} = (10,15,8,7,5,9,13,2)$. Os quadrados pretos representam os vértices intergênicos, e o número dentro de cada quadrado representa seu tamanho. Os números entre dois quadrados (e nas extremidades) representam vértices de permutação. Os retângulos arredondados em azul representam blocos. Em (a) existem três arestas de inversão no grafo $I(\pi,\breve{\pi},\breve{\iota})$, e temos que $\mathcal{B}(I(\pi,\breve{\pi},\breve{\iota})) = \mathcal{B}_i(I(\pi,\breve{\pi},\breve{\iota})) = 2$ dado que existem cinco vértices intergênicos em $\mathcal{b}_1$ e três vértices intergênicos em $\mathcal{b}_2$. Em (a) também listamos todos os valores para $\breve{\pi}_i - \breve{\iota}_i$ e $\Delta_i(\breve{\pi},\breve{\iota})$, com $1 \leq i \leq 8$. A instância $(\pi',\breve{\pi}',\breve{\iota})$ é tal que $(\pi',\breve{\pi}') = (\pi,\breve{\pi}) \cdot \rho^{(1,2)}_{(8,2)}$. Temos em (b) que, comparado com (a), $I(\pi',\breve{\pi}',\breve{\iota})$ possui um bloco a mais, e a aresta e_1 foi removida. Em (c) podemos ver que quando estamos na instância final $(\iota,\breve{\iota},\breve{\iota})$ o número de blocos é igual ao número de regiões intergênicas de $\breve{\iota}$ (ou seja, $\mathcal{B}(I(\iota,\breve{\iota},\breve{\iota})) = n+1 = 8$).

Lema 14. *Seja $(\pi',\breve{\pi}')$ o genoma resultante após a aplicação de uma 2-reversão ou uma 2-transposição em $(\pi,\breve{\pi})$. Temos que $\mathcal{B}(I(\pi',\breve{\pi}',\breve{\iota})) \leq \mathcal{B}(I(\pi,\breve{\pi},\breve{\iota})) + 2$.*

Demonstração. Se uma 2-reversão ou 2-transposição é aplicada sobre vértices π_i e π_{i+1} de dois blocos diferentes de $I(\pi,\breve{\pi},\breve{\iota})$, então necessariamente criamos uma nova inversão, e o grafo intergênico da instância $I(\pi',\breve{\pi}',\breve{\iota})$ é tal que ou $\mathcal{B}(I(\pi',\breve{\pi}',\breve{\iota})) = \mathcal{B}(I(\pi,\breve{\pi},\breve{\iota})) - 2$ (conforme mostrado na Figura 5.3(a)) ou $\mathcal{B}(I(\pi',\breve{\pi}',\breve{\iota})) = \mathcal{B}(I(\pi,\breve{\pi},\breve{\iota})) - 1$ (conforme mostrado na Figura 5.3(b)).

Considere agora que esta operação super curta é aplicada sobre vértices π_i e π_{i+1} de um mesmo bloco de $I(\pi,\breve{\pi},\breve{\iota})$, com $1 \leq i < n-1$. Se o grafo intergênico $I(\pi',\breve{\pi}',\breve{\iota}) = (V',E')$ possui $\{\breve{\pi}'_i,\breve{\pi}'_{i+2}\} \in E'$, então $\mathcal{B}(I(\pi',\breve{\pi}',\breve{\iota})) = \mathcal{B}(I(\pi,\breve{\pi},\breve{\iota}))$. Caso contrário, temos três casos a considerar:

1. $\mathcal{B}(I(\pi',\breve{\pi}',\breve{\iota})) = \mathcal{B}(I(\pi,\breve{\pi},\breve{\iota}))$, se $S_i(\breve{\pi}',\breve{\iota}) \neq 0$ e $S_{i+1}(\breve{\pi}',\breve{\iota}) \neq 0$ (como mostra a Figura 5.3(c));

2. $\mathcal{B}(I(\pi',\breve{\pi}',\breve{\iota})) = \mathcal{B}(I(\pi,\breve{\pi},\breve{\iota})) + 1$ se ou $S_i(\breve{\pi}',\breve{\iota}) = 0$ ou $S_{i+1}(\breve{\pi}',\breve{\iota}) = 0$ (como mostra a Figura 5.3(d));

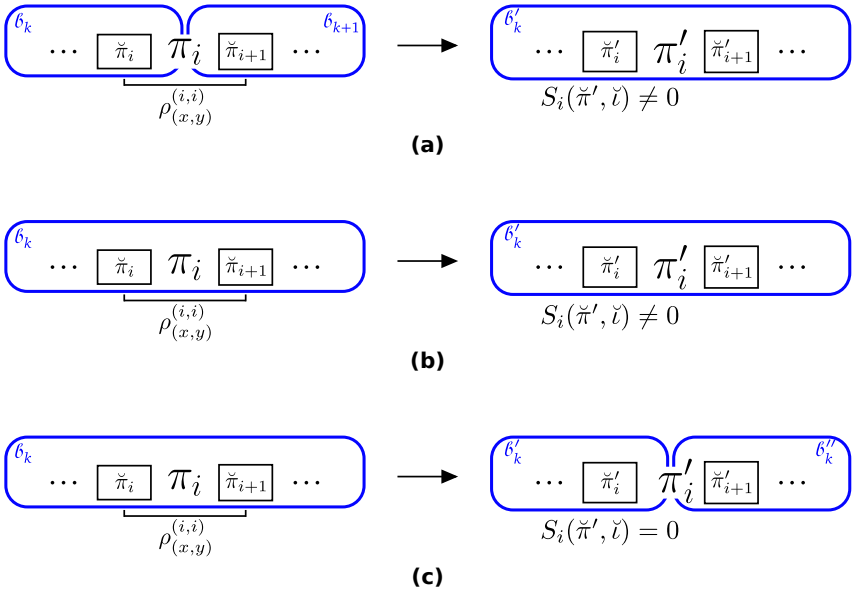


Figura 5.2: Exemplos de grafos intergênicos para as possíveis modificações no valor de $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota}))$ com relação a $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, onde $(\pi', \breve{\pi}')$ é o genoma resultante após a aplicação de uma 1-reversão em $(\pi, \breve{\pi})$. Quando a 1-reversão é aplicada sobre vértices intergênicos de dois blocos distintos e $\breve{\pi}'_i \neq \breve{\pi}_i$, temos que $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) - 1$, como mostrado em (a). Caso contrário, temos que $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + \ell$, com $\ell \in \{0, 1\}$, como mostrado em (b) e (c).

3. $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + 2$ se $S_i(\breve{\pi}', \breve{\iota}) = 0$ e $S_{i+1}(\breve{\pi}', \breve{\iota}) = 0$ (como mostra a Figura 5.3(e)). □

O lema a seguir nos ajudará posteriormente a lidar com blocos que não possuem arestas.

Lema 15. *Se um bloco não trivial $\mathcal{b}$ de um grafo intergênico $I(\pi, \breve{\pi}, \breve{\iota})$ não possui arestas, então sempre podemos aplicar uma* 1*-reversão em $(\pi, \breve{\pi})$ que quebra $\mathcal{b}$ em dois blocos $\mathcal{b}'$ e $\mathcal{b}''$ tal que $\mathcal{b}'_r + \mathcal{b}''_r = \mathcal{b}_r$.*

Demonstração. Seja p_i o índice em $\breve{\pi}$ do i-ésimo vértice intergênico dentro do bloco $\mathcal{b}$. O último vértice intergênico de $\mathcal{b}$ é a representação da região intergênica localizada na posição $p_{\mathcal{b}_r}$ de $\breve{\pi}$. Pela definição de bloco, e como $\mathcal{b}$ não possui arestas, para qualquer $p_1 \leq j < p_{\mathcal{b}_r}$ temos que $S_j(\breve{\pi}, \breve{\iota}) \neq 0$. Note que, como $\mathcal{b}_r > 1$, temos que $S_{p_1}(\breve{\pi}, \breve{\iota}) = \Delta_{p_1}(\breve{\pi}, \breve{\iota}) \neq 0$.

Se $S_{p_1}(\breve{\pi}, \breve{\iota}) > 0$, seja $p_i = p_1$ e seja k o índice do elemento de π cujo vértice de permutação correspondente está à direita do vértice intergênico $\breve{\pi}_{p_1}$. Aplique a reversão $\rho^{(k,k)}_{(\breve{\iota}_{p_1}, 0)}$ a $(\pi, \breve{\pi})$ gerando $(\pi', \breve{\pi}')$.

Caso contrário, temos que $S_{p_1}(\breve{\pi}, \breve{\iota}) < 0$ e precisamos encontrar dois vértices intergênicos $\breve{\pi}_{p_i}$ e $\breve{\pi}_{p_{i+1}}$ para $1 \leq i < \mathcal{b}_r$ tal que $S_{p_i}(\breve{\pi}, \breve{\iota}) < 0$ e $S_{p_{i+1}}(\breve{\pi}, \breve{\iota}) \geq 0$. Como, por definição de bloco, $S_{p_{\mathcal{b}_r}}(\breve{\pi}, \breve{\iota}) = 0$, este par sempre existe. Seja k o índice do elemento de π cujo vértice de permutação correspondente está à direita do vértice intergênico $\breve{\pi}_{p_i}$. Aplique a reversão $\rho^{(k,k)}_{(\breve{\pi}_{p_i}, -S_{p_i}(\breve{\pi}, \breve{\iota}))}$ a $(\pi, \breve{\pi})$ gerando $(\pi', \breve{\pi}')$.

Em ambos os casos, a instância $(\pi', \breve{\pi}', \breve{\iota})$ possui (i) $S_{p_i}(\breve{\pi}', \breve{\iota}) = 0$; (ii) $S_{p_{i+1}}(\breve{\pi}', \breve{\iota}) = S_{p_{i+1}}(\breve{\pi}, \breve{\iota}) + S_{p_i}(\breve{\pi}, \breve{\iota})$; e (iii) para qualquer $i + 2 \leq j \leq \mathcal{b}_r$ temos que $S_{p_j}(\breve{\pi}', \breve{\iota}) = S_{p_j}(\breve{\pi}, \breve{\iota})$,

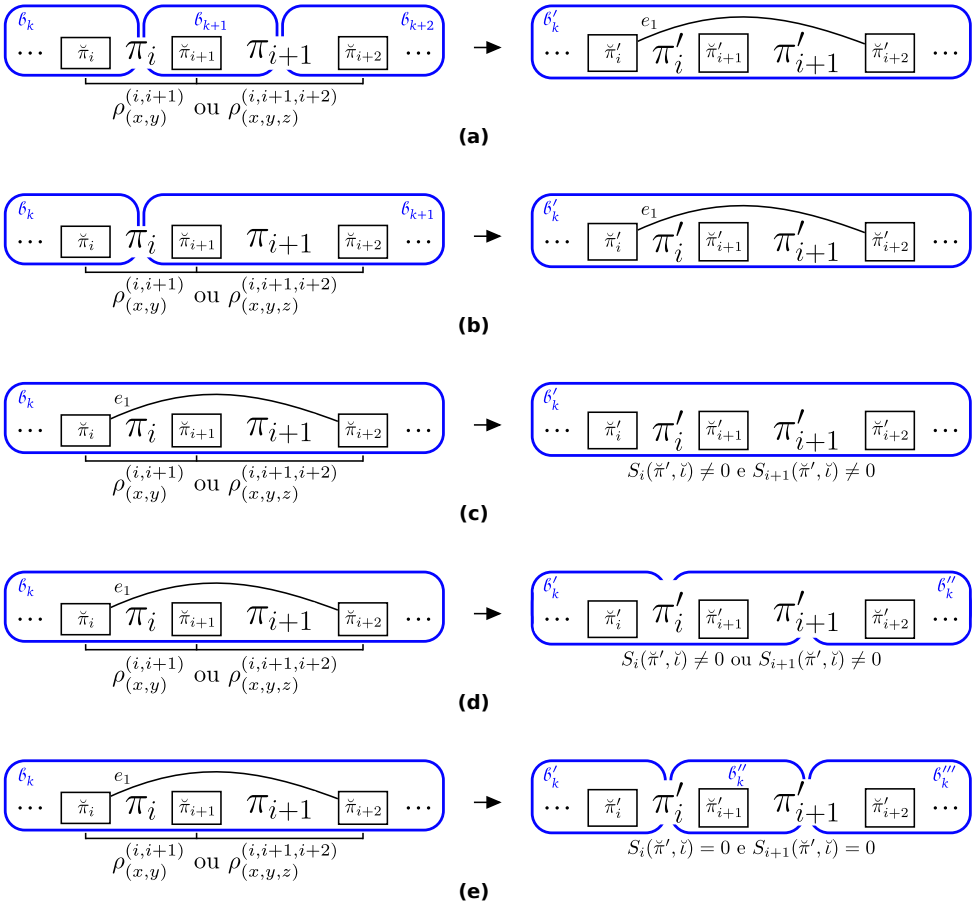


Figura 5.3: Exemplos de grafos intergênicos para as possíveis modificações no valor de $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota}))$ com relação a $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ onde $(\pi', \breve{\pi}')$ é o genoma resultante após a aplicação de uma 2-reversão ou 2-transposição em $(\pi, \breve{\pi})$. Quando a operação intergênica é aplicada sobre vértices intergênicos de dois blocos distintos ao mesmo tempo, temos que $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) < \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, como mostrado em (a) e (b). Caso contrário, temos que $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) \leq \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) \leq \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + 2$, como mostrado em (c), (d) e (e).

o que indica que, assim como antes, todos os vértices intergênicos de $\breve{\pi}'_{p_{i+1}}$ até $\breve{\pi}'_{p_{\mathcal{b}_r}}$ devem pertencer ao mesmo bloco.

Esta 1-reversão quebra $\mathcal{b}$ em dois blocos: $\mathcal{b}'$ com todos os vértices intergênicos entre as posições p_1 até p_i, e $\mathcal{b}''$ com todos os vértices intergênicos entre as posições p_{i+1} to $p_{\mathcal{b}_r}$. □

Vamos agora verificar como as 2-transposições quebram blocos não triviais de um grafo intergênico $I(\pi, \breve{\pi}, \breve{\iota})$.

Lema 16. *Se um bloco $\mathcal{b}$ de um grafo intergênico $I(\pi, \breve{\pi}, \breve{\iota})$ com $\mathcal{b}_r > 2$ (resp. $\mathcal{b}_r = 2$) não possui arestas, então podemos aplicar duas 2-transposições que quebram $\mathcal{b}$ em três blocos $\mathcal{b}'$, $\mathcal{b}''$ e $\mathcal{b}'''$ tais que $\mathcal{b}'_r + \mathcal{b}''_r + \mathcal{b}'''_r = \mathcal{b}_r$ (resp. dois blocos $\mathcal{b}'$ e $\mathcal{b}''$ tal que $\mathcal{b}'_r = \mathcal{b}''_r = 1$).*

Demonstração. Note que qualquer 2-transposição irá aumentar ou diminuir o número de inversões em uma unidade. Pelo Lema 14, uma 2-transposição que remove uma inversão pode aumentar o número de blocos em até duas unidades, e uma 2-transposição que cria uma inversão não pode aumentar o número de blocos. Como não existem inversões em

$\mathcal{b}$, para cada 2-transposição que remove uma inversão de $\mathcal{b}$ temos uma 2-transposição aplicada anteriormente criando aquele par de inversão.

Agora explicamos como aumentar o número de blocos em duas unidades quando $\mathcal{b}_r \geq 3$. Seja p_i o índice em $\breve{\pi}$ do i-ésimo vértice intergênico dentro do bloco $\mathcal{b}$. Se não existe um vértice intergênico $\breve{\pi}_j$ dentro de $\mathcal{b}$ no qual a soma cumulativa é negativa, aplique a transposição $\rho^{(p_1,p_2,p_3)}_{(x,y,0)}$ onde $x = \min\{\breve{\iota}_{p_1} + \breve{\iota}_{p_2}, \breve{\pi}_{p_1}\}$ e $y = \breve{\pi}_{p_2} + \breve{\iota}_{p_1} + \breve{\iota}_{p_2} - x$. Feito isso, aplique a transposição $\rho^{(p_1,p_2,p_3)}_{(\breve{\pi}_{p_1},0,0)}$. Estas duas 2-transposições quebram $\mathcal{b}$ em três blocos: $\mathcal{b}'$ com $\breve{\pi}_{p_1}$, $\mathcal{b}''$ com $\breve{\pi}_{p_2}$ e $\mathcal{b}'''$ com os vértices intergênicos restantes de $\mathcal{b}$. Note que $\mathcal{b}'$ e $\mathcal{b}''$ são blocos ímpares, e $\mathcal{b}'''$ possui a mesma paridade de $\mathcal{b}$.

Caso contrário, podemos encontrar um par de vértices intergênicos consecutivos $\breve{\pi}_{p_i}$ e $\breve{\pi}_{p_{i+1}}$ de $\mathcal{b}$ tal que $S_{p_i}(\breve{\pi}, \breve{\iota}) < 0$ e $S_{p_{i+1}}(\breve{\pi}, \breve{\iota}) \geq 0$, e como $S_{p_{\mathcal{b}_r}}(\breve{\pi}, \breve{\iota}) = 0$, este par sempre existe.

Se $\mathcal{b}_r$ ou p_i é par, aplique a transposição $\rho^{(p_{i-1},p_i,p_{i+1})}_{(x,y,0)}$ tal que $x = \breve{\pi}_{p_{i-1}}$ e $y = \breve{\pi}_{p_i} + S_{pi-1}(\breve{\pi}, \breve{\iota})$, seguida da transposição $\rho^{(p_{i-1},p_i,p_{i+1})}_{(x',0,y')}$ tal que $x' = \breve{\iota}_{p_{i-1}}$ e $y' = \breve{\iota}_{p_i}$.

Se $\mathcal{b}_r$ e p_i são ímpares, aplique a transposição $\rho^{(p_i,p_{i+1},p_{i+2})}_{(x,y,0)}$ tal que $x = \breve{\pi}_{p_i}$ e $y = \breve{\pi}_{p_{i+1}} + S_{pi}(\breve{\pi}, \breve{\iota})$, seguida da transposição $\rho^{(p_{i-1},p_i,p_{i+1})}_{(x',0,y')}$ tal que $x' = \breve{\iota}_{p_i}$ e $y' = \breve{\iota}_{p_{i+1}}$.

Estas duas transposições quebram $\mathcal{b}$ em três blocos, então se $\mathcal{b}_r$ é par, ele é transformado em dois blocos ímpares e um bloco par (dado que o bloco do meio é trivial e ímpar um dos blocos remanescentes deve ser ímpar também). Se $\mathcal{b}$ é ímpar, ele é transformado em três blocos impares devido à escolha da posição definida acima.

Se $\mathcal{b}_r = 2$ e $p_1 > 1$, aplicamos a transposição $\rho^{(p_1-1,p_1,p_2)}_{(x,y,0)}$ tal que $x = \breve{\pi}_{p_1-1}$ e $y = \breve{\pi}_{p_1}$, seguida da transposição $\rho^{(p_1-1,p_1,p_2)}_{(x',0,y')}$ tal que $x' = x$ e $y' = \breve{\iota}_{p_1}$. Se $\mathcal{b}_r = 2$ e $p_1 = 1$, aplicamos a transposição $\rho^{(p_1,p_2,p_2+1)}_{(\breve{\pi}_{p_1},0,0)}$ seguida da transposição $\rho^{(p_1,p_2,p_2+1)}_{(\breve{\iota}_{p_1},0,0)}$. Estas duas transposições transformam $\mathcal{b}$ em dois blocos triviais. □

Nas seções a seguir, exploramos cinco problemas que utilizam operações intergênicas super curtas, a saber:

- Ordenação de Permutações sem Sinais por Reversões Intergênicas Super Curtas (SbSSR);
- Ordenação de Permutações sem Sinais por Transposições Intergênicas Super Curtas (SbSST);
- Ordenação de Permutações sem Sinais por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas (SbSSO);
- Ordenação de Permutações com Sinais por Reversões Intergênicas Super Curtas (SbSigSSR);
- Ordenação de Permutações com Sinais por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas (SbSigSSO).

A Tabela 5.1 sumariza nossos resultados considerando:

- permutações gerais (PG);

- permutações com pelo menos n inversões (1IP);
- permutações com pelo menos $2n$ inversões (2IP);
- permutações com pelo menos $n\ell$ inversões, em que $\ell \geq 1$ (ℓIP).

Tabela 5.1: Sumário dos fatores de aproximação dos algoritmos considerando operações intergênicas super curtas apresentadas neste capítulo.

Problema	PG	1IP	2IP	ℓIP
SbSSR	3	2	1.5	$1+\frac{1}{\ell}$
SbSST	3	2	1.5	$1+\frac{1}{\ell}$
SbSSO	3	2	1.5	$1+\frac{1}{\ell}$
SbSigSSR	5	3	2	$1+\frac{2}{\ell}$
SbSigSSO	5	3	2	$1+\frac{2}{\ell}$

5.2 Ordenação de Permutações sem Sinais por Reversões Intergênicas Super Curtas (SbSSR)

Vamos investigar a versão do problema quando apenas reversões intergênicas super curtas (ou seja, 1-reversões e 2-reversões) são permitidas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação sem sinais.

Seja $S_i(\breve{\pi}, \breve{\iota}) = \sum_{i=1,\, i \pmod 2=1}^{n+1} \Delta_i(\breve{\pi}, \breve{\iota})$ a soma cumulativa dos desbalanços de regiões intergênicas de $\breve{\pi}$ e $\breve{\iota}$ localizadas em posições ímpares apenas. Utilizando os Lemas 13, 14 e 15, mostramos nos próximos dois lemas limitantes inferiores e superiores no número de reversões intergênicas super curtas necessárias para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

Lema 17. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, π é uma permutação sem sinais, e seja $\varphi = 0$ se $S_i(\breve{\pi}, \breve{\iota}) = 0$ e $\varphi = 1$ caso contrário. Temos que $d_r(\pi, \breve{\pi}, \breve{\iota}) \geq \max\{\frac{m-\mathcal{B}(I(\pi,\breve{\pi},\breve{\iota}))}{2},\ inv(\pi)+\varphi\}$.*

Demonstração. Para ordenar π, precisamos remover todas as inversões e, como uma 2-reversão pode remover apenas uma inversão, temos que $d_r(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi)$. Além disso, como 2-reversões permutam tamanhos de regiões de mesma paridade apenas, temos que $d_r(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi) + \varphi$, onde $\varphi = 1$ se $S_i(\breve{\pi}, \breve{\iota}) \neq 0$ (neste caso será necessário aplicar pelo menos uma 1-reversão para permutar os tamanhos entre duas regiões intergênicas consecutivas, tais que a paridade de uma seja diferente da paridade da outra), e $\varphi = 0$ caso contrário.

Por outro lado, pelos lemas 13 e 14, é possível aumentar o número de blocos em no máximo duas unidades a cada reversão super curta. Assim, para gerar os m blocos de $(\iota, \breve{\iota})$ precisamos de pelo menos $\frac{m-\mathcal{B}(I(\pi,\breve{\pi},\breve{\iota}))}{2}$ reversões super curtas. □

Lema 18. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Temos que $d_r(\pi, \breve{\pi}, \breve{\iota}) \leq inv(\pi) + m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$.*

Demonstração. Note que enquanto $\pi \neq \iota$, π possui pelo menos um par de elementos consecutivos (π_i, π_{i+1}) que é uma inversão. Suponha que primeiro removemos todas as inversões de π utilizando $inv(\pi)$ 2-reversões do tipo $\rho^{(i,i+1)}_{(\breve{\pi}_i,0)}$, ou seja, sem modificar $\breve{\pi}$. Se $(\pi', \breve{\pi}')$ é o genoma resultante, então sabemos que $\pi' = \iota$ e $\breve{\pi}' = \breve{\pi}$. Note que o número de blocos em $I(\pi', \breve{\pi}', \breve{\iota})$ não pode ser menor que $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, dado que qualquer 2-reversão que remove uma inversão deve ser aplicada dentro de um bloco. Pelo Lema 15, $m - \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) \leq m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ 1-reversões são suficientes para obter m blocos. □

Utilizando os lemas 17 e 18, mostramos que existe um algoritmo de 3-aproximação para este problema.

Teorema 5. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. O valor de $d_r(\pi, \breve{\pi}, \breve{\iota})$ é 3-aproximável.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, e seja $\varphi = 0$, se $S_i(\breve{\pi}, \breve{\iota}) = 0$, ou $\varphi = 1$ caso contrário. Se $\frac{m-k}{2} \geq inv(\pi) + \varphi$ então, pelo Lema 17, temos que $d_r(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{m-k}{2}$, e, pelo Lema 18, $d_r(\pi, \breve{\pi}, \breve{\iota}) \leq m - k + inv(\pi) \leq m - k + \frac{m-k}{2} \leq 3\frac{m-k}{2}$. Caso contrário, $\frac{m-k}{2} < inv(\pi) + \varphi$, então $m - k < 2inv(\pi) + 2\varphi$. Pelo Lema 17, $d_r(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi) + \varphi$, e, pelo Lema 18, $d_r(\pi, \breve{\pi}, \breve{\iota}) \leq m - k + inv(\pi) \leq 2inv(\pi) + 2\varphi + inv(\pi) \leq 3inv(\pi) + 2\varphi$. □

O Algoritmo 2 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ (onde π é uma permutação sem sinais) utilizando reversões intergênicas super curtas, e possui um fator de aproximação igual a 3. Computar $S_i(\breve{\pi}, \breve{\iota})$ e $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ custa tempo $O(n)$ e a atualização de ambos ocorre em tempo constante. Podemos computar $inv(\pi)$ em tempo $O(n\sqrt{\log n})$ [17]. Os laços nas linhas 5 e 15 iteram $O(n^2)$ vezes. Assim, a complexidade total do Algoritmo 2 é $O(n^2)$.

Vamos denotar por δ_n o conjunto de todas as permutações π de tamanho n e vamos denotar por $\delta_{n,k}$ o número de todas as permutações π em S_n tal que $inv(\pi) \leq k$. Para $n = 12$, existem 762.007 permutações em $\delta_{12,12}$, o que corresponde a 0.16% das 12! permutações de δ_{12}, e para $n > 12$ o número de permutações em $\delta_{n,n}$ nunca corresponde a mais de 0.05% das $n!$ permutações em δ_n [37]. Além disso, para $n > 18$, o número de permutações em $\delta_{n,2n}$ nunca corresponde a mais de 0.03% das $n!$ permutações em δ_n [37].

O Algoritmo 2 possui um fator de aproximação melhor quando o número de inversões é pelo menos n, como explicado no teorema a seguir.

Teorema 6. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) \geq n$, o Algoritmo 2 possui um fator de aproximação de $(1 + \frac{1}{\ell})$, onde $\ell = \frac{inv(\pi)}{n} \geq 1$.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, e seja $\varphi = 0$ se $S_i(\breve{\pi}, \breve{\iota}) = 0$ ou $\varphi = 1$, caso contrário. Suponha agora que $inv(\pi) = n\ell$ para algum $\ell \geq 1$. Como $\frac{m-k}{2} < n$, pelo Lema 17, temos que $d_r(\pi, \breve{\pi}, \breve{\iota}) \geq n\ell$. O Algoritmo 2 aplica $n\ell$ 2-reversões e até $m - k < n$ 1-reversões, o que resulta em não mais de $n\ell + n - 1 < n(\ell + 1)$ reversões intergênicas super curtas. □

Algoritmo 2: Uma 3-aproximação para a Ordenação de Permutações sem Sinais por Reversões Intergênicas Super Curtas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação sem sinais.
Resultado: uma sequência de reversões intergênicas super curtas $\rho_1, \rho_2, \ldots, \rho_v$ tal que $(\pi, \breve{\pi}) \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \rho_v = (\iota, \breve{\iota})$.

$j \leftarrow 1$
$\ell \leftarrow inv(\pi)$
Compute $S_i(\breve{\pi}, \breve{\iota})$
$k \leftarrow \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$
▷ Remova as inversões de π utilizando ℓ 2-reversões aplicadas sempre no mesmo bloco. Como uma heurística, sempre tentamos fazer com que $S_i(\breve{\pi}, \breve{\iota}) = 0$, o que pode aumentar o número de blocos.
enquanto *existe um par* $(\pi_i, \pi_{i+1}) \in \pi$ *que é uma inversão* **faça**
  se $S_i(\breve{\pi}, \breve{\iota}) \geq 0$ **então**
    $aux \leftarrow \max\{0, \breve{\pi}_i - S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1)}_{(aux,0)}$
  senão
    $aux \leftarrow \min\{\breve{\pi}_{i+2}, -S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1)}_{(\breve{\pi}_i,aux)}$
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Neste ponto, temos que $inv(\pi) = 0$ e podemos quebrar todo bloco não trivial em blocos triviais utilizando no máximo $m-k$ 1-reversões.
enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$ **faça**
  $op \leftarrow$ uma 1-reversão garantida pelo Lema 15
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
retorna $(\rho_1, \rho_2, \ldots, \rho_{j-1})$

Corolário 3. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) \geq n$ (resp. $inv(\pi) \geq 2n$), o Algoritmo 2 possui fator de aproximação igual a 2 (resp. 1.5).*

5.3 Ordenação de Permutações sem Sinais por Transposições Intergênicas Super Curtas (SbSST)

Vamos agora analisar a versão do problema em que apenas transposições super curtas são permitidas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação sem sinais.

O lema a seguir mostra o número de transposições necessárias para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ quando $inv(\pi) = 0$.

Lema 19. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) = 0$, temos que $d_t(\pi, \breve{\pi}, \breve{\iota}) = m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) +$*

$\mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$.

Demonstração. Se uma 2-transposição aplicada a um bloco $\mathcal{b}$ de $I(\pi, \breve{\pi}, \breve{\iota})$ aumenta o número de blocos em duas unidades, podemos assumir pela prova do Lema 16 que o bloco $\mathcal{b}$ é transformado em três blocos $\mathcal{b}'$, $\mathcal{b}''$, $\mathcal{b}'''$ tais que dois deles são ímpares e o terceiro possui a mesma paridade de $\mathcal{b}$.

Se um bloco $\mathcal{b}$ é ímpar, podemos sempre aumentar o número de blocos em duas unidades a partir deste bloco, terminando com um bloco trivial. Entretanto, se $\mathcal{b}$ é par, em algum momento teremos que aumentar o número de blocos em apenas uma unidade, criando dois blocos. Isto significa que para cada bloco par precisamos aplicar duas 2-transposições que aumentam o número de blocos em uma unidade apenas. Como sempre podemos aplicar pares de transposições que não aumentam o número de blocos pares, temos que $d_t(\pi, \breve{\pi}, \breve{\iota}) = m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$. □

Os lemas 20 e 21 mostram os limitantes inferiores e superiores para $d_t(\pi, \breve{\pi}, \breve{\iota})$, respectivamente.

Lema 20. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Temos que $d_t(\pi, \breve{\pi}, \breve{\iota}) \geq \max\{\frac{m-\mathcal{B}(I(\pi,\breve{\pi},\breve{\iota}))+\mathcal{B}_p(I(\pi,\breve{\pi},\breve{\iota}))}{2}, inv(\pi)\}$.*

Demonstração. Para transformar π em ι precisamos remover todas as inversões, e como uma 2-transposição pode remover apenas uma inversão temos, necessariamente, que $d_t(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi)$. Além disso, pelo Lema 14, podemos aumentar em no máximo duas unidades o número de blocos com uma transposição super curta. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) - \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$. Para chegar em m blocos triviais, e considerando também o Lema 19, precisamos de pelo menos $\frac{m-k}{2}$ transposições super curtas. Desta forma, $d_t(\pi, \breve{\pi}, \breve{\iota}) \geq \max\{\frac{m-k}{2}, inv(\pi)\}$. □

Lema 21. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Temos que $d_t(\pi, \breve{\pi}, \breve{\iota}) \leq inv(\pi) + m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$.*

Demonstração. Suponha que primeiro removemos todas as inversões de π utilizando $inv(\pi)$ 2-transposições do tipo $\rho^{(i,i+1,i+2)}_{(\breve{\pi}_i,0,0)}$, e seja $(\pi', \breve{\pi}')$ o genoma resultante. O valor de $\mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota}))$ não pode ser menor que $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ dado que as 2-transposições que removeram inversões são aplicadas dentro de um mesmo bloco. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) - \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$ e seja $k' = \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) - \mathcal{B}_p(I(\pi', \breve{\pi}', \breve{\iota}))$. Vamos analisar agora a paridade de qualquer bloco que uma 2-transposição quebrou: (i) se ela transforma um bloco ímpar em dois, então pelo menos um destes deve ser ímpar; (ii) se ela transforma um bloco par em dois, então ambos devem ser pares ou ímpares; (iii) se ela transforma um bloco par em três, então dois destes devem ser pares (lembre que ao transformar em três, pelo menos um deve ser trivial); e (iv) se ela transforma um bloco ímpar em três, então ou dois blocos são pares ou os três são ímpares. Isto implica que $k' \geq k$.

Pelo Lema 19, $m - k' \leq m - k$ 2-transposições são suficientes para obter m blocos. □

Utilizando os lemas 20 e 21, mostramos que existe um algoritmo de 3-aproximação para a ordenação de permutações sem sinais por transposições intergênicas super curtas.

Teorema 7. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. O valor de $d_t(\pi, \breve{\pi}, \breve{\iota})$ é 3-aproximável.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) - \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$. Se $\frac{m-k}{2} \geq inv(\pi)$ então, pelo Lema 20, $d_t(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{m-k}{2}$, e, pelo Lema 18, $d_t(\pi, \breve{\pi}, \breve{\iota}) \leq m - k + inv(\pi) \leq m - k + \frac{m-k}{2} \leq 3\frac{m-k}{2}$. Caso contrário, $\frac{m-k}{2} < inv(\pi)$, então $m - k < 2\ inv(\pi)$. Pelo Lema 20, $d_t(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi)$, e, pelo Lema 21, $d_t(\pi, \breve{\pi}, \breve{\iota}) \leq m - k + inv(\pi) \leq 2\ inv(\pi) + inv(\pi) \leq 3\ inv(\pi)$. □

O Algoritmo 3 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ (onde π é uma permutação sem sinais) utilizando transposições intergênicas super curtas, e possui um fator de aproximação igual a 3. De modo similar ao Algoritmo 2, a complexidade do Algoritmo 3 é $O(n^2)$.

Algoritmo 3: Uma 3-aproximação para a Ordenação de Permutações sem Sinais por Transposições Intergênicas Super Curtas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação sem sinais.
Resultado: uma sequência de transposições intergênicas super curtas $\rho_1, \rho_2, \ldots, \rho_v$ tal que $(\pi, \breve{\pi}) \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \rho_v = (\iota, \breve{\iota})$.

$j \leftarrow 1$
$\ell \leftarrow inv(\pi)$
Compute $S_i(\breve{\pi}, \breve{\iota})$
$k \leftarrow \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) - \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$
▷ Remova todas as inversões utilizando ℓ 2-transposições (que nunca diminuem o número de blocos). Como uma heurística, sempre tentamos fazer com que $S_i(\breve{\pi}, \breve{\iota}) = 0$, o que pode aumentar o número de blocos.
enquanto *existe um par* $(\pi_i, \pi_{i+1}) \in \pi$ *que é uma inversão* **faça**
  se $S_i(\breve{\pi}, \breve{\iota}) \geq 0$ **então**
    $aux \leftarrow \max\{0, \breve{\pi}_i - S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1,i+2)}_{(aux, \breve{\pi}_{i+1}, 0)}$
  senão
    $aux \leftarrow \min\{\breve{\pi}_{i+1}, -S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1,i+1)}_{(\breve{\pi}_i, aux, 0)}$
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Neste ponto temos que $inv(\pi) = 0$, e podemos quebrar todo bloco não trivial em blocos triviais utilizando no máximo $m - k$ 2-transposições.
enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$ **faça**
  $ops \leftarrow$ um par de 2-transposições garantido pelo Lema 19
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
retorna $(\rho_1, \rho_2, \ldots, \rho_{j-1})$

O Algoritmo 3 possui um fator de aproximação melhor quando o número de inversões é estritamente maior que n, como mostra o teorema a seguir.

Teorema 8. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) > n$, o Algoritmo 3 possui um fator de aproximação de $(1 + \frac{1}{\ell})$, onde $\ell = \frac{inv(\pi)}{n} \geq 1$.*

Demonstração. Similar à prova do Teorema 6 uma vez que o valor de $m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})) + \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$ é menor ou igual a $n + 1$. □

Corolário 4. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) \geq n$ (resp. $inv(\pi) \geq 2n$) o Algoritmo 3 possui fator de aproximação igual a 2 (resp. 1.5).*

5.4 Ordenação de Permutações sem Sinais por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas (SbSSO)

Vamos analisar a versão em que tanto reversões intergênicas super curtas quanto transposições intergênicas super curtas são permitidas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Como já sabemos como estas operações afetam o número de blocos de um grafo intergênico, os dois lemas a seguir definem limitantes inferiores e superiores para este problema.

Lema 22. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Temos que $d_{rt}(\pi, \breve{\pi}, \breve{\iota}) \geq \max\{\frac{m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))}{2}, inv(\pi)\}$.*

Demonstração. Diretamente dos lemas 14, 17 e 20. □

Lema 23. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Temos que $d_{rt}(\pi, \breve{\pi}, \breve{\iota}) \leq inv(\pi) + m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$.*

Demonstração. Suponha que começamos removendo todas as inversões em π utilizando $inv(\pi)$ 2-reversões do tipo $\rho^{(i,i+1)}_{(\breve{\pi}_i, 0)}$, e seja $(\pi', \breve{\pi}')$ o genoma resultante, onde sabemos que $\pi' = \iota$ e $\breve{\pi}' = \breve{\pi}$. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ e seja $k' = \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota}))$. Temos que $k' \geq k$, dado que as 2-reversões que removeram inversões foram aplicadas dentro de um mesmo bloco.

De maneira análoga ao Lema 21, e assumindo que $k' = k + \ell$ para algum $\ell \geq 0$, temos que $\mathcal{B}_p(I(\pi', \breve{\pi}', \breve{\iota})) \leq \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota})) + \ell$. Vamos utilizar as transposições descritas no Lema 21 para os blocos $\mathcal{b}$ onde $\mathcal{b}_r \geq 3$, aplicando assim duas 2-transposições que aumentam o número de blocos em duas unidades. Para os blocos $\mathcal{b}$ onde $\mathcal{b}_r = 2$, nós iremos aplicar uma 1-reversão conforme descrito no Lema 13, quebrando estes blocos em dois blocos triviais cada.

O procedimento descrito acima aplica $inv(\pi)$ 2-reversões, $n - k' - \mathcal{B}_p(I(\pi', \breve{\pi}', \breve{\iota}))$ 2-transposições e $\mathcal{B}_p(I(\pi', \breve{\pi}', \breve{\iota}))$ 1-reversões, o que resulta em não mais de $inv(\pi) + m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ operações. □

Agora provamos que existe um algoritmo de 3-aproximação para este problema.

Teorema 9. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. O valor de $d_{rt}(\pi, \breve{\pi}, \breve{\iota})$ é 3-aproximável.*

Demonstração. Similar à prova do Teorema 5, e utilizando os lemas 22 e 23. □

O Algoritmo 4 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ (onde π é uma permutação sem sinais) utilizando tanto reversões intergênicas super curtas quanto transposições intergênicas super curtas, e possui um fator de aproximação igual a 3. De modo similar aos algoritmos 2 e 3, este algoritmo possui complexidade de tempo de $O(n^2)$.

Como nos algoritmos apresentados anteriormente neste capítulo, o Algoritmo 4 possui um fator de aproximação melhor quando o número de inversões é pelo menos n, como explicado no teorema a seguir.

Teorema 10. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) \geq n$ o Algoritmo 4 possui um fator de aproximação de $(1 + \frac{1}{\ell})$, onde $\ell = \frac{inv(\pi)}{n} \geq 1$.*

Demonstração. Análoga à prova do Teorema 6. □

Corolário 5. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação sem sinais. Se $inv(\pi) \geq n$ (resp. $inv(\pi) \geq 2n$) o Algoritmo 4 possui fator de aproximação igual a 2 (resp. 1.5).*

5.5 Ordenação de Permutações com Sinais por Reversões Intergênicas Super Curtas (SbSigSSR)

Vamos novamente analisar a versão que permite reversões super curtas, mas agora a instância $(\pi, \breve{\pi}, \breve{\iota})$ possui π como um permutação com sinais. Note que o Lema 17, aplicado em instâncias sem sinais, pode ser utilizado como um limitante inferior para esta versão do problema.

Assim, dada a permutação com sinais π, seja S_π^{p-} o conjunto de elementos de π tal que $||\pi_i| - i|$ é par e $\pi_i < 0$, e seja S_π^{i+} o conjunto de elementos de π tal que $||\pi_i| - i|$ é ímpar e $\pi_i > 0$. Os conjuntos S_π^{p-} e S_π^{i+} capturam os elementos positivos e negativos de π que possuem sinais negativos após uma sequência de 2-reversões que colocam todos os elementos em suas posições corretas (ou seja, que remove todas as inversões). Como os elementos destes conjuntos terão sinal negativo, sabemos que operações ainda terão que ser aplicadas para torná-los positivos. Seja φ^{neg} o número de elementos em $S_\pi^{p-} \cup S_\pi^{i+}$.

O lema a seguir, provado por Galvão e coautores [27], define o número de reversões super curtas necessárias para transformar π em ι.

Lema 24. *Dada uma permutação π com sinais, são necessárias $inv(\pi) + \varphi^{neg}$ reversões super curtas para transformar π em ι.*

Este lema nos ajuda a definir um limitante inferior, como mostra o lema a seguir. Ademais, como blocos não levam em consideração os sinais dos elementos, o Lema 17 também pode ser utilizado como limitante inferior.

Lema 25. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Temos que $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota}) \geq \max\{\frac{m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))}{2}, inv(\pi) + \ell\}$, onde $\ell = \max\{\varphi, \varphi^{neg}\}$.*

Algoritmo 4: Uma 3-aproximação para a Ordenação de Permutações sem Sinais por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação sem sinais.
Resultado: uma sequência de reversões intergênicas super curtas e transposições intergênicas super curtas $\rho_1, \rho_2, \ldots, \rho_v$ tal que $(\pi, \breve{\pi}) \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \rho_v = (\iota, \breve{\iota})$.

$j \leftarrow 1$
$\ell \leftarrow inv(\pi)$
Compute $S_i(\breve{\pi}, \breve{\iota})$
▷ Remova todas as inversões de π utilizando ℓ 2-reversões (que nunca diminuem o número de blocos). Como uma heurística, sempre tentamos fazer com que $S_i(\breve{\pi}, \breve{\iota}) = 0$, o que pode aumentar o número de blocos.
enquanto *existe um par* $(\pi_i, \pi_{i+1}) \in \pi$ *que é uma inversão* **faça**
  se $S_i(\breve{\pi}, \breve{\iota}) \geq 0$ **então**
    $aux \leftarrow \max\{0, \breve{\pi}_i - S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1)}_{(aux,0)}$
  senão
    $aux \leftarrow \min\{\breve{\pi}_{i+2}, -S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1)}_{(\breve{\pi}_i,aux)}$
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Neste ponto, temos que $inv(\pi) = 0$.
$k \leftarrow \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$
$k' \leftarrow \mathcal{B}_p(I(\pi, \breve{\pi}, \breve{\iota}))$
▷ Vamos quebrar todo bloco não trivial $\mathcal{b}$ com $\mathcal{b}_r \geq 3$ em três blocos utilizando $m - k - k'$ 2-transposições (agrupada em pares).
enquanto *existe um bloco* $\mathcal{b}$ *com* $\mathcal{b}_r \geq 3$ **faça**
  $op \leftarrow$ um par de 2-transposições como explicado no Lema 19
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Agora quebramos todo bloco não trivial em blocos triviais utilizando k' 1-reversões.
enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$ **faça**
  $op \leftarrow$ uma 1-reversão como explicado no Lema 15
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
retorna $(\rho_1, \rho_2, \ldots, \rho_{j-1})$

Demonstração. Diretamente pelos lemas 17 e 24. □

O lema a seguir apresenta um limitante superior para este problema.

Lema 26. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Temos que $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota}) \leq inv(\pi) + \max\{\varphi, \varphi^{neg}\} + 2(m - \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})))$.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ e seja $\ell = \max\{\varphi, \varphi^{neg}\}$. Suponha que primeiro vamos remover todas as inversões de π usando $inv(\pi)$ 2-reversões do tipo $\rho^{(i,i+1)}_{(\breve{\pi}_i,0)}$ aplicadas em $(\pi, \breve{\pi})$, e seja $(\pi', \breve{\pi}')$ o genoma resultante.

Seja $k' = \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})$. Temos que $k' \geq k$. Agora, aplicamos $m - k' \leq m - k$ 1-reversões em $(\pi', \breve{\pi}')$ que quebram todos os blocos não triviais de $I(\pi', \breve{\pi}', \breve{\iota})$ em dois blocos conforme descrito no Lema 15, e seja $(\pi'', \breve{\pi}'')$ o genoma resultante.

Neste ponto temos que $\breve{\pi}'' = \breve{\iota}$, e π'' possui no máximo $\ell + (m - k') \leq \ell + (m - k)$ elementos negativos. Assim, aplicamos até $\ell + (m - k')$ 1-reversões do tipo $\rho^{(i,i)}_{(\breve{\pi}''_i,0)}$ (ou seja, sem modificar $\breve{\pi}$) em cada elemento negativo de π'', e o lema segue. □

Utilizando os lemas 25 e 26, provamos que existe um algoritmo de 5-aproximação para encontrar o valor de $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota})$.

Teorema 11. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. O valor de $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota})$ é 5-aproximável.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, e seja $\ell = \max\{\varphi, \varphi^{neg}\}$. Se $\frac{m-k}{2} \geq inv(\pi) + \ell$, então pelo Lema 25, $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\pi}) \geq \frac{m-k}{2}$, e, pelo Lema 26, $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota}) \leq 2(m-k) + inv(\pi) + \ell \leq 2(m-k) + \frac{m-k}{2} \leq 5\frac{m-k}{2}$.

Caso contrário, $\frac{m-k}{2} < inv(\pi) + \ell$, então $2(m-k) < 4(inv(\pi) + \ell)$. Pelo Lema 25, $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi) + \ell$, e, pelo Lema 26, $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota}) \leq 2(m-k) + inv(\pi) + \ell \leq 4(inv(\pi) + \ell) + inv(\pi) + \ell \leq 5(inv(\pi) + \ell)$. □

O Algoritmo 5 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ (onde π é uma permutação com sinais) utilizando reversões intergênicas super curtas, e possui um fator de aproximação igual a 5. Como nos algoritmos anteriores, a complexidade de tempo do Algoritmo 5 é $O(n^2)$.

O Algoritmo 5 possui um fator de aproximação melhor quando o número de inversões de uma instância $(\pi, \breve{\pi}, \breve{\iota})$ é pelo menos n, como explicamos no teorema a seguir.

Teorema 12. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Se $inv(\pi) \geq n$, o Algoritmo 4 possui fator de aproximação de $(1 + \frac{2}{\ell})$, onde $\ell = \frac{inv(\pi)}{n} \geq 1$.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$. Suponha agora que $inv(\pi) = n\ell$ para algum $\ell \geq 1$. Como $\frac{m-k}{2} < n$, pelo Lema 17, temos que $d_{\bar{r}}(\pi) \geq n\ell$. O Algoritmo 5 aplica $n\ell$ 2-reversões, até $m - k < n$ 1-reversões, e até n 1-reversões para trocar o sinal de cada elemento negativo, o que resulta em não mais de $n\ell + n - 1 + n < n(\ell + 2)$ reversões intergênicas super curtas. □

Corolário 6. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Se $inv(\pi) \geq n$ (resp. $inv(\pi) \geq 2n$) o Algoritmo 4 possui fator de aproximação igual a 3 (resp. 2).*

Algoritmo 5: Uma 5-aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas Super Curtas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação com sinais.
Resultado: uma sequência de reversões intergênicas super curtas $\rho_1, \rho_2, \ldots, \rho_v$ tal que $(\pi, \breve{\pi}) \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \rho_v = (\iota, \breve{\iota})$.

$j \leftarrow 1$
$\ell \leftarrow inv(\pi)$
Compute $S_i(\breve{\pi}, \breve{\iota})$
▷ Remova todas as inversões de π utilizando ℓ 2-reversões (que nunca diminuem o número de blocos). Como uma heurística, sempre tentamos fazer com que $S_i(\breve{\pi}, \breve{\iota}) = 0$, o que pode aumentar o número de blocos.
enquanto *existe um par* $(\pi_i, \pi_{i+1}) \in \pi$ *que é uma inversão* **faça**
  se $S_i(\breve{\pi}, \breve{\iota}) \geq 0$ **então**
    $aux \leftarrow \max\{0, \breve{\pi}_i - S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1)}_{(aux,0)}$
  senão
    $aux \leftarrow \min\{\breve{\pi}_{i+2}, -S_i(\breve{\pi}, \breve{\iota})\}$
    $op \leftarrow \rho^{(i,i+1)}_{(\breve{\pi}_i,aux)}$
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Neste ponto $inv(\pi) = 0$.
$k \leftarrow \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$
▷ Quebre todo bloco não trivial em dois utilizando $m - k$ 1-reversões.
enquanto *existe um bloco* $\mathcal{b}$ *não trivial em* $I(\pi, \breve{\pi}, \breve{\iota})$ **faça**
  $op \leftarrow$ uma 1-reversão garantida pelo Lema 15
  $\rho_j \leftarrow op$
  $j \leftarrow j + 1$
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Agora aplicamos até $\varphi^{neg} + m - k$ 1-reversões em cada elemento negativo de π.
para $\pi_i \in \pi$ **faça**
  se $\pi_i < 0$ **então**
    $op \leftarrow \rho^{(i,i)}_{(\breve{\pi}_i,0)}$
    $\rho_j \leftarrow op$
    $j \leftarrow j + 1$
    $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
retorna $(\rho_1, \rho_2, \ldots, \rho_{j-1})$

5.6 Ordenação de Permutações com Sinais por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas (SbSigSSO)

Vamos novamente analisar a versão que permite reversões intergênicas super curtas e transposições intergênicas super curtas, mas agora a instância $(\pi, \breve{\pi}, \breve{\iota})$ possui π como um permutação com sinais.

Seja φ^n o número de componentes ímpares do grafo de inversões $IG(\pi)$ definido na Seção 4.2.1. O lema a seguir, provado por Galvão e coautores [27], mostra o número exato de reversões super curtas e transposições super curtas suficientes para transformar π em ι.

Lema 27. *Dada uma permutação com sinais π, $inv(\pi) + \varphi^n$ operações super curtas são suficientes para transformar π em ι.*

Este lema e o Lema 17 nos ajudam a definir um limitante inferior para o problema, como mostra o lema a seguir.

Lema 28. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \geq \max\{\frac{m-\mathcal{B}(I(\pi,\breve{\pi},\breve{\iota}))}{2}, inv(\pi) + \ell\}$, onde $\ell = \max\{\varphi, \varphi^n\}$.*

Demonstração. Diretamente pelos lemas 17 e 24. □

O lema a seguir mostra um limitante superior para nosso problema.

Lema 29. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \leq inv(\pi)+\varphi^n+2(m-\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota})))$.*

Demonstração. Suponha que primeiro removemos todas as inversões de π utilizando o algoritmo exato em tempo polinomial apresentado por Galvão e coautores [27], que utiliza $inv(\pi)+\varphi^n$ operações super curtas tal que toda 1-reversão é ignorada, todas as 2-reversões são do tipo $\rho^{(i,i+1)}_{(\breve{\pi}_i,0)}$, e todas as 2-transposições são do tipo $\rho^{(i,i+1,i+2)}_{(\breve{\pi}_i,0,0)}$. Seja $(\pi', \breve{\pi}')$ o genoma resultante. Como as 1-reversões foram ignoradas, foram aplicadas exatamente $inv(\pi)$ operações.

O número de blocos em $I(\pi', \breve{\pi}', \breve{\iota})$ não pode ser menor que $\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, dado que as 2-reversões e 2-transposições são aplicadas dentro de um mesmo bloco. Seja $k' = \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota})) \geq \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$.

Pelo Lema 15, $m - k' \leq m-\mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$ 1-reversões são suficientes para obter m blocos. O genoma $(\pi'', \breve{\pi}'')$ é tal que π'' possui no máximo $\min\{m - k' + \varphi^n, n\}$ elementos negativos, e podemos aplicar $\min\{m - k' + \varphi^n, n\}$ 1-reversões do tipo $\rho^{(i,i)}_{(\breve{\iota}_i,0)}$ em $(\pi'', \breve{\pi}'')$ para cada elemento negativo de π'' na i-ésima posição. □

Utilizando os lemas 28 e 29, provamos que é possível obter um algoritmo de 5-aproximação para encontrar o valor de $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota})$.

Teorema 13. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. O valor de $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota})$ é 5-aproximável.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$, e seja $\ell = \max\{\varphi, \varphi^n\}$. Se $\frac{m-k}{2} \geq inv(\pi)+\ell$ então, pelo Lema 28, temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{m-k}{2}$, e, pelo Lema 29, temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \leq 2(m-k) + inv(\pi) + \ell \leq 2(m-k) + \frac{m-k}{2} \leq 5\frac{m-k}{2}$.

Caso contrário, $\frac{m-k}{2} < inv(\pi) + \ell$, então $2(m-k) < 4(inv(\pi) + \ell)$. Pelo Lema 28, temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \geq inv(\pi) + \ell$, e, pelo Lema 29, temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \leq 2(m-k) + inv(\pi) + \ell \leq 4(inv(\pi) + \ell) + inv(\pi) + \ell \leq 5(inv(\pi) + \ell)$. □

O Algoritmo 6 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ (onde π é uma permutação com sinais) utilizando reversões intergênicas super curtas e transposições intergênicas super curtas, e possui um fator de aproximação igual a 5. Com relação à complexidade, pelos algoritmos anteriores sabemos que as linhas 7-16 possuem complexidade de tempo de $O(n^2)$, e o laço da linha 3 possui complexidade de tempo de $O(n^3)$ [27], que é então a complexidade de tempo do Algoritmo 6.

Como nos demais algoritmos, o Algoritmo 6 também possui fator de aproximação melhor quando o número de inversões é pelo menos n, como explicamos no teorema a seguir.

Teorema 14. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Se $inv(\pi) \geq n$, o Algoritmo 6 possui um fator de aproximação de $(1 + \frac{2}{\ell})$, onde $\ell = \frac{inv(\pi)}{n} \geq 1$.*

Demonstração. Seja $k = \mathcal{B}(I(\pi, \breve{\pi}, \breve{\iota}))$. Suponha agora que $inv(\pi) = n\ell$. Como $\frac{m-k}{2} < n$, pelo Lema 17 temos que $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \geq \ell$. O Algoritmo 5 aplica $n\ell$ operações (entre 2-reversões e 2-transposições), até $m - k < n$ 1-reversões, e até n 1-reversões para trocar os sinais dos elementos negativos, o que resulta em não mais de $n\ell + n - 1 + n < n(\ell + 2)$ operações super curtas. □

Corolário 7. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem m regiões intergênicas cada, e π é uma permutação com sinais. Se $inv(\pi) \geq n$ (resp. $inv(\pi) \geq 2n$) o Algoritmo 6 possui fator de aproximação igual a 3 (resp. 2).*

5.7 Experimentos

Nós implementamos os cinco algoritmos propostos e os testamos em dados simulados para investigar como eles se comportam na prática. Nós geramos dois conjuntos de dados diferentes de instâncias, os quais definimos como instâncias totalmente aleatórias (FRI), e instâncias quase aleatórias (ARI). Cada conjunto de dados possui 1.000.000 de instâncias $(\pi, \breve{\pi}, \breve{\iota})$, onde π é uma permutação com 100 elementos e $\breve{\pi}$ e $\breve{\iota}$ são sequências de tamanhos das 101 regiões intergênicas.

O conjunto de dados FRI foi gerado da seguinte maneira: (i) seja $(\iota, \breve{\iota})$ um par inicial, sendo ι com 100 elementos e cada $\breve{\iota}_i$ um número inteiro aleatório no intervalo [0..100]; (ii) geramos uma instância $(\pi, \breve{\pi}, \breve{\iota})$ aplicando w operações intergênicas super curtas em $(\iota, \breve{\iota})$, cujos índices foram gerados aleatoriamente, tanto para as posições quanto tamanhos intergênicos, sempre respeitando os valores existentes a cada iteração. Criamos 10.000 instâncias para cada valor de $w \in \{10, 20, 30, \ldots, 990, 1000\}$.

Algoritmo 6: Uma 5-aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação com sinais.
Resultado: uma sequência de reversões intergênicas super curtas e transposições intergênicas super curtas $\rho_1, \rho_2, \ldots, \rho_v$ tal que $(\pi, \breve{\pi}) \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \rho_v = (\iota, \breve{\iota})$.

$j \leftarrow 1$
$\ell \leftarrow inv(\pi)$
▷ Remova todas as inversões de π utilizando ℓ operações intergênicas super curtas de acordo com o algoritmo apresentado por Galvão e coautores [27], ignorando qualquer 1-reversão desta sequência. Como uma heurística, sempre tentamos fazer com que $S_i(\breve{\pi}, \breve{\iota}) = 0$, o que pode aumentar o número de blocos.
enquanto *existe um par* $(\pi_i, \pi_{i+1}) \in \pi$ *que é uma inversão* **faça**
    $op \leftarrow$ uma 2-reversão ou uma 2-transposição que remove uma inversão [27]
    $\rho_j \leftarrow op$
    $j \leftarrow j + 1$
    $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot op$
▷ Seja $(\pi', \breve{\pi}')$ o genoma resultante. Temos que $inv(\pi') = 0$, e existem exatamente φ^n elementos negativos em π'.
$k \leftarrow \mathcal{B}(I(\pi', \breve{\pi}', \breve{\iota}))$
▷ Quebra blocos não triviais em dois utilizando $m - k < n$ 1-reversões.
enquanto *existe um bloco* $\mathcal{b}$ *não trivial em* $I(\pi, \breve{\pi}, \breve{\iota})$ **faça**
    $op \leftarrow$ uma 1-reversão como descrito no Lema 15
    $\rho_j \leftarrow op$
    $j \leftarrow j + 1$
    $(\pi', \breve{\pi}') \leftarrow (\pi', \breve{\pi}') \cdot op$
▷ Agora aplicamos no máximo n 1-reversões em cada elemento negativo de π'.
para cada $\pi'_i \in \pi'$ **faça**
    se $\pi'_i < 0$ **então**
        $op \leftarrow \rho^{(i,i)}_{(\breve{\pi}'_i, 0)}$
        $\rho_j \leftarrow op$
        $j \leftarrow j + 1$
        $(\pi', \breve{\pi}') \leftarrow (\pi', \breve{\pi}') \cdot op$
retorna $(\rho_1, \rho_2, \ldots, \rho_{j-1})$

Para a Ordenação de Permutações por Reversões Super Curtas, para ambos os casos (com e sem sinais), foram aplicadas $0.8w$ 2-reversões e $0.2w$ 1-reversões, e a cada iteração uma operação foi escolhida ao acaso enquanto ambas estavam disponíveis. Para a Ordenação de Permutações sem Sinais por Transposições Intergênicas Super Curtas nós aplicamos w 2-transposições. Já para a Ordenação de Permutações por Reversões Intergênicas Super Curtas e Transposições Intergênicas Super Curtas, para ambos os casos com e sem sinais, foram aplicadas $0.5w$ 2-transposições, $0.4w$ 2-reversões e $0.1w$ 1-reversões, e a cada iteração uma operação foi escolhida ao acaso enquanto mais de uma estava

disponível.

O conjunto de dados ARI foi gerado de maneira similar ao FRI. Entretanto, quando o algoritmo fosse aplicar uma 2-reversão ou uma 2-transposição um par era escolhido ao acaso considerando apenas os pares de elementos consecutivos que não eram uma inversão. Como $w < \max_{\pi \in \delta_n}\{inv(\pi)\} = \frac{n(n-1)}{2}$, pelo menos um par sempre existia.

Dada qualquer instância $(\pi, \breve{\pi}, \breve{\iota})$ de ARI criada utilizando w operações intergênicas super curtas, sabemos que o valor de $inv(\pi)$ é igual ao número de 2-reversões e 2-transposições aplicadas. Já o número de inversões das instâncias FRI não é conhecido, mas existe uma fórmula que nos dá o valor esperado de inversões em uma permutação de tamanho n após k trocas adjacentes randômicas (ou seja, 2-reversões e 2-transposições) aplicadas à permutação identidade [9]:

$$E[i_{n,k}] = \frac{n(n+1)}{4} - \frac{1}{8(n+1)^2} \sum_{i,j=0}^{n} \frac{(c_j + c_i)^2}{s_k^2 s_i^2} x_{ji}^k,$$

onde $c_a = \cos \alpha_a$, $s_a = \sin \alpha_a$, $x_{ab} = 1 - \frac{4}{n}(1 - c_a c_b)$, e $\alpha_a = \frac{(2a+1)\pi}{2n+2}$.

As figuras 5.4 e 5.5 mostram os resultados experimentais para as instâncias de FRI e ARI, respectivamente. Mostramos as distâncias médias retornadas para cada algoritmo apresentado neste capítulo, além do fator de aproximação experimental médio e máximo, utilizando como base os limitantes inferiores dos problemas de cada instância.

Nas figuras 5.4 e 5.5, o Algoritmo 2 é denotado por SbSSR, o Algoritmo 5 é denotado por SbSigSSR, o Algoritmo 3 é denotado por SbSST, o Algoritmo 4 é denotado por SbSSO, e o Algoritmo 6 é denotado por SbSigSSO.

Na Figura 5.4, a curva Inv1 denota o número esperado de inversões das instâncias de SbSSR e SbSigSSR, a curva Inv2 denota o número esperado de inversões das instâncias de SbSST, e a curva Inv3 denota o número esperado de inversões das instâncias de SbSSO e SbSigSSO. Estas três curvas foram geradas utilizando a fórmula $E[i_{n,k}]$ descrita acima. Na Figura 5.5, as curvas Inv1, Inv2 e Inv3 seguem a mesma ideia das curvas pontilhadas da Figura 5.4, mas ao invés do número esperado de inversões elas denotam o número exato de inversões.

Como as distâncias destes problemas estão diretamente relacionadas ao número de inversões, na Figura 5.4(a) podemos ver que, apesar de na prática aplicar até 1.000 operações para gerar as instâncias, os valores retornados pelos algoritmos não passam de 300 operações na média, e a distância média retornada para cada algoritmo de fato segue a tendência das linhas pontilhadas representando o número esperado de inversões para as instâncias. Os algoritmos para permutações com sinais retornaram distâncias em média um pouco maiores que os algoritmos para permutações sem sinais, o que é esperado dado que além das inversões e regiões intergênicas esses algoritmos também precisam lidar com sinais negativos.

Com relação às aproximações experimentais apresentadas nas figuras 5.4(b-c), podemos notar que apesar do fator de aproximação teórico igual a 3 e 5, as aproximações experimentais médias das soluções para as instâncias de FRI ficaram entre 1.0 e 2.2. Além disso, em nossos testes com os algoritmos SbSSR e SbSSO, cujo fator de aproximação teórico é igual a 3, eles não retornaram uma solução cujo fator de aproximação

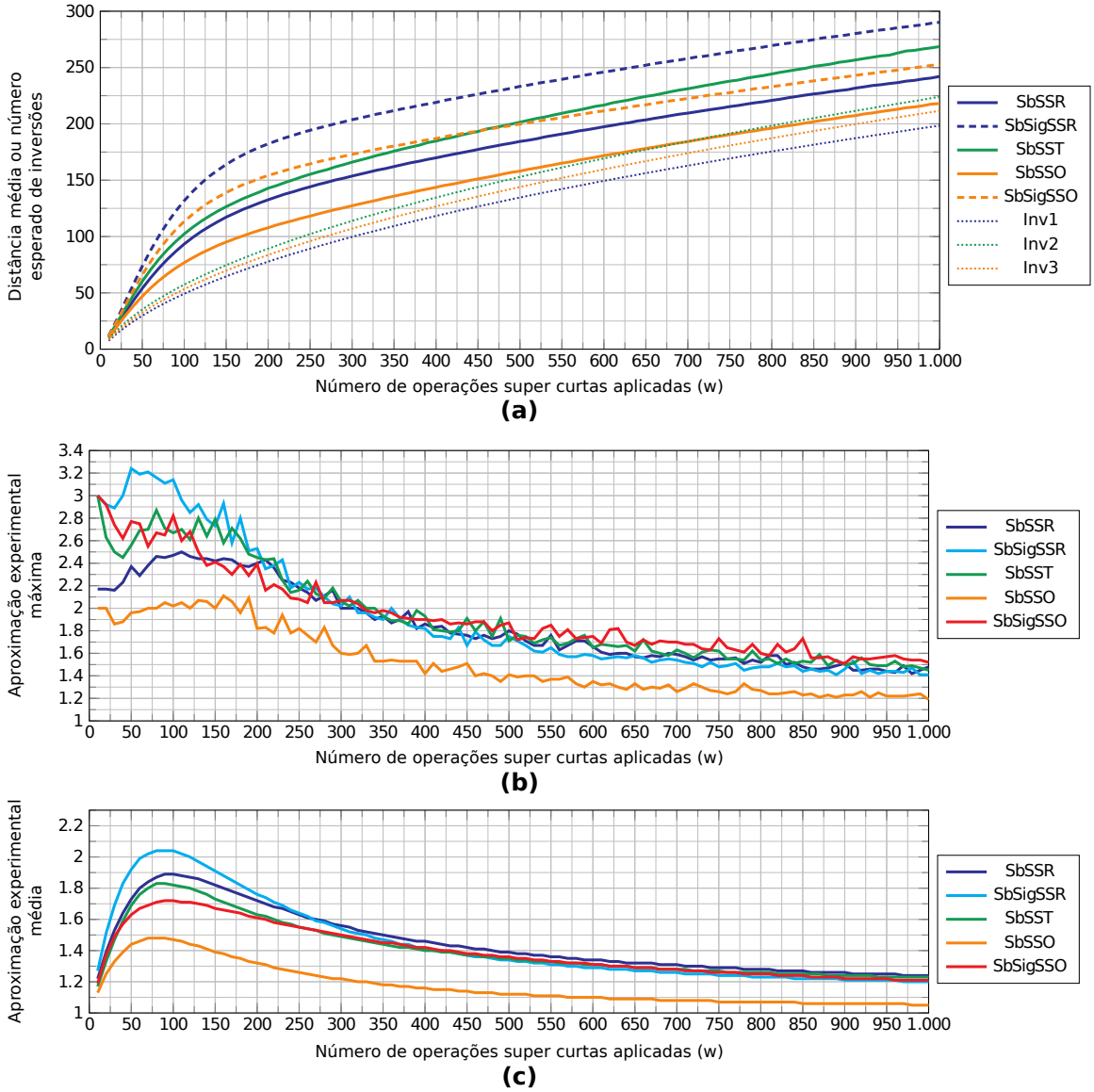


Figura 5.4: (a) distância média, (b) fator de aproximação experimental máximo, e (c) fator de aproximação experimental médio retornadas para instâncias de FRI. Em (a), as curvas pontilhadas representam o número esperado de inversões, as curvas tracejadas representam algoritmos para permutações com sinais e as curvas contínuas representam algoritmos para permutações sem sinais. As cores relacionam problemas com o mesmo número esperado de inversões dado pelas curvas pontilhadas de mesma cor. Isto significa que é que esperado que SbSSR e SBSIgSSR tenham o número de inversões dado pela curva Inv1; é esperado que SbSST tenha o número de inversões dado pela curva Inv2; e é esperado que SbSSO e SbSigSSO tenham número de inversões dado pela curva Inv3. As aproximações experimentais em (b) e (c) foram calculadas levando em consideração os limitantes inferiores dos problemas.

experimental máximo ficasse acima de 2.5 para nenhuma instância testada, e com os algoritmos SbSigSSR e SbSigSSO, cujo fator de aproximação teórico é 5, eles não retornaram uma solução cujo fator de aproximação experimental máximo ficasse acima de 3.3 e 3.0 para nenhuma instância testada, respectivamente.

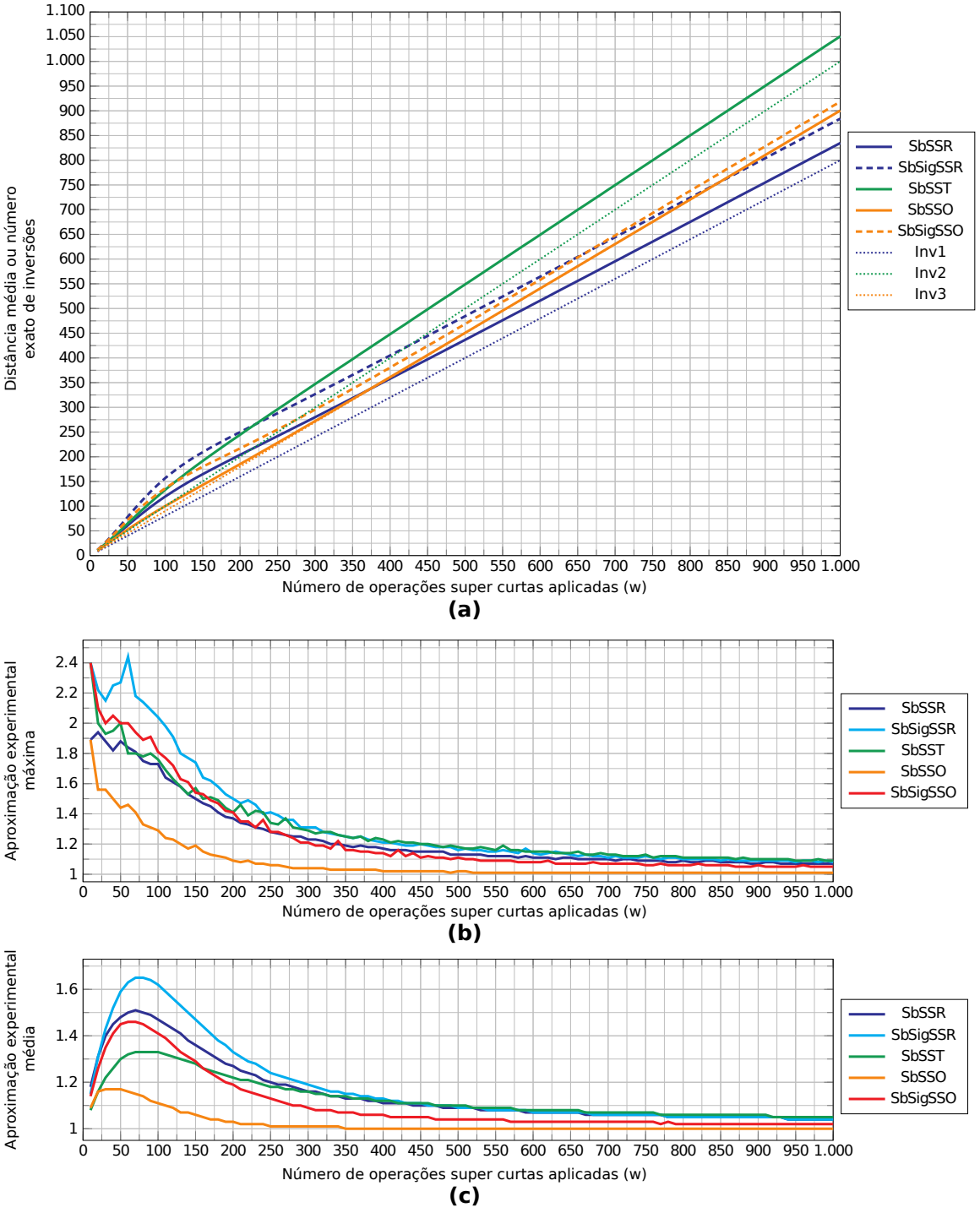


Figura 5.5: (a) distância média, (b) fator de aproximação experimental máximo, e (c) fator de aproximação experimental médio retornadas para instâncias de FRI. Em (a), as curvas pontilhadas representam o número exato de inversões, as curvas tracejadas representam algoritmos para permutações com sinais e as curvas contínuas representam algoritmos para permutações sem sinais. As cores relacionam problemas com o mesmo número esperado de inversões dado pelas curvas pontilhadas de mesma cor. Isto significa que SbSSR e SbSigSSR possuem o número de inversões dado pela curva Inv1; SbSST possui o número de inversões dado pela curva Inv2l e SbSSO e SbSigSSO possuem o número de inversões dado pela curva Inv3. As aproximações experimentais em (b) e (c) foram calculadas levando em consideração os limitantes inferiores dos problemas.

Na Figura 5.5(a), temos outro cenário em que as distâncias médias devolvidas seguem de fato o número de operações super curtas aplicadas para gerar as instâncias, mas este comportamento é devido a nossa escolha de aplicar apenas operações que não removem inversões previamente criadas. Um ponto interessante desta imagem é que as distâncias devolvidas pelos algoritmos ficaram muito próximas ao número de inversões, especialmente quando $w \geq 400$, o que não ocorreu com as instâncias de FRI. Na Figura 5.5(b-c), podemos ver que as soluções dos algoritmos com este conjunto de dados possuem aproximações experimentais sistematicamente melhores do que as instâncias de FRI: os algoritmos não obtiveram um fator de aproximação experimental máximo acima de 2.5 para nenhuma instância, e todos os algoritmos possuem um fator aproximação experimental médio abaixo de 1.7. Para $w \geq 120$ (resp. $w \geq 240$), onde esperamos que as instâncias possuam cerca de n (resp. $2n$) inversões, nenhum algoritmo obteve um fator de aproximação experimental máximo acima de 2 (resp. 1.5), o que já havia sido mostrado nos lemas 6, 8, 10, 12 e 14.

5.8 Conclusões

Neste capítulo analisamos o número mínimo de operações intergênicas super curtas necessárias para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação com ou sem sinais.

Nós definimos alguns limitantes e uma estrutura em grafo que nos permitiu construir cinco algoritmos (sendo um para cada problema) que garantem uma aproximação de 3 em permutações sem sinais (utilizando reversões intergênicas super curtas, transposições intergênicas super curtas ou ambos) e 5 em permutações com sinais (utilizando reversões intergênicas super curtas com ou sem transposições intergênicas super curtas). Estes algoritmos possuem um fator de aproximação melhor se o número de inversões é pelo menos n ou $2n$. No primeiro caso eles se tornam 2 em permutações sem sinais e 3 em permutações com sinais; no segundo caso eles se tornam 1.5 em permutações sem sinais e 2 em permutações com sinais.

Os cinco algoritmos foram implementados e testados em conjuntos de instâncias simuladas, mostrando que as aproximações experimentais médias dos algoritmos é melhor que seus fatores de aproximação teóricos.

Capítulo 6

Ordenação de Permutações por Operações Intergênicas

Este capítulo apresenta resultados para a Ordenação de Permutações em modelos que consideram operações intergênicas, mas desta vez sem a restrição de aplicar apenas operações super curtas. Começamos com as definições utilizadas neste capítulo e, em seguida, apresentamos algoritmos quando o modelo permite (i) reversões em permutações com sinais, (ii) transposições em permutações sem sinais e (iii) reversões e transposições em permutações com sinais. Parte dos resultados deste capítulo foi publicado nos anais da *7th International Conference on Algorithms for Computational Biology* em 2020 [49].

6.1 O Grafo de Breakpoints Ponderado

O *Grafo de Breakpoints* [29] é uma estrutura muito utilizada em problemas de ordenação de permutações por rearranjos de genomas. Nós o adaptamos para representar, em um único grafo, uma instância $(\pi, \breve{\pi}, \breve{\iota})$. Todas as definições propostas aqui são exemplificadas nas figuras 6.1 e 6.2.

O *Grafo de Breakpoints Ponderado*, denotado por $G(\pi, \breve{\pi}, \breve{\iota}) = (V, E, w)$, é um grafo onde V é o conjunto $\{-(n+1), -n, \ldots, -2, -1, 0, 1, 2, \ldots, n\}$, E é o conjunto de arestas pretas e cinzas, e $w : E \to \mathbb{N}$ é uma função que mapeia arestas aos tamanhos das regiões intergênicas correspondentes.

O conjunto de arestas pretas é dado por $\{e_i = (-\pi_i, +\pi_{i-1}) : 1 \leq i \leq n+1\}$, e o *peso* de uma aresta e_i é dado por $w(e_i) = \breve{\pi}_i$. O conjunto de arestas cinzas é dado por $\{e'_i = (+(i-1), -i) : 1 \leq i \leq n+1\}$, e o *peso* de uma aresta e'_i é dado por $w(e'_i) = \breve{\iota}_i$. Note que esta definição utiliza permutações em sua forma estendida (veja a Figura 6.2(a)).

Este grafo pode ser desenhado de muitas maneiras, mas seguiremos um padrão de colocar seus vértices em uma linha horizontal na mesma ordem que os elementos de π. Além disso, para cada $\pi_i \in \pi$, o vértice $-\pi_i \in V(G(\pi, \breve{\pi}, \breve{\iota}))$ é desenhado à esquerda do vértice $+\pi_i$. Como as arestas pretas estão diretamente relacionadas a π, elas são desenhadas como linhas horizontais, e nós rotulamos a aresta preta e_i como i. Já as arestas cinzas são desenhadas como arcos.

Para cada vértice v de $V(G(\pi, \breve{\pi}, \breve{\iota}))$, existe exatamente uma aresta cinza e uma aresta

preta incidentes a v, o que permite uma decomposição única das arestas em ciclos de cores alternadas. Cada ciclo C com k arestas pretas é representado como uma lista $(c^1, c^2, \ldots, c^k)$ dos rótulos de suas arestas pretas, e, de modo a tornar esta notação única, assumimos que c^1 é o rótulo da aresta preta mais à direita (utilizando a representação proposta acima) de C, e nós percorremos esta aresta da direita para a esquerda. Além disso, se ao percorrer um ciclo começando o trajeto na aresta e_{c^1} da direita para a esquerda encontrarmos uma aresta preta e_{c^i} que é percorrida da esquerda para a direita, colocamos um sinal negativo ('-') antes de seu rótulo na lista e dizemos que e_{c^i} é *divergente*; uma aresta preta percorrida da direita para a esquerda é chamada *convergente*. Como convencionamos iniciar o trajeto na aresta e_{c^1} da direita para a esquerda, esta aresta será sempre convergente. Vamos agora apresentar uma série de definições utilizando ciclos.

Um ciclo é *longo* caso possua três ou mais arestas pretas; um ciclo é *curto* caso possua duas arestas pretas; um ciclo é *trivial* caso possua apenas uma aresta preta; um ciclo é *não trivial* se for curto ou longo.

Um ciclo não trivial $C = (c^1, \ldots, c^k)$ é *convergente* se todas as arestas pretas $e_{c^i} \in C$ são convergentes; e C é *divergente* caso contrário. Dado um ciclo divergente C, um par de arestas pretas (e_{c^x}, e_{c^y}) de C é dito um *par divergente* se uma aresta é divergente e a outra é convergente. Todo ciclo divergente possui pelo menos um par divergente (por exemplo, e_{c^1}, que sempre é convergente, e qualquer aresta divergente e_{c^i}, que deve existir em C por definição).

Um ciclo não trivial convergente $C = (c^1, \ldots, c^k)$ é *não orientado* se $c^1, \ldots, c^k$ forma uma sequência decrescente; caso contrário, o ciclo C é *orientado*.

Dado um ciclo não trivial convergente $C = (c^1, \ldots, c^k)$, um par de arestas pretas e_{c^i} e $e_{c^{(i+1)}}$, com $1 \leq i < k$, é um *open gate* se para todo $c^j \in C$ com $j \notin \{i, i{+}1\}$ ou $c^j > c^i$ ou $c^j < c^{i+1}$. Além disso, o par de arestas pretas e_{c^1} e e_{c^k} é um *open gate* se $c^1 > c^j > c^k$ para qualquer $c^j \in C$ tal que $j \notin \{1, k\}$. Bafna e Pevzner [3] mostraram que para todo open gate e_{c^i} e e_{c^j} de um ciclo C com $c^i > c^j$ existe outro ciclo não trivial D com arestas pretas e_{d^i} e e_{d^j} tal que (i) $d^i > d^j$; (ii) existe uma aresta cinza entre os extremos de e_{d^i} e e_{d^j}; e (iii) $c^i > d^i > c^j > d^j$ ou $d^i > c^i > d^j > c^j$. Neste caso, dizemos que o ciclo D *fecha* este open gate de C (veja exemplos na Figura 6.1).

Ciclos de $G(\pi, \breve{\pi}, \breve{\iota})$ são classificados em *balanceados* e *desbalanceados*. Um ciclo $C = (c^1, \ldots, c^k)$ é dito *balanceado* se $\sum_{i=1}^{k}[w(e'_{c^i}) - w(e_{c^i})] = 0$, e é dito *desbalanceado* caso contrário. Em outras palavras, um ciclo é balanceado se a soma dos pesos de suas arestas cinzas é igual a soma dos pesos de suas arestas pretas, e é desbalanceado caso contrário. Um ciclo desbalanceado é chamado de *positivo* se $\sum_{i=1}^{k}[w(e'_{c^i}) - w(e_{c^i})] > 0$, e de *negativo* caso contrário.

Denotamos por $c(\pi, \breve{\pi}, \breve{\iota})$ e $c_b(\pi, \breve{\pi}, \breve{\iota})$ o número de ciclos e ciclos balanceados em $G(\pi, \breve{\pi}, \breve{\iota})$, respectivamente.

Propriedade 5. *A instância $(\iota, \breve{\iota}, \breve{\iota})$ possui duas propriedades não existentes em qualquer outra instância: (i) $c(\iota, \breve{\iota}, \breve{\iota}) = n + 1$, e (ii) $c_b(\iota, \breve{\iota}, \breve{\iota}) = n + 1$.*

Dada uma sequência de operações $\mathcal{S}_\rho = (\rho_1, \ldots, \rho_k)$, seja $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot \mathcal{S}_\rho$. Denotamos por $\Delta c(\pi, \breve{\pi}, \breve{\iota}, \mathcal{S}_\rho) = c(\pi', \breve{\pi}', \breve{\iota}) - c(\pi, \breve{\pi}, \breve{\iota})$ e $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \mathcal{S}_\rho) = c_b(\pi', \breve{\pi}', \breve{\iota}) - c_b(\pi, \breve{\pi}, \breve{\iota})$

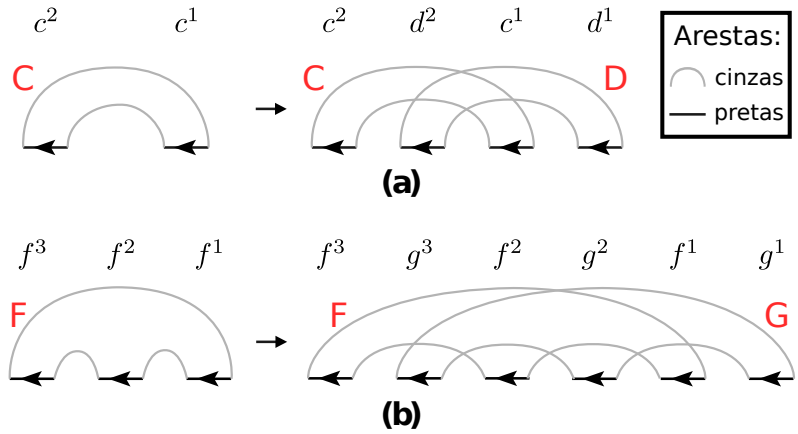


Figura 6.1: **(a)** O ciclo $C = (c^1, c^2)$ à esquerda, no qual o par (e_{c^1}, e_{c^2}) é um open gate fechado pelo ciclo $D = (d^1, d^2)$ à direita. **(b)** O ciclo $F = (f^1, f^2, f^3)$ à esquerda no qual os pares (e_{f^1}, e_{f^2}), (e_{f^2}, e_{f^3}) e (e_{f^3}, e_{f^1}) são open gates; estes open gates são fechados pelo ciclo $G = (g^1, g^2, g^3)$ à direita.

a variação no número de ciclos e ciclos balanceados, respectivamente, quando a sequência $\mathcal{S}_\rho$ é aplicada em $(\pi, \breve{\pi})$.

Lema 30. $\Delta c(\pi, \breve{\pi}, \breve{\iota}, \rho) \in \{1, 0, -1\}$ *para qualquer reversão* ρ.

Demonstração. Diretamente do Teorema 1 de Bafna e Pevzner [2]. □

Lema 31. $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) \leq 1$ *para qualquer reversão* ρ.

Demonstração. Pelo Lema 30, sabemos que podemos aumentar o número de ciclos em no máximo uma unidade. Neste cenário, um ciclo C é dividido em dois por ρ. Se C é um ciclo balanceado, o melhor que podemos esperar é que ρ gere dois ciclos balanceados e, desta forma, $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 1$. Caso contrário, C é desbalanceado e pelo menos um dos ciclos resultantes deve ser desbalanceado também, e o melhor que podemos esperar é que o outro ciclo gerado seja balanceado, e $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 1$.

Quando ρ não altera o número de ciclos temos que $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 0$. Por fim, se ρ diminui o número de ciclos em uma unidade, o melhor que podemos esperar é que esta reversão gere um ciclo balanceado a partir de dois ciclos desbalanceados (um negativo e outro positivo), e $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 1$. □

Lema 32. $\Delta c(\pi, \breve{\pi}, \breve{\iota}, \rho) \in \{2, 0, -2\}$ *para qualquer transposição* ρ.

Demonstração. Diretamente de Bafna e Pevzner [3]. □

Lema 33. $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) \leq 2$ *para qualquer transposição* ρ.

Demonstração. Pelo Lema 32, sabemos que podemos aumentar o número de ciclos em no máximo duas unidades com uma transposição. Neste cenário, um ciclo C é quebrado em três com uma transposição ρ. Se C é um ciclo balanceado, o melhor que podemos esperar é que ρ gera três ciclos balanceados, então $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 2$. Caso contrário, C é desbalanceado e pelo menos um dos ciclos gerados deve ser desbalanceado, dado que a soma dos pesos das arestas pretas dos três ciclos é diferente da soma das arestas cinzas destes ciclos. Desta forma, o melhor que podemos esperar é que os outros dois ciclos sejam balanceados, e, assim, $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 2$. □

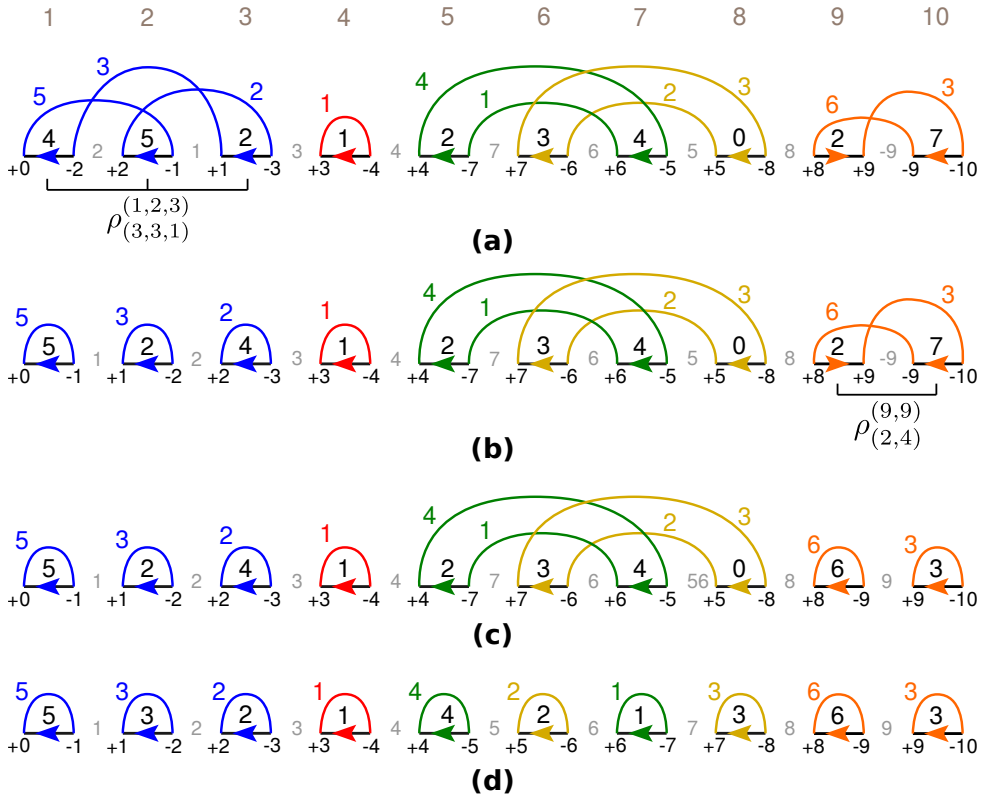


Figura 6.2: **(a)** O grafo de breakpoints ponderado $G(\pi, \breve{\pi}, \breve{\iota})$ onde $\pi = (+2\ +1\ +3\ +4\ +7\ +6\ +5\ +8\ -9)$, $\breve{\pi} = (4\ 5\ 2\ 1\ 2\ 3\ 4\ 0\ 2\ 7)$, e $\breve{\iota} = (5\ 3\ 2\ 1\ 4\ 2\ 1\ 3\ 6\ 3)$. Nós colorimos o grafo para melhor visualização dos ciclos. As arestas pretas são representadas pelas linhas horizontais e as arestas cinzas por arcos. As setas sobre as arestas pretas indicam como elas são percorridas e os números na cor cinza no topo da figura indicam os rótulos das arestas pretas abaixo deles. Números na cor preta sobre arestas pretas (resp. cinzas) indicam o tamanho das regiões intergênicas de $\breve{\pi}$ (resp. $\breve{\iota}$) que elas representam; números na cor preta abaixo das arestas pretas representas os vértices de $G(\pi, \breve{\pi}, \breve{\iota})$; e π é descrita em vermelho abaixo dos vértices. Note que $G(\pi, \breve{\pi}, \breve{\iota})$ possui cinco ciclos: $A = (10, -9)$ é um ciclo balanceado divergente curto; $B = (8, 6)$ é um ciclo positivo não orientado curto; $C = (7, 5)$ é um ciclo negativo não orientado curto; $D = (4)$ é um ciclo trivial balanceado e $E = (3, 1, 2)$ é um ciclo longo não orientado negativo. **(b)** O grafo de breakpoints ponderado $G(\pi', \breve{\pi}', \breve{\iota})$ onde $(\pi, \breve{\pi}) = (\pi, \breve{\pi}) \cdot \rho^{(1,2,3)}_{(3,3,1)}$, com $\Delta c(\pi, \breve{\pi}, \breve{\iota}, \rho) = 2$ e $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) = 1$. A transposição aplicada no ciclo negativo E de **(a)** o transformou em três ciclos triviais $E' = (3)$, $E'' = (2)$ e $E''' = (1)$ tal que o ciclo E''' é balanceado. **(c)** O grafo de breakpoints ponderado $G(\pi'', \breve{\pi}'', \breve{\iota})$ onde $(\pi'', \breve{\pi}'') = (\pi', \breve{\pi}') \cdot \rho^{(9,9)}_{(2,4)}$, com $\Delta c(\pi', \breve{\pi}', \breve{\iota}, \rho) = 1$ e $\Delta c_b(\pi', \breve{\pi}', \breve{\iota}, \rho) = 1$. A reversão aplicada ao ciclo balanceado A o transformou em dois ciclos triviais balanceados $A' = (10)$ e $A'' = (9)$. **(d)** O grafo de breakpoints ponderado $G(\iota, \breve{\iota}, \breve{\iota})$, onde $c(\iota, \breve{\iota}, \breve{\iota}) = c_b(\iota, \breve{\iota}, \breve{\iota}) = n + 1$.

Seja $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota})$, $d_t(\pi, \breve{\pi}, \breve{\iota})$ e $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota})$ o número mínimo de reversões intergênicas, transposições intergênicas, e reversões intergênicas e transposições intergênicas, respectivamente, necessárias para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

Teorema 15. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem $n + 1$ regiões intergênicas cada, e π é uma permutação com sinais. $d_{\bar{r}}(\pi, \breve{\pi}, \breve{\iota}) \geq n + 1 - c_b(\pi, \breve{\pi}, \breve{\iota})$.*

Demonstração. Pela Propriedade 5 temos que $c_b(\iota, \breve{\iota}, \breve{\iota}) = n + 1$. Logo, nosso objetivo é aumentar o número de ciclos balanceados de $c_b(\pi, \breve{\pi}, \breve{\iota})$ para $n + 1$. Como pelo Lema 31 temos que este número aumenta em no máximo uma unidade após cada reversão, o lema segue. □

Teorema 16. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem $n + 1$ regiões intergênicas cada, e π é uma permutação sem sinais. $d_t(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{n+1-c_b(\pi,\breve{\pi},\breve{\iota})}{2}$.*

Demonstração. Pela Propriedade 5 temos que $c_b(\iota, \breve{\iota}, \breve{\iota}) = n + 1$. Desde modo, nosso objetivo é aumentar o número de ciclos de $c_b(\pi, \breve{\pi}, \breve{\iota})$ para $n+1$. Como, pelo Lema 33 este número aumenta em no máximo duas unidades para cada transposições, o lema segue. □

Teorema 17. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância onde $\breve{\pi}$ e $\breve{\iota}$ possuem $n + 1$ regiões intergênicas cada, e π é uma permutação com sinais. $d_{\overline{rt}}(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{n+1-c_b(\pi,\breve{\pi},\breve{\iota})}{2}$.*

Demonstração. Pela Propriedade 5 temos que $c_b(\iota, \breve{\iota}, \breve{\iota}) = n+1$, logo, queremos aumentar o número de ciclos de $c_b(\pi, \breve{\pi}, \breve{\iota})$ para $n + 1$. Pelo Lema 31 (resp. Lema 33) este número aumenta em no máximo uma unidade (resp. duas unidades) após cada reversão (resp. transposição), e o lema segue. □

Por fim, vamos introduzir uma função que computa o peso de um caminho entre duas arestas pretas. Sejam e_{c^x} e e_{c^y} duas arestas pretas arbitrárias de um mesmo ciclo $C = (c^1, \ldots, c^\ell)$, com $x, y \in [1..\ell]$. A função $f : E \times E \to \mathbb{Z}$ é definida como $f(e_{c^x}, e_{c^y}) = \sum_{i=x+1 \pmod \ell}^{y-1} [w(e'_{c^i}) - w(e_{c^i})] + w(e'_{c^x})$.

Em outras palavras, $f(e_{c^x}, e_{c^y})$ é a soma dos pesos de todas as arestas cinzas no caminho que vai de e_{c^x} para e_{c^y}, menos a soma dos pesos de todas as arestas pretas deste caminho com exceção das arestas e_{c^x} e e_{c^y}.

Note que, dado um ciclo C, $f(e_{c^x}, e_{c^y}) + f(e_{c^y}, e_{c^x}) - w(e_{c^x}) - w(e_{c^y})$ representa a soma dos pesos de todas as arestas cinzas menos a soma de todas as arestas pretas de um ciclo C. Desta forma, um ciclo C é balanceado se $f(e_{c^x}, e_{c^y}) + f(e_{c^y}, e_{c^x}) - w(e_{c^x}) - w(e_{c^y}) = 0$, positivo se $f(e_{c^x}, e_{c^y}) + f(e_{c^y}, e_{c^x}) - w(e_{c^x}) - w(e_{c^y}) > 0$ e negativo se $f(e_{c^x}, e_{c^y}) + f(e_{c^y}, e_{c^x}) - w(e_{c^x}) - w(e_{c^y}) < 0$.

6.2 A Complexidade dos Problemas Envolvendo Operações Intergênicas

Nesta seção mostramos a complexidade de três problemas envolvendo operações intergênicas: Ordenação de Permutações com Sinais por Reversões Intergênicas, Ordenação de Permutações sem Sinais por Transposições Intergênicas e Ordenação de Permutações com Sinais por Reversões Intergênicas e Transposições Intergênicas.

6.2.1 Reversões Intergênicas

Vamos mostrar que decidir se é possivel ordenar uma instância $(\pi, \breve{\pi}, \breve{\iota})$, onde π é uma permutação com sinais, utilizando no máximo k reversões intergênicas (**IRD**) é um problema

NP-difícil. Utilizaremos uma redução do problema de decisão fortemente NP-completo da 3 Partição (**3-PART**) [28].

Uma instância para o problema **3-PART** é um conjunto de inteiros positivos $A = \{a_1, a_2, ..., a_{3n}\}$ tal que $\sum_{i=1}^{3n} a_i = Bn$ para algum $B \in \mathbb{Z}^+$ e $\frac{B}{4} < a_i < \frac{B}{2}$ para qualquer $i \in [1..3n]$, e o objetivo é decidir se é possível particionar A em n triplas $A_1, \ldots, A_n$ tal que $\sum_{a_j \in A_i} a_j = B$ para cada tripla A_i, com $1 \leq i \leq n$.

Teorema 18. *Decidir se é possível ordenar uma instância $(\pi, \breve{\pi}, \breve{\iota})$ utilizando no máximo k reversões intergênicas é* NP*-difícil.*

Demonstração. Dada uma instância A para o problema **3-PART**, construímos uma instância $\langle(\iota, \breve{\iota}', \breve{\iota}''), k\rangle$ para **IRD** onde ι é a permutação identidade com $4n-1$ elementos, $\breve{\iota}'$ e $\breve{\iota}''$ são regiões intergênicas com $4n$ elementos cada, tal que $\breve{\iota}'_i = a_i$ e $\breve{\iota}''_i = 0$ para $1 \leq i \leq 3n$, e $\breve{\iota}'_j = 0$ e $\breve{\iota}''_j = B$ para $3n < j \leq 4n$, e tome $k = 6n$. Note que precisamos apenas consetar as regiões intergênicas dessa instância.

Note que $G(\iota, \breve{\iota}', \breve{\iota}'')$ possui $4n$ ciclos triviais desbalanceados tais que os primeiros $3n$ ciclos são negativos (qualquer ciclo $C = (c^1)$ desses é tal que $w(e_{c^1}) = a_i$ e $w(e'_{c^1}) = 0$), e os últimos n ciclos são positivos (qualquer ciclo $C = (c^1)$ desses é tal que $w(e_{c^1}) = 0$ e $w(e'_{b^1}) = B$). Vamos mostrar que uma instância A de **3-PART** é satisfeita se, e somente se, $d_{\bar{r}}(\iota, \breve{\iota}', \breve{\iota}'') \leq 6n$.

($\Rightarrow$) Suponha que a instância $A = \{a_1, ..., a_{3n}\}$ de **3-PART** pode ser particionada em n triplas $A_1, ..., A_n$ tal que, para cada tripla A_i, $\sum_{a_j \in A_i} a_j = B$, com $i \in [1..n]$. Seja $\{a_x, a_y, a_z\}$ a tripla A_i para qualquer $i \in [1..n]$. O algoritmo a seguir transforma $(\iota, \breve{\iota}')$ em $(\iota, \breve{\iota}'')$ utilizando $6n$ reversões.

Aplique a reversão $\rho_{(0,0)}^{(x,3n+i)}$ duas vezes em $(\iota, \breve{\iota}')$. Note que, dado que aplicamos duas vezes a mesma reversão, continuamos com uma instância cuja permutação é ι. Além disso, nossa instância agora é $(\iota, \breve{\iota}^*, \breve{\iota}'')$ tal que $\breve{\iota}^*_x = 0$; $\breve{\iota}^*_{3n+1} = \breve{\iota}'_{3n+1} + \breve{\iota}'_x = \breve{\iota}'_x$; e $\breve{\iota}^*_j = \breve{\iota}'_j$ para qualquer $j \neq \{x, 3n+1\}$. Como $\breve{\iota}^*_x = 0$, o ciclo trivial de rótulo x em $G(\iota, \breve{\iota}^*, \breve{\iota}'')$ se torna balanceado.

Aplicando duas vezes seguidas $\rho_{(0,0)}^{(y,3n+i)}$ e duas vezes seguidas $\rho_{(0,0)}^{(z,3n+i)}$ terminamos com 6 reversões aplicadas e 4 ciclos balanceados (a saber, os ciclos triviais rotulados como x, y, z e também $(3n+i)$, dado que $a_x + a_y + a_z = B$). Repetir este processo para todas as n triplas resulta em aplicar $6n$ reversões e obter $4n$ ciclos balanceados, logo $d_{\bar{r}}(\iota, \breve{\iota}', \breve{\iota}'') \leq 6n$.

($\Leftarrow$) Suponha agora que $d_{\bar{r}}(\iota, \breve{\iota}', \breve{\iota}'') \leq 6n$. Vamos definir agora dois tipos diferentes de reversões. Dizemos que uma reversão é de *fusão* caso seja aplicada em dois ciclos diferentes (gerando assim um único ciclo), e é dita de *divisão* caso ela quebre um ciclo em dois. Temos os seguintes fatos:

1. Como $c(\iota, \breve{\iota}', \breve{\iota}'') = 4n$, que também é o número máximo de ciclos, qualquer sequência que transforma $(\iota, \breve{\iota}')$ em $(\iota, \breve{\iota}'')$ possui o mesmo número de reversões de fusão e de divisão.

2. Qualquer reversão que não seja de fusão ou de divisão não altera o número de ciclos balanceados.

3. Uma reversão de fusão aumenta o número de ciclos balanceados se, e somente se, ela é aplicada sobre um ciclo positivo e um ciclo negativo.

4. Existem n ciclos positivos em $G(\iota, \breve{\iota}', \breve{\iota}'')$, todos ele triviais e com uma única aresta cinza com peso B. Como o peso de qualquer outra aresta cinza é 0, o numero máximo de ciclos positivos é sempre n.

Como $a_i < \frac{B}{2}$ para qualquer i, não existem dois ciclos negativos em $G(\iota, \breve{\iota}', \breve{\iota}'')$ tal que a soma dos pesos de suas arestas pretas seja igual a B. Assim, para uma reversão de fusão ser capaz de aumentar o número de ciclos balanceados, duas outras reversões de fusão que não aumentaram o número de ciclos balanceados devem ter sido aplicadas entre dois ciclos negativos e um ciclo positivo, ou entre três ciclos negativos com soma dos pesos das arestas pretas B. Isto significa que pelo menos três reversões de fusão são necessárias para aumentar em uma unidade o número de ciclos balanceados. Observe que reversões de divisão podem ocorrer entre as reversões de fusão e também aumentar o número de ciclos balanceados.

No melhor cenário, todas as reversões de divisão aumentam o número de ciclos balanceados em uma unidade. Seja S_ρ uma sequência de reversões tal que $(\iota, \breve{\iota}') \cdot S_\rho = (\iota, \breve{\iota}'')$. Suponha que S_ρ possui q reversões de divisão, q reversões de fusão (dado pelo Fato 1 acima), e $\ell \geq 0$ reversões que não são nem de fusão nem de divisão.

Como $c_b(\iota, \breve{\iota}', \breve{\iota}'') = 0$, e como três reversões de fusão são necessárias para aumentar o número de ciclos balanceados, $q + \frac{q}{3} \geq 4n$, o que resulta em $\frac{4q}{3} \geq 4n$, logo $q \geq 3n$. Segue então que S_ρ possui pelo menos $2q + \ell \geq 6n + \ell$ reversões.

Se $d_{\bar{r}}(\iota, \breve{\iota}', \breve{\iota}'') \leq 6n$, pelos fatos 1-4 temos que $\ell = 0$ e $q = 3n$. Além disso, como argumentado acima, deve sempre ser possível a união de quatro ciclos utilizando três reversões de fusão gerando um ciclo balanceado, pois caso contrário $q > 3n$. Isto implica que n triplas de ciclos negativos podem ser unidos com um dos n ciclos positivos cada, e essas n triplas são exatamente os subconjuntos que satisfazem **3-PART**. □

6.2.2 Transposições Intergênicas

Agora vamos mostrar que decidir se é possivel ordenar uma instância $(\pi, \breve{\pi}, \breve{\iota})$, onde π é uma permutação sem sinais, utilizando no máximo k transposições intergênicas (**ITD**) é NP-difícil, utilizando uma redução do problema de Ordenação de Permutações por Transposições (**SbT**), cuja prova de que é NP-difícil foi apresentada por Bulteau e coautores [14].

Uma instância para o problema **SbT** é uma permutação π e um inteiro não negativo d, e o objetivo é decidir se é possível transformar π em ι aplicando no máximo d transposições.

Lema 34. *Decidir se é possível ordenar uma instância $(\pi, \breve{\pi}, \breve{\iota})$ utilizando no máximo k transposições intergênicas é* NP*-difícil.*

Demonstração. A prova vem do fato de que **ITD** é uma generalização de **SbT**: para qualquer instância $\langle \pi, d \rangle$ para **SbT**, podemos criar a instância $\langle (\pi', \breve{\pi}', \breve{\iota}), k \rangle$ de **IDT** tal que $\pi' = \pi$, $\breve{\pi}' = \breve{\iota} = (0\ 0\ ...\ 0)$ e $k = d$. Note que é possível transformar π em ι aplicando no máximo d transposições se, e somente se, $d_t(\pi', \breve{\pi}', \breve{\iota}) \leq d$.

Além disso, podemos obter uma sequência de transposições que transforma π em ι a partir de uma sequência de transposições intergênicas $\mathcal{S}_\tau$ que transforma $(\pi', \breve{\pi}')$ em $(\iota, \breve{\iota})$, ambas de mesmo tamanho, utilizando os índices dos rótulos das arestas pretas onde

são aplicadas e ignorando os índices dos tamanhos/pesos (que, por definição, devem ser sempre 0). □

6.2.3 Reversões Intergênicas e Transposições Intergênicas

Por fim, vamos mostrar que decidir se é possivel ordenar uma instância $(\pi, \breve{\pi}, \breve{\iota})$, onde π é uma permutação com sinais, utilizando no máximo k reversões intergênicas e transposições intergênicas (**IRITD**) também é NP-difícil. Para isso utilizamos o problema **SRT**, que mostramos ser NP-difícil no Corolário 1.

Lema 35. *Decidir se é possível ordenar uma instância $(\pi, \breve{\pi}, \breve{\iota})$ utilizando no máximo k reversões intergênicas e transposições intergênicas é* NP*-difícil.*

Demonstração. Semelhante ao Lema 34, uma vez que **IRITD** é uma generalização do problema **SRT**. □

6.3 O Efeito de Reversões Intergênicas em Grafos de Breakpoints Ponderados

Começamos com o seguinte lema, que garante que todo ciclo divergente negativo ou balanceado admite uma reversão que aumenta o número de ciclos balanceados.

Lema 36. *Seja C um ciclo não trivial divergente negativo ou balanceado. Existe uma reversão que aumenta o número de ciclos balanceados em uma unidade.*

Demonstração. Seja $C = (c^1, \ldots, c^\ell)$ um ciclo não trivial divergente negativo ou balanceado. Se encontrarmos um par divergente de arestas pretas e_{c^i} e e_{c^j} tal que $w(e_{c^i})+w(e_{c^j}) \geq f(e_{c^i}, e_{c^j})$ e $f(e_{c^i}, e_{c^j}) \geq 0$, então é possível criar um ciclo balanceado aplicando uma reversão nestas arestas (note que se $w(e_{c^i}) + w(e_{c^j}) \geq f(e_{c^i}, e_{c^j})$, então o ciclo gerado que possui todas as arestas pretas e cinzas de $f(e_{c^i}, e_{c^j})$ pode ser balanceado usando os pesos $w(e_{c^i})$ e $w(e_{c^j})$).

Vamos provar que estas condições podem ser encontradas por contradição. Assuma que não importe qual par de vértices e_{c^i} e e_{c^j} que escolhamos, ou $f(e_{c^i}, e_{c^j}) > w(e_{c^i}) + w(e_{c^j})$ ou $f(e_{c^i}, e_{c^j}) < 0$.

Se um total de k arestas pretas de C são divergentes, então $\ell - k \geq 1$ são convergentes (lembre-se de que c^1 é sempre convergente por definição). Isto resulta em $k(\ell - k)$ pares divergentes, o que é sempre maior ou igual a $\ell - 1$.

Suponha, sem perda de generalidade, que apenas a aresta preta mais à esquerda e_{c^ℓ} é percorrida da esquerda para a direita. Neste caso temos como par divergente a aresta e_{c^ℓ} com cada uma das outras $\ell - 1$ arestas pretas de C. Uma análise similar para ciclos com mais de uma aresta preta divergente também resulta no mesmo tipo de contradição.

Vamos começar com o par e_{c^ℓ} e e_{c^1}. Note que $f(e_{c^\ell}, e_{c^1}) \geq 0$ pois existe apenas uma aresta cinza entre e_{c^ℓ} e e_{c^1}, o que significa que $f(e_{c^\ell}, e_{c^1}) = w(e'_{c^\ell})$. Este cenário nos força a estabelecer que $f(e_{c^\ell}, e_{c^1}) > w(e_{c^\ell}) + w(e_{c^1})$.

Vamos escolher agora e_{c^ℓ} e e_{c^2} (a ideia é continuar sempre com a próxima aresta preta disponível começando pela aresta preta divergente e_{c^ℓ}). Se $f(e_{c^\ell}, e_{c^2}) < 0$, então $w(e'_{c^\ell}) + w(e'_{c^1}) - w(e_{c^1}) < 0$. Em outras palavras, $f(e_{c^\ell}, e_{c^1}) + w(e'_{c^1}) - w(e_{c^1}) < 0$, o que significa que $w(e'_{c^1}) < w(e_{c^1}) - f(e_{c^\ell}, e_{c^1})$. Entretanto, já estabelecemos que $f(e_{c^\ell}, e_{c^1}) > w(e_{c^\ell}) + w(e_{c^1})$, o que nos leva também a $f(e_{c^\ell}, e_{c^1}) > w(e_{c^1})$. Neste caso, $f(e_{c^\ell}, e_{c^2}) < 0$ se, e somente se, $w(e'_{c^1})$ é negativo, o que não é possível por definição. Isto nos força novamente a estabelecer que $f(e_{c^\ell}, e_{c^2}) > w(e_{c^\ell}) + w(e_{c^2})$.

Este mesmo argumento pode ser feito para qualquer aresta preta existente neste caminho feito a partir de e_{c^ℓ}, dado que $f(e_{c^\ell}, e_{c^i}) = f(e_{c^\ell}, e_{c^{(i-1)}}) + w(e'_{c^{(i-1)}}) - w(e_{c^{(i-1)}})$. Assim, $f(e_{c^\ell}, e_{c^i}) < 0$ se, e somente se, $w(e_{c^{(i-1)}})$ é negativo. Por outro lado, se estabelecermos $f(e_{c^\ell}, e_{c^i}) > w(e_{c^\ell}) + w(e_{c^i})$ para todo $1 \leq i < m$, o ciclo é positivo, o que contradiz nossa primeira afirmação. □

O seguinte lema pode ser aplicado nos casos onde todos os ciclos negativos ou balanceados de $G(\pi, \breve{\pi}, \breve{\iota})$ são convergentes.

Lema 37. *Seja $C = (c^1, \ldots, c^\ell)$, $\ell > 1$ um ciclo convergente negativo ou balanceado em um grafo de breakpoints ponderado $G(\pi, \breve{\pi}, \breve{\iota})$ onde todos os ciclos que são negativos ou balanceados são convergentes. É possível aumentar o número de ciclos balanceados usando duas reversões.*

Demonstração. Seja C um ciclo convergente negativo ou balanceado e seja e_{c^i} e $e_{c^{i+1}}$ duas arestas pretas de C, com $1 \leq i < \ell$. Se existe uma aresta preta e_{c^j} com $j \notin \{i, i{+}1\}$ tal que ou $c^j \in [c^i..c^{i+1}]$, se $c^i < c^{i+1}$, ou $c^j \in [c^{i+1}..c^i]$, caso contrário, então uma reversão aplicada em e_{c^i} e e_{c^j} não irá quebrar o ciclo C, dado que este par é convergente, e transformará C em um ciclo divergente. Agora podemos utilizar o Lema 36 para encontrar uma reversão que, quando aplicada em C, aumenta o número de ciclos balanceados em uma unidade.

Se não existe uma aresta preta e_{c^j} como descrito acima temos que o par e_{c^i} e $e_{c^{i+1}}$ é um open gate e sabemos por Bafna e Pevzner [3] que existe um ciclo não trivial $D = (\ldots, d^i, \ldots, d^j, \ldots)$ que fecha este open gate, ou seja, ou c^i ou c^{i+1} está no intervalo entre d^i e d^j. Uma reversão aplicada em e_{d^i} e e_{d^j} transformará C em um ciclo divergente, e pelo Lema 36 podemos encontrar uma reversão que aplicada em C aumenta o número de ciclos balanceados em uma unidade. Note que se a primeira reversão quebra D, o que significa que D é divergente, mas como assumimos que todos os ciclos balanceados de $G(\pi, \breve{\pi}, \breve{\iota})$ são convergentes, então D é desbalanceado e o número de ciclos balanceados não pode diminuir com a aplicação desta reversão. □

Por fim, mostramos como aumentar o número de ciclos balanceados utilizando reversões em ciclos triviais negativos.

Lema 38. *Dado um ciclo trivial negativo $C = (c^1)$, é possivel transformar C em trivial balanceado com duas reversões.*

Demonstração. Seja $D = (d^1, \ldots)$ um ciclo positivo em $G(\pi, \breve{\pi}, \breve{\iota})$. Como C é um ciclo negativo, então D deve existir por definição. Aplique a reversão $\rho^{(c^1, d^1)}_{(e'_{c^1}, 0)}$ duas vezes: a primeira move $w(e_{c^1}) - w(e'_{c^1})$ de $w(e_{c^1})$ para $w(e_{d^1})$, e a segunda apenas gera o ciclo C novamente, que agora é balanceado. □

6.4 O Efeito de Transposições Intergênicas em Grafos de Breakpoints Ponderados

Nesta seção, apresentamos propriedades e lemas que irão auxiliar algoritmos que utilizam transposições intergênicas. Os próximos três lemas mostram como podemos aumentar o número de ciclos balanceados aplicando transposições em ciclos negativos não triviais. O Lema 39 mostra que sempre é possível aumentar o número de ciclos balanceados em uma unidade aplicando uma transposição em um ciclo longo não orientado. O Lema 40 mostra que sempre existe uma tripla de arestas pretas de um ciclo orientado C onde a aplicação de uma transposição transforma C em três ciclos tal que um dos ciclos é balanceado caso C seja não positivo. Usando este lema, o Lema 41 explica o caso específico em que C é negativo.

Lema 39. *Seja C um ciclo não trivial, não orientado e negativo. Existe uma transposição que aumenta o número de ciclos balanceados em uma unidade.*

Demonstração. Seja $C = (\ldots, c^x, \ldots, c^y, \ldots)$ um ciclo não trivial, não orientado e negativo, e seja $D = (\ldots, d^z, \ldots)$ um ciclo positivo. Note que se C existe, então D deve existir também, já que o grafo de breakpoints ponderado como um todo deve ser balanceado por definição.

Uma transposição aplicada nas arestas pretas e_{c^x}, e_{c^y} e e_{d^z} criará dois ciclos C' e D' tal que C' é formado pelo caminho que vai de e_{c^y} para e_{c^x} com uma nova aresta preta (que pode receber pesos de e_{c^x} e e_{c^y}). Queremos que C' seja balanceado, assim, esta nova aresta preta deve ter peso exatamente igual a $f(e_{c^y}, e_{c^x})$. Assim, precisamos provar que existem valores de c^x e c^y tal que $0 \leq f(e_{c^y}, e_{c^x}) \leq w(e_{c^y}) + w(e_{c^x})$.

Vamos provar que estas condições podem ser encontradas por contradição de modo muito similar ao Lema 36, mas agora para ciclos convergentes. Suponha que para quaisquer valores de c^x e c^y, ou $f(e_{c^y}, e_{c^x}) < 0$ ou $f(e_{c^y}, e_{c^x}) > w(e_{c^y}) + w(e_{c^x})$.

Vemos escolher c_x e c_y como as arestas pretas mais à direita e mais à esquerda de C. Em outras palavras, $x = 1$ e $y = \ell$, onde ℓ é o número de arestas pretas em C. Desta forma, $f(e_{c^\ell}, e_{c^1}) = w(e'_{c^\ell}) \geq 0$ dado que só existe uma aresta cinza entre e_{c^ℓ} e e_{c^1}. Isto nos força a considerar que $f(e_{c^\ell}, e_{c^1}) > w(e_{c^\ell}) + w(e_{c^1})$.

Vamos agora utilizar $x = 2$ (mantendo $y = \ell$). Se $f(e_{c^\ell}, e_{c^2}) < 0$, então $w(e'_{c^\ell}) + w(e'_{c^1}) - w(e_{c^1}) < 0$. Em outras palavras, $f(e_{c^\ell}, e_{c^1}) + w(e'_{c^1}) - w(e_{c^1}) < 0$, o que significa que $w(e'_{c^1}) < w(e_{c^1}) - f(e_{c^\ell}, e_{c^1})$. Entretanto, nós já estabelecemos que $f(e_{c^\ell}, e_{c^1}) > w(e_{c^\ell}) + w(e_{c^1})$, o que significa que $f(e_{c^\ell}, e_{c^1}) > w(e_{c^1})$. Neste caso, $f(e_{c^\ell}, e_{c^2}) < 0$ se, e somente se, $w(e'_{c^1})$ é negativo. Isto no força a considerar novamente que $f(e_{c^\ell}, e_{c^2}) > w(e_{c^\ell}) + w(e_{c^2})$.

Este mesmo argumento pode ser feito para quaisquer $x = i$, $2 \leq i < \ell$, já que $f(e_{c^\ell}, e_{c^i}) = f(e_{c^\ell}, e_{c^{(i-1)}}) + w(e'_{c^{(i-1)}}) - w(e_{c^{(i-1)}})$. Assim, $f(e_{c^\ell}, e_{c^i}) < 0$ se, e somente se, $w(e_{c^{(i-1)}})$ é negativo. Por outro lado, se permitirmos que $f(e_{c^\ell}, e_{c^i}) > w(e_{c^\ell}) + w(e_{c^i})$ para qualquer $1 \leq i < \ell$, o ciclo não pode ser negativo, o que contradiz nossa primeira afirmação. □

Lema 40. *Seja $C = (c^1, \ldots, c^\ell)$ um ciclo não trivial orientado. Se C é não positivo, então existe uma tripla $(e_{c^x}, e_{c^y}, e_{c^z})$ com $c^x > c^z > c^y$ e $1 \leq x < y < z \leq \ell$ tal que*

$0 \leq f(e_{c^x}, e_{c^y}) \leq w(c^x) + w(c^y)$, *ou* $0 \leq f(e_{c^y}, e_{c^z}) \leq w(c^y) + w(c^z)$, *ou* $0 \leq f(e_{c^z}, e_{c^x}) \leq w(c^z) + w(c^x)$.

Demonstração. Seja $C = (c^1, \ldots, c^\ell)$ um ciclo longo orientado não positivo com ℓ arestas pretas. Se encontrarmos três arestas pretas e_{c^x}, e_{c^y} e e_{c^z} em C tais que $c^x > c^z > c^y$ e $1 \leq x < y < z \leq \ell$, podemos aplicar uma transposição nestas três arestas que transformam C em três ciclos C', C'' e C''', tal que C' é formado pelo caminho que vai de e_{c^z} para e_{c^y} com uma nova aresta preta que pode receber pesos de e_{c^y} e e_{c^z}; C'' é formado pelo caminho que vai de e_{c^x} para e_{c^z} com uma nova aresta preta que pode receber pesos de e_{c^x} e e_{c^z}; e C''' é formado pelo caminho que vai de e_{c^y} para e_{c^x} com uma nova aresta preta que pode receber pesos de e_{c^y} e e_{c^x}.

Vamos começar com uma tripla (c^x, c^y, c^z) com $c^x > c^z > c^y$ e $1 \leq x < y < z \leq \ell$ tal que pelo menos uma das condições a seguir é verdade: (i) $y = x \pmod{\ell} + 1$; (ii) $z = y \pmod{\ell} + 1$; (iii) $x = z \pmod{\ell} + 1$. Ao invés de olhar para apenas um par de arestas pretas de cada vez, vamos olhar três pares: (c^x, c^y), (c^y, c^z) e (c^z, c^x). Pela escolha de tripla que fizemos, em pelo menos um destes três pares existe apenas uma aresta cinza no caminho que vai de uma aresta preta para a outra, de forma que a função f entre eles não pode ser negativa, forçando a estabelecer que o peso desta aresta cinza é maior que a soma dos pesos das duas arestas pretas ligadas a ela.

Se C possui apenas três arestas pretas, a condição acima deve ser verdadeira para os três pares, o que significa que C deve ser positivo. Se C possui mais de três arestas pretas, então existem mais triplas que atendem ao critério de (c^x, c^y, c^z) com $c^x > c^z > c^y$ e $1 \leq x < y < z \leq \ell$. Podemos avançar pelo caminho entre cada par de arestas pretas (como fizemos no Lema 39), tal que, ao final, temos que a soma dos pesos das arestas cinzas deve ser estritamente maior que a soma dos pesos das arestas pretas, e, desta forma, o ciclo deve ser positivo (veja os exemplos 1 e 2). □

Exemplo 1. *Suponha que temos um ciclo longo não orientado não positivo (ou seja, negativo ou balanceado)* $C = (c^1, \ldots, c^6) = (12, 10, 4, 2, 8, 6)$*, e suponha que não exista* (c^x, c^y, c^z) *com* $c^x > c^z > c^y$ *e* $1 \leq x < y < z \leq 6$*, tal que ou* $0 \leq f(e_{c^x}, e_{c^y}) \leq w(c^x) + w(c^y)$*, ou* $0 \leq f(e_{c^y}, e_{c^z}) \leq w(c^y) + w(c^z)$ *ou* $0 \leq f(e_{c^z}, e_{c^x}) \leq w(c^z) + w(c^x)$.

Existem oito triplas (c^x, c^y, c^z) *com* $c^x > c^z > c^y$ *e* $1 \leq x < y < z \leq 6$*:* $(12, 4, 8)$, $(12, 2, 8), (12, 4, 6), (12, 2, 6), (10, 4, 8), (10, 2, 8), (10, 4, 6)$ *e* $(10, 2, 6)$*. Dentre elas, vamos começar com aquelas que possuem pelo menos uma das condições a seguir: (i)* $y = x \pmod 6 + 1$*; (ii)* $z = y \pmod 6 + 1$*; e (iii)* $x = z \pmod 6 + 1$*. Para (i) podemos usar* $(c^x, c^{x+1}, c^z) = (c^2, c^3, c^5) = (10, 4, 8)$*; para (ii) podemos usar* $(c^x, c^y, c^{y+1}) = (c^2, c^4, c^5) = (10, 2, 8)$*; e para (iii) podemos usar* $(c^{z \pmod 6 + 1}, c^y, c^z) = (c^1, c^4, c^6) = (12, 2, 6)$.

Para $(c^2, c^3, c^5) = (10, 4, 8)$ *temos:*

1. $f(e_{c^2}, e_{c^3}) = w(e'_{c^2})$;
2. $f(e_{c^3}, e_{c^5}) = w(e'_{c^3}) - w(e_{c^4}) + w(e'_{c^4})$;
3. $f(e_{c^5}, e_{c^2}) = w(e'_{c^5}) - w(e_{c^6}) + w(e'_{c^6}) - w(e_{c^1}) + w(e'_{c^1})$.

Para $(c^2, c^4, c^5) = (10, 2, 8)$ *temos (3) e:*

4. $f(e_{c^2}, e_{c^4}) = w(e'_{c^2}) - w(e_{c^3}) + w(e'_{c^3})$;

5. $f(e_{c^4}, e_{c^5}) = w(e'_{c^4})$.

Para $(c^1, c^4, c^6) = (12, 2, 6)$ *temos:*

6. $f(e_{c^1}, e_{c^4}) = w(e'_{c^1}) - w(e_{c^2}) + w(e'_{c^2}) - w(e_{c^3}) + w(e'_{c^3})$;

7. $f(e_{c^4}, e_{c^6}) = w(e'_{c^4}) - w(e_{c^5}) + w(e'_{c^5})$;

8. $f(e_{c^6}, e_{c^1}) = w(e'_{c^6})$.

Em (1) temos que $f(e_{c^2}, e_{c^3}) = w(e'_{c^2}) \geq 0$, *então somos obrigados a assumir que* $w(e'_{c^2}) > w(e_{c^2}) + w(e_{c^3})$, *pois, caso contrário, teríamos que* $f(e_{c^2}, e_{c^3}) \leq w(e_{c^2}) + w(e_{c^3})$. *Pela mesma razão, em (5) somos obrigados a assumir que* $w(e'_{c^4}) > w(e_{c^4}) + w(e_{c^5})$, *e em (8) temos que assumir que* $w(e'_{c^6}) > w(e_{c^6}) + w(e_{c^1})$.

Somando as equações obtidas para (2), (5) e (8), temos que $w(e'_{c^2}) + w(e'_{c^4}) + w(e'_{c^6}) > w(e_{c^1}) + w(e_{c^2}) + w(e_{c^3}) + w(e_{c^4}) + w(e_{c^5}) + w(e_{c^6})$, *o que contradiz o fato de que* C *não é positivo.*

Exemplo 2. *Suponha que temos um ciclo não positivo (ou seja, negativo ou balanceado)* $C = (c^1, \ldots, c^5) = (10, 8, 6, 1, 2)$, *e suponha que não existe uma tripla* (c^x, c^y, c^z) *com* $c^x > c^z > c^y$ *e* $1 \leq x < y < z \leq 5\ell$ *tal que ou* $0 \leq f(e_{c^x}, e_{c^y}) \leq w(c^x) + w(c^y)$, *ou* $0 \leq f(e_{c^y}, e_{c^z}) \leq w(c^y) + w(c^z)$ *ou* $0 \leq f(e_{c^z}, e_{c^x}) \leq w(c^z) + w(c^x)$.

Existem três triplas (c^x, c^y, c^z) *com* $c^x > c^z > c^y$ *e* $1 \leq x < y < z \leq 5$*:* $(10, 1, 2)$, $(8, 1, 2)$ *e* $(6, 1, 2)$, *e em todas pelo menos uma das condições a seguir são satisfeitas: (i)* $y = x \pmod 5 + 1$*; (ii)* $z = y \pmod 5 + 1$*; e (iii)* $x = z \pmod 5 + 1$.

Para $(c^1, c^4, c^5) = (10, 1, 2)$ *temos que:*

1. $f(e_{c^1}, e_{c^4}) = w(e'_{c^1}) - w(e_{c^2}) + w(e'_{c^2}) - w(e_{c^3}) + w(e'_{c^3})$;

2. $f(e_{c^4}, e_{c^5}) = w(e'_{c^4})$;

3. $f(e_{c^5}, e_{c^1}) = w(e'_{c^5})$.

Para $(c^2, c^4, c^5) = (8, 1, 2)$ *temos (2), (3) e:*

4. $f(e_{c^2}, e_{c^4}) = w(e'_{c^2}) - w(e_{c^3}) + w(e'_{c^3})$.

Para $(c^3, c^4, c^5) = (6, 1, 2)$ *temos (2), (3) e:*

5. $f(e_{c^3}, e_{c^4}) = w(e'_{c^3})$.

Em (2) $f(e_{c^4}, e_{c^5}) = w(e'_{c^4}) \geq 0$, *logo somos obrigados a assumir que* $w(e'_{c^4}) > w(e_{c^4}) + w(e_{c^5})$, *pois, caso contrário, teríamos que* $f(e_{c^4}, e_{c^5}) \leq w(e_{c^4}) + w(e_{c^5})$. *Pela mesma razão, em (3) temos que estabelecer que* $w(e'_{c^5}) > w(e_{c^5}) + w(e_{c^1})$, *e em (5) temos que estabelecer que* $w(e'_{c^3}) > w(e_{c^3}) + w(e_{c^4})$.

Como $w(e'_{c^3}) > w(e_{c^3})$ *e* $w(e'_{c^2}) \geq 0$, *em (4) temos que* $f(e_{c^2}, e_{c^4}) = w(e'_{c^2}) - w(e_{c^3}) + w(e'_{c^3}) > 0$, *então devemos assumir que* $w(e'_{c^2}) - w(e_{c^3}) + w(e'_{c^3}) > w(e_{c^2}) + w(e_{c^4})$, *então* $w(e'_{c^2}) + w(e'_{c^3}) > w(e_{c^2}) + w(e_{c^3}) + w(e_{c^4})$. *Somando as equações (4) e (3) nós temos que* $w(e'_{c^2}) + w(e'_{c^3}) + w(e'_{c^5}) > w(e_{c^2}) + w(e_{c^3}) + w(e_{c^4}) + w(e_{c^5}) + w(e_{c^1})$, *o que significa que o ciclo deveria ser positivo.*

Lema 41. *Seja C um ciclo longo orientado negativo. Existe uma transposição que aumenta o número de ciclos balanceados em uma unidade e o número de ciclos em duas unidades.*

Demonstração. Seja $C = (c^1, \ldots, c^\ell)$ um ciclo orientado negativo com ℓ arestas pretas. Se encontrarmos três arestas pretas e_{c^x}, e_{c^y} e e_{c^z} de C tal que $c^x > c^z > c^y$ e $1 \leq x < y < z \leq \ell$, podemos aplicar uma transposição nestas arestas pretas que transforma C em três ciclos C', C'' e C''', e nosso objetivo é encontrar uma tripla que satisfaça a condição acima e que também gere pelo menos um ciclo positivo.

Como C não é positivo, pelo Lema 40 sabemos que podemos encontrar tal tripla. Deste modo, basta aplicar uma transposição nesta tripla de modo a gerar pelo menos um ciclo balanceado. □

O lema a seguir mostra que é possivel redistribuir os pesos entre três arestas pretas com quaisquer valores utilizando duas transposições, e este lema será utilizado com frequência até o final desta seção.

Lema 42. *É possível realizar qualquer redistribuição de pesos entre três arestas pretas distintas e_{b^1}, e_{b^2} e e_{b^3} utilizando duas transposições consecutivas.*

Demonstração. Vamos assumir que $w(e_{b^1}) = x$, $w(e_{b^2}) = y$ e $w(e_{b^3}) = z$, e que $b^1 < b^2 < b^3$. Vamos mostrar que podemos realizar qualquer redistribuição de valores (respeitando o valor máximo $x+y+z$) nestas três arestas pretas utilizando duas transposições, tal que ao final $w(e_{b^1}) = x'$, $w(e_{b^2}) = y'$ e $w(e_{b^3}) = z'$, com $x' + y' + z' = x + y + z$.

Suponha, sem perda de generalidade, que $x' = x_1 + y_1 + z_1$, $y' = x_2 + y_2 + z_2$ e $z' = x_3 + y_3 + z_3$ tal que: (i) $\{x_i, y_i, z_i\} \subset \mathbb{N}$, com $i \in \{1, 2, 3\}$; (ii) $x_1 + x_2 + x_3 = x$; (iii) $y_1 + y_2 + y_3 = y$; e (iv) $z_1 + z_2 + z_3 = z$.

Primeiro, aplique a transposição $\rho^{(b^1,b^2,b^3)}_{(x_1+x_2,y_2+y_3,z_1)}$, e sejam e'_{c^1}, e'_{c^2} e e'_{c^3} as arestas pretas resultantes, com $c^1 < c^2 < c^3$ (ainda temos que $c^1 = b^1$ e $c^3 = b^3$). Note que $w(e_{c^1}) = x_1 + x_2 + y_1$, $w(e_{c^2}) = x_3 + z_1$ e $w(e_{c^3}) = y_2 + y_3 + z_2 + z_3$, então $w(e_{c^1}) + w(e_{c^2}) + w(e_{c^3}) = x + y + z$. Agora aplique a transposição $\rho^{(c^1,c^2,c^3)}_{(x_1+y_1,x_3,y_2+z_2)}$. Esta segunda transposição irá gerar as arestas pretas e_{b^1}, e_{b^2} e e_{b^3}, mas agora $w(e_{b^1}) = x_1 + y_1 + z_1$, $w(e_{b^2}) = x_2 + y_2 + z_2$ e $w(e_{b^3}) = x_3 + y_3 + z_3$ conforme desejado.

Desta forma, sejam três arestas pretas quaisquer e_{b^1}, e_{b^2} e e_{b^3} com $w(e_{b^1}) = x$, $w(e_{b^2}) = y$ e $w(e_{b^3}) = z$. Dados três valores inteiros não negativos quaisquer x', y' e z' tal que $x' + y' + z' = x + y + z$, podemos redistribuir os pesos das três arestas para $w(e_{b^1}) = x'$, $w(e_{b^2}) = y'$ e $w(e_{b^3}) = z'$ aplicando as duas transposições acima com os seguintes valores:

- $x_1 = \min\{x, x'\}$, $y_1 = \min\{y, x' - x_1\}$ e $z_1 = \min\{z, x' - x_1 - y_1\}$;
- $x_2 = \min\{x - x_1, y'\}$, $y_2 = \min\{y - y_1, y' - x_2\}$ e $z_2 = \min\{z - z_1, y' - x_2 - y_2\}$;
- $x_3 = (x - x_1 - x_2)$, $y_3 = (y - y_1 - y_2)$ e $z_3 = (z - z_1 - z_2)$.

Como $x' \leq x + y + z$, temos que $x_1 + y_1 + z_1 = x'$. Como $y' \leq x - x' + y + z$, temos que $x_2 + y_2 + z_2 = y'$. Por fim, como $z' = x - x' + y - y' + z$, temos que $x_3 + y_3 + z_3 = z'$. □

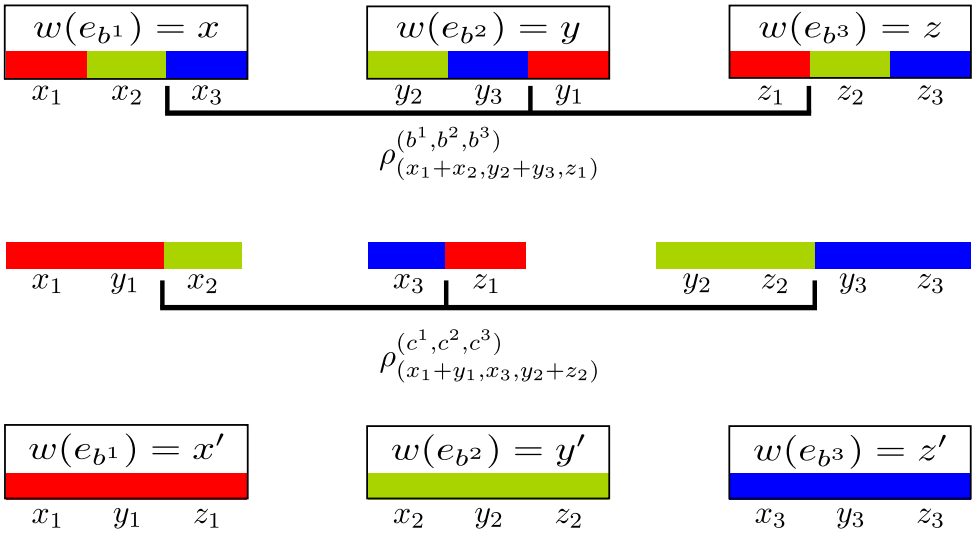


Figura 6.3: Redistribuição de pesos entre três arestas pretas e_{b^1}, e_{b^2} e e_{b^3}.

A Figura 6.3 mostra um exemplo desta redistribuição.

Utilizando o Lema 42, mostramos no lema a seguir quantas transposições são necessárias para transformar um ciclo trivial em balanceado.

Lema 43. *Seja G um grafo de breakpoints ponderado no qual todos os ciclos negativos são triviais. Existe uma sequência de duas transposições que aumenta o número de ciclos balanceados em duas unidades.*

Demonstração. Suponha que todos os ciclos negativos de G são triviais e seja $C = (c^1)$ um ciclo negativo de G. Como C é negativo, sabemos que existe um ciclo positivo D no grafo. Temos três casos:

1. Existe outro ciclo negativo $F = (f^1)$. Use as arestas pretas e_{c^1}, $e_{d^1} \in D$ e e_{f^1} para redistribuir os pesos usando duas transposições, como mostrado no Lema 42, tal que ambos C e F sejam transformados em ciclos balanceados.

2. C é o único ciclo negativo e existe outro ciclo positivo $F = (f^1, \ldots)$. Use as arestas pretas e_{c^1}, $e_{d^1} \in D$ e e_{f^1} para redistribuir os pesos utilizando duas transposições, conforme mostrado no Lema 42, tal que C e D sejam transformados em ciclos balanceados. Como C é o único ciclo negativo, seu peso é suficiente para transformar D em balanceado e passar o peso excedente (com relação à sua aresta cinza) para F, que será transformado em um ciclo balanceado ou negativo.

3. C e D são os únicos ciclos desbalanceados de G. Aqui temos mais dois casos:

   - $D = (\ldots, d^i, \ldots, d^j, \ldots)$ é um ciclo não trivial: aplique duas transposições nas arestas pretas e_{c^1}, e_{d^i} e e_{d^j} como mostrado no Lema 42, movendo o peso excedente de C para D balanceando ambos.
   - $D = (d^1)$ é trivial: neste caso, seja $F = (f^1, \ldots)$ um ciclo balanceado no grafo. Aplique duas transposições conforme mostrado no Lema 42, movendo o peso excedente de C para D tal que, ao final, $w(e_{f^1})$ possui o mesmo valor que

possuía antes das transposições serem aplicadas. Ao final temos que C e D são ciclos balanceados.

Nos três casos acima, nós aumentamos o número de ciclos balanceados em duas unidades aplicando duas transposições. □

Os lemas 44, 45 e 46 mostram como é possível aumentar o número de ciclos balanceados aplicando transposições em ciclos longos orientados balanceados, ciclos longos não orientados balanceados e ciclos curtos balanceados, respectivamente. A Figura 6.4 explica como, e em quais casos, cada um destes lemas é aplicado, conforme explicado em detalhes logo após os lemas.

Lema 44. *Seja C um ciclo longo orientado balanceado. É possível aumentar o número de ciclos balanceados em duas unidades utilizando no máximo três transposições.*

Demonstração. Seguimos a mesma ideia do Lema 41: encontrar três arestas pretas e_{c^x}, e_{c^y} e e_{c^z} em C tal que $c^x > c^z > c^y$ e $1 \leq x < y < z \leq \ell$, onde podemos aplicar uma transposição que transforma este ciclo em três ciclos C', C'' e C'''. Como C não é positivo, pelo Lema 40 sabemos que existe uma tripla tal que a transposição criará um ciclo balanceado.

Dado que C' será um ciclo balanceado, precisamos lidar com os outros dois ciclos C'' e C'''. Aqui temos três casos:

1. Se C'' é balanceado, então C''' também é balanceado (e vice-versa) e o resultado segue.

2. Se C'' é negativo, então C''' é positivo (e vice-versa). Vamos assumir que C'' é não trivial, e os lemas 39 e 41 garantem que podemos aplicar uma transposição para gerar um ciclo balanceado. Se escolhermos C''' para receber o peso excedente de C'', então ambos estarão balanceados após a transposição.

3. Se o ciclo negativo C'' é trivial, então o Lema 43 garante que podemos mover o peso excedente para outro ciclo utilizando duas transposições. Se escolhermos C''' para tal, ambos os ciclos estarão balanceados após as duas transposições aplicadas. □

Lema 45. *Seja C um ciclo longo não orientado balanceado em um grafo de breakpoints ponderado onde não existem ciclos orientados. É possível aumentar o número de ciclos balanceados em quatro unidades usando até sete transposições.*

Demonstração. Para provar este lema, usamos a seguinte afirmação provada por Bafna e Pevzner [3, Teorema 4.7]: se todos os ciclos de um grafo G são não orientados, convergentes, e existe um ciclo longo, então existe uma sequência de três transposições que aumenta o número de ciclos em quatro unidades (também conhecido como 0,2,2-movimento, ou seja a primeira transposição não aumenta o número de ciclos enquanto as próximas duas os aumentam em duas unidades cada).

Desta forma, vamos aplicar primeiramente uma transposição que transforma C em um ciclo orientado C' (o "0-movimento" de Bafna e Pevzner). Feito isso, usamos o Lema 44 duas vezes (para cada "2-movimento"), já que após a primeira aplicação do Lema 44 em

C' outro ciclo orientado deve existir. Por exemplo, após aplicar o Lema 44 pela primeira vez na Figura 6.4(l), o ciclo vermelho $B' = (5, 3, 1)$ que é não orientado se torna o ciclo orientado $B''(5, 1, 3)$.

O Lema 44 aplica até três transposições, assim, aplicar no máximo sete transposições aumenta em quatro unidades o número de ciclos balanceados.

A última coisa a se considerar é que a primeira transposição aplicada (0-movimento) não deve diminuir o número de ciclos balanceados. Como C é não orientado e longo, ele possui pelo menos dois open gates que devem ser fechados por um ou mais ciclos. Temos três casos dependendo de quais ciclos fecham estes open gates:

1. Se eles são fechados por um mesmo ciclo D, nós aplicamos uma transposição neste ciclo para transformar C em um ciclo orientado. Isto não mudará o número de ciclos balanceados, já que é aplicado apenas em um ciclo não orientado que não será quebrado em mais de um ciclo.

2. Se eles são fechados por dois ciclos desbalanceados D e F, aplicamos uma transposição nestes ciclos para transformar C em um ciclo orientado. Esta transposição não pode diminuir o número de ciclos balanceados, já que nem D nem F são balanceados.

3. Se eles são fechados por dois ciclos D e F, e pelo um deles é balanceado, vamos aplicar uma transposição que garante que pelo menos um dos ciclos resultantes D' e F' será balanceado (se D e F são ambos balanceados, então se D' é balanceado F' também será, e vice-versa). Seja $D = (\ldots, d^i, \ldots, d^j, \ldots)$ e seja $F = (\ldots, f^k, \ldots)$ tais que e_{d^i} e e_{f^k} estão localizados entre duas arestas pretas diferentes de C, e e_{d^j} não está localizado entre duas arestas pretas de C. Como não existem ciclos orientados, e como todo open gate deve ser fechado por outro ciclo, sempre podemos encontrar um ciclo C tal que existe um ciclo D como mencionado acima e e_{d^i} é a única aresta preta de D entre duas arestas pretas de C. Suponha, sem perda de generalidade, que D é um ciclo balanceado. Utilizando um argumento similar ao utilizado na prova do Lema 39, podemos concluir que existe uma aresta preta e_{d^j} tal que $w(e_{d^i}) + w(e_{c^j}) \geq f(e_{d^j}, e_{d^i})$, então podemos aplicar uma transposição criando um ciclo balanceado D' e, caso F seja balanceado, o novo ciclo F' também será balanceado. Além disso, agora temos que C é um ciclo orientado. □

Lema 46. *Seja G um grafo de breakpoints ponderado sem ciclos longos nem ciclos desbalanceados. Se G possui ciclos curtos, é possível aumentar o número de ciclos balanceados em duas unidades utilizando duas transposições.*

Demonstração. Seja $C = (c^1, c^2)$ um ciclo de G. Como o par e_c^1 e e_c^2 é um open gate, dado que não existem, existe um ciclo $D = (d^1, d^2)$ tal que $c^1 > d^1 > c^2 > d^2$. O próximo passo é aplicar uma transposição nas arestas pretas e_{c^1}, e_{c^2} e e_{d^2}. Utilizando a mesma ideia do Lema 39, esta transposição gera um ciclo trivial balanceado (com a aresta cinza e'_{c^2}) e um ciclo orientado balanceado.

Como D é um ciclo curto e estamos aplicando uma transposição em uma de suas arestas pretas, podemos aplicar esta transposição de modo que a próxima transposição aplicada no ciclo orientado gerado seja suficiente para gerar três ciclos balanceados. Seja

$F = (f^1, f^2, f^3)$ o ciclo orientado gerado. Note que $e_{f^2} = e_{d^1}$, pois a primeira transposição não agiu nesta aresta preta. Além disso, e_{f^2} é adjacente a ambas as arestas cinzas que estão em D. Se $k = w(e_{d^2}) - w(e'_{d^2}) > 0$, a primeira transposição deve mover $k + w(e'_{c^1})$ para $w(e_{f^1})$, ou para $w(e'_{c^1})$, caso contrário. Esta configuração para F é suficiente para gerar três ciclos balanceados com apenas uma transposição adicional.

Desta forma, aplicamos duas transposições e aumentamos o número de ciclos balanceados em duas unidades. □

A Figura 6.4 mostra uma sequência de transposições intergênicas usando os lemas 39-46. Na Figura 6.4(a) o ciclo azul $A = (6, 4, 2)$ é não orientado e negativo, e podemos aplicar o Lema 39 usando também o ciclo positivo $B = (7, 5)$, e a transposição intergênica $\rho^{(2,6,7)}_{(5,4,1)}$ gera na Figura 6.4(b) o ciclo trivial balanceado $C = (2)$ e o ciclo não trivial $D = (7, 5, 3, 6)$, que neste caso é negativo e orientado.

Na Figura 6.4(b), podemos utilizar o Lema 41 em D, o que nos leva a aplicar a transposição $\rho^{(3,6,7)}_{(2,1,0)}$ que gera na Figura 6.4(c) o ciclo trivial balanceado $E = (3)$, o ciclo trivial $F = (7)$ e um ciclo curto $G = (6, 4)$. Note que F é negativo e G é balanceado.

Na Figura 6.4(c), não existem ciclos longos, e podemos utilizar o Lema 43 no ciclo trivial negativo F. Este lema requer um ciclo positivo para interagir com o ciclo trivial negativo, sendo que $H = (8)$ é o único ciclo positivo. Como H também é trivial, precisamos utilizar uma aresta preta de um ciclo balanceado para aplicar a transposição, então vamos utilizar o ciclo curto balanceado G. Duas transposições consecutivas nestas arestas pretas (veja as figuras 6.4(c)-6.4(d)) geram dois ciclos balanceados $I = (7)$ e $J = (8)$ na Figura 6.4(e), sem modificar a aresta preta do ciclo G.

O grafo de breakpoints ponderado na Figura 6.4(e) possui apenas ciclos balanceados, e eles são curtos ou triviais. Neste caso, podemos utilizar o Lema 46 que aplica duas transposições intergênicas em dois ciclos curtos. Neste caso, $G = (6, 4)$ e $K = (5, 1)$. Estas duas transposições aplicadas em G e K (veja as figuras 6.4(e-f)) geram quatro ciclos balanceados, aumentando o número de ciclos balanceados em duas unidades e completando o processo de transformação de $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ — o grafo de breakpoints ponderado da Figura 6.4(g) possui apenas ciclos balanceados triviais.

Na Figura 6.4(h), o ciclo verde $A' = (9, 7, 8)$ é orientado e dado que também é balanceado podemos utilizar o Lema 44. As três transposições aplicadas nas figuras 6.4(h-j) transformam A' em três ciclos balanceados.

Na Figura 6.4(k), temos apenas ciclos não orientados, e podemos utilizar o Lema 45 que aplica até sete transposições e aumenta o número de ciclos balanceados em quatro unidades. A primeira transposição é aplicada nas arestas pretas de $B' = (5, 3, 1)$ transformando o ciclo (em azul na Figura 6.4(k)) em um ciclo orientado balanceado $C' = (6, 2, 4)$ (o 0-movimento).

Continuando a aplicação do Lema 45, podemos utilizar agora o primeiro 2-movimento (que nada mais é que o Lema 44) para transformar C' em três ciclos balanceados. Neste caso, apenas uma transposição é suficiente — note que esta transposição também transforma o ciclo não orientado B' em um ciclo orientado $B'' = (5, 1, 3)$.

Na Figura 6.4(m), podemos usar o segundo 2-movimento (ou seja, o Lema 44) que transforma o ciclo vermelho B'' em três ciclos balanceados triviais, novamente apenas uma

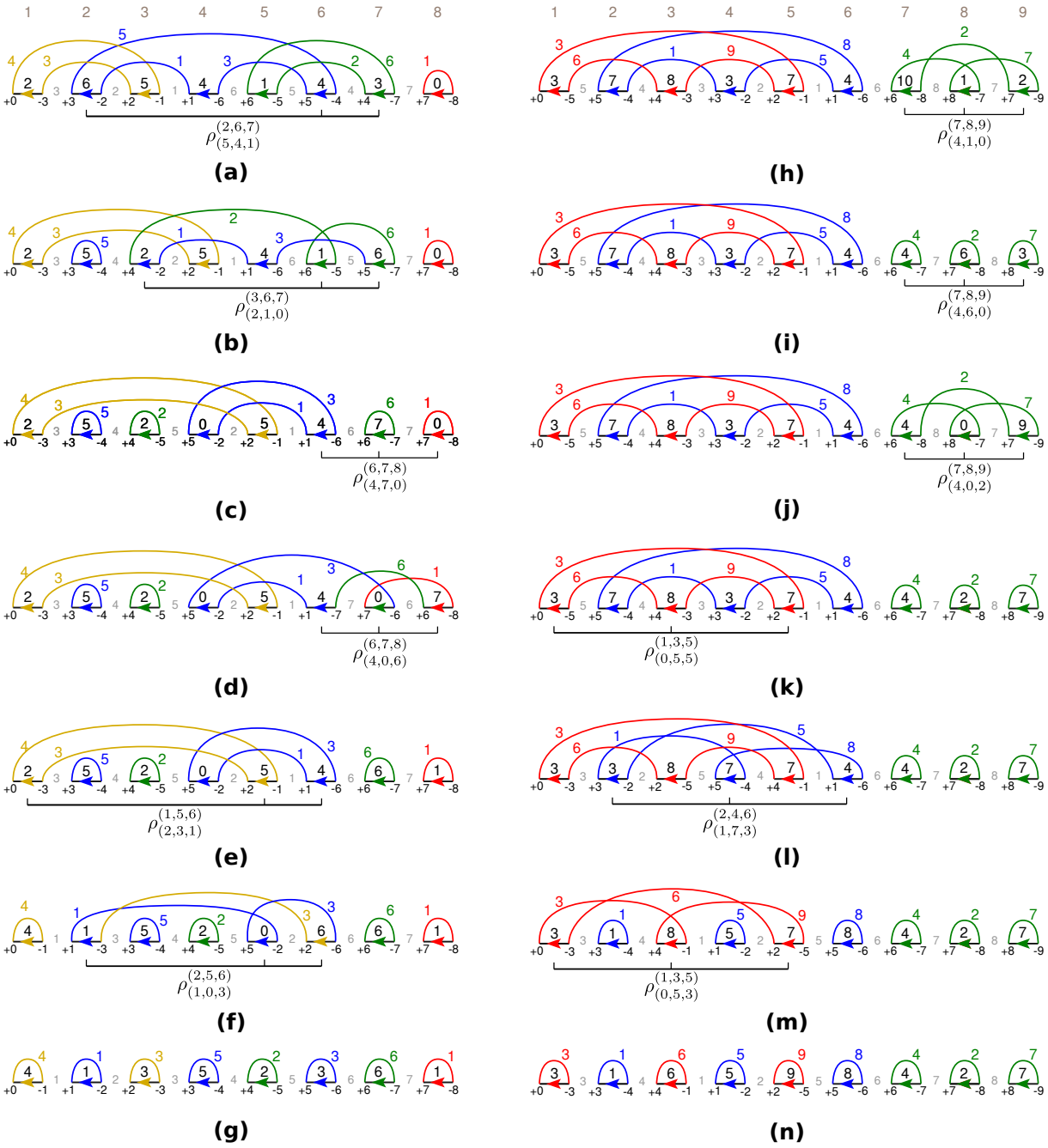


Figura 6.4: De **(a)-(g)** temos uma sequência de transposições intergênicas que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ com $\pi = (3\ 2\ 1\ 6\ 5\ 4\ 7), \breve{\pi} = (2, 6, 5, 4, 1, 4, 3, 0)$, e $\breve{\iota} = (4, 1, 3, 5, 2, 3, 6, 1)$ usando os lemas 39-43 e 46. De **(h)-(n)** temos uma sequência de transposições intergênicas que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ com $\pi = (5\ 4\ 3\ 2\ 1\ 6\ 8\ 7), \breve{\pi} = (3, 7, 8, 3, 7, 4, 10, 1, 2)$ e $\breve{\iota} = (3, 1, 6, 5, 9, 8, 4, 2, 7)$ usando os lemas 44 e 45.

transposição foi suficiente. Assim, aplicamos 3 transposições e aumentamos o número de ciclos balanceados em quatro unidades, completando o processo de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ — a Figura 6.4(m) possui apenas ciclos balanceados triviais.

6.5 Uma 2-Aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas

Utilizando os lemas apresentados na Seção 6.3 vamos apresentar um algoritmo de 2-aproximação para o problema de Ordenação de Permutações com Sinais por Reversões utilizando o grafo de breakpoints ponderado. Como uma reversão pode aumentar o número de ciclos balanceados em não mais de uma unidade, nosso objetivo é mostrar que sempre podemos aumentar o número de ciclos balanceados em uma unidade utilizando no máximo duas reversões.

O Algoritmo 7 utiliza os lemas 36-38 para encontrar uma sequência de reversões que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Vamos discutir brevemente sua corretude. Enquanto $(\pi, \breve{\pi})$ é diferente de $(\iota, \breve{\iota})$ (condição na linha 2), sabemos que pelo menos uma das seguintes condições é verdade com relação a $G(\pi, \breve{\pi}, \breve{\iota})$:

- Existe um ciclo não trivial C balanceado ou negativo em $G(\pi, \breve{\pi}, \breve{\iota})$, e C é divergente. Neste caso, o algoritmo irá aplicar a reversão recebida na linha 4;
- Existe um ciclo não trivial C negativo ou balanceado em $G(\pi, \breve{\pi}, \breve{\iota})$, e todo ciclo balanceado ou negativo de $G(\pi, \breve{\pi}, \breve{\iota})$ é convergente. Neste caso, o algoritmo aplicará as duas reversões recebidas na linha 5;
- Existe um ciclo trivial negativo C em $G(\pi, \breve{\pi}, \breve{\iota})$. Neste caso, o algoritmo aplicará as duas reversões recebidas na linha 6.

Toda vez que o algoritmo alcança a linha 7, ele aplica uma ou duas reversões que aumentam o número de ciclos balanceados em uma unidade, o que garante tanto a aproximação desejada, quanto que o algoritmo irá parar após no máximo k iterações (onde $k = c_b(\iota, \breve{\iota}, \breve{\iota}) - c_b(\pi, \breve{\pi}, \iota)$).

Podemos encontrar um ciclo conforme descrito em tempo $O(n)$, e em quais posições a reversão deve ser aplicada em tempo $O(n^2)$. Este processo é repetido por até $k \leq n$ vezes; assim, no pior caso, o algoritmo executa em tempo $O(n^3)$.

6.6 Uma 3.5-Aproximação para a Ordenação de Permutações sem Sinais por Transposições Intergênicas

Vamos agora apresentar um algoritmo de 3.5-aproximação para transposições intergênicas utilizando os lemas apresentados na Seção 6.4. O Algoritmo 8 utiliza os lemas 39, 41 e 43-46, para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação sem sinais.

Vamos explicar brevemente sua corretude. Enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$, temos que uma das seguintes situações deve ser verdadeira: (i) existe um ciclo orientado, caso em que quebramos tal ciclo se ele é balanceado ou negativo nas linhas 4-6; (ii) existe um ciclo negativo, caso em que aumentamos o número de ciclos balanceados se ele é não orientado

Algoritmo 7: Algoritmo de 2-aproximação para o problema de Ordenação de Permutações com Sinais por Reversões Intergênicas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação com sinais.
Resultado: uma sequência de reversões intergênicas $\mathcal{S}_\rho$ tal que $(\pi, \breve{\pi}) \cdot \mathcal{S}_\rho = (\iota, \breve{\iota})$.

```
1 seq ← ∅
2 enquanto (π, π̆) ≠ (ι, ῐ) faça
3 |   G ← o grafo de breakpoints ponderado G(π, π̆, ῐ)
4 |   se existe um ciclo divergente C em G que é ou balanceado ou negativo então
  |     S_ρ ← reversão garantida pelo Lema 36
5 |   senão se existe um ciclo não trivial C em G que é ou balanceado ou negativo
  |     então S_ρ ← reversões garantidas pelo Lema 37
6 |   senão S_ρ ← reversões garantidas pelo Lema 38
7 |   (π, π̆) ← (π, π̆) · S_ρ
8 |   seq.append(S_ρ)
9 retorna seq
```

ou trivial nas linhas 8-11; e (iii) existe um ciclo longo ou curto, caso em que quebramos este ciclo nas linhas 14 e 16, respectivamente.

Note que não consideramos em nenhum momento os ciclos positivos orientados, uma vez que eles se tornarão ou balanceados ou negativos antes do algoritmo chegar na condição (iii), e eles serão manipulados na condição (i) em algum momento. Se o algoritmo chega na condição (iii), então todos os ciclos estão balanceados, dado que ciclos negativos são manipulados em (i) e (ii).

Com relação à complexidade do Algoritmo 8, o laço nas linhas 2–18 é iterado por até $m = n + 1$ vezes, dado que cada vez que o algoritmo aplica um dos lemas ele aumenta o número de ciclos balanceados em pelo menos uma unidade. Encontrar qual lema utilizar (e em quais posições a transposição deve agir) requer $O(n^2)$. Desta forma, a complexidade total do Algoritmo 8 é $O(n^3)$.

Vamos agora discutir sobre a aproximação garantida pelo Algoritmo 8. Note que alguns passos requerem mais de uma transposição e, desta forma, a aproximação será computada da seguinte maneira.

Definição 4. *Seja $(\pi, \breve{\pi}, \breve{\iota})$ uma instância e seja $\mathcal{S}_\rho = (\rho_1, \rho_2, \ldots, \rho_d)$ uma sequência de transposições intergênicas tal que $(\pi, \breve{\pi}) \cdot \mathcal{S}_\rho = (\sigma, \breve{\sigma})$. Pelo Lema 33, $\mathcal{S}_\rho$ aumentará em até $2d$ o número de ciclos balanceados. Desta forma, o fator de aproximação será no máximo $\frac{2d}{c_b(\sigma,\breve{\sigma},\breve{\iota}) - c_b(\pi,\breve{\pi},\breve{\iota})}$.*

O lema a seguir mostra que cada passo do Algoritmo 8 garante um fator de aproximação menor ou igual a 3.5, o que nos leva ao fator de aproximação proposto.

Lema 47. *O Algoritmo 8 possui um fator de aproximação igual a* 3.5.

Demonstração. Utilizamos a fórmula dada pela Definição 4 para calcular a aproximação de cada passo e os resultados que são apresentados na Tabela 6.1, onde podemos ver que o fator de aproximação máximo obtido é igual a 3.5. □

Algoritmo 8: Algoritmo de 3.5-aproximação para a Ordenação de Permutações sem Sinais por Transposições Intergênicas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação sem sinais.
Resultado: uma sequência de transposições intergênicas $\mathcal{S}_\rho$ tal que $(\pi, \breve{\pi}) \cdot \mathcal{S}_\rho = (\iota, \breve{\iota})$.

```
seq ← ∅
enquanto (π, π̆) ≠ (ι, ῐ) faça
    G ← o grafo de breakpoints ponderado G(π, π̆, ῐ)
    se existe um ciclo longo orientado C em G que é balanceado ou negativo então
        se C é balanceado então
            S_ρ ← transposições garantidas pelo Lema 44
        senão S_ρ ← transposição garantida pelo Lema 41        ▷ C é negativo
    senão se existe um ciclo negativo C em G que é não orientado ou trivial então
        se C é não orientado então
            S_ρ ← transposição garantida pelo Lema 39
        senão S_ρ ← transposições garantidas pelo Lema 43      ▷ C é trivial
    senão                ▷ G só possui ciclos balanceados e não orientados
        se existe um ciclo longo C em G então
            S_ρ ← transposições garantidas pelo Lema 45
        senão            ▷ existem apenas ciclos curtos e triviais em G
            S_ρ ← transposições garantidas pelo Lema 46
    (π, π̆) ← (π, π̆) · S_ρ
    seq.append(S_ρ)
retorna seq
```

6.7 Uma 3-aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas e Transposições Intergênicas

Nesta seção, apresentamos um algoritmo de aproximação que usa tanto reversões intergênicas quanto transposições intergênicas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação com sinais. Para isso, utilizaremos lemas já definidos neste capítulo juntamente com o Lema 48, que trata o caso de um ciclo longo balanceado convergente C em um grafo $G(\pi, \breve{\pi}, \breve{\iota})$ em que todos os ciclos estão balanceados, são convergentes e não orientados. A ideia é substituir o Lema 45 por este novo lema que, conforme veremos mais tarde, fornece um fator de aproximação melhor.

Lema 48. *Seja $C = (c^1, \ldots, c^k)$, $k > 2$, um ciclo convergente balanceado de um grafo de breakpoints ponderado $G(\pi, \breve{\pi}, \breve{\iota})$ no qual todos os ciclos balanceados são convergentes. É possível aumentar o número de ciclos balanceados em duas unidades utilizando três reversões.*

Demonstração. Seja C um ciclo balanceado e sejam e_{c^i} e $e_{c^{i+1}}$ duas arestas pretas de C, com $1 \leq i < k$. O par e_{c^i} e $e_{c^{i+1}}$ é um open gate e sabemos por Bafna e Pevzner [3] que existe um ciclo não trivial $D = (\ldots, d^i, \ldots, d^j, \ldots)$ que fecha este open gate, ou seja, que

Tabela 6.1: Cálculo da aproximação máxima ($\mathbf{A_{max}}$) utilizando a Definição 4, de cada passo do Algoritmo 8, dado o número máximo de transposições existentes em $\mathcal{S}_\rho$ e o número mínimo de novos ciclos balanceados gerados.

Passo	Tamanho máximo de $\mathcal{S}_\rho$	$\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \mathcal{S}_\rho)$ mínimo	$\mathrm{A_{max}}$
Lema 44	3	2	$\frac{6}{2} = 3$
Lema 41	1	1	$\frac{2}{1} = 2$
Lema 39	1	2	$\frac{2}{1} = 2$
Lema 43	2	2	$\frac{4}{2} = 2$
Lema 45	7	4	$\frac{14}{4} = 3.5$
Lema 46	2	2	$\frac{4}{2} = 2$

ou c^i ou c^{i+1} está entre d^i e d^j. Uma reversão aplicada nas arestas pretas e_{d^i} e e_{d^j} faz com que C seja transformado em um ciclo divergente. Pelo Lema 36, podemos aplicar uma reversão em C que aumenta o número de ciclos balanceados em uma unidade. Após esta segunda reversão ser aplicada, temos que D será um ciclo divergente, então podemos aplicar uma reversão em D seguindo o mesmo lema que aumentará o número de ciclos balanceados em uma unidade. □

O Algoritmo 9 aplica o lema acima, o Lema 36, e os lemas do Algoritmo 8, com exceção do Lema 45 que fornece fator de aproximação igual a 3.5. Como os passos deste algoritmo são similares aos do Algoritmo 8, temos que a complexidade total do Algoritmo 9 também é $O(n^3)$.

Vamos discutir como o algoritmo para e ordena qualquer genoma $(\pi, \breve{\pi})$. O algoritmo começa aplicando reversões em ciclos divergentes existentes, exceto pelos positivos. A seguir, os ciclos orientados balanceados ou negativos são tratados, bem como os ciclos negativos restantes (neste ponto ciclos positivos são transformados em negativos ou balanceados). O último passo do algoritmo é quando não existem mais nem ciclos divergentes nem desbalanceados. Neste caso, o algoritmo aplicará ou uma sequência de três reversões, se existe um ciclo longo, ou uma sequência de duas transposições em dois ciclos curtos.

O lema a seguir mostra como garantimos o fator de aproximação do Algoritmo 9. Note que o limitante inferior para reversões e transposições é o mesmo que o limitante para transposições apenas, logo podemos reutilizar os cálculos apresentados para os passos do Algoritmo 8.

Lema 49. *O Algoritmo 9 possui um fator de aproximação igual a* 3.

Demonstração. Pela Tabela 6.1, sabemos a aproximação máxima garantida pelo Lema 44, que é igual a 3, bem como dos lemas 41, 39, 43 e 46 que são iguais a 2. O Lema 36 aplica uma reversão que aumenta o número de ciclos balanceados em uma unidade e sua aproximação segundo a Definição 4 é igual a $\frac{2}{1} = 2$. O Lema 48 utiliza três reversões que aumentam o número de ciclos balanceados em duas unidades, o que resulta, pela

Algoritmo 9: Algoritmo de 3-aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas e Transposições Intergênicas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação com sinais.
Resultado: uma sequência de reversões intergênicas e transposições intergênicas $\mathcal{S}_\rho$ tal que $(\pi, \breve{\pi}) \cdot \mathcal{S}_\rho = (\iota, \breve{\iota})$.

$seq \leftarrow \emptyset$
enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$ **faça**
  $G \leftarrow$ o grafo de breakpoints ponderado $G(\pi, \breve{\pi}, \breve{\iota})$
  se *existe um ciclo divergente C em G que é ou balanceado ou negativo* **então**
    $\mathcal{S}_\rho \leftarrow$ reversão garantida pelo Lema 36
  senão se *existe um ciclo orientado C em G que é balanceado ou negativo* **então**
    se *C é balanceado* **então**
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 44
    senão ▷ *C é negativo*
      $\mathcal{S}_\rho \leftarrow$ transposição garantida pelo Lema 41
  senão se *existe um ciclo negativo C em G que é não orientado ou trivial* **então**
    se *C é não orientado* **então**
      $\mathcal{S}_\rho \leftarrow$ transposição garantida pelo Lema 39
    senão ▷ *C é trivial*
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 43
  senão ▷ *G possui apenas ciclos balanceados*
    se *existe um ciclo longo balanceado C em G* **então**
      $\mathcal{S}_\rho \leftarrow$ reversões garantidas pelo Lema 48
    senão ▷ *G possui apenas ciclos curtos e triviais*
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 46
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot \mathcal{S}_\rho$
  $seq.append(\mathcal{S}_\rho)$
retorna seq

Definição 4, na aproximação $\frac{6}{2} = 3$, sendo este o fator de aproximação final do Algoritmo 9. □

6.8 Transposições Genéricas Intergênicas

Até este ponto, assumimos que uma transposição troca a posição de dois segmentos tal que cada segmento possui pelo menos um gene (elemento). Esta restrição é mandatória quando não consideramos as regiões intergênicas, mas quando as consideramos podemos também considerar o caso em que um pedaço de região intergênica (sem genes) troca de posição com um segmento adjacente com pelo menos um gene (e vice-versa) — note que apenas trocaríamos a posição de nucleotídeos de uma mesma região intergênica caso fosse permitido que os segmentos não possuíssem nenhum gene, o que não faz sentido em nosso problema já que resultaria no mesmo genoma.

Assim, uma *transposição genérica intergênica* $\rho^{(i,j,k)}_{(x,y,z)}$, com $i \neq k$ e $1 \leq i \leq j \leq k \leq n+1$, pode ser definida ou como uma *troca intergênica*, caso $i = j$ ou $j = k$, ou como uma *transposição intergênica* (como definida anteriormente, onde $i < j < k$).

Uma troca intergênica $\rho^{(i,i,k)}_{(x,y,z)}$, com $1 \leq i < k \leq n+1$, $x \in [0..\breve{\pi}_i - 1]$, $y \in [1..\breve{\pi}_i]$, $x < y$ e $z \in [0..\breve{\pi}_k]$, corta $\breve{\pi}_i$ após x nucleotídeos e também $\breve{\pi}_i$ após y nucleotídeos e insere o segmento de tamanho $y - x$ em $\breve{\pi}_k$ após os primeiros z nucleotídeos. Isto significa que $(\pi, \breve{\pi}) \cdot \rho$ resulta em $(\pi, \breve{\pi}')$, com $\breve{\pi}'_j = \breve{\pi}_j$ se $j \notin \{i, k\}$, $\breve{\pi}'_i = \breve{\pi}_i - y + x$ e $\breve{\pi}'_k = \breve{\pi}_k + y - x$.

De modo similar, uma troca intergênica $\rho^{(i,k,k)}_{(x,y,z)}$, com $1 \leq i < k \leq n+1$, $x \in [0..\breve{\pi}_i]$, $y \in [0..\breve{\pi}_k - 1]$, $z \in [1..\breve{\pi}_k]$ e $y < z$, corta $\breve{\pi}_k$ após y nucleotídeos e também $\breve{\pi}_k$ após z nucleotídeos e insere o segmento de tamanho $z - y$ em $\breve{\pi}_i$ após os primeiros x nucleotídeos. Isto significa que $(\pi, \breve{\pi}) \cdot \rho$ resulta em $(\pi, \breve{\pi}')$, com $\breve{\pi}'_j = \breve{\pi}_j$ se $j \notin \{i, k\}$, $\breve{\pi}'_i = \breve{\pi}_i + z - y$ e $\breve{\pi}'_k = \breve{\pi}_k - z + y$. A Figura 6.5 mostra um exemplo de uma troca intergênica.

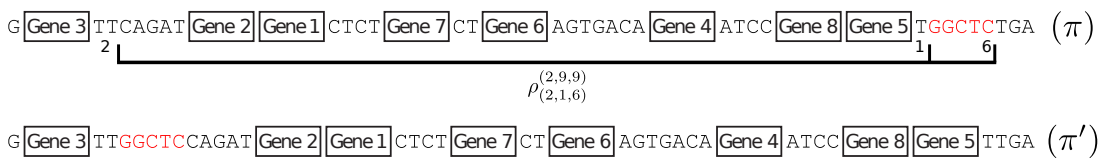


Figura 6.5: Um genoma (fictício) representado pela permutação $\pi = (3\ 2\ 1\ 7\ 6\ 4\ 8\ 5)$ com $\breve{\pi} = (1\ 7\ 0\ 4\ 2\ 7\ 4\ 0\ 9)$. A troca intergênica $\rho^{(2,9,9)}_{(2,1,3)}$ transforma $(\pi, \breve{\pi})$ em $(\pi', \breve{\pi}')$ com $\pi' = \pi$ e $\breve{\pi}' = (1\ 12\ 0\ 4\ 2\ 7\ 4\ 0\ 4)$, movendo um segmento de nucleotídeos de tamanho $6 - 1 = 5$ da região intergênica $\breve{\pi}_9$ para $\breve{\pi}_2$, após seu segundo nucleotídeo.

A *distância de transposição genérica intergênica*, denotada por $d_{gt}(\pi, \breve{\pi}, \breve{\iota})$, é o número mínimo m de transposições genéricas $\rho_1, \ldots, \rho_m$ que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação sem sinais. Já a *distância de reversão intergênica e transposição genérica intergênica*, denotada por $d_{\overline{rgt}}(\pi, \breve{\pi}, \breve{\iota})$, é o número mínimo m de reversões e transposições genéricas $\rho_1, \ldots, \rho_m$ que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, onde π é uma permutação com sinais.

Note que uma troca intergênica pode ser aplicada em $(\pi, \breve{\pi})$ se, e somente se, $\breve{\pi}_i > 0$. Como as reduções utilizadas nos lemas 34 e 35 utilizam $\breve{\pi}_i = 0$, nenhuma troca intergênica pode ser aplicada nestas instâncias, então podemos utilizar o mesmo argumento para provar que os problemas também permanecem NP-difíceis quando permitimos transposições genéricas.

Lema 50. *Encontrar a distância de ordenação por transposições genéricas é* NP*-difícil.*

Demonstração. Similar à prova do Lema 34. □

Lema 51. *Encontrar a distância de ordenação por reversões intergênicas e transposições intergênicas é* NP*-difícil.*

Demonstração. Similar à prova do Lema 35. □

6.9 O Efeito de Transposições Genéricas Intergênicas em Grafos de Breakpoints Ponderados

Os lemas a seguir mostram como trocas intergênicas modificam o número de ciclos e ciclos balanceados de $G(\pi, \breve{\pi}, \breve{\iota})$ quando aplicadas a um genoma $(\pi, \breve{\pi})$.

Lema 52. *Seja* $(\pi, \breve{\pi}, \breve{\iota})$ *uma instância. Para qualquer troca intergênica* ρ *temos que* $\Delta c(\pi, \breve{\pi}, \breve{\iota}, \rho) = 0$.

Demonstração. Como trocas intergênicas não alteram a ordem dos genes, elas não podem modificar o conjunto de arestas de $G(\pi, \breve{\pi}, \breve{\iota})$. □

Lema 53. *Seja* $(\pi, \breve{\pi}, \breve{\iota})$ *uma instância. Para qualquer troca intergênica* ρ *temos que* $\Delta c_b(\pi, \breve{\pi}, \breve{\iota}, \rho) \leq 2$.

Demonstração. Note que uma troca intergênica é aplicada em um ou dois ciclos — ela modifica duas arestas pretas apenas. Se ela é aplicada em um único ciclo, ela não modificará o número de ciclos balanceados. Caso contrário, como uma troca intergênica não altera os ciclos, o melhor cenário é mover o peso excedente de um ciclo para outro de modo a gerar dois ciclos balanceados. □

Assim, conseguimos definir os seguintes limitantes inferiores considerando transposições genéricas.

Teorema 19. $d_{gt}(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{n+1-c_b(\pi,\breve{\pi},\breve{\iota})}{2}$.

Demonstração. Pela Propriedade 5, temos que $c_b(\iota, \breve{\iota}, \breve{\iota}) = n + 1$. Assim, precisamos encontrar uma forma de aumentar o número de ciclos balanceados de $c_b(\pi, \breve{\pi}, \breve{\iota})$ para $n+1$. O lema segue pelo fato deste número aumentar em no máximo duas unidades para cada transposição (Lema 33) ou troca intergênica (Lema 53). □

Teorema 20. $d_{\overline{rgt}}(\pi, \breve{\pi}, \breve{\iota}) \geq \frac{n+1-c_b(\pi,\breve{\pi},\breve{\iota})}{2}$.

Demonstração. Sabemos que precisamos ir de $c_b(\pi, \breve{\pi}, \breve{\iota})$ para $c_b(\iota, \breve{\iota}, \breve{\iota}) = n + 1$. O lema segue pelo fato deste número aumentar em no máximo duas unidades para cada transposição (Lema 33), em no máximo uma unidade para cada reversão (Lema 31) e em no máximo duas unidades para cada troca intergênica (Lema 53). □

Agora vamos modificar parte dos lemas utilizados pelo Algoritmo 8 para tirar vantagem das trocas intergênicas. Começamos com mais uma melhoria dos lemas 43 e 44.

Lema 54. *Seja* C *um ciclo trivial negativo. É possível aumentar o número de ciclos balanceados em uma unidade utilizando uma troca intergênica.*

Demonstração. Como $C = (c^1)$ é negativo, sabemos que existe um ciclo positivo $D = (d^1)$ no grafo de breakpoints ponderado. Aplique a troca intergênica nas arestas pretas e_{c^1} e e_{d^1}, movendo o peso excedente de C para D. Esta troca intergênica transforma C em um ciclo balanceado. □

Utilizando o Lema 54, podemos diminuir o número de operações aplicadas no Lema 44, como explicamos no lema a seguir.

Lema 55. *Seja* C *um ciclo longo orientado balanceado. É possível aumentar o número de ciclos balanceados em duas unidades utilizando até duas transposições genéricas.*

Demonstração. A prova é similar à prova do Lema 44, com exceção do último caso, em que temos um ciclo trivial negativo C'' após a primeira transposição. Aqui, utilizamos o Lema 54 para transformar C'' em um ciclo balanceado com uma troca intergênica apenas. □

Como o Lema 45 utiliza o Lema 43, também podemos diminuir o número de operações aplicadas como explicado no lema a seguir.

Lema 56. *Seja C um ciclo longo não orientado balanceado em um grafo de breakpoints ponderados em que não existem ciclos orientados. É possível aumentar o número de ciclos balanceados em quatro unidades com a aplicação de até cinco transposições genéricas.*

Demonstração. A prova também é similar à prova do Lema 45, mas agora utilizamos o Lema 55 duas vezes ao invés do Lema 44, dado que este lema utiliza uma operação a menos. Desta forma, o número de transposições genéricas aplicadas diminui em duas unidades e o lema segue. □

6.10 Uma 2.5-aproximação para a Ordenação de Permutações sem Sinais por Transposições Genéricas Intergênicas

Note que o Algoritmo 8 pode ser utilizado para o problema de Ordenação de Permutações sem Sinais por Transposições Genéricas Intergênicas, mas é possível obter um algoritmo com um fator de aproximação de 2.5. Para isto, vamos ajustar o Algoritmo 8 para utilizar os novos lemas que aplicam trocas intergênicas, conforme os passos descritos no Algoritmo 10.

Dado que os passos dos algoritmos 8 e 10 utilizam a mesma ideia, as provas de corretude e de complexidade computacional são as mesmas. Vamos argumentar sobre o novo fator de aproximação.

Note que o limitante inferior definido no Teorema 19 é o mesmo que o definido no Teorema 16, assim, também podemos utilizar a fórmula da Definição 4 para calcular o fator de aproximação de cada passo do algoritmo, conforme mostra o Lema 57.

Lema 57. *O Algoritmo 10 possui um fator de aproximação igual a* 2.5.

Demonstração. Já sabemos que os lemas 39, 41 e 46 possuem fator de aproximação igual a 2. A troca intergênica aplicada pelo Lema 54 aumenta em uma unidade o número de ciclos balanceados, então seu fator de aproximação, segundo a Definição 4, é $\frac{2}{1} = 2$. As duas transposições genéricas aplicadas no Lema 55 aumentam em duas unidades o número de ciclos balanceados, o que resulta no fator de aproximação $\frac{4}{2} = 2$. Por fim, as cinco transposições genéricas aplicadas pelo Lema 56 aumentam em quatro unidades o número de ciclos balanceados, o que resulta no fator de aproximação $\frac{10}{4} = 2.5$, que é então o fator de aproximação do Algoritmo 10. □

Algoritmo 10: Uma 2.5-aproximação para a Ordenação de Permutações sem Sinais por Transposições Genéricas Intergênicas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação sem sinais.
Resultado: uma sequência de transposições genéricas intergênicas $\mathcal{S}_\rho$ tal que $(\pi, \breve{\pi}) \cdot \mathcal{S}_\rho = (\iota, \breve{\iota})$.

$seq \leftarrow \emptyset$
enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$ **faça**
  $G \leftarrow$ o grafo de breakpoints ponderado $G(\pi, \breve{\pi}, \breve{\iota})$
  se *existe um ciclo orientado C em G que é balanceado ou negativo* **então**
    se *C é balanceado* **então**
      $\mathcal{S}_\rho \leftarrow$ transposições genéricas garantidas pelo Lema 55
    senão ▷ *C é negativo*
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 41
  senão se *existe um ciclo negativo C em G que é não orientado ou trivial* **então**
    se *C é não orientado* **então**
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 39
    senão ▷ *C é trivial*
      $\mathcal{S}_\rho \leftarrow$ troca intergênica garantida pelo Lema 54
  senão ▷ *G possui apenas ciclos balanceados*
    se *existe um ciclo longo balanceado C em G* **então**
      $\mathcal{S}_\rho \leftarrow$ transposições genéricas garantidas pelo Lema 56
    senão ▷ *G possui apenas ciclos curtos e triviais*
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 46
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot \mathcal{S}_\rho$
  $seq.append(\mathcal{S}_\rho)$
retorna seq

6.11 Algoritmo de 2.5-aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas e Transposições Genéricas Intergênicas

Utilizando a mesma estratégia da Ordenação por Transposições Genéricas, vamos modificar alguns lemas do Algoritmo 9 para utilizar os novos lemas que aplicam as trocas intergênicas. Vamos então remover os Lemas 44, 43 e 48, que são os passos cujo fator de aproximação é 3, e utilizar os lemas 54-56, conforme os passos descritos no Algoritmo 11.

A corretude e a complexidade do Algoritmo 11 são as mesmas do Algoritmo 9, dado que também utilizam a mesma ideia. O lema a seguir define seu fator de aproximação utilizando novamente a Definição 4, dado que os limitantes inferiores dos teoremas 20 e 16 são os mesmos.

Lema 58. *O Algoritmo 11 possui um fator de aproximação igual a* 2.5.

Demonstração. Já sabemos, pelos algoritmos anteriores, que as operações aplicadas pelos lemas 39, 41, 46, 54 e 55 possuem fator de aproximação igual a 2, e as operações aplicadas pelo Lema 56 possui fator de aproximação igual a 2.5, que é então o fator de aproximação do Algoritmo 11. □

Algoritmo 11: Uma 2.5-aproximação para a Ordenação de Permutações com Sinais por Reversões Intergênicas e Transposições Genéricas Intergênicas.

Dados: uma instância $(\pi, \breve{\pi}, \breve{\iota})$ tal que π é uma permutação com sinais.
Resultado: uma sequência de reversões intergênicas e transposições genéricas intergênicas $\mathcal{S}_\rho$ tal que $(\pi, \breve{\pi}) \cdot \mathcal{S}_\rho = (\iota, \breve{\iota})$.

$seq \leftarrow \emptyset$
enquanto $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$ **faça**
  $G \leftarrow$ o grafo de breakpoints ponderado $G(\pi, \breve{\pi}, \breve{\iota})$
  se *existe um ciclo divergente* C *em* G *que é balanceado ou negativo* **então**
    $\mathcal{S}_\rho \leftarrow$ reversão garantida pelo Lema 36
  senão se *existe um ciclo orientado* C *em* G *que é balanceado ou negativo* **então**
    se C *é balanceado* **então**
      $\mathcal{S}_\rho \leftarrow$ transposição genérica garantida pelo Lema 55
    senão ▷ C é negativo
      $\mathcal{S}_\rho \leftarrow$ transposição garantida pelo Lema 41
  senão se *existe um ciclo negativo* C *em* G *que é não orientado ou trivial* **então**
    se C *é não orientado* **então**
      $\mathcal{S}_\rho \leftarrow$ transposição garantida pelo Lema 39
    senão ▷ C é trivial
      $\mathcal{S}_\rho \leftarrow$ troca intergênica garantida pelo Lema 54
  senão ▷ C possui apenas ciclos balanceados
    se *existe um ciclo longo balanceado* C *em* G **então**
      $\mathcal{S}_\rho \leftarrow$ transposições genéricas garantidas pelo Lema 56
    senão ▷ C possui apenas ciclos curtos e triviais
      $\mathcal{S}_\rho \leftarrow$ transposições garantidas pelo Lema 46
  $(\pi, \breve{\pi}) \leftarrow (\pi, \breve{\pi}) \cdot \mathcal{S}_\rho$
  $seq.append(\mathcal{S}_\rho)$
retorna seq

6.12 Experimentos

Nós criamos bases de dados com instâncias simuladas para verificar tanto a evidência da utilidade de considerar as regiões intergênicas quanto o desempenho dos algoritmos de aproximação descritos até o momento neste capítulo, e também implementamos os algoritmos 7-11.

Cinco conjuntos de dados foram gerados: para reversões intergênicas ($\mathbf{DB_{IR}}$), para transposições intergênicas ($\mathbf{DB_{IT}}$), para transposições genéricas ($\mathbf{DB_{GT}}$), para reversões intergênicas e transposições intergênicas ($\mathbf{DB_{IRIT}}$), e para reversões intergênicas e transposições genéricas ($\mathbf{DB_{IRGT}}$).

Para cada conjunto de dados, começamos com 100 pares $(\iota, \breve{\iota})$, tal que ι possui 100 elementos e cada $\breve{\iota}_i$ recebeu um valor aleatório no intervalo [0..100], para $i \in [1..101]$.

Depois, para cada par $(\iota, \breve{\iota})$ foram geradas 100 triplas $(\pi, \breve{\pi}, \breve{\iota})$ aplicando neste par:

- d reversões intergênicas, com d indo de 5 até 100 em intervalos de 5 ($\mathbf{DB_{IR}}$).
- d transposições intergênicas, com d indo de 5 até 100 em intervalos de 5 ($\mathbf{DB_{IT}}$).

- d transposições genéricas (sendo 80% transposições intergênicas e 20% trocas intergênicas), com d indo de 5 até 100 em intervalos de 5 ($\mathbf{DB_{GT}}$).
- d reversões intergênicas e transposições intergênicas (sendo 50% de cada), com d indo de 10 até 100 em intervalos de 10 ($\mathbf{DB_{IRIT}}$).
- d reversões intergênicas e transposições genéricas (sendo 50% reversões intergênicas, 40% transposições intergênicas e 10% trocas intergênicas), com d indo de 10 até 100 em intervalos de 10 ($\mathbf{DB_{IRGT}}$).

Consequentemente, os conjuntos de dados $\mathbf{DB_{IR}}$, $\mathbf{DB_{IT}}$ e $\mathbf{DB_{GT}}$ (resp. $\mathbf{DB_{IRIT}}$ e $\mathbf{DB_{IRGT}}$) possuem 200.000 (resp. 100.000) triplas $(\pi, \breve{\pi}, \breve{\iota})$ cada, sendo 10.000 triplas para cada valor de d. As posições tanto dos elementos onde a operação é aplicada quanto da localização dentro das regiões intergênicas foram geradas aleatoriamente (respeitando as restrições de cada operação). Além disso, nos conjuntos em que mais de uma operação pode ser aplicada, elas foram escolhidas também de maneira aleatória.

6.12.1 Algoritmos Existentes na Literatura

Para verificar a utilidade dos algoritmos que consideram regiões intergênicas, começamos nosso experimento testando dois algoritmos de aproximação existentes que não levam em consideração regiões intergênicas: um algoritmo para a Ordenação de Permutações por Transposições (SbT) e outro para a Ordenação de Permutações com Sinais por Reversões e Transposições (SbSRT). Nestes algoritmos utilizamos apenas o elemento π de cada tripla $(\pi, \breve{\pi}, \breve{\iota})$. Lembre que a Ordenação de Permutações com Sinais por Reversões (SbSR) admite algoritmo exato em tempo polinomial. Desta forma, optamos por não utilizá-lo, uma vez que as distâncias retornadas seriam sempre menores ou iguais ao número de operações aplicadas para gerar uma instância.

Para SbT, utilizamos o algoritmo **DD**, que é uma melhoria do algoritmo de Bafna e Pevzer [3], proposto por Dias e Dias [21] e possui fator de aproximação 1.5. Para SbSRT, nós utilizamos o algoritmo **ODD**, proposto por Oliveira e coautores [47], que possui um fator de aproximação igual a 2. Como trocas intergênicas não modificam a permutação, não utilizamos estes algoritmos nos conjuntos de dados que aplicam esta operação. Assim, **DD** foi testado com $\mathbf{DB_{IT}}$ e **ODD** foi testado com $\mathbf{DB_{IRIT}}$. A Tabela 6.2 mostra os resultados obtidos para estes dois algoritmos.

Podemos notar que estes algoritmos começam como bons estimadores, mas começam a subestimar a distância real (ou seja, retornar um valor abaixo do número d de operações aplicadas para gerar as instâncias) rapidamente. Por exemplo, o algoritmo **DD** começa a subestimar a distância em média quando $d \geq 20$, e nunca retorna distâncias maiores que 54. Já o algoritmo **ODD**, começa a subestimar a distância em média quando $d \geq 30$, e nunca retorna uma sequência de ordenação com mais de 60 operações para $\mathbf{DB_{IRIT}}$.

6.12.2 Resultados com o Conjunto de Dados $\mathbf{DB_{IR}}$

A Tabela 6.3 mostra os resultados obtidos pelo Algoritmo 7, um algoritmo com fator teórico de aproximação 2 para reversões intergênicas, utilizando o conjunto de dados

Tabela 6.2: Sumário para os algoritmos **DD** e **ODD** da distância média (**AVG**), mínima (**MIN**) e máxima (**MAX**) obtidas para cada valor de d nos conjuntos de dados $\mathbf{DB_{IT}}$ e $\mathbf{DB_{IRIT}}$, respectivamente.

DD com $\mathbf{DB_{IT}}$				**DD** com $\mathbf{DB_{IT}}$ (cont.)				**ODD** com $\mathbf{DB_{IRIT}}$			
d	**AVG**	**MIN**	**MAX**	**d**	**AVG**	**MIN**	**MAX**	**d**	**AVG**	**MIN**	**MAX**
5	4.99	3	7	55	40.66	33	47	10	10.49	7	14
10	10.01	8	13	60	42.04	35	49	20	20.28	16	26
15	15.04	12	19	65	43.27	35	49	30	28.37	23	34
20	19.98	16	24	70	44.32	37	50	40	35.04	28	42
25	24.59	19	29	75	45.18	39	52	50	40.25	33	49
30	28.57	22	34	80	45.92	38	53	60	44.37	36	52
35	31.91	25	37	85	46.54	39	53	70	47.36	40	56
40	34.71	28	42	90	47.07	41	53	80	49.69	42	58
45	37.06	30	44	95	47.54	40	54	90	51.47	42	60
50	39.01	32	46	100	47.94	41	54	100	52.77	45	60

$\mathbf{DB_{IR}}$.

Podemos ver que o algoritmo consegue realizar boas inferências, com distâncias médias muito próximas do número de operações utilizadas para gerar as instâncias em quase todos os valores de d. Apesar do fator teórico do algoritmo ser 2, o fator de aproximação experimental médio foi muito melhor na prática, com resultados muito próximos de 1 para quase todos os valores de d. Além disso, na média, o algoritmo começa a subestimar a distância apenas quando o número de reversões intergênicas aplicadas é maior ou igual a 75.

6.12.3 Resultados com os Conjuntos de Dados $\mathbf{DB_{IT}}$ e $\mathbf{DB_{GT}}$

Vamos comparar agora os algoritmos 8 e 10, que utilizam transposições intergênicas ou transposições genéricas e possuem fatores teóricos de aproximação iguais a 3.5 e 2.5, respectivamente. Apesar de apenas o Algoritmo 10 lidar com trocas intergênicas, nós executamos os dois algoritmos em ambos os conjuntos $\mathbf{DB_{IT}}$ e $\mathbf{DB_{GT}}$. A Tabela 6.4 mostra os resultados obtidos para os conjuntos de dados $\mathbf{DB_{IT}}$ e $\mathbf{DB_{GT}}$.

Podemos notar que a distância média e o fator de aproximação experimental médio dos resultados obtidos em ambos os conjuntos são menores no algoritmo com o menor fator de aproximação teórico, como esperado. No conjunto $\mathbf{DB_{IT}}$ o Algoritmo 10 foi o melhor estimador, mesmo aplicando trocas intergênicas, que não foram aplicadas na criação desde conjunto. Além disso, podemos perceber um leve aumento na distância média retornada pelo Algoritmo 8 entre os dois conjuntos, o que faz sentido dado que o segundo conjunto foi criado utilizando operações que este algoritmo não consegue reproduzir.

Nos dois conjuntos, o Algoritmo 8 começa a subestimar a distância quando $d > 65$ na

Tabela 6.3: Sumário para o Algoritmo 7 do número médio ($\mathbf{D_{avg}}$), máximo ($\mathbf{D_{max}}$), e mínimo ($\mathbf{D_{min}}$) de reversões intergênicas aplicadas, o fator de aproximação experimental médio do algoritmo ($\mathbf{AE_{avg}}$) e a porcentagem de permutações em que o algoritmo aplica exatamente o número de operações indicado pelo limitante inferior (**%SLB**) retornado no conjunto de dados para cada valor aplicado (**d**) para gerar as instâncias de $\mathbf{DB_{IR}}$.

d	$\mathbf{D_{avg}}$	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{AE_{avg}}$	%SLB	d	$\mathbf{D_{avg}}$	$\mathbf{D_{max}}$	$\mathbf{D_{min}}$	$\mathbf{AE_{avg}}$	%SLB
5	5.16	7	4	1.0318	84.70	55	55.59	66	49	1.0215	46.06
10	10.28	13	8	1.0281	74.86	60	60.59	75	53	1.0259	39.59
15	15.35	20	13	1.0237	69.89	65	65.46	84	57	1.0310	32.82
20	20.38	25	18	1.0203	66.72	70	70.09	86	59	1.0363	27.46
25	25.41	31	23	1.0178	65.16	75	74.32	94	63	1.0411	22.13
30	30.43	37	27	1.0166	62.41	80	77.97	94	65	1.0442	18.62
35	35.44	43	32	1.0155	60.91	85	81.27	103	66	1.0473	15.88
40	40.47	52	37	1.0159	57.75	90	84.02	105	70	1.0479	14.09
45	45.52	55	40	1.0169	54.92	95	86.43	107	72	1.0481	13.48
50	50.54	61	45	1.0184	49.91	100	88.45	110	74	1.0482	13.70

média, e o Algoritmo 10 quando $d > 60$ em média. Isto significa que considerar as regiões intergênicas faz com que os algoritmos comecem a subestimar a distância muito depois do algoritmo (**DD**) que as ignora ($d \geq 20$ contra $d \geq 70$).

No conjunto $\mathbf{DB_{IT}}$, o fator de aproximação experimental máximo retornado pelos algoritmos 8 e 10 foi de 2.42 e 1.96, respectivamente. No conjunto $\mathbf{DB_{GT}}$, o fator de aproximação experimental máximo retornado pelos algoritmos 8 e 10 foi de 2.6 e 2, respectivamente.

Considerando todos os resultados obtidos nos dois conjuntos, os algoritmos 8 e 10 possuem fator de aproximação experimental médio abaixo de 1.5 e 1.4, respectivamente.

6.12.4 Resultados com os Conjuntos de Dados $\mathbf{DB_{IRIT}}$ e $\mathbf{DB_{IRGT}}$

Por fim, vamos comparar os algoritmos 9 e 11, que utilizam reversões intergênicas juntamente com transposições intergênicas ou transposições genéricas e possuem fatores teóricos de aproximação iguais a 3 e 2.5, respectivamente. Aqui também executamos os dois algoritmos em ambos os conjuntos $\mathbf{DB_{IRIT}}$ e $\mathbf{DB_{IRGT}}$. A Tabela 6.5 mostra os resultados obtidos para os conjuntos de dados $\mathbf{DB_{IRIT}}$ e $\mathbf{DB_{IRGT}}$. Os resultados possuem um comportamento parecido com os algoritmos que utilizam transposições intergênicas ou genéricas: as distâncias retornadas pelo Algoritmo 11 são levemente menores que as distâncias retornadas pelo Algoritmo 9.

Nos dois conjuntos, o Algoritmo 9 começa a subestimar a distância quando $d > 80$ na média, e o Algoritmo 11 quando $d > 70$ em média. Note que o algoritmo que desconsidera regiões intergênicas testado (**ODD**) começa a subestimar mais cedo que os algoritmos propostos ($d \geq 30$ contra $d > 80$).

Tabela 6.4: Sumário para os algoritmos 8 e 10 (**A8** e **A10**, respectivamente) do número médio ($\mathbf{DIST_{avg}}$), mínimo ($\mathbf{DIST_{min}}$) e máximo ($\mathbf{DIST_{max}}$) de transposições intergênicas/genéricas aplicadas, e o fator de aproximação experimental médio ($\mathbf{AE_{avg}}$) dos algoritmos (utilizando os limitantes inferiores dos teoremas 16 e 19) para cada valor de operações aplicadas (**d**) para gerar as instâncias de $\mathbf{DB_{IT}}$ (tabela à esquerda) e $\mathbf{DB_{GT}}$ (tabela à direita).

$\mathbf{DB_{IT}}$	$\mathbf{DIST_{avg}}$		$\mathbf{DIST_{min}}$		$\mathbf{DIST_{max}}$		$\mathbf{AE_{avg}}$	
d	**A8**	**A10**	**A8**	**A10**	**A8**	**A10**	**A8**	**A10**
5	5.21	5.14	4	4	11	9	1.04	1.03
10	11.29	10.95	9	9	20	17	1.13	1.10
15	18.17	17.40	14	14	30	26	1.21	1.16
20	25.45	24.17	19	19	38	34	1.28	1.22
25	32.81	31.07	23	23	58	45	1.34	1.27
30	39.80	37.64	27	27	61	52	1.40	1.32
35	45.72	43.27	31	31	69	61	1.44	1.36
40	50.71	47.95	33	33	74	68	1.46	1.38
45	54.95	51.96	38	37	81	69	1.49	1.41
50	58.78	55.51	41	41	82	76	1.51	1.42
55	61.79	58.35	44	43	88	76	1.52	1.44
60	64.52	60.94	43	43	99	83	1.53	1.45
65	66.83	63.10	46	45	102	83	1.54	1.46
70	68.84	65.01	50	47	99	89	1.55	1.47
75	70.53	66.64	52	50	101	87	1.56	1.48
80	72.05	68.08	50	49	101	89	1.57	1.48
85	73.38	69.29	53	51	116	91	1.58	1.49
90	74.49	70.36	54	52	102	90	1.58	1.49
95	75.78	71.55	53	52	115	91	1.59	1.50
100	76.69	72.41	56	55	107	94	1.60	1.51

$\mathbf{DB_{GT}}$	$\mathbf{DIST_{avg}}$		$\mathbf{DIST_{min}}$		$\mathbf{DIST_{max}}$		$\mathbf{AE_{avg}}$	
d	**A8**	**A10**	**A8**	**A10**	**A8**	**A10**	**A8**	**A10**
5	6.14	5.11	4	4	13	10	1.23	1.02
10	13.00	11.25	9	9	23	18	1.31	1.13
15	20.53	18.20	14	13	36	27	1.39	1.23
20	28.23	25.41	19	18	48	37	1.46	1.31
25	35.37	32.29	24	23	58	43	1.50	1.37
30	41.74	38.37	28	28	65	52	1.52	1.40
35	47.17	43.59	31	31	72	58	1.54	1.42
40	51.79	48.02	35	35	79	66	1.55	1.44
45	55.68	51.81	39	38	86	71	1.56	1.45
50	59.10	55.05	41	41	84	72	1.57	1.46
55	61.93	57.83	42	42	91	78	1.57	1.47
60	64.57	60.33	44	42	99	80	1.58	1.47
65	66.83	62.58	48	47	99	83	1.58	1.48
70	68.77	64.41	48	47	95	83	1.59	1.49
75	70.48	66.08	50	48	100	89	1.59	1.49
80	71.96	67.48	52	51	104	87	1.59	1.50
85	73.16	68.72	53	51	108	88	1.60	1.50
90	74.31	69.80	52	51	107	91	1.60	1.50
95	75.21	70.70	50	50	105	92	1.60	1.51
100	76.46	71.91	56	54	105	93	1.61	1.52

Novamente, é possivel perceber um leve aumento na distância média retornada pelo Algoritmo 9 entre os dois conjuntos, o que faz sentido dado que o segundo conjunto foi criado utilizando operações que este algoritmo não consegue reproduzir.

No conjunto $\mathbf{DB_{IRIT}}$, o fator de aproximação experimental máximo retornado pelos algoritmos 9 e 11 foi de 2.4 e 2.13, respectivamente. No conjunto $\mathbf{DB_{GT}}$, o fator de aproximação experimental máximo retornado pelos algoritmos 9 e 11 foi de 2.53 e 2.14, respectivamente.

Considerando todos os resultados obtidos nos dois conjuntos, os algoritmos 9 e 11 possuem fator de aproximação experimental médio abaixo de 1.9 e 1.8, respectivamente.

6.13 Conclusões

Neste capítulo, investigamos a Ordenação por Operações Intergênicas sem restrições no tamanho destas operações. Conseguimos mostrar que as três versões estudadas (reversões, transposições e reversões e transposições) são NP-difíceis.

Tabela 6.5: Sumário para os algoritmos 9 e 11 (**A9** e **A11**, respectivamente) do número médio ($\mathbf{DIST_{avg}}$), mínimo ($\mathbf{DIST_{min}}$) e máximo ($\mathbf{DIST_{max}}$) de reversões intergênicas e transposições intergênicas/genéricas aplicadas, e o fator de aproximação experimental médio ($\mathbf{AE_{avg}}$) dos algoritmos (utilizando os limitantes inferiores dos teoremas 17 e 20) para cada valor de operações aplicadas (**d**) para gerar as instâncias de $\mathbf{DB_{IRIT}}$ (tabela à esquerda) e $\mathbf{DB_{IRGT}}$ (tabela à direita).

$\mathbf{DB_{IRIT}}$	$\mathbf{DIST_{avg}}$		$\mathbf{DIST_{min}}$		$\mathbf{DIST_{max}}$		$\mathbf{AE_{avg}}$	
d	**A9**	**A11**	**A9**	**A11**	**A9**	**A11**	**A9**	**A11**
10	12.60	12.31	8	8	18	16	1.68	1.64
20	26.37	25.76	20	20	34	31	1.77	1.72
30	40.23	39.20	30	30	52	46	1.81	1.77
40	53.29	51.78	41	41	68	60	1.85	1.80
50	63.78	61.84	51	51	81	73	1.87	1.82
60	71.88	69.63	54	54	90	85	1.89	1.83
70	77.83	75.34	63	62	99	91	1.90	1.84
80	82.45	79.71	66	66	99	92	1.91	1.84
90	85.98	83.07	70	69	107	98	1.92	1.85
100	88.78	85.68	71	70	108	100	1.93	1.86

$\mathbf{DB_{IRGT}}$	$\mathbf{DIST_{avg}}$		$\mathbf{DIST_{min}}$		$\mathbf{DIST_{max}}$		$\mathbf{AE_{avg}}$	
d	**A9**	**A11**	**A9**	**A11**	**A9**	**A11**	**A9**	**A11**
10	13.03	11.79	8	8	19	16	1.74	1.57
20	27.01	24.97	20	19	35	31	1.82	1.68
30	40.92	38.21	30	28	52	46	1.87	1.74
40	53.56	50.47	42	41	70	61	1.90	1.79
50	63.78	60.46	49	48	80	72	1.91	1.81
60	71.62	68.17	58	55	93	81	1.92	1.83
70	77.64	74.06	62	60	98	89	1.93	1.84
80	82.30	78.68	67	65	102	92	1.93	1.85
90	85.82	82.09	71	68	107	94	1.94	1.85
100	88.63	84.85	74	72	106	98	1.94	1.86

Utilizando uma adaptação do grafo de breakpoints para levar em consideração as regiões intergênicas e investigando como as operações intergênicas modificam este grafo, fomos capazes de propor um algoritmo de aproximação para cada versão, sendo o fator de aproximação 2 para reversões intergênicas, 3.5 para transposições intergênicas e 3 para reversões intergênicas e transposições intergênicas.

Além disso, foi proposta uma generalização das transposições, em que um de seus segmentos não precisa necessariamente conter genes. Esta generalização permite a construção de um algoritmo de 2.5-aproximação tanto para a versão que considera transposições quanto a versão que considera reversões e transposições.

Os cinco algoritmos apresentados foram implementados e testados, o que nos permitiu constatar que o fator de aproximação experimental médio dos algoritmos tendem a ser melhores que seus fatores de aproximação teóricos. O algoritmo que considera apenas reversões retornou os resultados mais próximos tanto do real quanto do limitante inferior.

Capítulo 7

Considerações Finais

Nesta tese, apresentamos resultados obtidos durante o doutorado. Estes resultados contemplam tanto a prova de NP-dificuldade de diversos modelos, previamente existentes ou propostos neste trabalho, bem como a criação de diversos algoritmos de aproximação ou exatos para problemas que utilizam reversões e transposições.

Estes resultados geraram, durante o período, os seguintes artigos:

- "*Sorting signed circular permutations by super short operations*", publicado na revista Algorithms for Molecular Biology, e em coautoria com Guillaume Fertin, Ulisses Dias e Zanoni Dias [48].

- "*On the Complexity of Sorting by Reversals and Transpositions Problems*", publicado na revista Journal of Computational Biology, e em coautoria com Klairton Lima Brito, Ulisses Dias e Zanoni Dias [44].

- "*Super Short Operations on Both Gene Order and Intergenic Sizes*", publicado na revista Algorithms for Molecular Biology, e em coautoria com Géraldine Jean, Guillaume Fertin, Ulisses Dias e Zanoni Dias [51]. Uma versão preliminar deste artigo foi apresentada no Brazilian Symposium on Bioinformatics (BSB) realizado em Niterói, Brasil, no mês de Outubro de 2018 [50].

- "*A 3.5-Approximation Algorithm for Sorting by In-tergenic Transpositions*", aceito para publicação na International Conference on Algorithms for Computational Biology (AlCoB) que será realizada em Missoula, Montana, USA, no mês de Abril de 2020 [49].

Além dos resultados listados acima, diretamente relacionados a esta tese, outras contribuições foram apresentadas nos seguintes artigos:

- "*Sorting by Weighted Reversals and Transpositions*", publicado na revista Journal of Computational Biology, e em coautoria com Klairton Lima Brito, Ulisses Dias e Zanoni Dias [46]. Uma versão preliminar deste artigo foi apresentada no Brazilian Symposium on Bioinformatics (BSB) realizado em Niterói, Brasil, no mês de Outubro de 2018 [45].

- "*Heuristics for the Reversal and Transposition Distance Problem*", publicado na revista IEEE/ACM Transactions on Computational Biology and Bioinformatics, e em coautoria com Klairton Lima Brito, Ulisses Dias e Zanoni Dias [13]. Uma versão preliminar deste artigo foi apresentada na International Conference on Algorithms for Computational Biology (AlCoB) realizado em Hong Kong, China, no mês de Junho de 2018 [12].

- "*Sorting by Genome Rearrangements on both Gene Order and Intergenic Sizes.*", publicado na revista Journal of Computational Biology, e em coautoria com Klairton Lima Brito, Géraldine Jean, Guillaume Fertin, Ulisses Dias e Zanoni Dias [11]. Uma versão preliminar deste artigo foi apresentada no International Symposium on Bioinformatics Research and Applications (ISBRA) realizado em Barcelona, Espanha, no mês de Junho de 2019 [10].

Estas contribuições deixam em aberto diversas possibilidades de trabalhos futuros. A primeira delas seria estudar a complexidade da abordagem ponderada de Ordenação por Reversões e Transposições quando o custo de uma transposição é maior que 1,5 vezes o custo de uma reversão.

Além disso, a complexidade dos algoritmos intergênicos considerando operações super curtas continua em aberto. Outra possibilidade seria incorporar inserções e deleções de regiões intergênicas aos modelos, considerando operações intergênicas super curtas ou não, para que seja possível comparar dois genomas que compartilham o mesmo conjunto de genes, mas não o mesmo tamanho total de regiões intergênicas.

Outra métrica bastante investigada em problemas de ordenação por rearranjos é o *diâmetro*, denotado por $D_n(\pi)$, que é o maior valor de distância de ordenação, entre todas as permutações de tamanho n. As abordagens intergênicas apresentadas aqui não possuem diâmetros conhecidos até o momento.

Sobre os algoritmos de aproximação propostos, uma possível investigação futura poderia incluir o estudo de novos algoritmos com fatores de aproximação menores, ou então algoritmos ou heurísticas cujos resultados práticos sejam melhores que os apresentados nesta tese.

Referências Bibliográficas

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

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

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

[4] Eugeni Belda, Andrés Moya, and Francisco J. Silva. Genome Rearrangement Distances and Gene Order Phylogeny in γ-Proteobacteria. *Molecular Biology and Evolution*, 22(6):1456–1467, 2005.

[5] 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, 2002.

[6] 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.

[7] Priscila Biller, Carole Knibbe, Guillaume Beslon, and Eric Tannier. Comparative Genomics on Artificial Life. In *Proceedings of the 12th Conference on Computability in Europe (CiE'2016)*, volume 9709 of *Lecture Notes in Computer Science*, pages 35–44. Springer International Publishing, 2016.

[8] Mathieu Blanchette, Takashi Kunisawa, and David Sankoff. Parametric Genome Rearrangement. *Gene*, 172(1):GC11–GC17, 1996.

[9] Mireille Bousquet-Melou. The Expected Number of Inversions after n Adjacent Transpositions. *Discrete Mathematics and Theoretical Computer Science*, 12(2):65–88, 2010.

[10] Klairton Lima Brito, Géraldine Jean, Guillaume Fertin, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Sorting by reversals, transpositions, and indels on both gene order and intergenic sizes. In *Proceedings of the 15th International Symposium*

on Bioinformatics Research and Applications (ISBRA'2019), volume 11490 of *Lecture Notes in Computer Science*, pages 28–39. Springer International Publishing, 2019.

[11] 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.

[12] Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Heuristics for the sorting signed permutations by reversals and transpositions problem. In *Proceedings of the 5th International Conference on Algorithms for Computational Biology (AlCoB'2018)*, volume 10849 of *Lecture Notes in Computer Science*, pages 65–75, Cham, 2018. Springer International Publishing.

[13] Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Heuristics for the reversal and transposition distance problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(1):2–13, 2020.

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

[15] Laurent Bulteau, Guillaume Fertin, and Eric Tannier. Genome rearrangements with indels in intergenes restrict the scenario space. *BMC Bioinformatics*, 17(S14):225–231, 2016.

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

[17] Timothy M. Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2010)*, pages 161–173. Society for Industrial and Applied Mathematics, 2010.

[18] David A. Christie. Sorting Permutations by Block-Interchanges. *Information Processing Letters*, 60(4):165–169, 1996.

[19] Thiago da Silva Arruda, Ulisses Dias, and Zanoni Dias. A grasp-based heuristic for the sorting by length-weighted inversions problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(2):352–363, 2018.

[20] 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.

[21] Ulisses Dias and Zanoni Dias. Extending Bafna-Pevzner Algorithm. In *Proceedings of the International Symposium on Biocomputing (ISB'2010)*, pages 23:1–23:8, New York, NY, USA, 2010. ACM.

[22] 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.

[23] Niklas Eriksen. $(1+\epsilon)$-Approximation of Sorting by Reversals and Transpositions. *Theoretical Computer Science*, 289(1):517–529, 2002.

[24] 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(16):1–11, 2017.

[25] Guillaume Fertin, Anthony Labarre, Irena Rusu, Éric Tannier, and Stéphane Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. The MIT Press, London, England, 2009.

[26] Gustavo Rodrigues Galvão, Christian Baudet, and Zanoni Dias. Sorting Circular Permutations by Super Short Reversals. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 14(3):620–633, 2017.

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

[28] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

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

[30] 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.

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

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

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

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

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

[36] Donald Ervin Knuth. *The art of Computer Programming: Fundamental Algorithms*. Addison-Wesley, Reading, Massachusetts, 1973.

[37] Donald Ervin Knuth. *The Art of Computer Programming, volume 3: Sorting and Searching*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1998.

[38] Claire Lemaitre, Lamia Zaghloul, Marie-France Sagot, Christian Gautier, Alain Arneodo, Eric Tannier, and Benjamin Audit. Analysis of fine-scale mammalian evolutionary breakpoints provides new insight into their relation to genome organisation. *BMC Genomics*, 10(1):1–12, 2009.

[39] Yu Lin, Vaibhav Rajan, and Bernard M. E. Moret. TIBA: a tool for phylogeny inference from rearrangement data with bootstrap analysis. *Bioinformatics*, 28(24):3324–3325, 2012.

[40] Carla N. Lintzmayer, Guillaume Fertin, and Zanoni Dias. Approximation Algorithms for Sorting by Length-Weighted Prefix and Suffix Operations. *Theoretical Computer Science*, 593:26–41, 2015.

[41] Carla Negri Lintzmayer, Guillaume Fertin, and Zanoni Dias. Sorting permutations and binary strings by length-weighted rearrangements. *Theoretical Computer Science*, 715:35–59, 2018.

[42] Aoife McLysaght, Cathal Seoighe, and Kenneth H. Wolfe. High frequency of inversions during eukaryote gene order evolution. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families*, pages 47–58, Dordrecht, 2000. Springer Netherlands.

[43] Guilherme Henrique Santos Miranda, Carla Negri Lintzmayer, and Zanoni Dias. Sorting permutations by λ-operations. *Journal of Universal Computer Science*, 25(2):98–121, 2019.

[44] 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(11):1223–1229, 2019.

[45] Andre Rodrigues Oliveira, Klairton Lima Brito, Zanoni Dias, and Ulisses Dias. Sorting by weighted reversals and transpositions. In *Proceedings of the 11th Brazilian Symposium on Bioinformatics (BSB'2018)*, volume 11228 of *Lecture Notes in Computer Science*, pages 38–49. Springer International Publishing, 2018.

[46] Andre Rodrigues Oliveira, Klairton Lima Brito, Zanoni Dias, and Ulisses Dias. Sorting by Weighted Reversals and Transpositions. *Journal of Computational Biology*, 26(5):420–431, 2019.

[47] Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. On the Sorting by Reversals and Transpositions Problem. *Journal of Universal Computer Science*, 23(9):868–906, 2017.

[48] Andre Rodrigues Oliveira, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Sorting signed circular permutations by super short operations. *Algorithms for Molecular Biology*, 13(13):1–16, 2018.

[49] 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)*, volume 12099 of *Lecture Notes in Computer Science*, pages 1–13. Springer International Publishing, 2020.

[50] Andre Rodrigues Oliveira, Géraldine 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)*, volume 11228 of *Lecture Notes in Computer Science*, pages 14–25. Springer International Publishing, 2018.

[51] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Super Short Operations on Both Gene Order and Intergenic Sizes. *Algorithms for Molecular Biology*, 14(21):1–17, 2019.

[52] Doron Rotem and Jorge Urrutia. Circular Permutation Graphs. *Networks*, 12(4):429–437, 1982.

[53] Cathal Seoighe, Nancy Federspiel, Ted Jones, Nancy Hansen, Vesna Bivolarovic, Ray Surzycki, Raquel Tamse, Caridad Komp, Lucas Huizar, Ronald W. Davis, Stewart Scherer, Evelyn Tait, Duncan J. Shaw, David Harris, Lee Murphy, Karen Oliver, Kate Taylor, Marie-Adele Rajandream, Bart G. Barrell, and Kenneth H. Wolfe. Prevalence of small inversions in yeast gene order evolution. *Proceedings of the National Academy of Sciences*, 97(26):14433–14437, 2000.

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

[55] 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.

[56] Li-San Wang, Tandy Warnow, Bernard M. E. Moret, Robert K. Jansen, and Linda A. Raubeson. Distance-Based Genome Rearrangement Phylogeny. *Journal of Molecular Evolution*, 63(4):473–483, 2006.

[57] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.