Universidade Estadual de Campinas
Instituto de Computação

Klairton de Lima Brito

Modelos com Proporção entre Operações e Regiões Intergênicas Rígidas e Flexíveis

CAMPINAS
2022

Klairton de Lima Brito

Modelos com Proporção entre Operações e Regiões Intergênicas Rígidas e Flexíveis

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 Klairton de Lima Brito e orientada pelo Prof. Dr. Zanoni Dias.

CAMPINAS
2022

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

B777m

Brito, Klairton de Lima, 1991-
Modelos com proporção entre operações e regiões intergênicas rígidas e flexíveis / Klairton de Lima Brito. – Campinas, SP : [s.n.], 2022.

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

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

Informações Complementares

Título em outro idioma: Models with a proportion ratio between operations and rigid and flexible intergenic regions
Palavras-chave em inglês:
Computational biology
Genome rearrangements
Approximation algorithms
Theory of computing
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
Rafael Crivellari Saliba Schouery
Data de defesa: 16-12-2022
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-5287-2925
- Currículo Lattes do autor: http://lattes.cnpq.br/2415128546847155

Universidade Estadual de Campinas
Instituto de Computação

Klairton de Lima Brito

Modelos com Proporção entre Operações e Regiões Intergênicas Rígidas e Flexíveis

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. Rafael Crivellari Saliba Schouery
  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, 16 de dezembro de 2022

Agradecimentos

A vida é um constante processo de aprendizado e esta tese representa o fim de uma etapa muito importante na minha vida. Inúmeras pessoas, direta ou indiretamente, fizeram parte dessa conquista. É impossível listar todas elas em uma única página, mas gostaria de deixar registrado algumas aqui.

Agradeço aos membros da minha família, em especial aos meus pais, José Airton e Raimunda, por todo amor, incentivo e apoio que me deram. Sei que ambos não mediram esforços para que eu pudesse me dedicar inteiramente aos estudos.

À Camila, minha companheira que, definitivamente, sem ela não teria sido possível chegar até aqui. Não tenho como agradecer todo o carinho e companheirismo. Somos bem diferentes, mas você me entende, me aceita, me complementa e me faz muito feliz. Obrigado por estar todo esse tempo ao meu lado, me ajudando e apoiando.

Ao meu orientador Zanoni Dias, pela dedicação, tempo e apoio que me fizeram crescer tanto profissionalmente como pessoalmente. Não tenho palavras para agradecer a confiança depositada em mim desde que aceitou me orientar no mestrado. Ao meu coorientador Ulisses Dias, que assumiu esse papel durante o mestrado, mesmo que não oficialmente. Obrigado aos dois pelas dicas e amizade.

Como se não fosse suficiente o orientador e coorientador, ainda tive a felicidade de fazer parte de um grupo de pesquisa que dispensa comentários. As pessoas desse grupo são: André, Alexsandro e Gabriel. Obrigado pelas ajudas e conversas que tivemos e, mais que isso, obrigado pela amizade.

Durante o doutorado tive a grande oportunidade de viver durante um ano, em Nantes, na França, colaborando com os professores Guillaume Fertin e Géraldine Jean. Deixo aqui o meu muito obrigado pela dedicação.

Aos membros do Laboratório de Otimização e Combinatória (LOCo), pelas conversas, dicas e pelo ambiente acolhedor. Gostaria de agradecer também a todos do Instituto de Computação da Unicamp.

O meu muito obrigado àqueles que direta ou indiretamente fizeram parte dessa conquista.

Por fim, mas não menos importante, gostaria de deixar meu agradecimento ao ser peludo apelidado de Paco(lino), por todas as lambidas que fizeram meus dias bem mais alegres.

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 140272/2020-8, e do acordo CAPES/COFECUB - Projeto 831/15 - Processo 88887.185411/2018-00.

Resumo

A genômica comparativa é um campo de pesquisa da biologia com foco na comparação de características genéticas entre organismos. Dentre as características genéticas comumente utilizadas, podemos citar a sequência de genes dos genomas. O número de mutações genéticas capazes de transformar uma sequência genética em outra é uma das métricas amplamente utilizada para a comparação de dois genomas. Os rearranjos de genoma são eventos mutacionais que podem afetar grandes trechos de um genoma. A reversão e a transposição são dois dos eventos de rearranjo mais estudados na literatura. Uma reversão inverte um segmento do genoma, enquanto uma transposição troca a posição de dois segmentos adjacentes.

Um modelo de rearranjo determina o conjunto de eventos de rearranjo que podem ser utilizados para transformar um genoma em outro. Os primeiros estudos apresentaram resultados considerando modelos de rearranjo constituídos exclusivamente por um único tipo de evento de rearranjo. Estudos posteriores apresentaram resultados considerando modelos de rearranjo compostos por múltiplos tipos de eventos de rearranjo. Quando consideramos apenas o número de eventos de rearranjo necessários para transformar um genoma em outro, assumimos que cada evento tem a mesma probabilidade de ocorrer em um cenário evolutivo, sendo essa abordagem chamada de não ponderada. No entanto, quando assumimos que determinados tipos de eventos ocorrem mais do que outros, é possível atribuir um custo a cada tipo de evento de rearranjo. Nessa nova versão, o objetivo do problema consiste em buscar uma sequência de eventos de rearranjo que transforme um genoma em outro com custo mínimo, sendo essa abordagem chamada de ponderada. Nesta tese, apresentamos uma abordagem que considera uma proporção mínima entre a quantidade de eventos de reversão e o tamanho da sequência de eventos de rearranjo que transforma um genoma em outro. Nós mostramos que a abordagem de proporção naturalmente contorna problemas que podem surgir adotando uma abordagem ponderada ou não ponderada. Além disso, realizamos uma análise de complexidade do problema e apresentamos algoritmos de aproximação com fatores constantes.

Estudos têm destacado a importância da informação presente nas regiões intergênicas, que são estruturas presentes entre cada par de genes e nas extremidades de um genoma, e que podem levar a modelos mais realistas considerando um contexto evolutivo. O tamanho de cada região intergênica é dado pelo número de nucleotídeos presentes nela. Desde então, foram apresentados estudos considerando tanto a sequência de genes quanto o tamanho das regiões intergênicas para representar um genoma. Nesta tese, mostramos resultados nesse mesmo contexto, mas consideramos diferentes modelos de rearranjo. Além disso, introduzimos uma generalização dos problemas possibilitando atribuir um grau de flexibilidade em relação ao tamanho das regiões intergênicas desejadas no genoma alvo. Para ambos os casos, realizamos uma análise de complexidade dos problemas, desenvolvemos algoritmos e conduzimos experimentos para verificar o desempenho prático.

Abstract

Comparative genomics is a field of research in biology focusing on comparing genetic features between organisms. Among the genetic features commonly used, we can mention the sequence of genes in genomes. The number of genetic mutations capable of transforming one gene sequence into another is a widely used metric to compare genomes. Genome rearrangement events are mutational events that can affect large stretches of a genome. Reversal and transposition are two of the most studied rearrangement events in the literature. A reversal inverts a genome segment, while a transposition exchanges the position of two adjacent segments.

A rearrangement model determines the set of rearrangement events that can be used to transform one genome into another. Early studies presented results considering a rearrangement model consisting exclusively of a single type of rearrangement event. However, subsequent studies presented results considering rearrangement models composed of multiple rearrangement events. When we consider only the number of rearrangement events that are required to transform one genome into another, we assume that each event has the same probability of occurring in an evolutionary scenario; this is called the unweighted approach. However, when we want certain types of events to occur more than others, it is possible to assign a cost to each type of rearrangement event. The problem goal changes to search for a sequence of rearrangement events that transforms one genome into another with minimal cost; this is called the weighted approach. In this work, we introduce an approach that considers a minimum proportion between the number of reversal events and the size of the rearrangement sequence that transforms one genome into another. We show that the proportion approach naturally overcomes problems that may arise by adopting a weighted or unweighted approach. In addition, we perform a complexity analysis of the problem and present approximation algorithms with constant factors.

Another genetic feature that studies pointed out as relevant in a genetic comparison context is the intergenic regions, which are structures between each pair of genes and at the extremities of a genome. The size of each intergenic region is the number of nucleotides within it. Since then, studies were presented considering both the sequence of genes and the size of the intergenic regions to represent a genome. In this work, we show results in this same context but we consider different rearrangement models. In addition, we introduce a generalization of the problems such that it is possible to assign a degree of flexibility regarding the size of the intergenic regions desired in the target genome. For both cases, we conducted a complexity analysis of the problems, developed algorithms, and performed experiments to verify the practical performance.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

O estudo da evolução dos organismos é uma tarefa de fundamental importância no campo da biologia. No decorrer da evolução, organismos podem sofrer mutações. Na natureza, os organismos mais bem adaptados têm mais oportunidades de procriar e consequentemente passar suas características genéticas para seus descendentes. As mudanças genéticas são uma das características utilizadas no campo da *genômica comparativa* para estimar a proximidade de dois organismos com base na similaridade de seus materiais genéticos. O genoma pode sofrer modificações a partir de mutações pontuais ou que afetam grandes trechos do genoma, chamadas de *eventos de rearranjos de genomas*. Tais eventos afetam o genoma modificando, inserindo ou removendo material genético [44]. Uma forma bem aceita de estimar a proximidade de dois organismos é determinando uma sequência de eventos de rearranjos de genomas com tamanho mínimo capaz de transformar o genoma de um organismo em outro. O tamanho de tal sequência é chamada de *distância de rearranjo*.

Reversão e transposição são os eventos de rearranjo mais estudados na literatura [27, 29, 48, 55]. Uma *reversão* atua em um segmento do genoma invertendo a posição e a orientação dos genes contidos no segmento, enquanto uma *transposição* troca dois segmentos consecutivos do genoma, mas sem afetar a posição e a orientação dos genes nos segmentos. Os eventos de reversão e transposição são chamados de *conservativos*, pois não alteram a quantidade de material genético do genoma. Existem também eventos não conservativos, como é o caso dos eventos de inserção, deleção e duplicação [7, 39, 49, 54, 73], que inserem, removem e duplicam material genético de uma região específica do genoma, respectivamente. Um *modelo de rearranjo* é caracterizado pelo conjunto de eventos de rearranjo permitidos para transformar um genoma em outro.

Um genoma pode ser representado computacionalmente de diferentes maneiras. Quando o genoma é tratado como uma sequência ordenada de genes, podemos encontrar casos em que determinados genes apresentam múltiplas cópias, sendo comum utilizarmos uma representação na forma de uma *cadeia de caracteres* (*string*), onde cada caractere é associado a um gene. Por outro lado, se existir apenas uma cópia de cada gene, podemos associar um número inteiro para cada gene e a representação é dada na forma de uma *permutação*. Em ambos os casos, quando a orientação dos genes é conhecida, um sinal positivo (+) ou negativo (−) é atribuído para cada elemento e a representação é chamada *com sinais* (string com sinais e permutação com sinais). Caso contrário, o sinal é omi-

tido e a representação é chamada *sem sinais* (string sem sinais e permutação sem sinais). Chamamos a representação de um genoma através de uma permutação de *representação clássica* e a representação de um genoma por meio de uma string de *representação por strings*.

Ao utilizar a representação clássica, podemos simplificar o problema como sendo um *problema de ordenação*. Nesse caso, o objetivo consiste em transformar uma permutação qualquer em uma permutação específica na qual os elementos encontram-se ordenados de maneira crescente e com sinal positivo para o caso com sinais, sendo essa permutação chamada de *identidade*.

Quando consideramos um modelo de rearranjo composto apenas pelo evento de reversão e utilizamos uma representação clássica com sinais, temos a variação com sinais do problema de Ordenação de Permutações por Reversões (**Sorting by Reversals – SbR**). Hannenhalli e Pevzner [48] apresentaram o primeiro algoritmo exato polinomial para o problema, sendo posteriormente simplificado por Bergeron [13]. Atualmente, existe um algoritmo com complexidade subquadrática para determinar a sequência de reversões capaz de ordenar uma permutação com sinais [69]. Entretanto, se estivermos interessados somente na distância de reversão, existe um algoritmo que executa em tempo linear [6]. Quando consideramos uma representação clássica sem sinais, temos a variação sem sinais do problema **SbR**. Caprara [30] provou que o problema faz parte da classe de problemas NP-Difícil. Um dos primeiros algoritmos de aproximação para o problema tem um fator de aproximação 1.75 [11]. Em seguida, Christie [34] apresentou um algoritmo com fator de aproximação 1.5. Atualmente, o melhor algoritmo para o problema tem um fator de aproximação 1.375 [14].

Quando consideramos um modelo de rearranjo composto apenas pelo evento de transposição, a orientação dos genes não é considerada, tendo em vista que o evento de transposição não altera a orientação dos genes. Dessa forma, ao adotar uma representação clássica sem sinais, temos o problema de Ordenação de Permutações por Transposições (**Sorting by Transpositions – SbT**). O problema também pertence à classe de problemas NP-Difícil, sendo a prova apresentada por Bulteau *et al.* [27]. O primeiro algoritmo de aproximação para o problema foi proposto por Bafna e Pevzner [12] com fator de aproximação 1.5. Atualmente, o melhor algoritmo para o problema tem fator de aproximação 1.375 [40,64] e heurísticas foram apresentadas por Dias e Dias [36] visando a obtenção de resultados práticos melhores.

Outro evento de rearranjo que podemos mencionar é o *block-interchange*, que troca a posição de dois segmentos do genoma (não necessariamente consecutivos). Note que com o evento de *block-interchange* é possível simular uma transposição. Considerando um modelo de rearranjo composto exclusivamente pelo evento de *block-interchange* e adotando uma representação clássica sem sinais, temos o problema de Ordenação de Permutações por Block-Interchanges (**Sorting by Block-Interchanges – SbBI**). Em 1996, foi apresentado um algoritmo exato polinomial para o problema [33].

Ao considerar um modelo de rearranjo composto pelos eventos de reversão e transposição adotando uma representação clássica, obtemos o problema de Ordenação de Permutações por Reversões e Transposições (**Sorting by Reversals and Transpositions – SbRT**). O problema possui a variação com e sem sinais e ambas pertencem à classe de

problemas NP-Difícil [55]. Os melhores algoritmos de aproximação para o problema apresentam fatores de aproximação 2 [71] e $2k$ [62] (onde k é o fator de aproximação para a decomposição de ciclos [31]) para as variações com e sem sinais, respectivamente. Diversas heurísticas considerando esses problemas foram apresentadas na literatura [26, 37].

Além dos eventos citados anteriormente, podemos mencionar ainda o evento de rearranjo *double cut and join* (DCJ) [74], que atua cortando o genoma em dois pontos e, em seguida, as extremidades dos segmentos resultantes são unidas obedecendo certas restrições. A Tabela 1.1 sumariza os resultados de complexidade e fator de aproximação, considerando a representação clássica de um genoma e adotando um modelo de rearranjo composto pelos eventos de reversão, transposição, *block-interchange*, DCJ e algumas combinações dos mesmos.

Tabela 1.1: Resultados de complexidade e fator de aproximação dos modelos considerando uma representação clássica.

Representação Clássica com Sinais		
Modelo	**Complexidade**	**Aproximação**
DCJ	P [74]	Exato [74]
Reversão	P [48]	Exato [48]
Reversão e Transposição	NP-difícil [55]	2 [71]

Representação Clássica sem Sinais		
Modelo	**Complexidade**	**Aproximação**
DCJ	NP-difícil [31]	$\frac{17}{12} + \epsilon$ [31]
Reversão	NP-difícil [30]	1.375 [14]
Transposição	NP-difícil [27]	1.375 [40, 64]
Block-Interchange	P [33]	Exato [33]
Reversão e Transposição	NP-difícil [55]	$2k$ [31, 62]

Quando consideramos uma representação por strings, em 2001, Christie e Irving [35] mostraram que a variação sem sinais do problema de Distância de Strings por Reversões pertence à classe de problemas NP-Difícil, mesmo se considerarmos um alfabeto binário (ou seja, os caracteres das strings comparadas pertencem a um conjunto com apenas dois elementos). Para isso, os autores apresentaram uma redução do problema *3-partition* [46]. Em 2005, Radcliffe *et al.* [61] mostraram que a variação com sinais do problema de Distância de Strings por Reversões e o problema de Distância de Strings por Transposições também pertencem à classe de problemas NP-Difícil, mesmo se considerarmos um alfabeto binário. Outra contribuição importante do trabalho foi a caracterização de um conjunto de instâncias em que é possível obter uma solução ótima em tempo polinomial.

Uma relação entre o problema de Distância de Strings por Reversões e o problema de Partição Mínima em Strings foi apresentada por Chen *et al.* [32]. Com essa relação entre os problemas, foi apresentado por Kolman e Waleń [51] um algoritmo de aproximação com fator $\Theta(k)$ para ambas as variações do problema de Distância de Strings por Reversões, onde k representa o número máximo de cópias de um caractere nas strings consideradas. Uma relação similar entre o problema de Distância de Strings por Transposições e o problema de Partição Mínima em Strings foi apresentada por Shapira e Storer [63], que

no mesmo trabalho apresentaram um algoritmo de aproximação com fator de $\mathcal{O}(\log n)$ para o problema, onde n é o número de caracteres das strings comparadas.

A representação do genoma como uma sequência de genes é uma abordagem simples e prática, mas acarreta na perda de informação referente às estruturas genéticas que não fazem parte da sequência de genes. Estudos apontaram que considerar informações adicionais contidas no genoma, além da sequência de genes, pode tornar a comparação entre genomas mais realista [15, 16]. Em particular, os pesquisadores abordaram a importância de considerar o tamanho das regiões presentes entre cada par de genes consecutivos e nas extremidades do genoma, chamadas de *regiões intergênicas*.

Trabalhos que levam em conta a sequência de genes e que também consideram os tamanhos das regiões intergênicas começaram a ser apresentados recentemente. Assumindo que em um genoma exista uma cópia de cada gene e que o tamanho de cada região intergênica seja uma informação conhecida, então sua representação computacional pode ser dada por uma permutação (com ou sem sinais), que representa a ordem e a orientação dos genes, e uma lista de números naturais representando o tamanho de cada região intergênica do genoma. Essa representação é chamada de *intergênica*. Os problemas de rearranjo de genomas considerando uma representação intergênica consistem em ordenar os genes, levando em conta as suas posições relativas e orientações, e transformar o tamanho das regiões intergênicas de acordo com o desejado no genoma alvo.

Fertin *et al.* [43], adotando uma representação intergênica com sinais, apresentaram um modelo composto pelo evento de rearranjo DCJ e mostraram que o problema pertence à classe de problemas NP-Difícil. Além disso, os autores desenvolveram um algoritmo de aproximação com fator $\frac{4}{3}$. Bulteau *et al.* [28] apresentaram um modelo que permite o uso do evento DCJ juntamente com os eventos não conservativos de inserção e deleção restritos a atuarem apenas sobre as regiões intergênicas. Para esse problema, os autores apresentaram um algoritmo exato polinomial.

Considerando um modelo composto exclusivamente pelo evento de reversão e adotando uma representação intergênica, temos o problema de Ordenação de Permutações por Reversões Intergênicas (**Sorting by Intergenic Reversals – $\mathbf{Sb_IR}$**). As variações com e sem sinais do problema pertencem à classe de problemas NP-Difícil [22, 57], sendo que os melhores algoritmos de aproximação conhecidos para o problema possuem fatores de aproximação 2 [57] e 4 [22] para as variações com e sem sinais, respectivamente. Considerando um modelo composto exclusivamente pelo evento de transposição, temos o problema de Ordenação de Permutações por Transposições Intergênicas (**Sorting by Intergenic Transpositions – $\mathbf{Sb_IT}$**). O problema também pertence à classe de problemas NP-Difícil [59], sendo que o melhor algoritmo de aproximação conhecido possui fator de aproximação 3.5 [59]. Considerando um modelo composto exclusivamente pelo evento de *block-interchange*, temos o problema de Ordenação de Permutações por Block-Interchanges Intergênicos (**Sorting by Intergenic Block-Interchanges – $\mathbf{Sb_IBI}$**). A complexidade do problema ainda permanece desconhecida e o melhor algoritmo de aproximação para o problema possui fator de aproximação 2 [38].

Quando utilizamos um modelo composto pelos eventos de reversão e transposição e adotamos uma representação intergênica, temos o problema de Ordenação de Permutações por Operações Intergênicas de Reversão e Transposição (**Sorting by Intergenic**

Operations of Reversal and Transposition – $\mathbf{Sb_IRT}$). As variações com e sem sinais do problema pertencem à classe de problemas NP-Difícil [22, 59], sendo que os melhores algoritmos de aproximação conhecidos para o problema possuem fatores de aproximação 3 [59] e 4 [23] para as variações com e sem sinais, respectivamente.

Em um cenário em que é adotada a representação intergênica, Oliveira *et al.* [59] introduziram o evento de rearranjo chamado de *move*. Esse evento é similar ao evento de transposição, mas um dos segmentos afetados é composto exclusivamente por uma região intergênica. Além disso, os autores apresentaram os problemas de Ordenação de Permutações por Operações Intergênicas de Transposição e Move (**Sorting by Intergenic Operations of Transposition and Move – $\mathbf{Sb_ITM}$**) e Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Move (**Sorting by Intergenic Operations of Reversal, Transposition, and Move – $\mathbf{Sb_IRTM}$**). Para o problema $\mathbf{Sb_ITM}$ os autores apresentaram um algoritmo de aproximação com fator de 2.5 [59]. Para as variações com e sem sinais do problema $\mathbf{Sb_IRTM}$ existem algoritmos de aproximação com fatores 2.5 [59] e 3 [23], respectivamente.

Permitindo o uso dos eventos de reversão e move, temos o problema de Ordenação de Permutações por Operações Intergênicas de Reversão e Move (**Sorting by Intergenic Operations of Reversal and Move – $\mathbf{Sb_IRM}$**). A variação com sinais desse problema pertence à classe de problemas NP-Difícil e o melhor algoritmo de aproximação para o problema possui fator de aproximação 2 [25].

Considerando uma restrição adicional no número máximo de genes afetados pelos eventos de reversão e transposição, Oliveira *et al.* [60] apresentaram modelos compostos pela combinação dos eventos super curtos de reversão e transposição. Um evento de rearranjo *super curto* pode afetar segmentos do genoma com no máximo dois genes. Adotando uma representação intergênica sem sinais, os autores mostraram algoritmos de aproximação com fator 3, enquanto para uma representação intergênica com sinais apresentaram algoritmos de aproximação com fator 5.

A Tabela 1.2 sumariza os resultados de complexidade e fator de aproximação dos modelos, considerando uma representação intergênica de um genoma.

Estudos recentes apresentaram resultados em que a representação adotada de um genoma considera tanto a estrutura das regiões intergênicas como também a possibilidade de mapear genes com múltiplas cópias. Os resultados abordam modelos compostos pela combinação dos eventos de reversão e transposição em cenários em que a orientação dos genes é conhecida [65] ou desconhecida [66]. Para todos os problemas, foram apresentados algoritmos que garantem um fator de aproximação $\Theta(k)$, onde k é o número máximo de cópias de um gene nos genomas comparados.

O restante dessa tese está dividida da seguinte forma. No Capítulo 2 são apresentadas definições, conceitos e estruturas que serão utilizadas nos três capítulos subsequentes desta tese, e que são fundamentais para a obtenção de resultados para os problemas investigados. No Capítulo 3 um novo problema de rearranjo é introduzido, onde uma restrição de proporção entre operações é adicionada ao modelo. No Capítulo 4 são investigados problemas em que a informação referente ao tamanho das regiões intergênicas e a ordem dos genes é levada em consideração. No Capítulo 5 é apresentada uma generalização dos problemas que consideram tanto a ordem dos genes como o tamanho estrito das regiões

Tabela 1.2: Resultados de complexidade e fator de aproximação dos modelos considerando uma representação intergênica.

Representação Intergênica com Sinais		
Modelo	**Complexidade**	**Aproximação**
DCJ	NP-difícil [43]	$\frac{4}{3}$ [43]
DCJ, Inserção e Deleção	P [28]	Exato [28]
Reversão	NP-difícil [57]	2 [57]
Reversão (Super Curta)	Desconhecida	5 [60]
Reversão e Move	NP-difícil [25]	2 [25]
Reversão e Transposição	NP-difícil [59]	3 [57]
Reversão e Transposição (Super Curta)	Desconhecida	5 [60]
Reversão, Transposição e Move	NP-difícil [59]	$\frac{5}{2}$ [59]

Representação Intergênica sem Sinais		
Modelo	**Complexidade**	**Aproximação**
Reversão	NP-difícil [22]	4 [22]
Reversão (Super Curta)	Desconhecida	3 [60]
Transposição	NP-difícil [59]	$\frac{7}{2}$ [59]
Transposição (Super Curta)	Desconhecida	3 [60]
Transposição e Move	NP-difícil [59]	$\frac{5}{2}$ [59]
Block-Interchange	Desconhecida	2 [38]
Reversão e Transposição	NP-difícil [22]	4 [23]
Reversão e Transposição (Super Curta)	Desconhecida	3 [60]
Reversão, Transposição e Move	NP-difícil [23]	3 [23]

intergênicas, permitindo adicionar um grau de flexibilidade para o tamanho desejado das regiões intergênicas no genoma alvo. Por fim, no Capítulo 6, as conclusões finais desta tese são apresentadas.

Capítulo 2

Definições

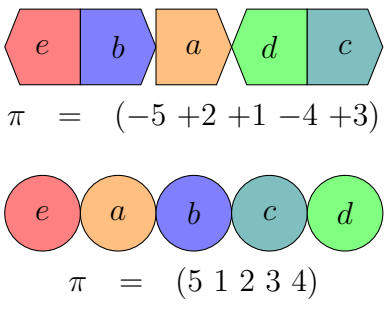
Neste capítulo, apresentaremos as formas como um genoma será representado e como os eventos de rearranjo de genomas podem afetá-lo. Além disso, definiremos o formato das instâncias utilizadas pelos problemas investigados nos capítulos seguintes e apresentaremos definições e conceitos utilizados para obtenção de resultados.

2.1 Representação de Genomas

Nesta seção, apresentamos três representações de genomas que diferem quanto às estruturas genéticas incorporadas na representação computacional.

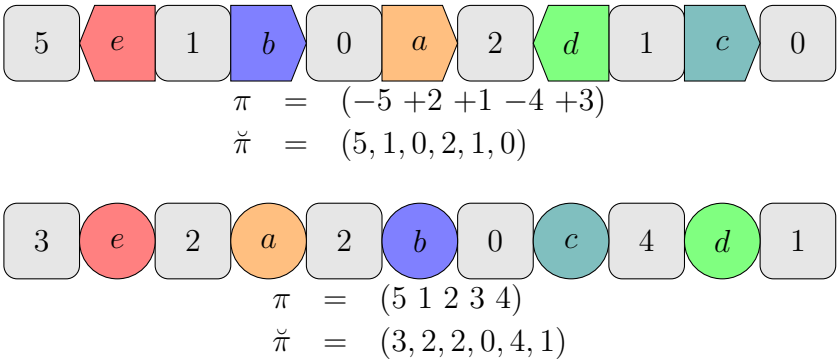
Dado um genoma $\mathcal{G} = (\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n)$ com n genes não repetidos, utilizamos uma representação através de uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, de forma que cada elemento π_i, com $1 \leq i \leq n$, da permutação π representa o gene $\mathcal{G}_i$ do genoma $\mathcal{G}$. Caso a orientação dos genes no genoma $\mathcal{G}$ seja conhecida, associamos um sinal "+" ou "−" a cada elemento $\pi_i \in \pi$ para representar a orientação de cada um dos genes de $\mathcal{G}$. Caso contrário, o sinal é omitido. Quando representamos um genoma utilizando apenas as informações obtidas com base no posicionamento dos genes, denominamos de *representação clássica*. Além disso, denotamos por *representação clássica com sinais* quando a orientação dos genes é conhecida e por *representação clássica sem sinais* caso contrário. O Exemplo 2.1.1 mostra representações clássicas com sinais e sem sinais de genomas fictícios. Os elementos coloridos com letras no interior representam os genes. Na parte superior os elementos possuem orientação e na parte inferior não possuem.

Exemplo 2.1.1.



Dado um genoma $\mathcal{G} = (\mathfrak{R}_1, \mathcal{G}_1, \mathfrak{R}_2, \mathcal{G}_2 \ldots, \mathfrak{R}_n, \mathcal{G}_n, \mathfrak{R}_{n+1})$ com n genes não repetidos $\mathcal{G}_1$, $\mathcal{G}_2$, $\ldots$, $\mathcal{G}_n$ e $n+1$ regiões intergênicas $\mathfrak{R}_1$, $\mathfrak{R}_2$, $\ldots$, $\mathfrak{R}_{n+1}$, utilizamos essas duas características para representar um genoma. As regiões intergênicas estão presentes nas extremidades do genoma e entre cada par de genes consecutivos. Definimos o *tamanho* de uma região intergênica como sendo a quantidade de nucleotídeos contidos nela. Representamos o genoma $\mathcal{G}$ utilizando dois elementos, sendo o primeiro elemento uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, de forma que cada elemento π_i, com $1 \leq i \leq n$, da permutação π representa o gene $\mathcal{G}_i$ do genoma $\mathcal{G}$. Caso a orientação dos genes no genoma $\mathcal{G}$ seja conhecida, associamos um sinal "+" ou "−" a cada elemento $\pi_i \in \pi$ para representar a orientação de cada um dos genes de $\mathcal{G}$. Caso contrário, o sinal é omitido. O segundo elemento da representação é uma lista de número inteiros não negativos $\breve{\pi} = (\breve{\pi}_1, \breve{\pi}_2, \ldots, \breve{\pi}_{n+1})$, de forma que cada elemento $\breve{\pi}_i$, com $1 \leq i \leq n+1$, da lista $\breve{\pi}$ representa o tamanho da região intergênica $\mathfrak{R}_i$ do genoma $\mathcal{G}$. Quando representamos um genoma utilizando a informação do posicionamento dos genes e os tamanhos das regiões intergênicas, denominamos de *representação intergênica rígida*. Além disso, denotamos por *representação intergênica rígida com sinais* quando a orientação dos genes é conhecida e por *representação intergênica rígida sem sinais* caso contrário. O Exemplo 2.1.2 mostra representações intergênicas rígidas com sinais e sem sinais de genomas fictícios. Os elementos coloridos com letras no interior representam os genes, sendo que na parte superior eles possuem orientação e na parte inferior não possuem. Os retângulos com bordas arredondadas, localizados nas extremidades e entre cada par de genes, representam as regiões intergênicas, com o número no interior indicando seu tamanho.

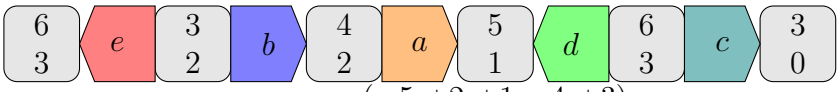
Exemplo 2.1.2.



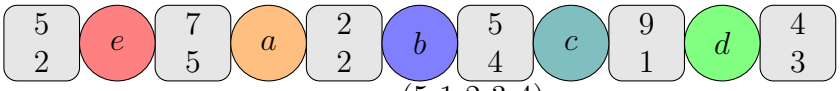
Para tornar a especificação em relação ao tamanho de cada região intergênica menos rígida, criamos uma representação denominada de *representação intergênica flexível*. Para isso, representamos um genoma $\mathcal{G} = (\mathfrak{R}_1, \mathcal{G}_1, \mathfrak{R}_2, \mathcal{G}_2 \ldots, \mathfrak{R}_n, \mathcal{G}_n, \mathfrak{R}_{n+1})$ com n genes não repetidos $\mathcal{G}_1$, $\mathcal{G}_2$, $\ldots$, $\mathcal{G}_n$ e $n+1$ regiões intergênicas $\mathfrak{R}_1$, $\mathfrak{R}_2$, $\ldots$, $\mathfrak{R}_{n+1}$ utilizando três elementos. O primeiro elemento é uma permutação $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, de forma que cada elemento π_i, com $1 \leq i \leq n$, da permutação π representa o gene $\mathcal{G}_i$ do genoma $\mathcal{G}$. Caso a orientação dos genes no genoma $\mathcal{G}$ seja conhecida, associamos um sinal "+" ou "−" a cada elemento $\pi_i \in \pi$ para representar a orientação de cada um dos genes de $\mathcal{G}$. Caso contrário, o sinal é omitido. Os demais elementos são duas listas de número inteiros não negativos $\breve{\pi}^{\min} = (\breve{\pi}_1^{\min}, \breve{\pi}_2^{\min}, \ldots, \breve{\pi}_{n+1}^{\min})$ e $\breve{\pi}^{\max} = (\breve{\pi}_1^{\max}, \breve{\pi}_2^{\max}, \ldots, \breve{\pi}_{n+1}^{\max})$, de forma que $\breve{\pi}_i^{\min} \leq \breve{\pi}_i^{\max}$ e o tamanho da região intergênica $\mathfrak{R}_i$ pertence ao intervalo $[\breve{\pi}_i^{\min}..\breve{\pi}_i^{\max}]$, com $1 \leq i \leq n+1$.

Isso faz com que o tamanho de cada região intergênica seja flexível, tornando possível especificar um intervalo de valores aceitáveis para o tamanho de cada uma delas ao invés de apenas um único valor. Por fim, denotamos por *representação intergênica flexível com sinais* quando a orientação dos genes é conhecida e por *representação intergênica flexível sem sinais* caso contrário. O Exemplo 2.1.3 mostra representações intergênicas flexíveis com sinais e sem sinais de genomas fictícios. Os elementos coloridos com letras no interior representam os genes, sendo que na parte superior eles possuem orientação e na parte inferior não possuem. Os retângulos com bordas arredondadas, localizados nas extremidades e entre cada par de genes, representam as regiões intergênicas. O número na parte superior de cada região intergênica indica o tamanho máximo permitido, enquanto o número na parte inferior indica o tamanho mínimo permitido.

Exemplo 2.1.3.



$$\begin{aligned} \pi &= (-5\ +2\ +1\ -4\ +3) \\ \breve{\pi}^{\min} &= (3, 2, 2, 1, 3, 0) \\ \breve{\pi}^{\max} &= (6, 3, 4, 5, 6, 3) \end{aligned}$$



$$\begin{aligned} \pi &= (5\ 1\ 2\ 3\ 4) \\ \breve{\pi}^{\min} &= (2, 5, 2, 4, 1, 3) \\ \breve{\pi}^{\max} &= (5, 7, 2, 5, 9, 4) \end{aligned}$$

Dada a representação $\mathcal{R}$ de um genoma $\mathcal{G}$, seja na forma clássica $\mathcal{R} = (\pi)$, intergênica rígida $\mathcal{R} = (\pi, \breve{\pi})$ ou intergênica flexível $\mathcal{R} = (\pi, \breve{\pi}^{\min}, \breve{\pi}^{\max})$, obtemos sua versão *estendida* adicionando dois novos elementos em π, $\pi_0 = 0$ e $\pi_{n+1} = n + 1$, no início e no fim da permutação π, respectivamente. Esses dois novos elementos adicionados em π representam genes fictícios que não serão afetados por nenhum evento de rearranjo de genomas, e serão utilizados apenas para tornar algumas definições mais simples. De agora em diante, assumimos que qualquer representação de genoma estará na sua forma estendida, a não ser que seja dito expressamente o contrário.

2.2 Eventos de Rearranjo

Nesta seção, apresentamos os eventos de rearranjo considerados nesta tese e como eles podem afetar o genoma dependendo da representação utilizada.

Os eventos de rearranjo de genomas são classificados em *conservativos* ou *não conservativos*. Os eventos de rearranjo conservativos não alteram a quantidade de material genético do genoma, enquanto os eventos de rearranjo não conservativos, sim. Dado um evento de rearranjo γ e uma representação $\mathcal{R}$ de um genoma, denotamos por $\mathcal{R} \cdot \gamma$ o genoma resultante após a aplicação do evento de rearranjo γ em $\mathcal{R}$. De maneira similar, quando temos uma sequência de eventos de rearranjo $S = (\gamma_1, \gamma_2, \ldots, \gamma_k)$ e uma representação $\mathcal{R}$ de um genoma, denotamos por $\mathcal{R} \cdot S = \mathcal{R} \cdot \gamma_1 \cdot \gamma_2 \cdot \ldots \cdot \gamma_k$ como sendo o genoma

resultante após a aplicação da sequência S em $\mathcal{R}$. A seguir, mostramos como os eventos de rearranjo conservativos de reversão e transposição afetam a representação clássica de um genoma.

Definição 2.2.1 (Reversão)**.** Seja $\mathcal{R} = (\pi)$ uma representação clássica de um genoma e sejam i e j números inteiros tais que $1 \leq i \leq j \leq n$. Uma *reversão* $\rho^{(i,j)}$ inverte o segmento $(\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}\ \pi_j)$ de π. Caso a representação $\mathcal{R}$ do genoma seja clássica com sinais, o sinal de cada elemento no segmento $(\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}\ \pi_j)$ também é invertido.

Os exemplos 2.2.1 e 2.2.2 mostram uma reversão $\rho^{(i,j)}$ genérica sendo aplicada em uma representação clássica com e sem sinais de um genoma, respectivamente.

Exemplo 2.2.1.

$$\begin{aligned} \pi &= (\pi_0\ \pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}\ \pi_j}\ \pi_{j+1}\ \ldots\ \pi_n\ \pi_{n+1}) \\ \pi \cdot \rho^{(i,j)} &= (\pi_0\ \pi_1\ \ldots\ \pi_{i-1}\ \underline{-\pi_j\ -\pi_{j-1}\ \ldots\ -\pi_{i+1}\ -\pi_i}\ \pi_{j+1}\ \ldots\ \pi_n\ \pi_{n+1}) \end{aligned}$$

Exemplo 2.2.2.

$$\begin{aligned} \pi &= (\pi_0\ \pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}\ \pi_j}\ \pi_{j+1}\ \ldots\ \pi_n\ \pi_{n+1}) \\ \pi \cdot \rho^{(i,j)} &= (\pi_0\ \pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \pi_{j-1}\ \ldots\ \pi_{i+1}\ \pi_i}\ \pi_{j+1}\ \ldots\ \pi_n\ \pi_{n+1}) \end{aligned}$$

O Exemplo 2.2.3 mostra uma reversão $\rho^{(2,4)}$ sendo aplicada na representação clássica com sinais $\mathcal{R} = (\pi) = (+0\ -3\ +2\ -4\ +1\ +5\ +6)$ de um genoma, enquanto o Exemplo 2.2.4 mostra uma reversão $\rho^{(1,5)}$ sendo aplicada na representação clássica sem sinais $\mathcal{R} = (\pi) = (0\ 4\ 5\ 3\ 2\ 1\ 6)$ de um genoma.

Exemplo 2.2.3.

$$\begin{aligned} \pi &= (+0\ -3\ \underline{+2\ -4\ +1}\ +5\ +6) \\ \pi \cdot \rho^{(2,4)} &= (+0\ -3\ \underline{-1\ +4\ -2}\ +5\ +6) \end{aligned}$$

Exemplo 2.2.4.

$$\begin{aligned} \pi &= (0\ \underline{4\ 5\ 3\ 2\ 1}\ 6) \\ \pi \cdot \rho^{(1,5)} &= (0\ \underline{1\ 2\ 3\ 5\ 4}\ 6) \end{aligned}$$

Definição 2.2.2 (Transposição)**.** Seja $\mathcal{R} = (\pi)$ uma representação clássica de um genoma e sejam i, j e k números inteiros tais que $1 \leq i < j < k \leq n+1$. Uma *transposição* $\tau^{(i,j,k)}$ troca a posição dos segmentos consecutivos $(\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1})$ e $(\pi_j\ \pi_{j+1}\ \ldots\ \pi_{k-1})$ de π.

O Exemplo 2.2.5 mostra uma transposição $\tau^{(i,j,k)}$ genérica sendo aplicada em uma representação clássica de um genoma. Note que a transposição pode ser aplicada em ambas as representações clássicas, com e sem sinais.

Exemplo 2.2.5.

$$\begin{aligned} \pi &= (\pi_0\ \pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}}\ \underline{\pi_j\ \pi_{j+1}\ \ldots\ \pi_{k-1}}\ \pi_k\ \ldots\ \pi_n\ \pi_{n+1}) \\ \pi \cdot \tau^{(i,j,k)} &= (\pi_0\ \pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \pi_{j+1}\ \ldots\ \pi_{k-1}}\ \underline{\pi_i\ \pi_{i+1}\ \ldots\ \pi_{j-1}}\ \pi_k\ \ldots\ \pi_n\ \pi_{n+1}) \end{aligned}$$

O Exemplo 2.2.6 mostra uma transposição $\tau^{(1,3,5)}$ sendo aplicada na representação clássica com sinais $\mathcal{R} = (\pi) = (+0\ -4\ -3\ +1\ +2\ +5\ +6)$ de um genoma, enquanto o Exemplo 2.2.7 mostra uma transposição $\tau^{(4,5,6)}$ sendo aplicada na representação clássica sem sinais $\mathcal{R} = (\pi) = (0\ 3\ 2\ 1\ 5\ 4\ 6)$ de um genoma.

Exemplo 2.2.6.

$$\begin{aligned} \pi &= (+0\ \underline{-4\ -3}\ \underline{+1\ +2}\ +5\ +6) \\ \pi \cdot \tau^{(1,3,5)} &= (+0\ \underline{+1\ +2}\ \underline{-4\ -3}\ +5\ +6) \end{aligned}$$

Exemplo 2.2.7.

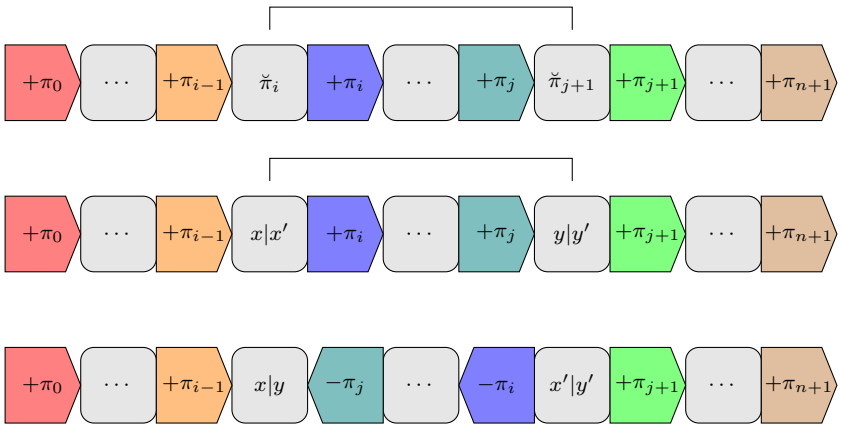
$$\begin{aligned} \pi &= (0\ 3\ 2\ 1\ \underline{5}\ \underline{4}\ 6) \\ \pi \cdot \tau^{(4,5,6)} &= (0\ 3\ 2\ 1\ \underline{4}\ \underline{5}\ 6) \end{aligned}$$

A seguir, mostramos como os eventos de rearranjo conservativos de reversão intergênica, transposição intergênica e move intergênico afetam a representação intergênica rígida de um genoma.

Definição 2.2.3 (Reversão Intergênica)**.** Seja $\mathcal{R} = (\pi, \breve{\pi})$ uma representação intergênica rígida de um genoma e sejam i, j, x e y números inteiros tais que $1 \leq i \leq j \leq n$, $0 \leq x \leq \breve{\pi}_i$ e $0 \leq y \leq \breve{\pi}_{j+1}$. Uma *reversão intergênica* $\rho^{(i,j)}_{(x,y)}$ divide as regiões intergênicas $\breve{\pi}_i$ e $\breve{\pi}_{j+1}$ da seguinte forma: $\breve{\pi}_i$ em duas partes com tamanhos x e x', onde $x' = \breve{\pi}_i - x$; e $\breve{\pi}_{j+1}$ em duas partes com tamanhos y e y', onde $y' = \breve{\pi}_{j+1} - y$. Em seguida, o segmento $(x', \pi_i, \breve{\pi}_{i+1} \ldots \breve{\pi}_j, \pi_j, y)$ do genoma é invertido. Caso a representação seja com sinais, os sinais dos elementos de π_i até π_j também são invertidos. Por fim, os segmentos do genoma são remontados com os pares de partes (x, y) e (x', y') fundindo-se e formando as novas regiões intergênicas $\breve{\pi}_i$ e $\breve{\pi}_{j+1}$ com tamanhos $x + y$ e $x' + y'$, respectivamente.

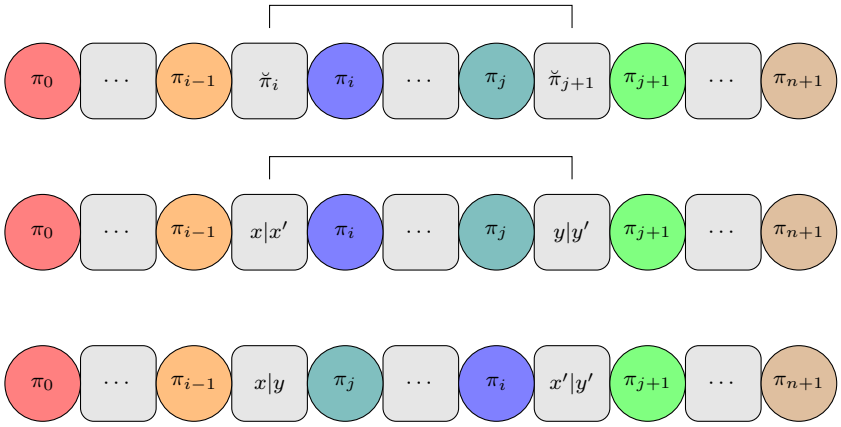
O Exemplo 2.2.8 mostra uma reversão intergênica $\rho^{(i,j)}_{(x,y)}$ genérica sendo aplicada em uma representação intergênica rígida com sinais de um genoma.

Exemplo 2.2.8.



O Exemplo 2.2.9 mostra uma reversão intergênica $\rho^{(i,j)}_{(x,y)}$ genérica sendo aplicada em uma representação intergênica rígida sem sinais de um genoma.

Exemplo 2.2.9.



O Exemplo 2.2.10 mostra uma reversão intergênica $\rho^{(2,4)}_{(2,0)}$ sendo aplicada na representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (1, 4, 4, 2, 0, 3))$ de um genoma, enquanto o Exemplo 2.2.11 mostra uma reversão intergênica $\rho^{(1,5)}_{(1,2)}$ sendo aplicada na representação intergênica rígida sem sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((0\ 4\ 5\ 3\ 2\ 1\ 6), (1, 1, 7, 3, 0, 2))$ de um genoma. As regiões intergênicas marcadas com sobrescrito podem ter seus tamanhos alterados pelo evento, enquanto as regiões intergênicas marcadas com subscrito sofrem apenas uma troca de posição.

Exemplo 2.2.10.

$$\begin{aligned}(\pi, \breve{\pi}) &= ((+0\ -3\ \underline{+2\ -4\ +1}\ +5\ +6), (1, \overline{4}, \underline{4, 2}, \overline{0}, 3))\\ (\pi, \breve{\pi}) \cdot \rho^{(2,4)}_{(2,0)} &= ((+0\ -3\ \underline{-1\ +4\ -2}\ +5\ +6), (1, \overline{2}, \underline{2, 4}, \overline{2}, 3))\end{aligned}$$

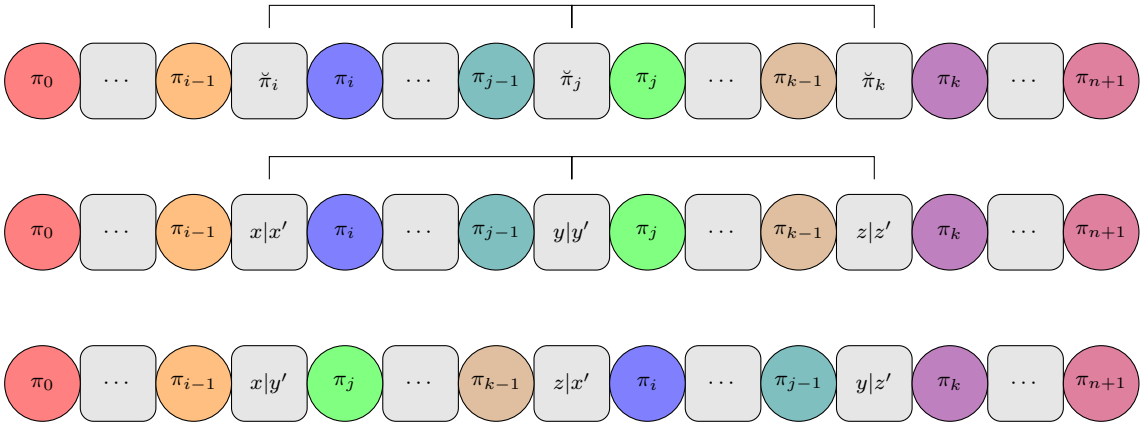
Exemplo 2.2.11.

$$\begin{aligned}(\pi, \breve{\pi}) &= ((0\ \underline{4\ 5\ 3\ 2\ 1}\ 6), (\overline{1}, \underline{1, 7, 3, 0}, \overline{2}))\\ (\pi, \breve{\pi}) \cdot \rho^{(1,5)}_{(1,2)} &= ((0\ \underline{1\ 2\ 3\ 5\ 4}\ 6), (\overline{3}, \underline{0, 3, 7, 1}, \overline{0}))\end{aligned}$$

Definição 2.2.4 (Transposição Intergênica)**.** Seja $\mathcal{R} = (\pi, \breve{\pi})$ uma representação intergênica rígida de um genoma e sejam i, j, k, x, y e z números inteiros tais que $1 \leq i < j < k \leq n + 1$, $0 \leq x \leq \breve{\pi}_i$, $0 \leq y \leq \breve{\pi}_j$ e $0 \leq z \leq \breve{\pi}_k$. Uma *transposição intergênica* $\tau^{(i,j,k)}_{(x,y,z)}$ divide as regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$ da seguinte forma: $\breve{\pi}_i$ em duas partes com tamanhos x e x', onde $x' = \breve{\pi}_i - x$; $\breve{\pi}_j$ em duas partes com tamanhos y e y', onde $y' = \breve{\pi}_j - y$; e $\breve{\pi}_k$ em duas partes com tamanhos z e z', onde $z' = \breve{\pi}_k - z$. Em seguida, os segmentos consecutivos $(x', \pi_i, \breve{\pi}_{i+1}, \ldots \breve{\pi}_{j-1}, \pi_{j-1}, y)$ e $(y', \pi_j, \breve{\pi}_{j+1} \ldots \breve{\pi}_{k-1}, \pi_{k-1}, z)$ trocam de posição sem alterar a orientação dos genes contidos nos segmentos. Por fim, os segmentos do genoma são remontados com os pares de partes (x, y'), (z, x') e (y, z') fundindo-se e formando as novas regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_{k+i-j}$, e $\breve{\pi}_k$ com tamanhos $x + y'$, $z + x'$ e $y + z'$, respectivamente.

O Exemplo 2.2.12 mostra uma transposição intergênica $\tau^{(i,j,k)}_{(x,y,z)}$ genérica sendo aplicada em uma representação intergênica rígida de um genoma. Note que, caso a representação utilizada seja com sinais, o evento não altera a orientação dos genes nos segmentos afetados.

Exemplo 2.2.12.



O Exemplo 2.2.13 mostra uma transposição intergênica $\tau^{(1,3,6)}_{(1,1,3)}$ sendo aplicada na representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((+0\ -4\ -3\ +1\ +2\ +5\ +6), (3, 0, 2, 2, 4, 7))$ de um genoma, enquanto o Exemplo 2.2.14 mostra uma transposição intergênica $\tau^{(4,5,6)}_{(0,0,1)}$ sendo aplicada na representação intergênica rígida sem sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((0\ 3\ 2\ 1\ 5\ 4\ 6), (3, 2, 4, 1, 0, 2))$ de um genoma. As regiões intergênicas marcadas com sobrescrito podem ter seus tamanhos alterados pelo evento, enquanto as regiões intergênicas marcadas com subscrito sofrem apenas uma troca de posição.

Exemplo 2.2.13.

$$\begin{aligned}(\pi, \breve{\pi}) &= ((+0\ \underline{-4\ -3}\ \underline{+1\ +2\ +5}\ +6), (\overline{3}, \underline{0}, \overline{2}, \underline{2, 4}, \overline{7}))\\ (\pi, \breve{\pi}) \cdot \tau^{(1,3,6)}_{(1,1,3)} &= ((+0\ \underline{+1\ +2\ +5}\ \underline{-4\ -3}\ +6), (\overline{2}, \underline{2, 4}, \overline{5}, \underline{0}, \overline{5}))\end{aligned}$$

Exemplo 2.2.14.

$$\begin{aligned}(\pi, \breve{\pi}) &= ((0\ 3\ 2\ 1\ \underline{5}\ \underline{4}\ 6), (3, 2, 4, \overline{1}, \overline{0}, \overline{2}))\\ (\pi, \breve{\pi}) \cdot \tau^{(4,5,6)}_{(0,0,1)} &= ((0\ 3\ 2\ 1\ \underline{4}\ \underline{5}\ 6), (3, 2, 4, \overline{0}, \overline{2}, \overline{1}))\end{aligned}$$

Definição 2.2.5 (Move Intergênico)**.** Seja $\mathcal{R} = (\pi, \breve{\pi})$ uma representação intergênica rígida de um genoma e sejam i, j e x números inteiros tais que $1 \leq i, j \leq n$ e $0 \leq x \leq \breve{\pi}_i$. Um *move intergênico* $\mu^{(i,j)}_{(x)}$ transfere x nucleotídeos da região intergênica $\breve{\pi}_i$ para a região intergênica $\breve{\pi}_j$.

O Exemplo 2.2.15 mostra um move intergênico $\mu^{(2,5)}_{(3)}$ sendo aplicado na representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (1, 4, 4, 2, 0, 3))$ de um genoma, enquanto o Exemplo 2.2.16 mostra um move intergênico $\mu^{(3,5)}_{(5)}$ sendo aplicado na representação intergênica rígida sem sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((0\ 4\ 5\ 3\ 2\ 1\ 6), (1, 1, 7, 3, 0, 2))$ de um genoma. As regiões intergênicas marcadas com sobrescrito sofrem alteração no seu tamanho causada pelo evento.

Exemplo 2.2.15.

$$\begin{aligned}(\pi, \breve{\pi}) &= ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (1, \overline{4}, 4, 2, \overline{0}, 3))\\ (\pi, \breve{\pi}) \cdot \mu^{(2,5)}_{(3)} &= ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (1, \overline{1}, 4, 2, \overline{3}, 3))\end{aligned}$$

Exemplo 2.2.16.

$$\begin{array}{lcl}(\pi, \breve{\pi}) & = & ((0\ 4\ 5\ 3\ 2\ 1\ 6), (1, 1, \overline{7}, 3, \overline{0}, 2)) \\ (\pi, \breve{\pi}) \cdot \mu_{(5)}^{(3,5)} & = & ((0\ 4\ 5\ 3\ 2\ 1\ 6), (1, 1, \overline{2}, 3, \overline{5}, 2))\end{array}$$

A seguir, mostramos como o evento de rearranjo não conservativo de indel intergênico afeta a representação intergênica rígida de um genoma. O evento de rearranjo indel intergênico é uma forma compacta para definir os eventos de inserção e deleção utilizando a mesma notação.

Definição 2.2.6 (Indel Intergênico)**.** Seja $\mathcal{R} = (\pi, \breve{\pi})$ uma representação intergênica rígida de um genoma e sejam i e x números inteiros tais que $1 \leq i \leq n$ e $x \geq -\breve{\pi}_i$. Um indel intergênico $\delta_{(x)}^{(i)}$ remove x nucleotídeos da região intergênica $\breve{\pi}_i$ caso x seja negativo. Caso contrário, um indel intergênico $\delta_{(x)}^{(i)}$ insere x nucleotídeos na região intergênica $\breve{\pi}_i$.

O Exemplo 2.2.17 mostra um indel intergênico $\delta_{(9)}^{(5)}$ sendo aplicado na representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (3, 5, 1, 0, 2, 1))$ de um genoma, enquanto o Exemplo 2.2.18 mostra um indel intergênico $\delta_{(-6)}^{(6)}$ sendo aplicado na representação intergênica rígida sem sinais $\mathcal{R} = (\pi, \breve{\pi}) = ((0\ 4\ 5\ 3\ 2\ 1\ 6), (3, 3, 2, 1, 0, 7))$ de um genoma. As regiões intergênicas marcadas com sobrescrito sofrem alteração no seu tamanho causada pelo evento.

Exemplo 2.2.17.

$$\begin{array}{lcl}(\pi, \breve{\pi}) & = & ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (3, 5, 1, 0, \overline{2}, 1)) \\ (\pi, \breve{\pi}) \cdot \delta_{(9)}^{(5)} & = & ((+0\ -3\ +2\ -4\ +1\ +5\ +6), (3, 5, 1, 0, \overline{11}, 1))\end{array}$$

Exemplo 2.2.18.

$$\begin{array}{lcl}(\pi, \breve{\pi}) & = & ((0\ 4\ 5\ 3\ 2\ 1\ 6), (3, 3, 2, 1, 0, \overline{7})) \\ (\pi, \breve{\pi}) \cdot \delta_{(-6)}^{(6)} & = & ((0\ 4\ 5\ 3\ 2\ 1\ 6), (3, 3, 2, 1, 0, \overline{1}))\end{array}$$

2.3 Caracterização das Instâncias

Os problemas investigados nesta tese têm como principal objetivo transformar uma representação de um genoma de origem $\mathcal{R}_o$ em uma representação de um genoma alvo $\mathcal{R}_a$ utilizando eventos de rearranjo de genoma para essa finalidade. Um *modelo de rearranjo* $\mathcal{M}$ é um conjunto de eventos de rearranjo que podem ser utilizados para transformar um genoma em outro. Os problemas de rearranjo de genomas diferenciam-se pela representação do genoma de origem e alvo utilizada e pelo modelo de rearranjo adotado. A seguir descrevemos os tipos de instâncias que os problemas investigados posteriormente recebem como entrada.

- Uma *instância clássica* é caracterizada por um par de representações clássicas de genomas (π, ι) que compartilham o mesmo conjunto de genes, sendo que ambas as representações podem ser com ou sem sinais. Por padrão, em uma instância clássica utilizaremos π e ι como as representações dos genomas de origem e alvo, respectivamente. O objetivo principal dos problemas que utilizam esse tipo de instância consiste em transformar π em ι.

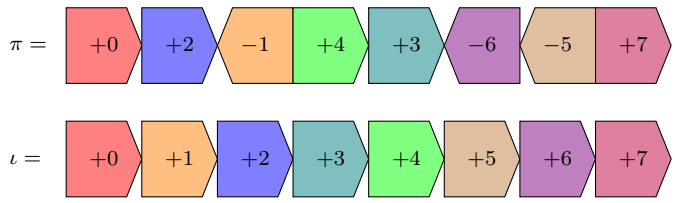
- Uma *instância intergênica rígida* é caracterizada por um par de representações intergênicas rígidas de genomas $((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ que compartilham o mesmo conjunto de genes, sendo que ambas as representações podem ser com ou sem sinais. Por padrão, em uma instância intergênica rígida utilizaremos $(\pi, \breve{\pi})$ e $(\iota, \breve{\iota})$ como as representações dos genomas de origem e alvo, respectivamente. O objetivo principal dos problemas que utilizam esse tipo de instância consiste em transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

- Uma *instância intergênica flexível* é caracterizada por um par de representações de genomas $((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ que compartilham o mesmo conjunto de genes, sendo a primeira representação intergênica rígida e a segunda intergênica flexível. Ambas as representações podem ser com ou sem sinais. Por padrão, em uma instância intergênica flexível utilizaremos $(\pi, \breve{\pi})$ e $(\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max})$ como as representações dos genomas de origem e alvo, respectivamente. O objetivo principal dos problemas que utilizam esse tipo de instância consiste em transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota}')$, tal que $\forall i \in \{1, 2, \ldots, n+1\} : \breve{\iota}_i^{\min} \leq \breve{\iota}'_i \leq \breve{\iota}_i^{\max}$.

Pelo fato dos genes serem representados por uma permutação e os genomas origem e alvo compartilharem o mesmo conjunto de genes, podemos determinar uma permutação padrão ι para os genes do genoma alvo e mapear a permutação do genoma de origem π de acordo com os valores utilizados em ι. A permutação padrão para os genes do genoma alvo é $\iota = (+1\ +2\ \ldots\ +n)$ para uma representação com sinais e $\iota = (1\ 2\ \ldots\ n)$ para uma representação sem sinais. Os elementos $\breve{\iota}$, $\breve{\iota}^{\min}$ e $\breve{\iota}^{\max}$ podem assumir quaisquer valores inteiros não negativos, sendo seus valores definidos a cada instância.

O Exemplo 2.3.1 mostra uma instância clássica com sinais.

Exemplo 2.3.1.



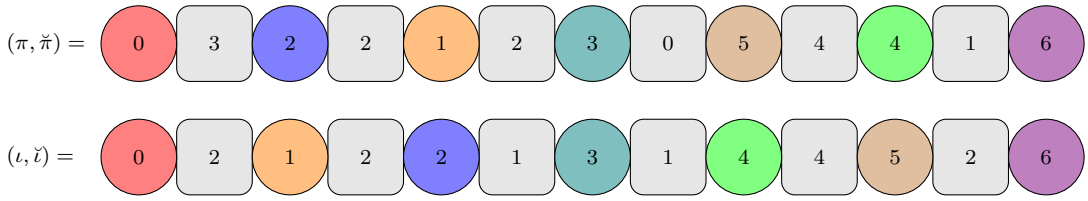
Definição 2.3.1. Uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ é chamada de *balanceada* se a seguinte igualdade é satisfeita:

$$\sum_{\breve{\pi}_i \in \breve{\pi}} \breve{\pi}_i = \sum_{\breve{\iota}_i \in \breve{\iota}} \breve{\iota}_i.$$

Caso contrário, $\mathcal{I}$ é chamada de *desbalanceada*.

O Exemplo 2.3.2 mostra uma instância intergênica rígida balanceada sem sinais.

Exemplo 2.3.2.



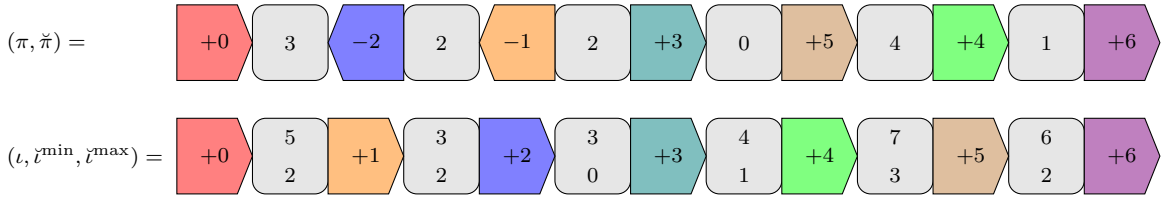
Definição 2.3.2. Uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é chamada de *balanceada* se a seguinte condição é satisfeita:

$$\sum_{\breve{\iota}_i^{\min} \in \breve{\iota}^{\min}} \breve{\iota}_i^{\min} \leq \sum_{\breve{\pi}_i \in \breve{\pi}} \breve{\pi}_i \leq \sum_{\breve{\iota}_i^{\max} \in \breve{\iota}^{\max}} \breve{\iota}_i^{\max}.$$

Caso contrário, $\mathcal{I}$ é chamada de *desbalanceada*.

O Exemplo 2.3.3 mostra uma instância intergênica flexível balanceada com sinais.

Exemplo 2.3.3.



Note que instâncias intergênicas rígidas e flexíveis balanceadas possuem, no genoma de origem, um total de nucleotídeos em que é possível atender todas as restrições referentes aos tamanhos permitidos para cada região intergênica no genoma alvo. Por outro lado, em instâncias intergênicas rígidas e flexíveis desbalanceadas, é necessário inserir ou remover nucleotídeos nas regiões intergênicas do genoma de origem para ser possível transformá-lo no genoma alvo.

2.4 Breakpoints

Nesta seção, apresentamos os conceitos de *breakpoints* em instâncias clássicas e intergênicas rígidas. Esses conceitos são importantes para obtenção de limitantes inferiores e para o desenvolvimento de algoritmos.

2.4.1 Breakpoint Clássico

Nesta seção, apresentamos o conceito de breakpoint para instâncias clássicas.

Definição 2.4.1 (Breakpoint Clássico Tipo Um)**.** Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, um par de elementos (π_i, π_{i+1}), de forma que $0 \leq i \leq n$, é um *breakpoint clássico tipo um* se $|\pi_{i+1} - \pi_i| \neq 1$.

Definição 2.4.2 (Breakpoint Clássico Tipo Dois)**.** Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, um par de elementos (π_i, π_{i+1}), de forma que $0 \leq i \leq n$, é um *breakpoint clássico tipo dois* se $\pi_{i+1} - \pi_i \neq 1$.

Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, o número total de breakpoints clássicos tipo um é denotado por $b_1(\mathcal{I})$. A variação no número de breakpoints clássicos tipo um após a aplicação de uma sequência de eventos de rearranjo S em π é denotada por $\Delta b_1(\mathcal{I}, S) = b_1(\mathcal{I}') - b_1(\mathcal{I})$, onde $\mathcal{I}' = (\pi', \iota)$ e $\pi' = \pi \cdot S$. O número total de breakpoints clássicos tipo dois é denotado por $b_2(\mathcal{I})$. A variação no número de breakpoints clássicos tipo dois após a aplicação de uma sequência de eventos de rearranjo S em π é denotada por $\Delta b_2(\mathcal{I}, S) = b_2(\mathcal{I}') - b_2(\mathcal{I})$, onde $\mathcal{I}' = (\pi', \iota)$ e $\pi' = \pi \cdot S$.

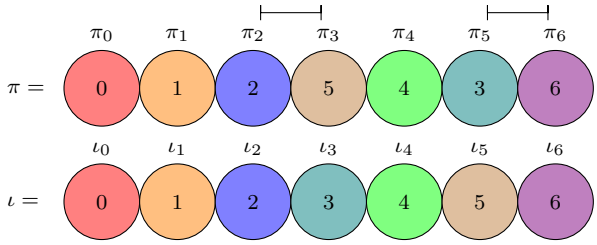
Observação 2.4.1. A única instância clássica $\mathcal{I}$ com $b_1(\mathcal{I}) = 0$ ou $b_2(\mathcal{I}) = 0$ é $\mathcal{I} = (\iota, \iota)$.

Definição 2.4.3 (Strip)**.** Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, *strips* são sequências maximais de elementos de π sem breakpoints clássicos.

Uma strip obtida de uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ com apenas um elemento π_i é chamada de *singleton* e é definida como *crescente* caso $i \in \{0, n\}$. Caso contrário, é definida como *decrescente*. Strips com mais de um elemento são chamadas de crescentes caso os elementos formem uma sequência crescente. Caso contrário, são chamadas de decrescentes. Uma strip obtida de uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ é definida como *positiva* caso todos os elementos da strips tenham sinal positivo. Caso contrário, a strip é definida como *negativa*.

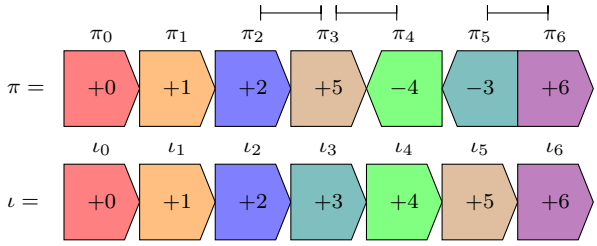
O Exemplo 2.4.1 mostra uma instância clássica sem sinais $\mathcal{I} = ((0\ 1\ 2\ 5\ 4\ 3\ 6), (0\ 1\ 2\ 3\ 4\ 5\ 6))$. Note que a instância possui dois breakpoints clássicos tipo um ($b_1(\mathcal{I}) = 2$), sendo eles (π_2, π_3) e (π_5, π_6). Além disso, obtemos as seguintes strips da instância $\mathcal{I}$: (0 1 2), (5 4 3) e (6), sendo que (0 1 2) e (6) são strips crescentes, enquanto (5 4 3) é uma strip decrescente.

Exemplo 2.4.1.



O Exemplo 2.4.2 mostra uma instância clássica com sinais $\mathcal{I} = ((+0\ +1\ +2\ +5\ -4\ -3\ +6), (+0\ +1\ +2\ +3\ +4\ +5\ +6))$. Note que a instância possui três breakpoints clássicos tipo dois ($b_2(\mathcal{I}) = 3$), sendo eles (π_2, π_3), (π_3, π_4) e (π_5, π_6). As strips obtidas dessa instância com esses breakpoints clássicos tipo dois são: $(+0\ +1\ +2)$, $(+5)$, $(-4\ -3)$ e $(+6)$. Sendo que $(+0\ +1\ +2)$, $(+5)$ e $(+6)$ são positivas enquanto a strip $(-4\ -3)$ é negativa.

Exemplo 2.4.2.



2.4.2 Breakpoint Intergênico

Nesta seção, apresentamos o conceito de breakpoint para instâncias intergênicas.

Definição 2.4.4 (Breakpoint Intergênico Tipo Um)**.** Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, um par de elementos (π_i, π_{i+1}), de forma que $0 \leq i \leq n$, é um *breakpoint intergênico tipo um* se um dos seguintes casos ocorrer:

- $|\pi_{i+1} - \pi_i| \neq 1$
- $|\pi_{i+1} - \pi_i| = 1$ e $\breve{\pi}_{i+1} \neq \breve{\iota}_x$, tal que $x = \max(\pi_i, \pi_{i+1})$.

Definição 2.4.5 (Adjacência Intergênica)**.** Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, um par de elementos (π_a, π_b) é uma *adjacência intergênica* se $|a - b| = 1$ e o par $(\pi_{\min(a,b)}, \pi_{\max(a,b)})$ não é um breakpoint intergênico tipo um.

Note que um breakpoint intergênico tipo um indica um ponto no genoma de origem que deve ser afetado por algum rearranjo de genoma com o objetivo de transformá-lo no genoma alvo. Por outro lado, uma adjacência intergênica mostra um ponto no genoma de origem em que o par de genes considerados também são consecutivos no genoma alvo. Além disso, a região intergênica entre os genes tem o mesmo tamanho no genoma de origem e alvo.

Definição 2.4.6 (Breakpoint Intergênico Sobrecarregado e Subcarregado)**.** Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, um breakpoint intergênico tipo um (π_i, π_{i+1}), tal que $|\pi_{i+1} - \pi_i| = 1$, é chamado de *sobrecarregado* se $\breve{\pi}_{i+1} > \breve{\iota}_x$, com $x = \max(\pi_i, \pi_{i+1})$. Caso contrário, o breakpoint intergênico tipo um (π_i, π_{i+1}) é chamado de *subcarregado*.

Observe que um breakpoint intergênico sobrecarregado é formado por um par de genes que são consecutivos nos genomas de origem e alvo. Contudo, o tamanho da região intergênica entre o par de genes do genoma de origem é maior do que entre o mesmo par de genes no genoma alvo. Já um breakpoint intergênico subcarregado é justamente o cenário oposto: o par de genes são consecutivos nos genomas de origem e alvo, mas a região intergênica entre o par de genes do genoma de origem é menor do que entre o mesmo par de genes no genoma alvo.

Definição 2.4.7 (Breakpoint Intergênico Forte e Suave)**.** Um breakpoint intergênico tipo um (π_i, π_{i+1}) é chamado de *forte* se (π_i, π_{i+1}) é um breakpoint intergênico sobrecarregado ou subcarregado. Caso contrário, o breakpoint intergênico tipo um (π_i, π_{i+1}) é chamado de *suave*.

Definição 2.4.8 (Breakpoint Intergênico Super Forte)**.** Um breakpoint intergênico forte (π_i, π_{i+1}) é chamado de *super forte* se um dos seguintes casos ocorrer:

- $i \in \{0, n\}$
- (π_{i-1}, π_i) ou (π_{i+1}, π_{i+2}) é um breakpoint intergênico forte ou uma adjacência intergênica.

Note que um breakpoint intergênico super forte está em uma das extremidades do genoma de origem ou imediatamente antes ou depois existe um breakpoint intergênico forte ou uma adjacência intergênica.

Definição 2.4.9 (Par Conectado de Breakpoints Intergênicos)**.** Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, um par de breakpoints intergênicos tipo um (π_i, π_{i+1}) e (π_j, π_{j+1}) é chamado de *conectado* se ambas as condições a seguir são satisfeitas:

1. Existe um par de elementos dentre (π_i, π_{i+1}), (π_j, π_{j+1}), (π_i, π_j), (π_i, π_{j+1}), (π_{i+1}, π_j) e (π_{i+1}, π_{j+1}) que são consecutivos em ι e tal par não forma uma adjacência intergênica.

2. $\breve{\pi}_{i+1}+\breve{\pi}_{j+1} \geq \breve{\iota}_k$, tal que $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos (que satisfaz a condição 1) em ι.

Um par de breakpoints intergênicos conectados indica a possibilidade de criar uma adjacência intergênica utilizando apenas o material de ambos os breakpoints intergênicos tipo um (genes e nucleotídeos das regiões intergênicas).

Definição 2.4.10 (Par Suavemente Conectado de Breakpoints Intergênicos)**.** Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, um par de breakpoints intergênicos conectados (π_i, π_{i+1}) e (π_j, π_{j+1}) é chamado de *suavemente conectado* se ambos os breakpoints intergênicos são suaves.

Definição 2.4.11 (Strip Suave)**.** Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida, *strips suaves* são sequências maximais de elementos de π sem breakpoints intergênicos suaves.

Uma strip suave com apenas um elemento π_i é chamada de *singleton* e é definida como crescente caso $i \in \{0, n\}$. Caso contrário, é definida como *decrescente*. Strips suaves com mais de um elemento são chamadas de *crescentes* caso os elementos formem uma sequência crescente. Caso contrário, são chamadas de decrescentes.

Definição 2.4.12 (Breakpoint Intergênico Tipo Dois)**.** Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, um par de elementos (π_i, π_{i+1}), de forma que $0 \leq i \leq n$, é um *breakpoint intergênico tipo dois* se um dos seguintes casos ocorrer:

- $\pi_{i+1} - \pi_i \neq 1$
- $\pi_{i+1} - \pi_i = 1$ e $\breve{\pi}_{i+1} \neq \breve{\iota}_x$, tal que $x = \max(|\pi_i|, |\pi_{i+1}|)$.

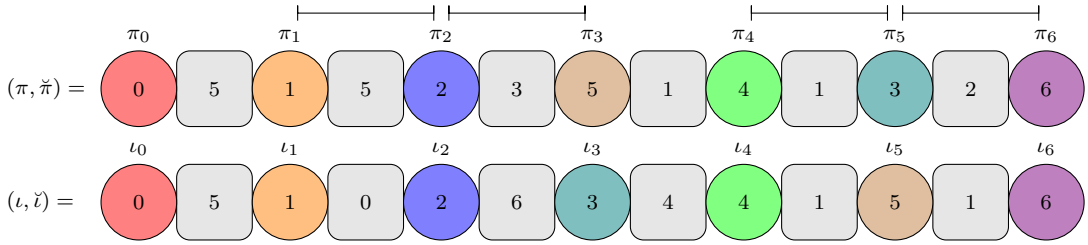
Os breakpoints intergênicos tipo um e dois são utilizados dependendo do tipo da instância intergênica rígida (com ou sem sinais) e do modelo de rearranjo que é considerado, mas ambos os conceitos indicam a mesma informação: os pontos que devem ser afetados no genoma de origem para transformá-lo no genoma alvo.

Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, o número total de breakpoints fortes e suaves são denotados por $ib_f(\mathcal{I})$ e $ib_s(\mathcal{I})$, respectivamente. O número total de breakpoints intergênicos tipo um é denotado por $ib_1(\mathcal{I})$ e note que $ib_1(\mathcal{I}) = ib_f(\mathcal{I}) + ib_s(\mathcal{I})$. A variação no número de breakpoints intergênicos tipo um após a aplicação de uma sequência de eventos de rearranjo S em $(\pi, \breve{\pi})$ é denotada por $\Delta ib_1(\mathcal{I}, S) = ib_1(\mathcal{I}') - ib_1(\mathcal{I})$, onde $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}))$ e $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$. O número total de breakpoints intergênicos tipo dois é denotado por $ib_2(\mathcal{I})$. A variação no número de breakpoints intergênicos tipo dois após a aplicação de uma sequência de eventos de rearranjo S em $(\pi, \breve{\pi})$ é denotada por $\Delta ib_2(\mathcal{I}, S) = ib_2(\mathcal{I}') - ib_2(\mathcal{I})$, onde $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}))$ e $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$.

Observação 2.4.2. A única instância intergênica rígida $\mathcal{I}$ com $ib_1(\mathcal{I}) = 0$ ou $ib_2(\mathcal{I}) = 0$ é $\mathcal{I} = ((\iota, \breve{\iota}), (\iota, \breve{\iota}))$.

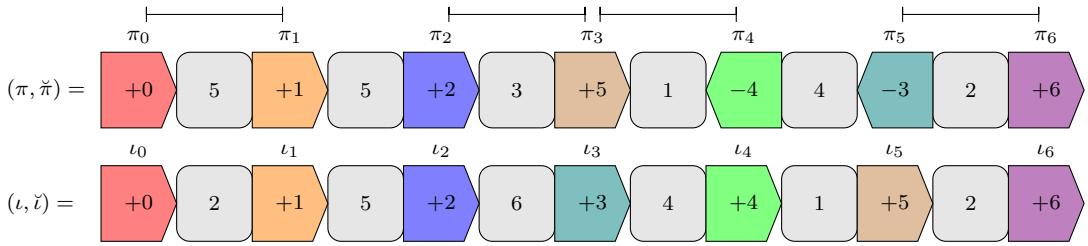
O Exemplo 2.4.3 mostra uma instância intergênica rígida sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 5, 3, 1, 1, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (5, 0, 6, 4, 1, 1)))$. Note que a instância possui quatro breakpoints intergênicos tipo um ($ib_1(\mathcal{I}) = 4$), sendo que $ib_f(\mathcal{I}) = 2$ e $ib_s(\mathcal{I}) = 2$. Os breakpoints intergênicos tipo um (π_1, π_2) e (π_4, π_5) são fortes, sendo que (π_1, π_2) é super forte e sobrecarregado, enquanto (π_4, π_5) é subcarregado. Os breakpoints intergênicos tipo um (π_2, π_3) e (π_5, π_6) são suaves. Entre os pares de breakpoints intergênicos que estão conectados na instância, podemos citar o par de breakpoints intergênicos tipo um $((\pi_1, \pi_2), (\pi_2, \pi_3))$, que está conectado, e o par de breakpoints intergênicos tipo um $((\pi_1, \pi_2), (\pi_4, \pi_5))$, que está suavemente conectado. Além disso, obtemos as seguintes strips suaves da instância $\mathcal{I}$: (0 1 2), (5 4 3) e (6), sendo que (0 1 2) e (6) são strips suaves crescentes, enquanto (5 4 3) é uma strip suave decrescente.

Exemplo 2.4.3.



O Exemplo 2.4.4 mostra uma instância intergênica rígida com sinais $\mathcal{I} = (((+0\ +1\ +2\ +5\ -4\ -3\ +6), (5, 5, 3, 1, 4, 2)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (2, 5, 6, 4, 1, 2)))$. Note que a instância possui quatro breakpoints intergênicos tipo dois ($ib_2(\mathcal{I}) = 4$), sendo eles (π_0, π_1), (π_2, π_3), (π_3, π_4) e (π_5, π_6).

Exemplo 2.4.4.



2.5 Regiões Intergênicas Flexíveis

Nesta seção, apresentamos alguns conceitos relacionados às regiões intergênicas em instâncias intergênicas flexíveis sem sinais. Esses conceitos são importantes para o desenvolvimento de algoritmos e limitantes inferiores para os problemas investigados nos capítulos seguintes.

Definição 2.5.1 (Região Intergênica Estável e Instável)**.** Dada uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, uma região intergênica $\breve{\pi}_i$ é chamada de *estável* se $|\pi_i - \pi_{i-1}| = 1$ e $\breve{\iota}_k^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_k^{\max}$, tal que $k = \max(\pi_{i-1}, \pi_i)$. Caso contrário, a região intergênica $\breve{\pi}_i$ é chamada de *instável*.

Uma região intergênica instável deve necessariamente ser afetada por um evento de rearranjo, seja para unir genes consecutivos no genoma alvo ou para alterar a quantidade de nucleotídeos na região intergênica. Dada uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, os conjuntos de regiões intergênicas estáveis e instáveis em $\mathcal{I}$ são denotados por $\mathcal{S}_e(\mathcal{I})$ e $\mathcal{S}_i(\mathcal{I})$, respectivamente. O número de regiões intergênicas estáveis e instáveis em $\mathcal{I}$ é denotado por $ir_e(\mathcal{I})$ e $ir_i(\mathcal{I})$, respectivamente. A variação no número de regiões intergênicas instáveis após a aplicação de uma sequência de eventos de rearranjo S em $(\pi, \breve{\pi})$ é denotada por $\Delta ir_i(\mathcal{I}, S) = ir_i(\mathcal{I}') - ir_i(\mathcal{I})$, onde $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ e $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$.

Sejam $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ uma instância intergênica flexível sem sinais e $\breve{\pi}_i$ uma região intergênica estável. Denotamos por $gap_{\min}(\breve{\pi}_i) = \breve{\pi}_i - \breve{\iota}_k^{\min}$ e $gap_{\max}(\breve{\pi}_i) = \breve{\iota}_k^{\max} - \breve{\pi}_i$, tal que $k = \max(\pi_{i-1}, \pi_i)$, a quantidade de nucleotídeos que podem ser, respectivamente, removidos e adicionados mantendo $\breve{\pi}_i$ estável.

Observação 2.5.1. Dada uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ tal que $ir_i(\mathcal{I}) = 0$, então $\pi = \iota$ e $\forall \breve{\pi}_i \in \breve{\pi} : \breve{\iota}_i^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_i^{\max}$.

De agora em diante, as definições e conceitos apresentados referem-se às instâncias intergênicas flexíveis balanceadas sem sinais, com a adoção de modelos compostos exclusivamente por eventos de rearranjo conservativos. Note que, dada uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, todas as regiões intergênicas instáveis precisam ser removidas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\pi}')$, tal que $\forall \breve{\pi}_i' \in \breve{\pi}' : \breve{\iota}_i^{\min} \leq \breve{\pi}_i' \leq \breve{\iota}_i^{\max}$. Entretanto, nesse caso em particular, temos que algumas regiões intergênicas estáveis também podem ser afetadas com esse objetivo, dependendo do total de nucleotídeos nas regiões intergênicas instáveis. Regiões intergênicas estáveis devem obrigatoriamente ser afetadas por algum evento de rearranjo conservativo se algum dos seguintes cenários ocorrer:

$$\texttt{(i)} \quad \sum_{\breve{\pi}_i \in \mathcal{S}_i(\mathcal{I})} \breve{\pi}_i < \sum_{\breve{\iota}_i^{\min} \in \breve{\iota}^{\min}} \breve{\iota}_i^{\min} - \sum_{\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})} (\breve{\pi}_i - gap_{\min}(\breve{\pi}_i)).$$

$$\texttt{(ii)} \quad \sum_{\breve{\pi}_i \in \mathcal{S}_i(\mathcal{I})} \breve{\pi}_i > \sum_{\breve{\iota}_i^{\max} \in \breve{\iota}^{\max}} \breve{\iota}_i^{\max} - \sum_{\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})} (\breve{\pi}_i + gap_{\max}(\breve{\pi}_i)).$$

No cenário `(i)`, chamado de *fonte*, a quantidade de nucleotídeos nas regiões intergênicas instáveis não é suficiente para torná-las estáveis. Dessa forma, nucleotídeos das regiões intergênicas estáveis devem ser transferidos para as regiões intergênicas instáveis. No cenário `(ii)`, chamado de *sorvedouro*, a quantidade de nucleotídeos nas regiões intergênicas instáveis excede o limite total permitido para essas regiões intergênicas. Dessa forma, nucleotídeos das regiões intergênicas instáveis devem ser transferidos para as regiões intergênicas estáveis. Uma instância intergênica flexível balanceada sem sinais que não pertence ao cenário fonte ou sorvedouro pertence ao cenário de *equilíbrio*. Nesse caso, a quantidade de nucleotídeos nas regiões intergênicas instáveis é suficiente para torná-las estáveis. Com a possível necessidade de afetar algumas regiões intergênicas estáveis, temos a seguinte definição.

Definição 2.5.2 (Região Intergênica Auxiliar e Definitiva)**.** Dada uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, uma região intergênica

estável $\breve{\pi}_i$ é chamada de *auxiliar* se $\breve{\pi}_i$ deve receber nucleotídeos de regiões intergênicas instáveis ou transferir nucleotídeos para regiões intergênicas instáveis. Caso contrário, $\breve{\pi}_i$ é chamada de *definitiva*.

O número total de regiões intergênicas auxiliares depende do cenário da instância $\mathcal{I}$. No caso do cenário fonte, o conjunto de regiões intergênicas auxiliares $\mathcal{S}_a(\mathcal{I})$ é tal que seu tamanho é mínimo e a seguinte restrição é satisfeita:

$$\sum_{\breve{\pi}_i \in \mathcal{S}_a(\mathcal{I})} gap_{\min}(\breve{\pi}_i) \geq \sum_{\breve{\iota}_i^{\min} \in \breve{\iota}^{\min}} \breve{\iota}_i^{\min} - \sum_{\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})} (\breve{\pi}_i - gap_{\min}(\breve{\pi}_i)) - \sum_{\breve{\pi}_i \in \mathcal{S}_i(\mathcal{I})} \breve{\pi}_i.$$

Note que o conjunto $\mathcal{S}_a(\mathcal{I})$ com tamanho mínimo pode ser facilmente obtido ordenando as regiões intergênicas estáveis em ordem decrescente pelo valor de $gap_{\min}$. Em seguida, cada região intergênica é rotulada como auxiliar até que a restrição seja satisfeita. No caso do cenário sorvedouro, o conjunto de regiões intergênicas auxiliares $\mathcal{S}_a(\mathcal{I})$ é tal que seu tamanho é mínimo e a seguinte restrição é satisfeita:

$$\sum_{\breve{\pi}_i \in \mathcal{S}_a(\mathcal{I})} gap_{\max}(\breve{\pi}_i) \geq \sum_{\breve{\pi}_i \in \mathcal{S}_i(\mathcal{I})} \breve{\pi}_i - \sum_{\breve{\iota}_i^{\max} \in \breve{\iota}^{\max}} \breve{\iota}_i^{\max} + \sum_{\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})} (\breve{\pi}_i + gap_{\max}(\breve{\pi}_i)).$$

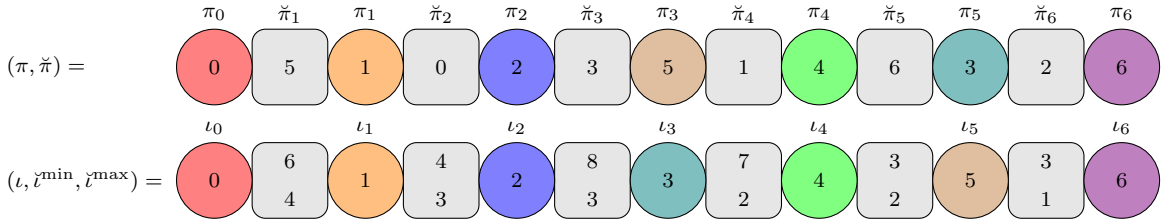
Semelhante ao cenário anterior, o conjunto $\mathcal{S}_a(\mathcal{I})$ com tamanho mínimo também pode ser facilmente obtido ordenando as regiões intergênicas estáveis em ordem decrescente pelo valor de $gap_{\max}$ e efetuando a rotulação das regiões intergênicas como auxiliares até que a restrição seja satisfeita. Obtendo o conjunto $\mathcal{S}_a(\mathcal{I})$, temos que o conjunto das regiões intergênicas definitivas $\mathcal{S}_d(\mathcal{I})$ pode ser obtido pela operação $\mathcal{S}_e(\mathcal{I}) \setminus \mathcal{S}_a(\mathcal{I})$. Note que $\mathcal{S}_a(\mathcal{I}) \cup \mathcal{S}_d(\mathcal{I}) = \mathcal{S}_e$.

Caso o cenário de equilíbrio ocorra, então temos que $\mathcal{S}_a(\mathcal{I}) = \emptyset$ e $\mathcal{S}_d(\mathcal{I}) = \mathcal{S}_e$, ou seja, o total de nucleotídeos nas regiões intergênicas instáveis é suficiente para torná-las estáveis sem ser preciso afetar as regiões intergênicas estáveis. Dada uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, o número de regiões intergênicas auxiliares em $\mathcal{I}$ é denotado por $ir_a(\mathcal{I})$. A variação no número de regiões intergênicas auxiliares após aplicar uma sequência de eventos de rearranjo S em $(\pi, \breve{\pi})$ é denotada por $\Delta ir_a(\mathcal{I}, S) = ir_a(\mathcal{I}') - ir_a(\mathcal{I})$, onde $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ e $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$.

Observação 2.5.2. Dada uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ tal que $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = 0$, então $\pi = \iota$ e $\forall \breve{\pi}_i \in \breve{\pi} : \breve{\iota}_i^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_i^{\max}$.

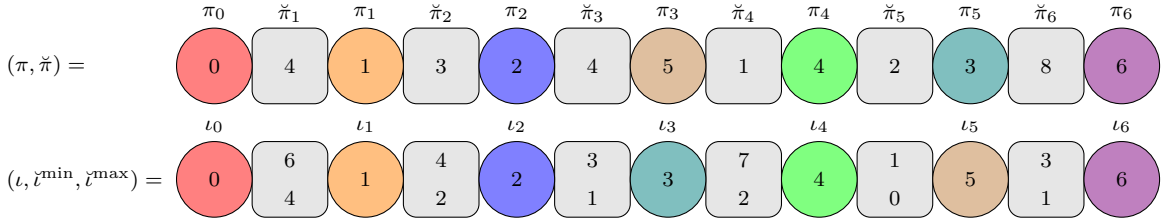
O Exemplo 2.5.1 mostra uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 0, 3, 1, 6, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 3, 3, 2, 2, 1), (6, 4, 8, 7, 3, 3)))$ que pertence ao cenário fonte. Note que a instância $\mathcal{I}$ possui quatro regiões intergênicas instáveis ($ir_i(\mathcal{I}) = 4$, com $\mathcal{S}_i = \{\breve{\pi}_2, \breve{\pi}_3, \breve{\pi}_4, \breve{\pi}_6\}$) e duas regiões intergênicas estáveis ($\mathcal{S}_e = \{\breve{\pi}_1, \breve{\pi}_5\}$). No exemplo, temos apenas uma região intergênica auxiliar ($ir_a(\mathcal{I}) = 1$ e $\mathcal{S}_a = \{\breve{\pi}_5\}$). Note que $gap_{\min}(\breve{\pi}_1) = 1$ e $gap_{\min}(\breve{\pi}_5) = 4$.

Exemplo 2.5.1.



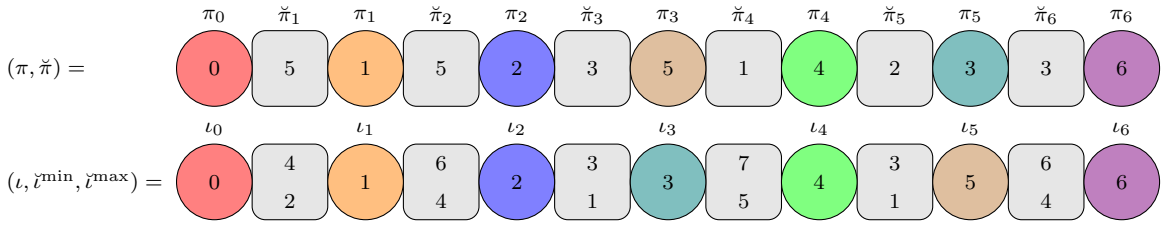
O Exemplo 2.5.2 mostra uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (4, 3, 4, 1, 2, 8)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 2, 1, 2, 0, 1), (6, 4, 3, 7, 1, 3)))$ que pertence ao cenário sorvedouro. Note que a instância $\mathcal{I}$ possui duas regiões intergênicas instáveis ($ir_i(\mathcal{I}) = 2$ e $\mathcal{S}_i = \{\breve{\pi}_3, \breve{\pi}_6\}$) e quatro regiões intergênicas estáveis ($\mathcal{S}_e = \{\breve{\pi}_1, \breve{\pi}_2, \breve{\pi}_4, \breve{\pi}_5\}$). No exemplo, temos duas regiões intergênica auxiliares ($ir_a(\mathcal{I}) = 2$ e $\mathcal{S}_a = \{\breve{\pi}_1, \breve{\pi}_5\}$). Note que $gap_{\max}(\breve{\pi}_1) = 2$, $gap_{\max}(\breve{\pi}_2) = 1$, $gap_{\max}(\breve{\pi}_4) = 0$ e $gap_{\max}(\breve{\pi}_5) = 5$.

Exemplo 2.5.2.



O Exemplo 2.5.3 mostra uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 5, 3, 1, 2, 3)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (2, 4, 1, 5, 1, 4), (4, 6, 3, 7, 3, 6)))$ que não pertence ao cenário fonte ou sorvedouro. Note que por esse motivo a instância não possui regiões intergênicas auxiliares, ou seja, $ir_a(\mathcal{I}) = 0$ e $\mathcal{S}_a = \emptyset$. A instância $\mathcal{I}$ possui cinco regiões intergênicas instáveis ($ir_i(\mathcal{I}) = 5$ e $\mathcal{S}_i = \{\breve{\pi}_1, \breve{\pi}_3, \breve{\pi}_4, \breve{\pi}_5, \breve{\pi}_6\}$) e uma região intergênica estável ($\mathcal{S}_e = \{\breve{\pi}_2\}$).

Exemplo 2.5.3.



2.6 Grafo de Ciclos

Grafos são estruturas amplamente utilizadas em problemas de rearranjo de genomas para obtenção de limitantes inferiores e algoritmos. Nesta seção, apresentamos os grafos de ciclos clássico, ponderado e ponderado flexível.

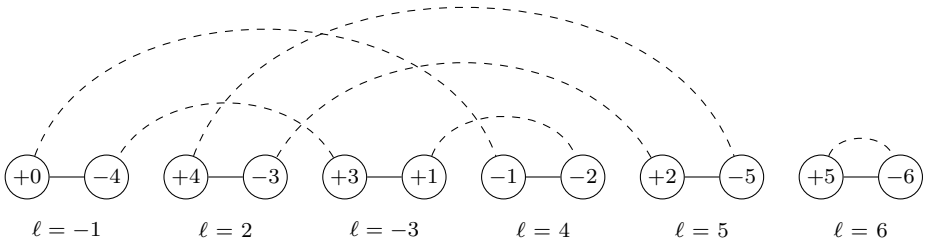
2.6.1 Grafo de Ciclos Clássico

O *grafo de ciclos clássico*, também chamado de *grafo de breakpoints* [12,47], tem seu uso bastante difundido em problemas de rearranjo de genomas que utilizam instâncias clássicas. Esse grafo evidencia em uma mesma estrutura as adjacências presentes no genoma de origem e as adjacências desejadas no genoma alvo. A seguir definimos formalmente o grafo de ciclos clássico.

Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, definimos o *grafo de ciclos clássico* por $G(\mathcal{I}) = (V, E, \ell)$, tal que V, E e ℓ representam o conjunto de vértices, o conjunto de arestas e uma função de rotulação de arestas, respectivamente. O conjunto de vértices V é dado por $\{+\pi_0, -\pi_1, +\pi_1, -\pi_2, +\pi_2, \ldots, -\pi_n, +\pi_n, -\pi_{n+1}\}$. Note que para cada elemento π_i, com $1 \leq i \leq n$, adicionamos em V os vértices $-\pi_i$ e $+\pi_i$. Por fim, adicionamos em V os vértices $+\pi_0$ e $-\pi_{n+1}$. O conjunto de arestas $E = E_p \cup E_c$ é dividido nos conjuntos de arestas pretas (E_p) e arestas cinzas (E_c), onde $E_p = \{\{-\pi_i, +\pi_{i-1}\} \,|\, 1 \leq i \leq n+1\}$ e $E_c = \{\{+(i-1), -i\} \,|\, 1 \leq i \leq n+1\}$. Perceba que as arestas pretas representam os elementos adjacentes na permutação π, enquanto as arestas cinzas representam os elementos adjacentes em ι.

Existem diferentes formas de desenhar o grafo de ciclos clássico. Entretanto, utilizaremos a forma que chamamos de *padrão*. Para essa forma de desenhar o grafo, os vértices são posicionados horizontalmente da esquerda para direita, seguindo a ordem $+\pi_0, -\pi_1, +\pi_1, -\pi_2, +\pi_2, \ldots, -\pi_n, +\pi_n, -\pi_{n+1}$. As arestas pretas são desenhadas formando uma linha horizontal contínua, enquanto as arestas cinzas formam arcos com linhas tracejadas sobre os vértices. O Exemplo 2.6.1 mostra o grafo de ciclos clássico construído a partir da instância clássica $\mathcal{I} = ((+0\ +4\ +3\ -1\ +2\ +5\ +6), (+0\ +1\ +2\ +3\ +4\ +5\ +6))$.

Exemplo 2.6.1.



Pelo Exemplo 2.6.1, podemos perceber que o grafo de ciclos clássico possui $2n + 2$ vértices e $2n + 2$ arestas ($n + 1$ pretas e $n + 1$ cinzas), sendo que em cada vértice duas arestas são incidentes, uma preta e uma cinza. Por esse motivo, há uma decomposição única de $G(\mathcal{I})$ em ciclos com arestas de cores alternadas.

A função de rotulação $\ell : E_p \to \{-(n+1), -n, \ldots, -2, -1, 1, 2, \ldots, n, (n+1)\}$ atribui um rótulo para cada aresta preta no grafo em função da direção em que a aresta é percorrida. Dada uma aresta preta $e_p = \{-\pi_i, +\pi_{i-1}\} \in E_p$, a função ℓ atribui o rótulo i em e_p caso ela seja percorrida de $-\pi_i$ até $+\pi_{i-1}$. Caso contrário, e_p é rotulada com $-i$. Por padrão, cada ciclo de $G(\mathcal{I})$ é representado pela sequência de rótulos de suas arestas pretas na ordem em que elas são percorridas, sendo que a primeira aresta preta de um ciclo é aquela que encontra-se mais à direita no grafo, seguindo o desenho padrão, e é percorrida

da direita para esquerda, ou seja, de $-\pi_i$ até $+\pi_{i-1}$. Essa representação utilizada para os ciclos faz com que eles sejam representados unicamente. No Exemplo 2.6.1, $G(\mathcal{I})$ possui três ciclos: $C_1 = (4, -1, -3)$, $C_2 = (5, 2)$ e $C_3 = (6)$.

O tamanho de um ciclo C em $G(\mathcal{I})$ é dado pela quantidade de arestas pretas do ciclo. Um ciclo de tamanho um é chamado de *trivial*. Um ciclo com tamanho menor que três é chamado de *curto*. Caso contrário, é chamado de *longo*.

Definição 2.6.1 (Arestas Divergentes e Convergentes)**.** Duas arestas pretas de um ciclo C em $G(\mathcal{I})$ são chamadas de *divergentes* se elas são percorridas em direções opostas. Caso contrário, são chamadas de *convergentes*.

Definição 2.6.2 (Ciclo Divergente e Convergente)**.** Um ciclo C em $G(\mathcal{I})$ é chamado de *divergente* se pelo menos um par de arestas pretas de C são divergentes. Caso contrário, C é chamado de *convergente*.

Podemos ainda classificar ciclos convergentes como *orientados* ou *não orientados*.

Definição 2.6.3 (Ciclo Orientado e Não Orientado)**.** Um ciclo convergente $C = (c_1, c_2, \ldots, c_k)$ em $G(\mathcal{I})$ é classificado como *não orientado* se $c_i > c_{i+1}$, para todo i com $1 \leq i < k$. Caso contrário, C é classificado como *orientado*.

Dois ciclos de mesmo tamanho $C = (c_1, c_2, \ldots, c_k)$ e $D = (d_1, d_2, \ldots, d_k)$, ambos pertencentes ao grafo $G(\mathcal{I})$, são *entrelaçados* se $|c_1| > |d_1| > |c_2| > |d_2| > \ldots > |c_k| > |d_k|$ ou $|d_1| > |c_1| > |d_2| > |c_2| > \ldots > |d_k| > |c_k|$. Seja g_1 uma aresta cinza adjacente às arestas pretas com rótulos x_1 e y_1 tais que $|x_1| < |y_1|$, e seja g_2 uma aresta cinza adjacente às arestas pretas com rótulos x_2 e y_2 tais que $|x_2| < |y_2|$. Dizemos que duas arestas cinzas g_1 e g_2 *cruzam-se* caso $|x_1| < |x_2| \leq |y_1| < |y_2|$. Dois ciclos C e D *cruzam-se* caso uma aresta cinza de C cruza-se com uma aresta cinza de D. Um *open gate* é uma aresta cinza de um ciclo não trivial C em $G(\mathcal{I})$ que não se cruza com nenhuma outra aresta cinza de C. Um open gate g_1 de C é fechado se outra aresta cinza (de um ciclo diferente de C) cruza com g_1.

Observação 2.6.1. Todos os *open gates* em $G(\mathcal{I})$ são fechados [11].

No Exemplo 2.6.1, os ciclos $C_1 = (4, -1, -3)$, $C_2 = (5, 2)$ e $C_3 = (6)$ são, respectivamente, longo divergente, curto convergente orientado e trivial. Note que o ciclo C_1 possui o open gate $(+3, -4)$, enquanto o ciclo C_2 possui os seguintes open gates: $(+2, -3)$ e $(+4, -5)$.

Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, denotamos por $c(G(\mathcal{I}))$ o número de ciclos em $G(\mathcal{I})$. Dada uma sequência de eventos de rearranjo S, denotamos por $\Delta c(G(\mathcal{I}), S) = c(G(\mathcal{I}')) - c(G(\mathcal{I}))$, tal que $\mathcal{I}' = (\pi \cdot S, \iota)$, a variação no número de ciclos após a aplicação da sequência S no genoma de origem π de $\mathcal{I}$.

Observação 2.6.2. A única instância clássica $\mathcal{I}$ com $c(G(\mathcal{I})) = n + 1$ é $\mathcal{I} = (\iota, \iota)$.

2.6.2 Grafo de Ciclos Ponderado Rígido

O grafo de ciclos ponderado rígido é uma extensão do grafo de ciclos clássico, incorporando na sua estrutura, através de pesos nas arestas, informações referentes ao tamanho

das regiões intergênicas do genoma de origem e do genoma alvo. A seguir, definimos formalmente o grafo de ciclos ponderado rígido.

Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, definimos o *grafo de ciclos ponderado rígido* por $G(\mathcal{I}) = (V, E = E_p \cup E_c, \ell, w_p, w_c)$, tal que V, E e ℓ representam, respectivamente, o conjunto de vértices, o conjunto de arestas e uma função de rotulação de arestas, enquanto w_p e w_c são funções de peso. Pelo fato do grafo de ciclos ponderado rígido tratar-se de uma extensão do grafo de ciclos clássico, V, E e ℓ comportam-se exatamente como anteriormente descrito. Além disso, todos os conceitos, definições e representações que foram apresentados no contexto de grafo de ciclos clássico também são válidos e utilizados no grafo de ciclos ponderado rígido.

A função de peso $w_p : E_p \to \mathbb{N}_0$ associa os tamanhos das regiões intergênicas no genoma de origem com pesos nas arestas pretas do grafo. A função de peso $w_c : E_c \to \mathbb{N}_0$ funciona de maneira similar, mas associando os tamanhos das regiões intergênicas no genoma alvo com pesos nas arestas cinzas do grafo. Para cada aresta preta $e_i = \{-\pi_i, +\pi_{i-1}\} \in E_p$, temos que $w_p(e_i) = \breve{\pi}_i$. Para cada aresta cinza $e'_i = \{+(i-1), -i\} \in E_c$, temos que $w_c(e'_i) = \breve{\iota}_i$. Dado um ciclo $C \in G(\mathcal{I})$, denotamos por $E_p(C)$ e $E_c(C)$, respectivamente, os conjuntos de arestas pretas e cinzas que pertencem ao ciclo C.

Definição 2.6.4 (Ciclo Balanceado e Desbalanceado)**.** Um ciclo C em $G(\mathcal{I})$ é chamado de *balanceado* caso $\sum_{e'_i \in E_c(C)} w_c(e'_i) - \sum_{e_i \in E_p(C)} w_p(e_i) = 0$. Caso contrário, o ciclo C é chamado de *desbalanceado*.

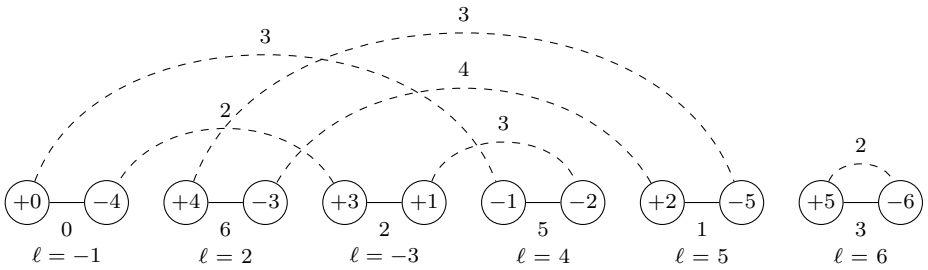
Em outras palavras, em um ciclo balanceado a soma dos pesos em suas arestas pretas é a mesma que a soma dos pesos em suas arestas cinzas.

Definição 2.6.5 (Ciclo Negativo e Positivo)**.** Um ciclo desbalanceado C em $G(\mathcal{I})$ é chamado de *negativo* quando $\sum_{e'_i \in E_c(C)} w_c(e'_i) - \sum_{e_i \in E_p(C)} w_p(e_i) < 0$. Caso contrário, o ciclo C é chamado de *positivo*.

Note que os eventos de rearranjo afetam apenas as arestas pretas no grafo de ciclos e suas extensões. Considerando um contexto composto por eventos conservativos, um ciclo negativo após ser afetado por evento de rearranjo apresenta um potencial maior de gerar novos ciclos balanceados, uma vez que existe peso suficiente em suas arestas pretas para atender o peso exigido em cada uma de suas arestas cinzas. Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, denotamos por $c(G(\mathcal{I}))$ e $c_b(G(\mathcal{I}))$ o número de ciclos e o número ciclos balanceados em $G(\mathcal{I})$, respectivamente. Dada uma sequência de eventos de rearranjo S, denotamos por $\Delta c(G(\mathcal{I}), S) = c(G(\mathcal{I}')) - c(G(\mathcal{I}))$ e $\Delta c_b(G(\mathcal{I}), S) = c_b(G(\mathcal{I}')) - c_b(G(\mathcal{I}))$, tal que $\mathcal{I}' = ((\pi, \breve{\pi}) \cdot S, (\iota, \breve{\iota}))$, a variação no número de ciclos e no número ciclos balanceados, respectivamente, após aplicar a sequência S no genoma de origem $(\pi, \breve{\pi})$ de $\mathcal{I}$.

O Exemplo 2.6.2 mostra o grafo de ciclos ponderado rígido construído a partir da instância intergênica rígida $\mathcal{I} = (((+0\ +4\ +3\ -1\ +2\ +5\ +6), (0, 6, 2, 5, 1, 3)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (3, 3, 4, 2, 3, 2)))$.

Exemplo 2.6.2.



No Exemplo 2.6.2, os ciclos $C_1 = (4, -1, -3)$, $C_2 = (5, 2)$ e $C_3 = (6)$ são, respectivamente, longo positivo, curto balanceado e trivial negativo.

Observação 2.6.3. A única instância intergênica rígida $\mathcal{I}$ com $c(G(\mathcal{I})) = n+1$ e $c_b(G(\mathcal{I})) = n+1$ é $\mathcal{I} = ((\iota, \breve{\iota}), (\iota, \breve{\iota}))$.

2.6.3 Grafo de Ciclos Ponderado Flexível

O grafo de ciclos ponderado flexível também é uma extensão do grafo de ciclos clássico, incorporando na sua estrutura, através de pesos nas arestas, informações referentes ao tamanho das regiões intergênicas do genoma de origem e os tamanhos mínimos e máximos permitidos para cada região intergênica no genoma alvo. A seguir, definimos formalmente o grafo de ciclos ponderado flexível.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, definimos o *grafo de ciclos ponderado flexível* por $G(\mathcal{I}) = (V, E = E_p \cup E_c, \ell, w_p, w_c^{\min}, w_c^{\max})$, tal que V, E e ℓ representam, respectivamente, o conjunto de vértices, o conjunto de arestas e uma função de rotulação de arestas, enquanto w_p, $w_c^{\min}$ e $w_c^{\max}$ são funções de peso. Pelo fato do grafo de ciclos ponderado flexível também tratar-se de uma extensão do grafo de ciclos clássico, V, E e ℓ comportam-se exatamente como já descrito. Além disso, todos os conceitos, definições e representações que foram apresentados no contexto de grafo de ciclos clássico também são válidos e utilizados no grafo de ciclos ponderado flexível.

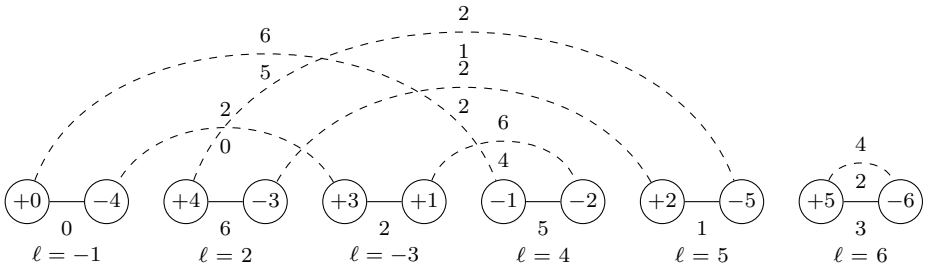
A função de peso $w_p : E_p \to \mathbb{N}_0$ associa os tamanhos das regiões intergênicas no genoma de origem com pesos nas arestas pretas do grafo. As funções de peso $w_c^{\min} : E_c \to \mathbb{N}_0$ e $w_c^{\max} : E_c \to \mathbb{N}_0$ associam, respectivamente, os tamanhos mínimos e máximos permitidos para as regiões intergênicas no genoma alvo com pesos nas arestas cinzas do grafo. Para cada aresta preta $e_i = \{-\pi_i, +\pi_{i-1}\} \in E_p$, temos que $w_p(e_i) = \breve{\pi}_i$. Para cada aresta cinza $e'_i = \{+(i-1), -i\} \in E_c$, temos que $w_c^{\min}(e'_i) = \breve{\iota}_i^{\min}$ e $w_c^{\max}(e'_i) = \breve{\iota}_i^{\max}$. Dado um ciclo C em $G(\mathcal{I})$, denotamos por $E_p(C)$ e $E_c(C)$, respectivamente, os conjuntos de arestas pretas e cinzas que pertencem ao ciclo C. Dado um ciclo C em $G(\mathcal{I})$, denotamos por $W_p(C) = \sum_{e_i \in E_p(C)} w_p(e_i)$, $W_c^{\min}(C) = \sum_{e'_i \in E_c(C)} w_c^{\min}(e'_i)$ e $W_c^{\max}(C) = \sum_{e'_i \in E_c(C)} w_c^{\max}(e'_i)$ o *peso total*, *peso mínimo total* e *peso máximo total* de C, respectivamente. Note que o peso total de um ciclo é a soma dos pesos em suas arestas pretas. Já os pesos mínimo total e máximo total são, respectivamente, a soma dos pesos mínimos e máximos em suas arestas cinzas.

Definição 2.6.6 (Ciclos Estável e Instável)**.** Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, um ciclo C em $G(\mathcal{I})$ é chamado de *estável* caso $W_c^{\min}(C) \leq W_p(C) \leq W_c^{\max}(C)$. Caso contrário, o ciclo C é chamado de *instável*.

Em outras palavras, um ciclo estável indica que o peso total é suficiente para satisfazer as restrições relativas aos pesos mínimos e máximos em cada uma de suas arestas cinzas. Denotamos os conjuntos de ciclos estáveis e instáveis em $G(\mathcal{I})$ como $\mathcal{S}_e(G(\mathcal{I}))$ e $\mathcal{S}_i(G(\mathcal{I}))$, respectivamente. Dado um ciclo C em $G(\mathcal{I})$, denotamos por $gap_{\min}(C) = W_p(C) - W_c^{\min}(C)$ e $gap_{\max}(C) = W_c^{\max}(C) - W_p(C)$ os valores que se subtraídos e adicionados do peso total de C resultam, respectivamente, nos pesos mínimo total e máximo total de C.

O Exemplo 2.6.3 mostra o grafo de ciclos ponderado flexível construído a partir da instância intergênica flexível $\mathcal{I} = (((+0\ +4\ +3\ -1\ +2\ +5\ +6), (0, 6, 2, 5, 1, 3)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (5, 4, 2, 0, 1, 2), (6, 6, 2, 2, 2, 4)))$.

Exemplo 2.6.3.



No Exemplo 2.6.3, os ciclos $C_1 = (4, -1, -3)$, $C_2 = (5, 2)$ e $C_3 = (6)$ são, respectivamente, longo instável, curto instável e trivial estável.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, denotamos por $c(G(\mathcal{I}))$ e $c_e(G(\mathcal{I}))$ o número de ciclos e o número ciclos estáveis em $G(\mathcal{I})$, respectivamente. Dada uma sequência de eventos de rearranjo S, denotamos por $\Delta c(G(\mathcal{I}), S) = c(G(\mathcal{I}')) - c(G(\mathcal{I}))$ e $\Delta c_e(G(\mathcal{I}), S) = c_e(G(\mathcal{I}')) - c_e(G(\mathcal{I}))$, tal que $\mathcal{I}' = ((\pi, \breve{\pi}) \cdot S, (\iota, \breve{\iota}))$, a variação no número de ciclos e no número ciclos estáveis, respectivamente, após a aplicação da sequência S no genoma de origem $(\pi, \breve{\pi})$ de $\mathcal{I}$.

Observação 2.6.4. Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ tal que $c(G(\mathcal{I})) = c_e(G(\mathcal{I})) = n + 1$, temos que $\pi = \iota$ e $\breve{\iota}_i^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_i^{\max}$ para todo $\breve{\pi}_i \in \breve{\pi}$.

De agora em diante, as definições e conceitos apresentados referem-se às instâncias intergênicas flexíveis balanceadas cujos modelos são compostos exclusivamente por eventos de rearranjo conservativos. Note que, dada uma instância intergênica flexível balanceada $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, todos os ciclos instáveis devem ser removidos e $G(\mathcal{I})$ deve possuir $n+1$ ciclos estáveis para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\pi}')$, tal que $\forall \breve{\pi}_i' \in \breve{\pi}', \breve{\iota}_i^{\min} \leq \breve{\pi}_i' \leq \breve{\iota}_i^{\max}$. Dependendo da distribuição dos nucleotídeos e das restrições de tamanho mínimo e máximo nas arestas cinzas, alguns dos ciclos estáveis também devem ser afetados para realizar essa tarefa. Na verdade, existem dois cenários em que isso ocorre:

$$\texttt{(i)} \quad \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C) < \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\min}(C).$$

$$\texttt{(ii)} \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C) > \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\max}(C).$$

No cenário `(i)`, chamado de *fonte*, a soma do peso total de todos os ciclos instáveis é menor que a soma do peso mínimo total dos mesmos ciclos, ou seja, não é possível atender todas as restrições de peso mínimo e máximo nas arestas cinzas dos ciclos instáveis sem que alguns ciclos estáveis transfiram uma determinada quantidade de peso de suas arestas pretas para os ciclos instáveis. No cenário `(ii)`, chamado de *sorvedouro*, a soma do peso total de todos os ciclos instáveis é maior que a soma do peso máximo total dos mesmos ciclos. Nesse caso, alguns ciclos instáveis precisam transferir uma determinada quantidade de peso de suas arestas pretas para alguns dos ciclos estáveis. Uma instância intergênica flexível balanceada que não pertence ao cenário fonte ou sorvedouro, pertence ao cenário de *equilíbrio*. Neste caso, é possível transformar os ciclos instáveis em estáveis sem afetar os demais ciclos estáveis da instância.

Definição 2.6.7 (Ciclo Auxiliar e Definitivo)**.** Dada uma instância intergênica flexível balanceada $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, um ciclo estável C em $G(\mathcal{I})$ é chamado de *auxiliar* se ele deve receber ou transferir peso de suas arestas pretas para outro ciclo, e é chamado de *definitivo* caso contrário.

Denotamos os conjuntos de ciclos auxiliares e definitivos em $G(\mathcal{I})$ como $\mathcal{S}_a(G(\mathcal{I}))$ e $\mathcal{S}_d(G(\mathcal{I}))$, respectivamente. Observe que os cenários fonte e sorvedouro não ocorrem se $\sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\min}(C) \leq \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C) \leq \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\max}(C)$, onde temos que $\mathcal{S}_a(G(\mathcal{I})) = \emptyset$ e $\mathcal{S}_d(G(\mathcal{I})) = \mathcal{S}_e(G(\mathcal{I}))$ (Exemplo 2.6.3). Note que os cenários fonte e sorvedouro não podem ocorrer simultaneamente. Caso um deles ocorra, então é necessário determinar os conjuntos $\mathcal{S}_a(G(\mathcal{I}))$ e $\mathcal{S}_d(G(\mathcal{I}))$, que são dependentes do cenário em que a instância se encaixa.

Considerando o cenário fonte, um conjunto $\mathcal{S}_a(G(\mathcal{I}))$ de tamanho mínimo pode ser composto do menor número de ciclos de forma que a seguinte restrição seja cumprida:

$$\sum_{C \in \mathcal{S}_a(G(\mathcal{I}))} gap_{\min}(C) + \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} gap_{\min}(C) \geq 0.$$

Considerando o cenário sorvedouro, um conjunto $\mathcal{S}_a(G(\mathcal{I}))$ de tamanho mínimo pode ser composto do menor número de ciclos de forma que a seguinte restrição seja cumprida:

$$\sum_{C \in \mathcal{S}_a(G(\mathcal{I}))} gap_{\max}(C) + \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} gap_{\max}(C) \geq 0.$$

Observe que, em ambos os cenários, o conjunto $\mathcal{S}_a(G(\mathcal{I}))$ pode ser facilmente obtido após a ordenação, de forma decrescente, dos ciclos estáveis pelos valores $gap_{\min}$ e $gap_{\max}$, considerando os casos fonte e sorvedouro, respectivamente. Então, seguindo a ordem decrescente, os ciclos são rotulados como auxiliares até que a restrição seja satisfeita. O conjunto de ciclos definitivos $\mathcal{S}_d(G(\mathcal{I}))$ é obtido por $\mathcal{S}_e(G(\mathcal{I})) \setminus \mathcal{S}_a(G(\mathcal{I}))$, note que $\mathcal{S}_a(G(\mathcal{I})) \cup \mathcal{S}_d(G(\mathcal{I})) = \mathcal{S}_e(G(\mathcal{I}))$.

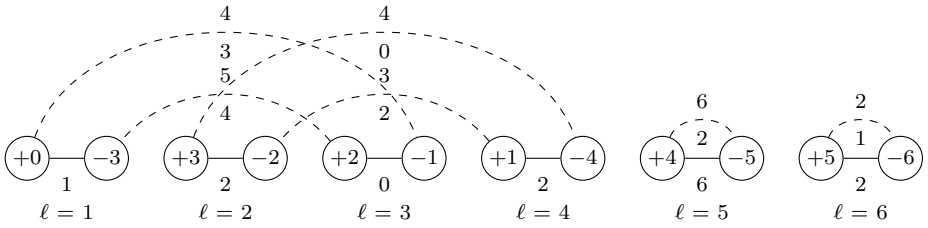
Dada uma instância intergênica flexível balanceada $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, denotamos por $c_d(G(\mathcal{I}))$ o número de ciclos definitivos em $G(\mathcal{I})$. Dada uma sequência de

eventos de rearranjo S, denotamos por $\Delta c_d(G(\mathcal{I}), S) = c_d(G(\mathcal{I}')) - c_d(G(\mathcal{I}))$, tal que $\mathcal{I}' = ((\pi, \breve{\pi}) \cdot S, (\iota, \breve{\iota}))$, a variação no número de ciclos definitivos após a aplicação da sequência S no genoma de origem $(\pi, \breve{\pi})$ de $\mathcal{I}$.

Observação 2.6.5. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ uma instância intergênica flexível balanceada tal que $c(G(\mathcal{I})) = c_d(G(\mathcal{I})) = n + 1$. Então temos que $\pi = \iota$ e $\breve{\iota}_i^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_i^{\max}$ para todo $\breve{\pi}_i \in \breve{\pi}$.

O Exemplo 2.6.4 mostra o grafo de ciclos ponderado flexível construído a partir da instância intergênica flexível $\mathcal{I} = (((+0\ +3\ +2\ +1\ +4\ +5\ +6), (1, 2, 0, 2, 6, 2)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (3, 2, 4, 0, 2, 1), (4, 3, 5, 4, 6, 2)))$.

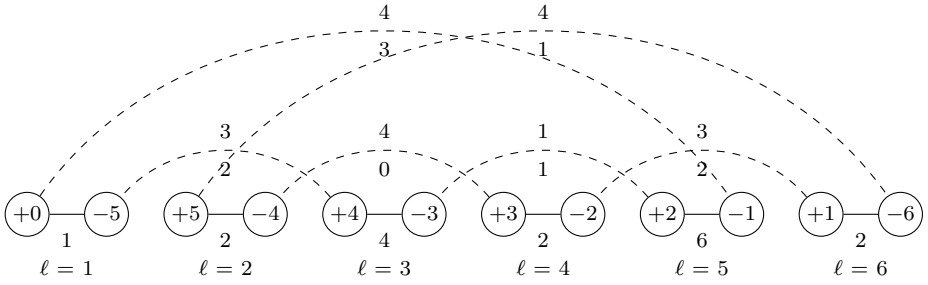
Exemplo 2.6.4.



No Exemplo 2.6.4, $G(\mathcal{I})$ possui quatro ciclos, sendo eles: $C_1 = (3, 1)$, $C_2 = (4, 2)$, $C_3 = (5)$ e $C_4 = (6)$. Além disso, temos os conjuntos $\mathcal{S}_i(G(\mathcal{I})) = \{C_1\}$ e $\mathcal{S}_e(G(\mathcal{I})) = \{C_2, C_3, C_4\}$. Observe que a instância intergênica flexível $\mathcal{I}$ do Exemplo 2.6.4 pertence ao caso fonte, onde apenas o ciclo instável C_1 precisa aumentar o seu peso total para ser transformado em um ciclo estável ($1 = W_p(C_1) < W_c^{\min}(C_1) = 7$). Note que $gap_{\min}(C_2) = 2$, $gap_{\min}(C_3) = 4$ e $gap_{\min}(C_4) = 1$. Portanto, temos que $\mathcal{S}_a(G(\mathcal{I})) = \{C_2, C_3\}$ e $\mathcal{S}_d(G(\mathcal{I})) = \{C_4\}$.

O Exemplo 2.6.5 mostra o grafo de ciclos ponderado flexível construído a partir da instância intergênica flexível $\mathcal{I} = (((+0\ +5\ +4\ +3\ +2\ +1\ +6), (1, 2, 4, 2, 6, 2)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (3, 2, 1, 0, 2, 1), (4, 3, 1, 4, 3, 4)))$.

Exemplo 2.6.5.



No Exemplo 2.6.5, $G(\mathcal{I})$ possui dois ciclos, sendo eles: $C_1 = (5, 3, 1)$ e $C_2 = (6, 4, 2)$. Além disso, temos os conjuntos $\mathcal{S}_i(G(\mathcal{I})) = \{C_1\}$ e $\mathcal{S}_e(G(\mathcal{I})) = \{C_2\}$. Observe que a instância intergênica flexível $\mathcal{I}$ do Exemplo 2.6.5 pertence ao caso sorvedouro, onde apenas o ciclo instável C_1 precisa reduzir o seu peso total para ser transformado em um ciclo estável ($11 = W_p(C_1) > W_c^{\max}(C_1) = 8$). Note que $gap_{\max}(C_2) = 5$. Portanto, temos que $\mathcal{S}_a(G(\mathcal{I})) = \{C_2\}$ e $\mathcal{S}_d(G(\mathcal{I})) = \emptyset$.

Capítulo 3

Modelos com Proporção entre Operações

Os problemas de distância entre genomas podem utilizar uma abordagem *não ponderada*, ou seja, cada evento de rearranjo utilizado para transformar o genoma de origem no genoma alvo contribui em uma unidade para a distância. Essa abordagem tem como característica que cada tipo de evento de rearranjo pertencente ao modelo de rearranjo adotado possui a mesma probabilidade de ocorrer em um cenário evolutivo. Outra abordagem que surgiu para possibilitar uma contribuição diferente para cada evento de rearranjo é chamada de *ponderada*. Nessa abordagem, cada tipo de evento de rearranjo possui um peso associado que é contabilizado na distância evolutiva entre os genomas, onde busca-se minimizar a soma total dos pesos. A abordagem ponderada geralmente é utilizada para mapear um cenário em que queremos que determinados eventos de rearranjo tenham uma probabilidade maior de ocorrer do que outros. Para isso, basta atribuir um peso menor para os eventos de rearranjo mais frequentes. Esses pesos podem ser atribuídos com base em observações empíricas de determinados organismos ou através de análises realizadas especificamente para esse objetivo [8, 41].

Os eventos de rearranjo de reversão e transposição são dois dos eventos mais estudados na literatura [14, 40, 64]. Considerando uma representação clássica e uma abordagem não ponderada, temos o problema de Ordenação de Permutações por Reversões e Transposições (**SbRT**), sendo que o problema possui a variação com e sem sinais. Ambas as variações pertencem à classe NP-difícil de problemas [55]. Para a variação com sinais do problema existe um algoritmo de aproximação com fator 2 [71], enquanto para a variação sem sinais existe um algoritmo de aproximação com fator 2α [62], onde α é o fator de aproximação do algoritmo utilizado para a decomposição em ciclos do grafo de ciclos [30, 31].

Considerando uma abordagem ponderada, temos o problema de Ordenação de Permutações por Reversões e Transposições Ponderadas ($\mathbf{Sb_WRT}$) na variação com e sem sinais. Em 2002, Eriksen [42] apresentou um algoritmo com fator de aproximação $\frac{7}{6}$ para a variação com sinais do problema, utilizando os pesos 1 e 2 para os eventos de reversão e transposição, respectivamente. Oliveira *et al.* [56] desenvolveram um algoritmo de aproximação com fator 1.5 para a variação com sinais do problema $\mathbf{Sb_WRT}$, adotando os pesos 2 e 3 para os eventos de reversão e transposição, respectivamente. Além disso, os

autores mostraram que as variações com e sem sinais do problema $\mathbf{Sb_WRT}$ pertencem à classe NP-difícil quando a razão entre os pesos dos eventos de transposição e reversão é maior ou igual a 1.5.

Em 2007, Bader e Ohlebusch [9] apresentaram o problema de Ordenação de Permutações por Reversões, Transposições e Transposições Inversa Ponderadas ($\mathbf{Sb_WRTIT}$). A *transposição inversa* é um evento similar ao evento de transposição, mas com um dos segmentos afetados sendo invertido. Para a variação com sinais do problema os autores apresentaram um algoritmo de aproximação com fator 1.5, utilizando o peso 1 para o evento de reversão e o mesmo peso para os eventos de transposição e transposição inversa no intervalo [1..2]. Em 2020, Alexandrino *et al.* [5] mostraram que as variações com e sem sinais do problema $\mathbf{Sb_WRTIT}$ pertencem à classe NP-difícil quando os eventos de transposição e transposição inversa possuem o mesmo peso e a razão entre os pesos dos eventos de transposição e reversão é maior ou igual a 1.5.

A abordagem ponderada possui vantagens em comparação com a abordagem não ponderada quando queremos mapear um cenário evolutivo dando maior prioridade para determinados tipos de eventos de rearranjo. Entretanto, ela não garante que os rearranjos de menor peso, supostamente os mais frequentes em um cenário evolutivo, serão realmente os mais utilizados pelos algoritmos. Para contornar esse obstáculo, propomos e investigamos o problema de Ordenação de Permutações por Reversões e Transposições com Restrição de Proporção ($\mathbf{Sb_PRT}$) em instâncias clássicas com e sem sinais. Nesse cenário, buscamos uma sequência de reversões e transposições S, capaz de transformar o genoma de origem no genoma alvo, com uma restrição adicional na qual a relação entre o número de reversões e o tamanho da sequência S deve ser maior ou igual a um determinado parâmetro $k \in [0..1]$.

Observe que tanto a abordagem ponderada como a de proporção tentam incorporar no modelo a frequência com que os eventos de rearranjo afetam o genoma de um determinado organismo. É importante notar que, do ponto de vista biológico, a frequência e o conjunto de eventos de rearranjo podem variar dependendo do organismo considerado. De um ponto de vista teórico, as abordagens possuem objetivos diferentes, apesar de compartilharem finalidades comuns. Uma característica que difere da abordagem de proporção é que, uma vez conhecida a frequência com que os eventos afetam o genoma, a proporção pode ser facilmente derivada dessa informação, enquanto que na abordagem ponderada o peso associado a cada tipo de evento precisa ser ajustado e validado através de testes experimentais.

O Exemplo 3.0.1 mostra uma solução ótima S para a instância clássica com sinais $((+0\ -1\ +4\ -8\ +3\ +5\ +2\ -7\ -6\ +9), (+0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9))$ considerando os problemas $\mathbf{SbRT}$ e $\mathbf{Sb_WRT}$ (utilizando os pesos 2 e 3 para os eventos de reversão e transposição, respectivamente). Note que metade dos eventos de rearranjo de S são reversões e a outra metade transposições, mesmo utilizando um peso maior para o evento de transposição.

Exemplo 3.0.1.

$$\begin{array}{lcll}
\pi & = & & (+0\ \underline{-1\ +4\ -8\ +3\ +5}\ +2\ -7\ -6\ +9) \\
\pi^1 & = & \pi \cdot \rho^{(1,5)} & = (+0\ -5\ \underline{-3\ +8}\ \underline{-4\ +1\ +2\ -7\ -6}\ +9) \\
\pi^2 & = & \pi^1 \cdot \tau^{(2,4,9)} & = (+0\ \underline{-5\ -4}\ \underline{+1\ +2\ -7\ -6}\ -3\ +8\ +9) \\
\pi^3 & = & \pi^2 \cdot \tau^{(1,3,7)} & = (+0\ +1\ +2\ \underline{-7\ -6\ -5\ -4\ -3}\ +8\ +9) \\
\pi^4 & = & \pi^3 \cdot \rho^{(3,7)} & = (+0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9) \\
S & = & (\rho^{(1,5)}, \tau^{(2,4,9)}, \tau^{(1,3,7)}, \rho^{(3,7)}) &
\end{array}$$

O Exemplo 3.0.2 mostra uma solução ótima S' para a mesma instância clássica com sinais apresentada no Exemplo 3.0.1 considerando o problema **Sb$_\mathbf{P}$RT** e adotando um valor de $k = 0.6$, ou seja, pelo menos 60% dos eventos de rearranjo em S' devem ser reversões. Quando comparamos com o Exemplo 3.0.1, podemos perceber que S' possui apenas um evento a mais que S, mas a proporção mínima de reversões em relação ao tamanho da sequência S' é garantida.

Exemplo 3.0.2.

$$\begin{array}{lcll}
\pi & = & & (+0\ -1\ \underline{+4\ -8\ +3\ +5\ +2\ -7\ -6}\ +9) \\
\pi^1 & = & \pi \cdot \rho^{(2,8)} & = (+0\ -1\ \underline{+6\ +7\ -2}\ -5\ -3\ +8\ -4\ +9) \\
\pi^2 & = & \pi^1 \cdot \rho^{(2,4)} & = (+0\ -1\ +2\ -7\ -6\ -5\ \underline{-3\ +8}\ \underline{-4}\ +9) \\
\pi^3 & = & \pi^2 \cdot \tau^{(6,8,9)} & = (+0\ \underline{-1}\ +2\ -7\ -6\ -5\ -4\ -3\ +8\ +9) \\
\pi^4 & = & \pi^3 \cdot \rho^{(1,1)} & = (+0\ +1\ +2\ \underline{-7\ -6\ -5\ -4\ -3}\ +8\ +9) \\
\pi^5 & = & \pi^4 \cdot \rho^{(3,7)} & = (+0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9) \\
S' & = & (\rho^{(2,8)}, \rho^{(2,4)}, \tau^{(6,8,9)}, \rho^{(1,1)}, \rho^{(3,7)}) &
\end{array}$$

Dada uma sequência de eventos de rearranjo S, denotamos por $|S|$ o tamanho da sequência S, ou seja, a quantidade de eventos em S. Além disso, denotamos por $|S_\rho|$ a quantidade de eventos de reversão em S. A seguir, descrevemos formalmente o problema de Ordenação de Permutações por Reversões e Transposições com Restrição de Proporção.

Ordenação de Permutações por Reversões e Transposições com Restrição de Proporção (Sb$_\mathbf{P}$RT)
Entrada: Uma instância clássica com ou sem sinais $\mathcal{I} = (\pi, \iota)$ e um número racional $k \in [0..1]$.
Objetivo: Com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau\}$, determinar uma sequência de eventos de rearranjo S de tamanho mínimo capaz de transformar π em ι, tal que $\frac{|S_\rho|}{|S|} \geq k$.

Dada uma instância clássica com ou sem sinais $\mathcal{I} = (\pi, \iota)$ e um número racional $k \in [0..1]$, a *distância de proporção* entre π e ι, denotada por $dp_k(\mathcal{I})$, é o tamanho da menor sequência de eventos de rearranjo S tal que todo evento de S pertence ao modelo $\mathcal{M} = \{\rho, \tau\}$, $\pi \cdot S = \iota$ e $\frac{|S_\rho|}{|S|} \geq k$. Por praticidade, neste capítulo nos referiremos a um breakpoint clássico simplesmente por breakpoint. Além disso, faremos uso do grafo de ciclos clássico.

Neste capítulo, provaremos que o problema **Sb$_\mathbf{P}$RT** pertence à classe NP-difícil em instâncias clássicas sem sinais para qualquer valor de k. Em instâncias clássicas com sinais,

mostraremos que existe um algoritmo exato polinomial para o problema quando $k = 1$ e provaremos que o problema pertence à classe NP-difícil quando $k < 1$. Para as variações com e sem sinais do problema $\mathbf{Sb_PRT}$, apresentaremos algoritmos de aproximação com fatores $3 - \frac{3k}{2}$ e $3 - k$, respectivamente. Além disso, apresentaremos um algoritmo de aproximação assintótico com um fator teórico melhor para instâncias clássicas com sinais. Por fim, realizaremos experimentos comparando o desempenho prático dos algoritmos propostos.

Os resultados apresentados neste capítulo foram publicados em 2021 na revista *Journal of Bioinformatics and Computational Biology* [18].

3.1 Limitantes Inferiores

Nesta seção, apresentaremos limitantes inferiores para as variações com e sem sinais do problema $\mathbf{Sb_PRT}$. Os limitantes inferiores fazem uso do conceito de breakpoint clássico (Seção 2.4.1) e da estrutura do grafo de ciclos clássico (Seção 2.6.1).

Lema 3.1.1 (Kececioglu e Sankoff [50])**.** *Seja $\mathcal{I}$ uma instância clássica sem sinais. Para qualquer reversão ρ temos que $\Delta b_1(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 3.1.2 (Walter *et al.* [71])**.** *Seja $\mathcal{I}$ uma instância clássica sem sinais. Para qualquer transposição τ temos que $\Delta b_1(\mathcal{I}, S = (\tau)) \geq -3$.*

Lema 3.1.3. *Seja $\mathcal{I}$ uma instância clássica sem sinais para o problema $\boldsymbol{Sb_PRT}$ considerando a proporção $k \in [0..1]$, e seja S uma sequência ótima de eventos de rearranjo para o problema. O número de breakpoints tipo um removidos por cada evento de S, em média, é menor ou igual a $3 - k$.*

Demonstração. Como S é uma sequência ótima para a instância $\mathcal{I}$ com base na proporção k, temos que pelo menos $|S|k$ eventos presentes em S são reversões. Pelos lemas 3.1.1 e 3.1.2, temos que uma reversão pode remover até dois breakpoints tipo um enquanto uma transposição pode remover até três. Seja $\phi b(S)$ o número médio de breakpoints tipo um removidos por um evento de S, temos que:

$$\phi b(S) \leq \frac{(2|S|k) + (3|S|(1-k))}{|S|} = 2k + 3(1-k) = 3 - k.$$

□

Lema 3.1.4 (Hannenhalli e Pevzner [48])**.** *Seja $\mathcal{I}$ uma instância clássica com sinais. Para qualquer reversão ρ temos que $\Delta b_2(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 3.1.5. *Seja $\mathcal{I}$ uma instância clássica com sinais. Para qualquer transposição τ temos que $\Delta b_2(\mathcal{I}, S = (\tau)) \geq -3$.*

Demonstração. Note que uma transposição pode afetar no máximo três breakpoints tipo dois de $\mathcal{I}$. Logo, no melhor cenário, os três breakpoints são removidos e o lema segue. □

Lema 3.1.6. *Seja $\mathcal{I}$ uma instância clássica com sinais para o problema* ***Sb$_P$RT*** *considerando a proporção $k \in [0..1]$, e seja S uma sequência ótima de eventos de rearranjo para o problema. O número de breakpoints tipo dois removidos por cada evento de S, em média, é menor ou igual a $3 - k$.*

Demonstração. A prova é similar à descrita no Lema 3.1.3, mas considerando os lemas 3.1.4 e 3.1.5. □

Lema 3.1.7 (Hannenhalli e Pevzner [48])**.** *Seja $\mathcal{I}$ uma instância clássica com sinais. Para qualquer reversão ρ temos que $\Delta c(G(\mathcal{I}), S = (\rho)) \leq 1$.*

Lema 3.1.8 (Bafna e Pevzner [10]; Walter *et al.* [71])**.** *Seja $\mathcal{I}$ uma instância clássica com ou sem sinais $\mathcal{I}$. Para qualquer transposição τ temos que $\Delta c(G(\mathcal{I}), S = (\tau)) \leq 2$.*

Lema 3.1.9. *Seja $\mathcal{I}$ uma instância clássica com sinais para o problema* ***Sb$_P$RT*** *considerando a proporção $k \in [0..1]$, e seja S uma sequência ótima de eventos de rearranjo para o problema. A variação no número de ciclos para cada evento de S, em média, é menor ou igual a $2 - k$.*

Demonstração. Como S é uma sequência ótima para a instância $\mathcal{I}$ com base na proporção k, temos que pelo menos $|S|k$ eventos presentes em S são reversões. Pelos lemas 3.1.7 e 3.1.8, temos que uma reversão pode criar até um novo ciclo, enquanto uma transposição pode criar até dois. Seja $\phi c(S)$ o número médio de ciclos criados por um evento de S, temos que:

$$\phi c(S) \leq \frac{(1|S|k) + (2|S|(1-k))}{|S|} = 1k + 2(1-k) = 2 - k.$$

□

Teorema 3.1.10. *Seja $\mathcal{I}$ uma instância clássica sem sinais para o problema* ***Sb$_P$RT*** *considerando uma proporção $k \in [0..1]$. Temos que $dp_k(\mathcal{I}) \geq \frac{b_1(\mathcal{I})}{3-k}$.*

Demonstração. Note que, pela Observação 2.4.1, $b_1(\mathcal{I})$ breakpoints tipo um devem ser removidos para transformar π em ι e, pelo Lema 3.1.3, até $3 - k$ breakpoints tipo um são removidos, em média, por cada operação de uma sequência ótima para o problema. Logo, o teorema segue. □

Teorema 3.1.11. *Seja $\mathcal{I}$ uma instância clássica com sinais para o problema* ***Sb$_P$RT*** *considerando uma proporção $k \in [0..1]$. Temos que $dp_k(\mathcal{I}) \geq \frac{b_2(\mathcal{I})}{3-k}$.*

Demonstração. A prova é similar à descrita no Teorema 3.1.10, mas considerando o número de breakpoints tipo dois em $\mathcal{I}$ e o Lema 3.1.6. □

Teorema 3.1.12. *Seja $\mathcal{I}$ uma instância clássica com sinais para o problema* ***Sb$_P$RT*** *considerando uma proporção $k \in [0..1]$. Temos que $dp_k(\mathcal{I}) \geq \frac{n+1-c(G(\mathcal{I}))}{2-k}$.*

Demonstração. Note que, pela Observação 2.6.2, $n + 1 - c(G(\mathcal{I}))$ novos ciclos precisam ser criados para transformar π em ι. Pelo Lema 3.1.9, até $2 - k$ novos ciclos são criados, em média, por cada operação de uma sequência ótima para o problema. Logo, o teorema segue. □

3.2 Análise de Complexidade

Nesta seção, apresentamos uma análise de complexidade do problema **Sb$_\mathbf{P}$RT** em instâncias clássicas com e sem sinais para todos os possíveis valores de k. Os problemas citados e investigados nessa seção referem-se a suas respectivas versões de decisão. A seguir, descrevemos formalmente a versão de decisão do problema **Sb$_\mathbf{P}$RT**.

> **Sb$_\mathbf{P}$RT** (Versão de Decisão)
> **Entrada:** Uma instância clássica com ou sem sinais $\mathcal{I} = (\pi, \iota)$, um número racional $k \in [0..1]$ e um número natural t.
> **Pergunta:** Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau\}$, capaz de transformar π em ι, tal que $\frac{|S_\rho|}{|S|} \geq k$ e $|S| = t$?

Note que para o problema **Sb$_\mathbf{P}$RT** é possível fornecer como entrada diferentes valores para k. Entretanto, a complexidade do problema pode variar dependendo do tipo da instância e do valor adotado para k. Com base nessa característica do problema obtemos os seguintes lemas.

Lema 3.2.1. *O problema **Sb$_P$RT** em instâncias clássicas com sinais pertence à classe NP-difícil quando $k = 0$ e admite um algoritmo exato polinomial quando $k = 1$.*

Demonstração. Quando $k = 0$, o problema **Sb$_\mathbf{P}$RT** em instâncias clássicas com sinais torna-se a variação com sinais do problema **SbRT**, que é NP-difícil [55]. Por outro lado, quando $k = 1$, o problema **Sb$_\mathbf{P}$RT** em instâncias clássicas com sinais torna-se a variação com sinais do problema **SbR**, que possui um algoritmo exato polinomial [48]. □

Lema 3.2.2. *O problema **Sb$_P$RT** em instâncias clássicas sem sinais pertence à classe NP-difícil quando $k \in \{0, 1\}$.*

Demonstração. Quando $k = 0$ e $k = 1$ o problema **Sb$_\mathbf{P}$RT** em instâncias clássicas sem sinais torna-se a variação sem sinais dos problemas **SbRT** e **SbR**, respectivamente. Ambos os problemas pertencem à classe NP-difícil [30, 55]. □

A seguir, investigamos a complexidade do problema **Sb$_\mathbf{P}$RT** quando k pertence ao intervalo $(0..1)$. Para isso, apresentamos definições utilizadas para provar a complexidade do problema para esse intervalo de valores de k. As transformações de *duplicação*, *orientação*, *extensão bridge* e *extensão gadget* descritas a seguir utilizam uma representação clássica de um genoma na sua forma não estendida. Caso a representação esteja na forma estendida, os elementos π_0 e π_{n+1} são ignorados, a transformação é aplicada e a nova representação clássica resultante é então estendida.

Definição 3.2.1. Dada uma representação clássica sem sinais π de tamanho n, a *duplicação* cria uma representação clássica sem sinais π' de tamanho $2n$ de forma que cada elemento $\pi_i \in \pi$ é mapeado em dois novos valores, com $\pi'_{2i-1} = 2\pi_i - 1$ e $\pi'_{2i} = 2\pi_i$, para $i \in [1..n]$.

O Exemplo 3.2.1 mostra a transformação de duplicação aplicada na representação clássica sem sinais $\pi = (4\ 1\ 5\ 3\ 2)$.

Exemplo 3.2.1.

$$\begin{aligned}\pi &= (4\ 1\ 5\ 3\ 2)\\ \pi' &= (7\ 8\ 1\ 2\ 9\ 10\ 5\ 6\ 3\ 4)\end{aligned}$$

Definição 3.2.2. Dada uma representação clássica sem sinais π de tamanho n, a *orientação* cria uma representação clássica com sinais π', também de tamanho n, de forma que $\pi'_i = +\pi_i$, para $i \in [1..n]$.

O Exemplo 3.2.2 mostra a transformação de orientação aplicada na representação clássica sem sinais $\pi = (4\ 1\ 5\ 3\ 2)$.

Exemplo 3.2.2.

$$\begin{aligned}\pi &= (4\ 1\ 5\ 3\ 2)\\ \pi' &= (+4\ +1\ +5\ +3\ +2)\end{aligned}$$

Definição 3.2.3. Dada uma representação clássica com ou sem sinais π de tamanho n, a *extensão bridge* cria uma representação clássica π' de tamanho $n+3$. Caso π seja uma representação com sinais, π' é gerado da seguinte forma: (i) $\pi'_i = \pi_i$ e (ii) $\pi'_{n+j} = +(n+j)$, para $i \in [1..n]$ e $j \in [1..3]$. Caso contrário, π' é gerado da seguinte forma: (i) $\pi'_i = \pi_i$ e (ii) $\pi'_{n+j} = n+j$, para $i \in [1..n]$ e $j \in [1..3]$.

O Exemplo 3.2.3 mostra a transformação de extensão bridge aplicada na representação clássica com sinais $\pi = (+4\ +1\ +5\ -3\ -2)$.

Exemplo 3.2.3.

$$\begin{aligned}\pi &= (+4\ +1\ +5\ -3\ -2)\\ \pi' &= (+4\ +1\ +5\ -3\ -2\ +6\ +7\ +8)\end{aligned}$$

O Exemplo 3.2.4 mostra a transformação de extensão bridge aplicada na representação clássica sem sinais $\pi = (4\ 1\ 5\ 3\ 2)$.

Exemplo 3.2.4.

$$\begin{aligned}\pi &= (4\ 1\ 5\ 3\ 2)\\ \pi' &= (4\ 1\ 5\ 3\ 2\ 6\ 7\ 8)\end{aligned}$$

Definição 3.2.4. Dada uma representação clássica com ou sem sinais π de tamanho n, a *extensão gadget* cria uma representação clássica π' de tamanho $n+6$. Caso π seja uma representação com sinais, π' é gerado da seguinte forma: (i) $\pi'_i = \pi_i$; (ii) $\pi'_{n+j} = -(n+4-j)$; (iii) $\pi'_{n+k} = +(n+k)$, para $i \in [1..n]$, $j \in [1..3]$ e $k \in [4..6]$. Caso contrário, π' é gerado da seguinte forma: (i) $\pi'_i = \pi_i$; (ii) $\pi'_{n+j} = n+4-j$; (iii) $\pi'_{n+k} = n+k$, para $i \in [1..n]$, $j \in [1..3]$ e $k \in [4..6]$.

O Exemplo 3.2.5 mostra a transformação de extensão gadget aplicada na representação clássica com sinais $\pi = (+4\ +1\ +5\ -3\ -2)$.

Exemplo 3.2.5.

$$\begin{aligned}\pi &= (+4\ +1\ +5\ -3\ -2)\\ \pi' &= (+4\ +1\ +5\ -3\ -2\ -8\ -7\ -6\ +9\ +10\ +11)\end{aligned}$$

O Exemplo 3.2.6 mostra a transformação de extensão gadget aplicada na representação clássica sem sinais $\pi = (4\ 1\ 5\ 3\ 2)$.

Exemplo 3.2.6.

$$\begin{aligned} \pi &= (4\ 1\ 5\ 3\ 2) \\ \pi' &= (4\ 1\ 5\ 3\ 2\ 8\ 7\ 6\ 9\ 10\ 11) \end{aligned}$$

A seguir, descrevemos formalmente a versão de decisão do problema de Ordenação de Permutações por 3-Transposições (**B3T**).

B3T (Versão de Decisão)
Entrada: Uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$, tal que $b_2(\mathcal{I}) = 3s$ e s é um número natural positivo.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\tau\}$, capaz de transformar π em ι, tal que $|S| = \frac{b_2(\mathcal{I})}{3}$?

Bulteau e coautores [27] provaram que o problema **B3T** pertence à classe NP-difícil. Utilizaremos uma redução do problema **B3T** para provar que o problema $\mathbf{Sb_PRT}$ é NP-difícil quando k pertence ao intervalo $(0..1)$.

Lema 3.2.3 (Oliveira *et al.* [55])**.** *Se uma instância clássica com sinais $\mathcal{I}$ possui apenas strips positivas, para qualquer reversão ρ temos que $\Delta b_2(\mathcal{I}, S = (\rho)) \geq 0$.*

Teorema 3.2.4. *O problema $\boldsymbol{Sb_PRT}$ em instâncias clássicas com sinais pertence à classe NP-difícil quando $k \in (0..1)$.*

Demonstração. Dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ para o problema **B3T**, definimos $\ell = \frac{b_2(\mathcal{I})}{3} \geq 1$. Criamos uma instância clássica com sinais $\mathcal{I}' = (\pi', \iota')$ para o problema $\mathbf{Sb_PRT}$ da seguinte maneira:

1. Seja σ uma representação clássica com sinais de tamanho $n + 3$ obtida através do processo de orientação aplicado em π, seguido da extensão bridge.

2. Seja k um número racional no intervalo $(0..1)$, definimos $p = \lceil \frac{\ell k}{1-k} \rceil \geq 1$, ou seja, p é o menor número inteiro tal que $\frac{p}{p+\ell} \geq k$.

3. Seja π' uma representação clássica com sinais de tamanho $n+3+6p$ obtida através da aplicação consecutiva de p extensões gadget em σ.

4. Seja ι' uma representação clássica com sinais de tamanho $n+3+6p$. Caso π esteja na sua forma estendida, $\iota'_i = +i$ para $i \in [1..(n+3+6p)]$. Caso contrário, $\iota'_i = +i$ para $i \in [0..(n+3+6p-1)]$.

O Exemplo 3.2.7 mostra o processo de criação de uma instância clássica com sinais $\mathcal{I}' = (\pi', \iota')$ para o problema $\mathbf{Sb_PRT}$ a partir de uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ para o problema **B3T**. Note que, em ambas as instâncias, os genomas de origem e alvo são representados na forma estendida. Além disso, é importante lembrar que o problema **B3T** e a variação com sinais do problema $\mathbf{Sb_PRT}$ utilizam breakpoints tipo dois. Note que a transformação de orientação preserva o número de breakpoints tipo dois, já que adicionamos apenas um sinal positivo aos elementos da permutação. A extensão

bridge também preserva o número de breakpoints tipo dois, já que adiciona apenas três elementos consecutivos ao final da permutação. Por outro lado, cada extensão gadget adiciona dois novos breakpoints tipo dois (ou seja, as extremidades de cada strip negativa), então $b_2(\mathcal{I}') = b_2(\mathcal{I}) + 2p$.

Agora mostramos que a instância $\mathcal{I}$ do problema **B3T** é satisfeita se, e somente se, $dp_k(\mathcal{I}') \leq \ell + p$.

($\Rightarrow$) Suponha que exista uma sequência S com ℓ transposições, tal que $\pi \cdot S = \iota$. Isso significa que cada transposição de S remove exatamente três breakpoints tipo dois de $\mathcal{I}$. Considere a sequência S' como sendo uma cópia de S incluindo p reversões, de forma que cada reversão é aplicada sobre uma strip negativa de $\mathcal{I}'$. Como $\pi'_i = +\pi_i$ para $i \in [1..n]$, cada transposição de S' também remove exatamente três breakpoints tipo dois, restando apenas $2p$ breakpoints tipo dois para serem removidos. Contudo, cada reversão $\rho \in S'$ remove dois breakpoints tipo dois (criados pela extensão gadget). Logo, $|S'| = \ell + p$, $\pi' \cdot S' = \iota'$ e $\frac{|S'_\rho|}{|S'|} = \frac{p}{p+\ell} \geq k$.

($\Leftarrow$) Pelo Teorema 3.1.11, temos que $dp_k(\mathcal{I}') \geq \frac{b_2(\mathcal{I}')}{3-k} = \frac{b_2(\mathcal{I})+2p}{3-k}$. Como temos por construção que $b_2(\mathcal{I}) = 3\ell$ e $\frac{p-1}{\ell+(p-1)} < k \leq \frac{p}{\ell+p}$, segue que $dp_k(\mathcal{I}') > \frac{(\ell+p-1)(3\ell+2p)}{3\ell+2p-2}$. Além disso, $\ell \geq 1$ e $p \geq 1$, então $\frac{3\ell+2p}{3\ell+2p-2} > 1$ e $dp_k(\mathcal{I}') > \ell+p-1$, o que resulta em $dp_k(\mathcal{I}') \geq \ell+p$. Suponha que exista uma sequência de eventos de rearranjo S' de tamanho $\ell + p$, tal que $\pi' \cdot S' = \iota'$ e $\frac{|S'_\rho|}{|S'|} \geq k$.

Como $b_2(\mathcal{I}') = 3\ell + 2p$, então deve existir pelo menos ℓ transposições em S' com cada uma removendo três breakpoints tipo dois. Caso contrário, S' não seria capaz de transformar π' em ι'. Além disso, deve existir no máximo ℓ transposições em S'. Caso contrário, a proporção $\frac{|S'_\rho|}{|S'|}$ não seria satisfeita. Dessa forma, temos que existem ℓ transposições em S' com cada uma removendo três breakpoints tipo dois. Logo, restam $|S'| - \ell = \ell + p - \ell = p$ reversões em S', e cada reversão deve remover dois breakpoints tipo dois. Caso contrário, S' não seria capaz de transformar π' em ι'.

Vamos definir três tipos de elementos em π'. Dizemos que um dado elemento π'_i é (i) original se $i \in [1..n]$; (ii) transitório se $i \in [n+1..n+3]$; e (iii) estendido se $i > n+3$. Como os elementos originais e transitórios são todos positivos, as strips nas primeiras $n+3$ posições são todas positivas. Pelo Lema 3.2.3, nenhuma reversão ρ aplicada nesses elementos remove breakpoints tipo dois, e isto permanece verdadeiro enquanto as transposições afetam apenas os elementos originais.

Como não é aplicada nenhuma reversão aos elementos originais, os 3ℓ breakpoints tipo dois (π'_i, π'_{i+1}), tal que pelo menos π'_i é um elemento original, devem ser removidos por transposições. Dessa forma, S' possui ℓ transposições $\tau^{(i,j,k)}$ de tal maneira que $1 \leq i < j < k \leq n+1$ (ou seja, as transposições afetam apenas os elementos originais).

Os eventos de rearranjo restantes de S', ou seja, as p reversões, devem remover $2p$ breakpoints tipo dois (π'_i, π'_{i+1}), de tal forma que pelo menos π'_{i+1} seja um elemento estendido (ou seja, $i \geq n+3$). A cada iteração, as únicas reversões que removem dois breakpoints tipo dois são aquelas aplicadas nas duas extremidades de uma strip negativa, implicando que cada reversão de S' é aplicada em uma das p strips negativas adicionadas pelas extensões gadget.

Perceba que S' possui ℓ transposições que removem 3ℓ breakpoints tipo dois (π'_i, π'_{i+1}),

tal que $i \leq n$. Seja S uma sequência de transposições criada a partir das transposições de S' mantendo a mesma ordem relativa. Como $\pi'_i = +\pi_i$ para $i \in [1..n]$, $\pi \cdot S = \iota$, e o teorema segue. □

Exemplo 3.2.7. Dada a instância clássica sem sinais $\mathcal{I} = ((0\ 3\ 5\ 1\ 4\ 2\ 6), (0\ 1\ 2\ 3\ 4\ 5\ 6))$ para o problema **B3T**, temos que $b_2(\mathcal{I}) = 6$. Para a criação da instância clássica com sinais $\mathcal{I}' = (\pi', \iota')$ para o problema $\mathbf{Sb_PRT}$ temos, no passo 1, a obtenção da representação clássica com sinais $\sigma = (+0\ +3\ +5\ +1\ +4\ +2\ +6\ +7\ +8\ +9)$. Usando $k = 0.3$, temos que $p = \lceil \frac{2\times 0.3}{1-0.3} \rceil = \lceil \frac{0.6}{0.7} \rceil = 1$ no passo 2. No passo 3, obtemos a representação clássica com sinais $\pi' = (+0\ +3\ +5\ +1\ +4\ +2\ +6\ +7\ +8\ -11\ -10\ -9\ +12\ +13\ +14\ +15)$ após aplicar $p = 1$ extenções gadget em σ. No passo 4, obtemos a representação clássica com sinais $\iota' = (+0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9\ +10\ +11\ +12\ +13\ +14\ +15)$. Note que $b_2(\mathcal{I}') = b_2(\mathcal{I}) + 2p = 6 + 2 = 8$. A sequência $S = (\tau^{(1,3,6)}, \tau^{(2,3,5)})$ é tal que $\pi \cdot S = \iota$ e $|S| = 2 = \frac{b_2(\mathcal{I})}{3} = \ell$, e a sequência $S' = (\tau^{(1,3,6)}, \tau^{(2,3,5)}, \rho^{(9,11)})$, que possui a mesma sequência de transposições de S, é tal que (i) $\pi' \cdot S' = \iota'$; (ii) $\frac{|S'_\rho|}{|S'|} = 0.333 \geq 0.3 = k$; e (iii) $|S'| = 3 = \frac{b_2(\mathcal{I})}{3} + 1 = \ell + p$.

Lema 3.2.5 (Oliveira *et al.* [55])**.** *Se uma instância clássica sem sinais $\mathcal{I}$ possui apenas strips crescentes, para qualquer reversão ρ temos que $\Delta b_1(\mathcal{I}, S = (\rho)) \geq 0$.*

Teorema 3.2.6. *O problema $\boldsymbol{Sb_PRT}$ em instâncias clássicas sem sinais pertence à classe NP-difícil quando $k \in (0..1)$.*

Demonstração. Dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ para o problema **B3T**, definimos $\ell = \frac{b_2(\mathcal{I})}{3} \geq 1$. Criamos uma instância clássica com sinais $\mathcal{I}' = (\pi', \iota')$ para o problema $\mathbf{Sb_PRT}$ da seguinte maneira:

1. Seja σ uma representação clássica sem sinais de tamanho $2n + 3$ obtida através do processo de duplicação aplicado em π, seguido da extenção bridge.

2. Seja k um número racional no intervalo $(0..1)$, definimos $p = \lceil \frac{\ell k}{1-k} \rceil \geq 1$, ou seja, p é o menor número inteiro tal que $\frac{p}{p+\ell} \geq k$.

3. Seja π' uma representação clássica sem sinais de tamanho $2n+3+6p$ obtida através da aplicação consecutiva de p extensões gadget em σ.

4. Seja ι' uma representação clássica com sinais de tamanho $2n+3+6p$. Caso π esteja na sua forma estendida, $\iota'_i = i$ para $i \in [1..(n+3+6p)]$. Caso contrário, $\iota'_i = i$ para $i \in [0..(n+3+6p-1)]$.

O Exemplo 3.2.8 mostra o processo de criação de uma instância clássica sem sinais $\mathcal{I}' = (\pi', \iota')$ para o problema $\mathbf{Sb_PRT}$ a partir de uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ para o problema **B3T**. Note que, em ambas as instâncias, os genomas de origem e alvo são representados na forma estendida. Note que, exceto por σ_0, cada elemento em posições pares de σ é igual ao elemento à sua esquerda mais um. Isto significa que (i) exceto para a primeira e última strip, qualquer outra strip em σ deve ter pelo menos dois elementos, ou seja, não existem singletons, e (ii) cada strip de σ é crescente. Essas observações também

são válidas para as primeiras $2n+3$ posições de π'. Além disso, é importante lembrar que o problema **B3T** utiliza breakpoints tipo dois enquanto a variação sem sinais do problema $\mathbf{Sb_PRT}$ utiliza breakpoints tipo um. Note que (i) para cada breakpoint tipo dois (π_i, π_{i+1}) de $\mathcal{I}$ existe um breakpoint tipo um (π'_{2i}, π'_{2i+1}) em $\mathcal{I}'$ (criado durante a transformação de duplicação), (ii) os pares (π'_{2i-1}, π'_{2i}) não são breakpoints tipo um, para $i \in [1..n]$ e (iii) os pares $(\pi'_{2n+j}, \pi'_{2n+j+1})$ não são breakpoints tipo um, para $j \in [1..3]$. Por outro lado, cada extensão gadget adiciona dois novos breakpoints tipo um (ou seja, as extremidades de cada strip decrescente), então $b_2(\mathcal{I}') = b_1(\mathcal{I}) + 2p$.

Agora mostramos que a instância $\mathcal{I}$ do problema **B3T** é satisfeita se, e somente se, $dp_k(\mathcal{I}') \leq \ell + p$.

($\Rightarrow$) Suponha que exista uma sequência S com ℓ transposições, tal que $\pi \cdot S = \iota$. Isso significa que cada transposição de S remove exatamente três breakpoints tipo dois de $\mathcal{I}$. Considere a sequência S' criada da seguinte forma: (i) para cada transposição $\tau^{(i,j,k)}$ de S, seguindo a ordem relativa, adicione em S' a transposição $\tau^{(2i-1,2j-1,2k-1)}$; (ii) em seguida, adicione p reversões em S', de forma que cada reversão é aplicada sobre uma strip decrescente de $\mathcal{I}'$. Note que cada transposição de S remove três breakpoints tipo dois de $\mathcal{I}$. Como temos que para cada breakpoint tipo dois (π_i, π_{i+1}) em $\mathcal{I}$ temos um breakpoint tipo um (π'_{2i}, π'_{2i+1}) em $\mathcal{I}'$, isso significa que cada transposição de S' remove três breakpoints tipo um de $\mathcal{I}'$. Com isso, restam apenas $2p$ breakpoints tipo um para serem removidos em $\mathcal{I}'$. Contudo, cada reversão $\rho \in S'$ remove dois breakpoints tipo dois (criados pela extensão gadget). Logo, $|S'| = \ell + p$, $\pi' \cdot S' = \iota'$ e $\frac{|S'_\rho|}{|S'|} = \frac{p}{p+\ell} \geq k$.

($\Leftarrow$) Pelo Teorema 3.1.10, temos que $dp_k(\mathcal{I}') \geq \frac{b_1(\mathcal{I}')}{3-k} = \frac{b_2(\mathcal{I})+2p}{3-k}$. Como temos por construção que $b_2(\mathcal{I}) = 3\ell$ e $\frac{p-1}{\ell+(p-1)} < k \leq \frac{p}{\ell+p}$, segue que $dp_k(\mathcal{I}') > \frac{(\ell+p-1)(3\ell+2p)}{3\ell+2p-2}$. Além disso, $\ell \geq 1$ e $p \geq 1$, então $\frac{3\ell+2p}{3\ell+2p-2} > 1$ e $dp_k(\mathcal{I}') > \ell+p-1$, o que resulta em $dp_k(\mathcal{I}') \geq \ell+p$. Suponha que exista uma sequência de eventos de rearranjo S' de tamanho $\ell + p$, tal que $\pi' \cdot S' = \iota'$ e $\frac{|S'_\rho|}{|S'|} \geq k$.

Como $b_1(\mathcal{I}') = 3\ell + 2p$, então deve existir pelo menos ℓ transposições em S' com cada uma removendo três breakpoints tipo um, caso contrário, S' não seria capaz de transformar π' em ι'. Além disso, deve existir no máximo ℓ transposições em S', caso contrário, a proporção $\frac{|S'_\rho|}{|S'|}$ não seria satisfeita. Dessa forma, temos que existem ℓ transposições em S' com cada uma removendo três breakpoints tipo um. Logo, restam $|S'| - \ell = \ell + p - \ell = p$ reversões em S', e cada reversão deve remover dois breakpoints tipo um, caso contrário, S' não seria capaz de transformar π' em ι'.

Vamos definir três tipos de elementos em π'. Dizemos que um dado elemento π'_i é (i) original se $i \in [1..2n]$; (ii) transitório se $i \in [2n+1..2n+3]$; e (iii) estendido se $i > 2n+3$. Como todos elementos originais e transitórios fazem parte de uma strip crescente, pelo Lema 3.2.5, nenhuma reversão ρ aplicada nesses elementos remove breakpoints tipo um, e isto permanece verdadeiro enquanto as transposições afetam breakpoints tipo um entre os elementos originais.

Como não é aplicada nenhuma reversão aos elementos originais, os 3ℓ breakpoints tipo um (π'_i, π'_{i+1}), tal que pelo menos π'_i é um elemento original, devem ser removidos por transposições. Dessa forma, S' possui ℓ transposições $\tau^{(i,j,k)}$ de tal maneira que $1 \leq i < j < k \leq 2n+1$ (ou seja, as transposições afetam apenas os elementos originais).

Os eventos de rearranjo restantes de S', ou seja, as p reversões, devem remover $2p$ breakpoints tipo um (π'_i, π'_{i+1}), de tal forma que pelo menos π'_{i+1} seja um elemento estendido (ou seja, $i \geq 2n+3$). A cada iteração, as únicas reversões que removem dois breakpoints tipo um são aquelas aplicadas nas duas extremidades de uma strip decrescente, implicando que cada reversão de S' é aplicada em uma das p strips decrescentes adicionadas pelas extensões gadget.

Perceba que S' possui ℓ transposições que removem 3ℓ breakpoints tipo um (π'_i, π'_{i+1}), tal que $i \leq 2n$. Seja S uma sequência de transposições criada a partir das transposições de S' da seguinte forma: mantendo a mesma ordem relativa, para cada transposição $\tau^{(i,j,k)}$ de S' adicione em S a transposição $\tau^{(\frac{i+1}{2}, \frac{j+1}{2}, \frac{k+1}{2})}$. Como o mapeamento feito reflete que cada transposição em S remove três breakpoints tipo dois de $\mathcal{I}$, temos que $\pi \cdot S = \iota$, e o teorema segue. □

Exemplo 3.2.8. Dada a instância clássica sem sinais $\mathcal{I} = ((0\ 1\ 3\ 2\ 4\ 5), (0\ 1\ 2\ 3\ 4\ 5))$ para o problema **B3T**, temos que $b_2(\mathcal{I}) = 3$. Para a criação da instância clássica sem sinais $\mathcal{I}' = (\pi', \iota')$ para o problema $\mathbf{Sb_PRT}$ temos, no passo 1, a obtenção da representação clássica sem sinais $\sigma = (0\ 1\ 2\ 5\ 6\ 3\ 4\ 7\ 8\ 9\ 10\ 11\ 12)$. Usando $k = 0.6$, temos que $p = \lceil \frac{1\times 0.6}{1-0.6} \rceil = \lceil \frac{0.6}{0.4} \rceil = 2$ no passo 2. No passo 3, obtemos a representação clássica sem sinais $\pi' = (0\ 1\ 2\ 5\ 6\ 3\ 4\ 7\ 8\ 9\ 10\ 11\ 14\ 13\ 12\ 15\ 16\ 17\ 20\ 19\ 18\ 21\ 22\ 23\ 24)$ após aplicar $p = 2$ extenções gadget em σ. No passo 4, obtemos a representação clássica sem sinais $\iota' = (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21\ 22\ 23\ 24)$. Note que $b_1(\mathcal{I}') = b_2(\mathcal{I}) + 2p = 3 + 4 = 7$. A sequência $S = (\tau^{(2,3,4)})$ é tal que $\pi \cdot S = \iota$ e $|S| = 1 = \frac{b_2(\mathcal{I})}{3} = \ell$, e a sequência $S' = (\tau^{(3,5,7)}, \rho^{(12,14)}, \rho^{(18,20)})$, que possui a mesma quantidade de transposições de S, é tal que (i) $\pi' \cdot S' = \iota'$; (ii) $\frac{|S'_\rho|}{|S'|} = 0.666 \geq 0.6 = k$; e (iii) $|S'| = 3 = \frac{b_2(\mathcal{I})}{3} + 2 = \ell + p$.

3.3 Algoritmos de Aproximação

Nesta seção, apresentamos algoritmos de aproximação para as variações com e sem sinais do problema $\mathbf{Sb_PRT}$.

3.3.1 Instâncias Clássicas sem Sinais

Com base no conceito de breakpoint, apresentamos algoritmos de aproximação com fatores $3 - k$ para o problema $\mathbf{Sb_PRT}$ em instâncias clássicas sem sinais.

Lema 3.3.1 (Kececioglu e Sankoff [50])**.** *Seja $\mathcal{I} = (\pi, \iota)$ uma instância clássica sem sinais. É possível transformar π em ι utilizando no máximo $b_1(\mathcal{I})$ reversões.*

Teorema 3.3.2. *Existe um algoritmo de aproximação com fator $3 - k$ para o problema $\boldsymbol{Sb_PRT}$ em instâncias clássicas sem sinais para uma proporção $k \in [0..1]$.*

Demonstração. Pelo Lema 3.3.1, dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$, é possível transformar π em ι utilizando no máximo $b_1(\mathcal{I})$ reversões. Como somente reversões são utilizadas na sequência de rearranjo S, então a restrição $\frac{|S_\rho|}{|S|} \geq k$ nunca é violada.

Além disso, pelo Teorema 3.1.10, temos que $dp_k(\mathcal{I}) \geq \frac{b_1(\mathcal{I})}{3-k}$. Logo, $\frac{b_1(\mathcal{I})}{\frac{b_1(\mathcal{I})}{3-k}} = 3-k$, e o teorema segue. □

Note que o algoritmo de aproximação resultante do Teorema 3.3.2 utiliza somente reversões. Para evitar que as soluções sejam compostas exclusivamente por reversões, nós propomos o Algoritmo 1. Esse algoritmo também garante o fator de aproximação $3-k$ para instâncias clássicas sem sinais do problema **Sb$_\mathbf{P}$RT** e para qualquer valor de k. Além disso, a proporção entre a quantidade de reversões e o tamanho da sequência de eventos de rearranjo fornecida pelo algoritmo tende a ser um valor próximo de k.

Algoritmo 1: Um algoritmo de aproximação para o problema **Sb$_\mathbf{P}$RT** em instâncias clássicas sem sinais e para $k \in [0..1]$.

Entrada: Uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ e um valor de $k \in [0..1]$
Saída: Uma sequência de reversões e transposições S, tal que $\pi \cdot S = \iota$ e $\frac{|S_\rho|}{|S|} \geq k$

```
1  Seja S ← ()
2  enquanto π ≠ ι faça
3      se |S_ρ|/(|S|+1) ≥ k e existe uma transposição τ tal que Δb1(I,(τ)) ≤ −1 então
4          π ← π · τ
5          S ← S + (τ)
6      senão
7          Seja S' uma sequência de reversões (de tamanho um ou dois) que remove, na
           média, um breakpoint tipo um por operação [50]
8          π ← π · S'
9          S ← S + S'
10
11 retorne S
```

Observe que o Algoritmo 1 aplica uma transposição τ se duas restrições forem satisfeitas: (i) $\frac{|S_\rho|}{|S|+1} \geq k$, que garante que a sequência de eventos de rearranjo S construída pelo algoritmo obedecerá à restrição do problema que $\frac{|S_\rho|}{|S|} \geq k$; e (ii) $\Delta b_1(\mathcal{I}, (\tau)) \leq -1$, que garante que a sequência de ordenação conterá, no máximo, $b_1(\mathcal{I})$ operações, pois cada sequência de reversões adicionada da sequência S remove, em média, um ou mais breakpoints tipo um por operação. Como o Algoritmo 1 remove, em média, um ou mais breakpoints tipo um por iteração, ele garante que π será transformada em ι. Além disso, não mais do que $b_1(\mathcal{I})$ operações serão usadas para isso, mantendo o fator de aproximação $3-k$. Como a transposição τ (linhas 3-5) e a sequência de no máximo duas reversões S' (linhas 6-9) podem ser encontradas em tempo linear, o tempo de execução do Algoritmo 1 é $\mathcal{O}(n^2)$, considerando que $|S| \leq b_1(\mathcal{I}) \leq n+1$.

3.3.2 Instâncias Clássicas com Sinais

Com base na estrutura de grafo de ciclos clássico, apresentamos um algoritmo de aproximação com fator $3 - \frac{3k}{2}$ para o problema **Sb$_\mathbf{P}$RT** em instâncias clássicas com sinais.

Lema 3.3.3. *Seja $\mathcal{I} = (\pi, \iota)$ uma instância clássica com sinais. Existe uma sequência de reversões S que transforma π em ι e que o número de ciclos criados por cada reversão, em média, é maior ou igual a $\frac{2}{3}$.*

Demonstração. Se $G(\mathcal{I})$ possuir um ciclo divergente C, então existe uma reversão que, quando aplicada, aumenta o número de ciclos em uma unidade (Teorema 5 de [71]). Caso contrário, todos os ciclos não triviais devem ser convergentes. Isso significa que um dos seguintes cenários deve ocorrer obrigatoriamente [56]:

- Existe em $G(\mathcal{I})$ um ciclo longo e orientado C (Figura 3.1, Caso 1);
- Existe em $G(\mathcal{I})$ um ciclo curto C tal que os *open gates* são fechados por outro ciclo não trivial D (Figura 3.1, Caso 2);
- Existe em $G(\mathcal{I})$ um ciclo longo não orientado C tal que os *open gates* são fechados por um ou mais ciclos não triviais (Figura 3.1, Caso 3);

Se $G(\mathcal{I})$ possui um ciclo longo e orientado C, então podemos aplicar uma reversão em suas arestas pretas de maneira que C é transformado em divergente. Como C é um ciclo longo, então é possível aplicar, pelo menos, duas reversões de forma que cada uma aumenta o número de ciclos em uma unidade (Figura 3.1, Caso 1).

Se algum dos outros casos ocorrer, então podemos tornar o ciclo C divergente após aplicar uma reversão no(s) ciclo(s) que fecham os open gates de C. Se C for um ciclo curto, então podemos aplicar uma reversão em suas arestas pretas quebrando-o em dois ciclos triviais, o que aumenta o número de ciclos em uma unidade. Como resultado da segunda reversão, o ciclo D também passa a ser divergente, o que nos garante aplicar uma terceira reversão que aumenta o número de ciclos em uma unidade (Figura 3.1, Caso 2). Se C for um ciclo longo, então é possível aplicar pelo menos duas reversões de forma que cada uma aumenta o número de ciclos em uma unidade (Figura 3.1, Caso 3).

Nos três casos mencionados acima, aplicamos três reversões que aumentam em pelo menos duas unidades o número de ciclos, e o lema segue. □

Lema 3.3.4. *Seja $\mathcal{I} = (\pi, \iota)$ uma instância clássica com sinais. É possível transformar π em ι utilizando no máximo $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ reversões.*

Demonstração. O Lema 3.3.3 resulta em uma sequência de reversões que sempre aumenta o número de ciclos. Logo, podemos aplicar o Lema 3.3.3 até que $c(G(\mathcal{I}))$ seja igual a $n+1$. Consequentemente, π será transformada em ι. Além disso, cada sequência de reversões S obtidas através do Lema 3.3.3 garante que o número de ciclos criados por cada reversão de S é, em média, maior ou igual a $\frac{2}{3}$. Logo, não mais do que $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ reversões são utilizadas para transformar π em ι, e o lema segue. □

Teorema 3.3.5. *Existe um algoritmo de aproximação com fator $3 - \frac{3k}{2}$ para o problema **Sb$_P$RT** em instâncias clássicas com sinais para uma proporção $k \in [0..1]$.*

Demonstração. Pelo Lema 3.3.4, dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$, é possível transformar π em ι utilizando, no máximo, $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ reversões. Como somente

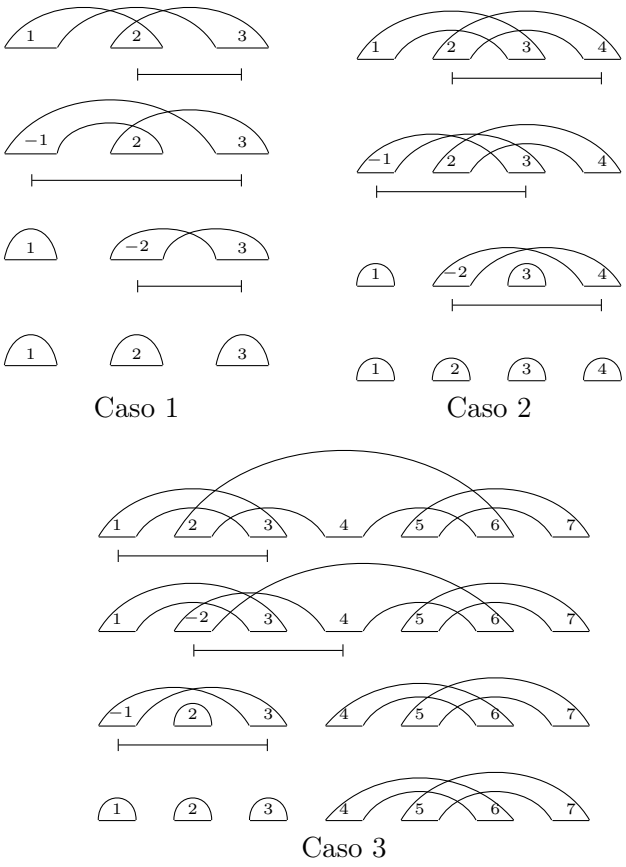


Figura 3.1: Configurações e respectivas sequências de reversões aplicadas em cada um dos casos do Lema 3.3.4. Indicamos, para cada um dos casos, o par de arestas pretas afetadas por cada reversão. No Caso 1, o ciclo $C = (3, 1, 2)$ é longo e orientado. No Caso 2, o ciclo curto $C = (3, 1)$ tem os open gates fechados pelo ciclo $D = (4, 2)$. Por fim, no Caso 3, o ciclo longo não orientado $C = (6, 4, 2)$ tem os open gates fechados pelos ciclos $D = (3, 1)$ e $E = (7, 5)$. Para os três casos, é mostrada uma sequência de três reversões que aumenta o número de ciclos em duas unidades.

reversões são utilizadas na sequência de rearranjo S, então a restrição $\frac{|S_\rho|}{|S|} \geq k$ nunca é violada. Além disso, pelo Teorema 3.1.12, temos que $dp_k(\mathcal{I}) \geq \frac{n+1-c(G(\mathcal{I}))}{2-k}$. Logo,

$$\frac{\frac{3(n+1-c(G(\mathcal{I})))}{2}}{\frac{n+1-c(G(\mathcal{I}))}{2-k}} = 3 - \frac{3k}{2}.$$

□

Note que o algoritmo de aproximação resultante do Teorema 3.3.5 aplica somente reversões. Para evitar que as soluções sejam compostas exclusivamente por reversões, nós propomos o Algoritmo 2. Esse algoritmo também garante um fator de aproximação de $3 - \frac{3k}{2}$ para instâncias clássicas com sinais do problema $\mathbf{Sb_PRT}$ e para qualquer valor de k.

Observe que o Algoritmo 2 aplica uma transposição τ se duas restrições forem satisfeitas: (i) $\frac{|S_\rho|}{|S|+1} \geq k$, que garante que a sequência de eventos de rearranjo S construída

Algoritmo 2: Um algoritmo de aproximação para o problema $\mathbf{Sb_PRT}$ em instâncias clássicas com sinais e $k \in [0..1]$.

Entrada: Uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ e um valor de $k \in [0..1]$
Saída: Uma sequência de reversões e transposições S, tal que $\pi \cdot S = \iota$ e $\frac{|S_\rho|}{|S|} \geq k$

1 Seja $S \leftarrow ()$
2 **enquanto** $\pi \neq \iota$ **faça**
3 **se** $\frac{|S_\rho|}{|S|+1} \geq k$ *e existe uma transposição* τ *tal que* $\Delta c(G(\mathcal{I}), (\tau)) \geq 1$ **então**
4 $\pi \leftarrow \pi \cdot \tau$
5 $S \leftarrow S + (\tau)$
6 **senão**
7 Seja S' uma sequência de reversões (de tamanho no máximo três), onde cada operação aumenta, em média, o número de ciclos em pelo menos 2/3 unidade (Lema 3.3.3)
8 $\pi \leftarrow \pi \cdot S'$
9 $S \leftarrow S + S'$
10
11 **retorne** S

pelo algoritmo obedecerá à restrição do problema que $\frac{|S_\rho|}{|S|} \geq k$; e (ii) $\Delta c(G(\mathcal{I}), (\tau)) \geq 1$, que garante que a sequência de ordenação conterá, no máximo, $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ operações, pois cada sequência de reversões adicionada à sequência S aumenta, em média, o número de ciclos em pelo menos 2/3 unidade. Dessa forma, o algoritmo garante o fator de aproximação de $3 - \frac{3k}{2}$. A transposição τ (linhas 3-5) pode ser encontrada em tempo linear, já a sequência de no máximo três reversões S' (linhas 6-9) pode ser encontrada em tempo $\mathcal{O}(n^2)$. Com isso, o tempo de execução do Algoritmo 2 é $\mathcal{O}(n^3)$, considerando que $|S| \leq \frac{3(n+1-c(G(\mathcal{I})))}{2} \leq \frac{3(n+1)}{2}$.

Algoritmo de Aproximação Assintótica

Nesta seção, apresentamos um algoritmo de aproximação assintótica com fator $\frac{2-k}{1-k/3}$ para o problema $\mathbf{Sb_PRT}$ em instâncias clássicas com sinais.

Definição 3.3.1. Seja $\mathcal{I} = (\pi, \iota)$ uma instância clássica com sinais e seja $\mathcal{A}_\rho$ o algoritmo descrito no Teorema 3.3.5 que transforma π em ι utilizando, no máximo, $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ reversões. Denotamos por $\mathcal{A}_\rho(\mathcal{I})$ a sequência de reversões obtidas através de $\mathcal{A}_\rho$ e que transforma π em ι.

Note que o algoritmo exato para o problema **SbR** em instâncias clássicas com sinais [48], com complexidade subquadrática ($\mathcal{O}(n^{\frac{3}{2}}\sqrt{(\log n)})$ [69]) para obter uma sequência ótima de reversões e com complexidade linear caso apenas a distância seja desejada [6], também pode ser utilizado como o algoritmo $\mathcal{A}_\rho$, uma vez que, para uma mesma instância, o tamanho da sequência de reversões obtida por tal algoritmo é menor ou igual ao tamanho da sequência de reversões dada por um algoritmo de acordo com o Teorema 3.3.5. Consequentemente, no máximo $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ reversões serão utilizadas.

Observação 3.3.1 (Oliveira *et al.* [56]). Dada uma representação clássica π, uma transposição $\tau^{(i,j,k)}$ pode ser reproduzida por uma sequência de três reversões consecutivas $S = (\rho^{(i,j-1)}, \rho^{(j,k-1)}, \rho^{(i,k-1)})$, ou seja, $\pi \cdot \tau^{(i,j,k)} = \pi \cdot S$.

Agora considere o Algoritmo 3. Note que podemos fazer uma análise considerando quatro sub-rotinas: (i) executar o algoritmo $\mathcal{A}_\rho$ (linhas 2 e 12, tempo de execução subquadrático), (ii) encontrar um ciclo divergente em $G(\mathcal{I})$ e determinar os parâmetros da reversão ρ que aumenta o número de ciclos em uma unidade (linhas 3-6, tempo de execução linear), (iii) determinar uma sequência de, no máximo, duas transposições que aumenta o número de ciclos em duas unidades (linhas 7-10, tempo de execução $\mathcal{O}(n^2)$) e (iv) substituir até duas transposições de S por uma sequência equivalente de reversões (linhas 13-14, tempo constante). Considerando que $|S| \leq n + 1$, o tempo de execução do Algoritmo 3 é $\mathcal{O}(n^3)$.

Algoritmo 3: Um algoritmo de aproximação assintótica para o problema $\mathbf{Sb_PRT}$ em instâncias clássicas com sinais e $k \in [0..1]$.

Entrada: Uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ e um valor de $k \in [0..1]$
Saída: Uma sequência de reversões e transposições S, tal que $\pi \cdot S = \iota$ e $\frac{|S_\rho|}{|S|} \geq k$

```
1  Seja S ← ()
2  enquanto |A_ρ(I)| > k(|S| + |A_ρ(I)|) faça
3      se em G(I) existe um ciclo divergente então
4          Seja ρ uma reversão que aumenta o número de ciclos em uma unidade
           (Teorema 5 de [71])
5          I = (π · ρ, ι)
6          S ← S + (ρ)
7      senão
8          Seja S' uma sequência de, no máximo, duas transposições que aumenta o
           número de ciclos em duas unidades (Teorema 3.4 de [12])
9          I = (π · S', ι)
10         S ← S + S'
11
12 S ← S + A_ρ(I)
13 se |S_ρ| < k|S| então
14     Substitua até duas transposições de S por uma sequência equivalente de
       reversões (Observação 3.3.1)
15 retorne S
```

Lema 3.3.6. *Seja $\mathcal{I} = (\pi, \iota)$ uma instância clássica com sinais e um valor $k \in [0..1]$. O Algoritmo 3 fornece uma sequência de operações S com, no máximo, $(n+1-c(G(\mathcal{I})))/(1-k/3)+4$ operações de reversões e transposições, tal que $\pi \cdot S = \iota$ e $\frac{|S_\rho|}{|S|} \geq k$.*

Demonstração. Note que a sequência fornecida pelo algoritmo $\mathcal{A}_\rho$ (linha 12) transforma π em ι. Consequentemente, o Algoritmo 3 também transforma π em ι. Seja S a sequência de operações gerada pelo Algoritmo 3 sem considerar a substituição de transposições por reversões feita na linha 14. Seja S' a subsequência de S criada durante o laço de repetição das linha 2 até 10. Note que, em média, cada operação em S' aumenta o

número de ciclos em pelo menos uma unidade. Além disso, em média, cada operação em $S \setminus S'$ (ou seja, as reversões utilizadas pelo algoritmo $\mathcal{A}_\rho$ na linha 12) aumenta o número de ciclos em pelo menos 2/3 unidade. Pela condição na linha 2, temos que $|S'| \geq (1-k)|S|$. Além disso, em média, cada operação de S aumenta o número de ciclos, em pelo menos, $\frac{(1-k)|S|+k|S|2/3}{|S|} = 1 - k/3$. Como para transformar π em ι é necessário aumentar o número de ciclos em $n + 1 - c(G(\mathcal{I}))$ unidades, temos que $|S| \leq \frac{n+1-c(G(\mathcal{I}))}{1-k/3}$. Note que a sequência S pode não satisfazer a restrição $\frac{|S_\rho|}{|S|} \geq k$. Caso isso ocorra, sabemos que o Algoritmo 3 adiciona transposições em S somente enquanto a condição da linha 2 for satisfeita e que, no máximo, duas transposições são adicionadas por iteração. No pior caso, garantimos que $\frac{|S_\rho|}{|S|} \geq k$ substituindo até duas transposições de S por seis reversões. Logo, $|S| \leq \frac{n+1-c(G(\mathcal{I}))}{1-k/3} + 4$ e o lema segue. □

Teorema 3.3.7. *O Algoritmo 3 é uma $\frac{2-k}{1-k/3}$-aproximação assintótica para o problema* ***Sb$_P$RT*** *em instâncias clássicas com sinais para uma proporção $k \in [0..1]$.*

Demonstração. Pelo Lema 3.3.6 e Teorema 3.1.12, dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ e um valor de $k \in [0..1]$, temos que a sequência de operações S obtida através do Algoritmo 3 satisfaz as seguintes condições: $\pi \cdot S = \iota$, $\frac{|S_\rho|}{|S|} \geq k$ e $|S| \leq (n + 1 - c(G(\mathcal{I})))/(1 - k/3) + 4 = \frac{2-k}{1-k/3} dp_k(\mathcal{I}) + 4$. Logo, o teorema segue. □

3.4 Resultados Práticos

Nesta seção, apresentamos os experimentos práticos e os resultados obtidos. Inicialmente, descrevemos os algoritmos utilizados como *baseline*, bem como as modificações realizadas para garantir que suas soluções sejam válidas para o problema **Sb$_P$RT**, ou seja, para garantir que a restrição de proporção seja satisfeita. Em seguida, apresentamos as bases de dados desenvolvidas e utilizadas como entrada pelos algoritmos. Por fim, discutimos os resultados.

3.4.1 Algoritmos Comparados

Para fins de comparação com os resultados fornecidos por nossos algoritmos, usamos seis algoritmos da literatura como baselines. Os algoritmos que descreveremos a seguir foram desenvolvidos para problemas específicos que não consideram a restrição de proporção entre os eventos de rearranjo e podem fornecer soluções inviáveis para o problema **Sb$_P$RT**. Para garantir que todas as soluções sejam viáveis, quando necessário, ajustamos a sequência de eventos de rearranjo substituindo transposições por reversões para atingir a proporção mínima dada como entrada. A seguir, apresentamos os algoritmos de baseline e o processo de modificação realizado.

- Instâncias clássicas sem sinais:
  - UR: Algoritmo de aproximação com fator 2 para o problema de Ordenação de Permutações por Reversões [50].

  - UT: Algoritmo de aproximação com fator 1.5 para o problema de Ordenação de Permutações por Transposições [12].
  - URT: Algoritmo de aproximação com fator 2α para o problema de Ordenação de Permutações por Reversões e Transposições [62], onde α é o fator de aproximação do algoritmo utilizado para a decomposição de ciclos do Grafo de Ciclos (valor adotado $\alpha = 1.4193 + \epsilon$ [52]).

- Instâncias clássicas com sinais:

  - SR: Algoritmo exato e polinomial para o problema de Ordenação de Permutações por Reversões [48].
  - SRT: Algoritmo de aproximação com fator 2 para o problema de Ordenação de Permutações por Reversões e Transposições [71].
  - SWRT: Algoritmo de aproximação com fator 1.5 para o problema de Ordenação de Permutações por Reversões e Transposições Ponderadas [56] (adotando os pesos 2 e 3 para os eventos de reversão e transposição, respectivamente).

O processo de modificação realizado na sequência de eventos de rearranjo para satisfazer a restrição de proporção mínima difere entre instâncias clássicas com e sem sinais. No caso de uma instância clássica com sinais, enquanto a proporção mínima não for atingida, uma transposição é substituída por uma sequência de três reversões seguindo o processo descrito na Observação 3.3.1. No caso de uma instância clássica sem sinais, esse processo segue regras para evitar o crescimento desnecessário da sequência S gerada pelos algoritmos de baseline: (i) se houver uma transposição $\tau^{(i,j,k)}$ tal que $k - i = 2$, então a substituição é realizada apenas por uma reversão $\rho^{(i,k-1)}$; (ii) se houver uma transposição $\tau^{(i,j,k)}$ tal que $j - i = 1$ ou $k - j = 1$, então a substituição é realizada por uma sequência de duas reversões $S = (\rho^{(i,k-1)}, \rho^{(i,k-2)})$ ou $S = (\rho^{(i,k-1)}, \rho^{(i+1,k-1)})$; e caso contrário, (iii) uma transposição é substituída por uma sequência de três reversões seguindo o processo descrito na Observação 3.3.1. Este processo se repete enquanto a proporção mínima não é atingida, seguindo a ordem das regras de substituição.

3.4.2 Base de Dados

Para verificar o desempenho dos algoritmos em diferentes cenários, criamos bases de dados de instâncias clássicas para simular cenários com proporções fixas entre eventos de reversão e transposição:

- DB1: Esta base de dados é dividida em grupos. Cada grupo tem um total de 10000 instâncias clássicas de tamanho 200 (ou seja, π e ι tem 200 elementos cada) e é identificado pela proporção k adotada para criar as instâncias. Uma sequência com 40 operações é gerada de forma que seja composta por $40k$ de reversões e $40(1 - k)$ de transposições. Os índices das reversões e transposições geradas são escolhidos aleatoriamente entre os valores possíveis. Em seguida, a sequência de operações é embaralhada e aplicada na permutação identidade ι. A permutação resultante π, a permutação identidade ι e a proporção k, formam uma instância do

grupo. Esse processo é repetido até que o grupo tenha um total de 10000 instâncias. As proporções utilizadas variaram de 0 a 1, em intervalos de 0.1, totalizando 11 grupos. Essa base de dados tem as versões com instâncias clássicas com e sem sinais. Considerando instâncias clássicas com e sem sinais, essa base de dados possui um total de 220000 instâncias.

- DB2: Esta base de dados foi desenvolvida para refletir cenários onde o número de reversões é 50% maior que o número de transposições. Assim, no processo de criação das instâncias, foi mantida uma proporção de $k = 0.6$. A base de dados é dividida em grupos com 10000 instâncias cada. Além disso, o identificador do grupo indica o tamanho das instâncias contidas nele e o número de operações utilizadas para criar as instâncias. Os tamanhos usados para as instâncias foram 100, 200, 300, 400 e 500. O número de operações foi baseado em uma porcentagem do tamanho da instância, sendo adotados: 10%, 20%, 30%, 40% e 50%. As etapas finais do processo são semelhantes ao que descrevemos anteriormente na base de dados DB1. Essa base de dados possui uma versão apenas para instâncias clássicas com sinais e um total de 250000 instâncias.

3.4.3 Comparação dos Algoritmos

Nesta seção, apresentamos os resultados fornecidos pelos algoritmos utilizando as bases de dados DB1 e DB2. Nas tabelas 3.1, 3.2, 3.3 e 3.4, as colunas Min, Avg e Max representam valores de mínimo, média e máximo, respectivamente. As colunas Proporção, Distância e Aproximação indicam a proporção obtida nas soluções, a estimativa para a distância de proporção fornecida pelos algoritmos e o fator de aproximação computado com base nos limitantes inferiores, respectivamente.

O objetivo principal dos testes experimentais é a análise do desempenho prático dos algoritmos propostos, comparando-os com as aproximações teóricas e com resultados fornecidos por outros algoritmos da literatura. As tabelas 3.1 e 3.2 mostram os resultados dos algoritmos considerando diferentes cenários de proporção, o que é útil para estudar o comportamento dos algoritmos variando a proporção desejada. As siglas UPRT e SPRT referem-se aos algoritmos 1 e 2, respectivamente. O Algoritmo 2 foi utilizado, ao invés do Algoritmo 3, por possuir uma característica de decisão semelhante a do Algoritmo 1, diferenciando-se apenas pela estrutura considerada (ciclos ao invés de breakpoints). Dessa forma, é possível obter um comparativo mais justo entre esses dois algoritmos.

A Tabela 3.1 mostra os resultados dos algoritmos UR, UT, URT e UPRT aplicados em instâncias sem sinais da base de dados DB1. Algumas soluções fornecidas por UT e URT foram modificadas seguindo o processo descrito na Seção 3.4.1 para ajustar a proporção mínima entre a quantidade de reversões e tamanho da sequência de rearranjo. Considerando todas as instâncias da base de dados, um total de 90.90% e 34.39% das soluções fornecidas por UT e URT, respectivamente, foram modificadas. A razão de aproximação foi calculada adotando-se o limite inferior apresentado no Teorema 3.1.10.

Pela Tabela 3.1, podemos ver que UR apresenta uma razão média de aproximação maior para os menores valores de k. No entanto, à medida que o valor de k aumenta, a razão de aproximação média tende a diminuir. A partir dos resultados práticos de UR, é

Tabela 3.1: Resultados dos algoritmos em instâncias clássicas sem sinais da base de dados DB1.

UR									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	1.000	1.000	1.000	69	84.746	99	2.47	2.77	3.00
0.1	1.000	1.000	1.000	68	81.358	94	2.31	2.63	2.90
0.2	1.000	1.000	1.000	63	78.096	91	2.13	2.49	2.78
0.3	1.000	1.000	1.000	62	74.598	90	2.09	2.37	2.67
0.4	1.000	1.000	1.000	58	71.054	85	1.91	2.24	2.53
0.5	1.000	1.000	1.000	55	67.264	79	1.76	2.10	2.42
0.6	1.000	1.000	1.000	50	63.265	74	1.65	1.96	2.27
0.7	1.000	1.000	1.000	49	59.089	70	1.46	1.81	2.13
0.8	1.000	1.000	1.000	45	54.682	67	1.32	1.66	2.03
0.9	1.000	1.000	1.000	40	50.249	64	1.24	1.52	1.93
1.0	1.000	1.000	1.000	38	45.613	56	1.11	1.37	1.74

UT									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	0.000	0.000	0.000	35	40.475	45	1.14	1.33	1.56
0.1	0.100	0.117	0.156	42	80.488	125	1.27	2.60	4.24
0.2	0.200	0.214	0.241	51	88.408	126	1.72	2.83	4.20
0.3	0.300	0.312	0.333	61	96.265	140	1.94	3.06	4.67
0.4	0.400	0.410	0.429	65	104.689	147	2.13	3.30	4.93
0.5	0.500	0.507	0.519	75	115.007	155	2.39	3.60	5.12
0.6	0.600	0.606	0.619	78	127.210	170	2.52	3.94	6.00
0.7	0.700	0.705	0.716	87	142.344	211	2.75	4.36	6.56
0.8	0.800	0.804	0.811	103	161.781	225	3.15	4.92	7.50
0.9	0.900	0.903	0.909	119	187.906	257	3.56	5.67	8.76
1.0	1.000	1.000	1.000	118	223.210	328	3.47	6.71	10.36

URT									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	0.188	0.642	0.919	43	59.872	78	1.37	1.96	2.57
0.1	0.267	0.651	0.899	44	57.611	73	1.36	1.86	2.45
0.2	0.222	0.657	0.901	40	55.277	71	1.22	1.77	2.38
0.3	0.306	0.660	0.909	39	52.658	67	1.22	1.68	2.20
0.4	0.400	0.663	0.905	39	49.953	64	1.20	1.58	2.10
0.5	0.500	0.667	0.933	36	47.097	62	1.14	1.47	2.04
0.6	0.600	0.683	0.930	37	44.624	57	1.09	1.38	1.84
0.7	0.700	0.732	0.925	37	43.719	57	1.08	1.34	1.78
0.8	0.800	0.815	0.918	37	44.750	60	1.08	1.36	1.88
0.9	0.900	0.912	0.977	38	47.017	64	1.05	1.42	1.97
1.0	1.000	1.000	1.000	39	48.809	75	1.08	1.47	2.27

UPRT									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	0.024	0.251	0.491	38	46.595	57	1.21	1.54	1.92
0.1	0.143	0.345	0.590	38	47.496	61	1.26	1.56	1.94
0.2	0.200	0.385	0.596	38	46.415	56	1.25	1.51	1.86
0.3	0.302	0.444	0.660	39	45.713	56	1.22	1.48	1.80
0.4	0.400	0.510	0.667	38	45.114	53	1.17	1.44	1.72
0.5	0.500	0.579	0.723	38	44.385	53	1.15	1.41	1.75
0.6	0.600	0.662	0.787	37	43.976	52	1.14	1.38	1.73
0.7	0.700	0.744	0.844	37	43.302	52	1.14	1.35	1.68
0.8	0.800	0.829	0.889	37	42.536	50	1.07	1.31	1.72
0.9	0.900	0.918	0.956	36	41.857	50	1.06	1.28	1.69
1.0	1.000	1.000	1.000	36	40.818	49	1.07	1.24	1.74

possível notar que a razão de aproximação média é muitas vezes melhor do que o fator de aproximação teórico de $3 - k$ provado para o problema (Teorema 3.3.2).

Analisando os resultados fornecidos por UT, é possível notar um comportamento oposto ao de UR, com a razão de aproximação média aumentando à medida que o valor de k aumenta. A razão de aproximação média foi menor que três apenas nos grupos em que k é menor ou igual a 0.2, e a razão de aproximação média no grupo em que $k = 1$ foi de 6.71. Vale ressaltar que todas as soluções fornecidas por UT para grupos em que $0.1 \leq k \leq 1.0$ foram modificadas para se adequarem à proporção mínima exigida pelo problema $\mathbf{Sb_PRT}$.

Considerando os grupos onde $k \geq 0.7$, a distância máxima fornecida por UT foi superior a cinco vezes o número de eventos utilizados para criar as instâncias (40 operações). Vale notar que, para esses grupos, a maior parte da sequência de eventos utilizada para construir as instâncias é composta de reversões. Além disso, para replicar o efeito causado por um evento de reversão pode ser necessário utilizar várias transposições. Essa é um possível explicação para os valores elevados para a métrica de distância observada nesses grupos adotando o algoritmo UT, que utiliza apenas transposições, juntamente com a modificação das soluções para adequarem-se ao parâmetro k exigido em cada grupo.

Considerando os resultados de URT, podemos observar que a razão média de aproximação foi menor ou igual a 1.96 para todos os grupos. Comparado com UR e UT, o algoritmo URT apresentou melhores resultados para a aproximação média para grupos onde $0.0 < k < 1.0$. Os algoritmos UR e UT apresentaram melhores resultados quando $k = 1.0$ e $k = 0.0$, respectivamente. O desempenho superior de UR e UT nesses cenários particulares ocorre porque a sequência de ordenação, composta apenas por reversões, encaixa-se perfeitamente no caso em que $k = 1.0$ e, quando $k = 0.0$, uma sequência de transposições não sofre nenhuma modificação para respeitar a restrição de proporção. Nesse caso, as sequências de eventos fornecidas pelos algoritmos não sofreram nenhuma modificação e resultaram em bons resultados, independentemente do grupo.

Observando os resultados do algoritmo UPRT, podemos ver que a razão de aproximação média tende a diminuir à medida que o valor de k aumenta e, em comparação com os demais algoritmos, sofre menor variação. Considerando a maior e a menor taxa de aproximação média entre todos os grupos, temos 1.56 e 1.24, respectivamente. Esta é uma variação de 0.32, o que mostra que o algoritmo é robusto, independentemente da proporção adotada para o cenário. Observe que a variação da aproximação média mostra o quanto o algoritmo oscila de acordo com as diferentes proporções. Deseja-se obter uma variação tão pequena quanto possível. Isso ajuda a gerar resultados práticos estáveis, independentemente da proporção desejada. A razão de aproximação média fornecida pelo algoritmo foi melhor que as demais, exceto no grupo onde $k = 0.0$. Isso provavelmente ocorre porque, quando $k = 0.0$, uma sequência composta exclusivamente por transposições se enquadra na restrição de proporção. Considerando a razão de aproximação máxima (pior caso prático), podemos observar que em todos os grupos o registro foi menor ou igual a 1.94. Outra característica interessante dos resultados desse algoritmo está relacionada às proporções obtidas nas soluções. A proporção média para todos os grupos é sempre muito próxima do valor mínimo especificado para a instância.

A Tabela 3.2 mostra os resultados dos algoritmos SR, SRT, SWRT e SPRT aplicados

em instâncias clássicas com sinais da base de dados DB1. Considerando todas as instâncias da base de dados, um total de 3.65% e 39.11% das soluções fornecidas por SRT e SWRT, respectivamente, foram modificadas para se adequarem à proporção mínima entre a quantidade de reversões e tamanho da sequência de rearranjo. A razão de aproximação foi calculada adotando-se o limite inferior apresentado no Teorema 3.1.12.

Pela Tabela 3.2, podemos ver que o algoritmo SR apresentou um comportamento semelhante ao algoritmo UR no caso sem sinal. No entanto, a aproximação média obtida foi exatamente $2 - k$, exceto para o grupo com $k = 0.0$. Observe que quando $k = 1.0$, temos o problema de Ordenação de Permutações por Reversões e o algoritmo SR fornece uma solução exata em tempo polinomial para o problema. Mantivemos esse cenário de proporção em nossos experimentos para verificar o desempenho dos outros algoritmos neste caso específico.

Os algoritmos SRT e SWRT não apresentam tendência de aumentar ou diminuir a razão média de aproximação considerando o valor de k. Comparando ambos, podemos ver que a variação média de aproximação do algoritmo SRT ($1.88 - 1.05 = 0.83$) é maior que a variação do algoritmo SWRT ($1.19 - 1.03 = 0.16$). Em comparação com o SWRT, a proporção média de soluções fornecidas pelo algoritmo SRT é maior. Exceto para o grupo com $k = 0.0$, a proporção média foi superior a 0.97. O fato do algoritmo SRT não ter aplicado nenhuma reversão no grupo em que $k = 0$ é explicado pelo próprio comportamento do algoritmo, pois ele aplica reversões apenas em ciclos divergentes, e as instâncias desses grupos foram geradas usando apenas transposições, o que não gera ciclos divergentes.

O algoritmo SPRT apresentou a aproximação média mais consistente para os diferentes valores de k. Observe que a aproximação média nos extremos quando k é igual a 0.0 e 1.0 foi 1.02 e 1.01, respectivamente. Além disso, a máxima aproximação média para os diferentes valores de k foi de 1.17, mostrando a robustez do algoritmo considerando diferentes cenários de proporção.

As tabelas 3.3 e 3.4 mostram, respectivamente, os resultados dos algoritmos SWRT e SPRT considerando o cenário de proporção específico onde $k = 0.6$. Como o algoritmo SWRT adota pesos 2 e 3 para eventos de reversão e transposição, respectivamente, uma forma indireta de fornecer uma comparação justa é usar a proporção $k = 0.6$ (o número de reversões em uma solução para o problema $\mathbf{Sb_PRT}$ é pelo menos 50% maior que o número de transposições).

A Tabela 3.3 mostra os resultados do algoritmo SWRT aplicado em instâncias clássicas com sinais da base de dados DB2. A coluna OP mostra o número de operações para criar as instâncias. Considerando os grupos de instâncias de tamanho 100, 200, 300, 400 e 500, um total de 37.00%, 69.46%, 79.98%, 84.81% e 87.59% das soluções fornecidas pelo algoritmo SWRT foram modificadas, respectivamente. Considerando todas as instâncias, um total de 71.76% das soluções fornecidas pelo algoritmo SWRT foram modificadas. A razão de aproximação foi calculada adotando-se o limite inferior apresentado no Teorema 3.1.12.

Pela Tabela 3.3, é possível notar que a variação da razão de aproximação média é pequena independentemente do tamanho da permutação ou do número de operações utilizadas para criar a instância. Considerando a maior e a menor razão de aproximação média, temos os valores 1.12 e 1.03, respectivamente. Além disso, a razão de aproximação

Tabela 3.2: Resultados dos algoritmos em instâncias clássicas com sinais da base de dados DB1.

SR									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	1.000	1.000	1.000	67	78.870	82	2.03	2.03	2.06
0.1	1.000	1.000	1.000	64	74.485	77	1.90	1.90	1.93
0.2	1.000	1.000	1.000	63	70.867	74	1.80	1.80	1.85
0.3	1.000	1.000	1.000	60	67.167	69	1.70	1.70	1.73
0.4	1.000	1.000	1.000	56	63.386	65	1.60	1.60	1.63
0.5	1.000	1.000	1.000	54	59.532	61	1.50	1.50	1.53
0.6	1.000	1.000	1.000	50	55.654	57	1.40	1.40	1.43
0.7	1.000	1.000	1.000	46	51.767	54	1.30	1.30	1.35
0.8	1.000	1.000	1.000	43	47.837	49	1.20	1.20	1.23
0.9	1.000	1.000	1.000	39	43.882	45	1.10	1.10	1.13
1.0	1.000	1.000	1.000	36	39.920	40	1.00	1.00	1.00

SRT									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	0.000	0.000	0.000	38	45.570	54	1.00	1.17	1.35
0.1	0.400	0.975	1.000	55	74.000	76	1.38	1.88	1.90
0.2	0.730	0.981	1.000	62	70.570	72	1.59	1.79	1.80
0.3	0.774	0.982	1.000	60	66.932	68	1.55	1.69	1.70
0.4	0.780	0.981	1.000	54	63.171	64	1.47	1.59	1.60
0.5	0.750	0.980	1.000	53	59.323	60	1.35	1.49	1.50
0.6	0.769	0.978	1.000	50	55.477	56	1.30	1.39	1.40
0.7	0.712	0.977	1.000	46	51.606	52	1.18	1.29	1.30
0.8	0.800	0.974	1.000	43	47.694	52	1.10	1.19	1.30
0.9	0.900	0.977	1.000	39	43.938	59	1.00	1.10	1.48
1.0	1.000	1.000	1.000	36	41.750	60	1.00	1.05	1.53

SWRT									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	0.000	0.000	0.000	35	40.153	45	1.00	1.03	1.14
0.1	0.100	0.334	0.737	37	45.021	57	1.00	1.15	1.50
0.2	0.200	0.422	0.727	37	45.141	55	1.00	1.14	1.41
0.3	0.300	0.478	0.778	37	44.345	54	1.00	1.12	1.39
0.4	0.400	0.526	0.792	37	43.248	53	1.00	1.09	1.33
0.5	0.500	0.575	0.816	37	42.116	50	1.00	1.06	1.26
0.6	0.600	0.638	0.833	37	41.576	48	1.00	1.04	1.20
0.7	0.700	0.720	0.894	36	42.131	54	1.00	1.06	1.35
0.8	0.800	0.814	0.909	37	43.511	56	1.00	1.09	1.40
0.9	0.900	0.911	0.952	38	45.516	60	1.00	1.14	1.50
1.0	1.000	1.000	1.000	38	47.336	66	1.00	1.19	1.65

SPRT									
k	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
0.0	0.000	0.014	0.244	35	39.696	48	1.00	1.02	1.24
0.1	0.125	0.334	0.582	38	45.691	55	1.02	1.17	1.43
0.2	0.219	0.394	0.627	39	45.103	55	1.01	1.15	1.38
0.3	0.300	0.455	0.640	38	44.406	54	1.01	1.12	1.35
0.4	0.400	0.520	0.694	38	43.732	52	1.00	1.10	1.32
0.5	0.500	0.587	0.735	38	42.965	50	1.00	1.08	1.25
0.6	0.600	0.665	0.792	37	42.514	50	1.00	1.07	1.28
0.7	0.700	0.745	0.844	39	42.010	47	1.00	1.06	1.20
0.8	0.800	0.828	0.889	37	41.430	47	1.00	1.04	1.18
0.9	0.900	0.918	0.954	36	40.941	46	1.00	1.03	1.15
1.0	1.000	1.000	1.000	36	40.059	42	1.00	1.01	1.05

Tabela 3.3: Resultados do algoritmo SWRT em instâncias clássicas com sinais da base de dados DB2.

Tamanho da Instância = 100									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
10	0.600	0.711	1.000	8	11.197	17	1.00	1.12	1.70
20	0.600	0.697	0.963	18	21.624	28	1.00	1.08	1.40
30	0.600	0.663	0.921	26	31.134	38	1.00	1.05	1.28
40	0.600	0.637	0.878	32	39.931	49	1.00	1.03	1.22
50	0.600	0.624	0.818	38	47.398	57	1.00	1.03	1.16

Tamanho da Instância = 200									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
20	0.600	0.693	0.963	19	21.808	28	1.00	1.09	1.40
40	0.600	0.637	0.857	37	41.565	49	1.00	1.04	1.22
60	0.600	0.614	0.738	55	62.137	71	1.00	1.04	1.18
80	0.600	0.610	0.645	70	82.220	91	1.00	1.06	1.15
100	0.600	0.608	0.618	83	98.919	111	1.00	1.07	1.17

Tamanho da Instância = 300									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
30	0.600	0.667	0.895	28	31.947	39	1.00	1.06	1.30
60	0.600	0.615	0.746	57	62.345	70	1.00	1.04	1.16
90	0.600	0.608	0.618	86	95.273	104	1.00	1.06	1.15
120	0.600	0.606	0.613	115	126.313	135	1.03	1.08	1.14
150	0.600	0.605	0.612	136	151.915	166	1.04	1.09	1.15

Tamanho da Instância = 400									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
40	0.600	0.647	0.857	38	41.928	49	1.00	1.04	1.22
80	0.600	0.610	0.698	77	83.992	94	1.00	1.05	1.17
120	0.600	0.606	0.613	120	128.804	139	1.02	1.08	1.15
160	0.600	0.605	0.610	155	170.892	181	1.05	1.10	1.14
200	0.600	0.604	0.608	189	205.193	224	1.07	1.11	1.14

Tamanho da Instância = 500									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
50	0.600	0.631	0.810	48	51.872	61	1.00	1.03	1.22
100	0.600	0.607	0.620	99	105.745	119	1.00	1.05	1.19
150	0.600	0.605	0.610	153	162.374	172	1.04	1.08	1.14
200	0.600	0.604	0.608	201	215.583	227	1.06	1.10	1.14
250	0.600	0.603	0.607	240	258.773	279	1.09	1.11	1.14

máxima foi de 1.70, observada no grupo de instâncias com tamanho 100. Note que o algoritmo apresenta bons resultados mesmo com 71.76% das soluções sendo modificadas para satisfazer a restrição de proporção mínima. Isso mostra que o processo de modificação pode produzir bons resultados dependendo do algoritmo utilizado.

A Tabela 3.4 mostra os resultados do algoritmo SPRT aplicado em instâncias clássicas com sinais da base de dados DB2. A coluna OP mostra o número de operações utilizadas para criar as instâncias. A razão de aproximação foi calculada adotando-se o limite inferior apresentado no Teorema 3.1.12.

A partir da Tabela 3.4, é possível notar que o algoritmo SPRT apresenta uma tendência de diminuir a razão de aproximação média à medida que o tamanho da permutação e o número de operações utilizadas para criar as instâncias (OP) crescem. Outro fato

Tabela 3.4: Resultados do algoritmo SPRT em instâncias clássicas com sinais da base de dados DB2.

Tamanho da Instância = 100									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
10	0.600	0.733	1.000	8	11.349	16	1.00	1.13	1.60
20	0.600	0.701	0.889	17	21.882	28	1.00	1.10	1.40
30	0.600	0.681	0.824	25	31.725	39	1.00	1.07	1.30
40	0.600	0.666	0.800	32	40.592	49	1.00	1.05	1.25
50	0.600	0.659	0.769	39	47.919	57	1.00	1.05	1.22

Tamanho da Instância = 200									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
20	0.600	0.700	0.923	19	22.155	29	1.00	1.10	1.45
40	0.600	0.666	0.809	38	42.556	50	1.00	1.06	1.25
60	0.600	0.649	0.742	55	62.097	69	1.00	1.04	1.16
80	0.600	0.639	0.711	70	80.046	89	1.00	1.03	1.14
100	0.600	0.633	0.697	83	94.773	105	1.00	1.03	1.10

Tamanho da Instância = 300									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
30	0.600	0.679	0.842	29	32.601	39	1.00	1.08	1.30
60	0.600	0.650	0.758	58	62.951	70	1.00	1.05	1.16
90	0.600	0.635	0.704	85	92.262	100	1.00	1.03	1.12
120	0.600	0.628	0.681	110	119.250	129	1.00	1.02	1.09
150	0.600	0.623	0.664	128	141.327	153	1.00	1.02	1.08

Tamanho da Instância = 400									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
40	0.600	0.666	0.816	39	42.867	50	1.00	1.07	1.25
80	0.600	0.640	0.727	76	83.148	93	1.00	1.04	1.16
120	0.600	0.627	0.683	115	122.314	130	1.00	1.02	1.09
160	0.600	0.622	0.656	149	158.546	168	1.00	1.02	1.07
200	0.600	0.619	0.647	175	187.099	198	1.00	1.01	1.06

Tamanho da Instância = 500									
OP	Proporção			Distância			Aproximação		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
50	0.600	0.656	0.793	49	53.064	60	1.00	1.06	1.20
100	0.600	0.633	0.697	98	103.272	111	1.00	1.03	1.12
150	0.600	0.622	0.673	145	152.283	162	1.00	1.02	1.08
200	0.601	0.618	0.645	186	197.490	204	1.00	1.01	1.05
250	0.603	0.617	0.634	226	233.744	239	1.00	1.01	1.04

importante é que em todos os grupos (considerando o tamanho da instância e o número de operações utilizadas para criar as instâncias), o algoritmo SPRT conseguiu encontrar, para pelo menos uma instância do grupo, uma solução ótima. Podemos confirmar esse comportamento observando a coluna da razão de aproximação mínima. Além disso, considerando os grupos de instâncias com tamanho maior que 100 e os casos em que foram utilizadas sequências de operações maiores que 20% do tamanho das instâncias, o algoritmo SPRT, em comparação com o algoritmo SWRT, apresentou resultados equivalentes ou melhores ao observar a razão de aproximação média.

A partir dos resultados, observamos que os algoritmos propostos apresentam um excelente desempenho na prática, tanto na variação sem sinais do problema $\mathbf{Sb_PRT}$, como também na variação com sinais. Vale ressaltar que o processo proposto de modificação

da solução para viabilizar soluções obtidas através de algoritmos para outros problemas também apresentou bons resultados, principalmente quando aplicado ao algoritmo SWRT.

3.5 Conclusões

Nesse capítulo, investigamos o problema de Ordenação de Permutações por Reversões e Transposições com Restrição de Proporção. Como resultado, apresentamos uma análise de complexidade do problema para qualquer valor permitido de k e consideramos as variações com e sem sinais. Apresentamos algoritmos de aproximação com fatores $3 - \frac{3k}{2}$ e $3 - k$ para as variações com e sem sinais, respectivamente. Além disso, apresentamos um algoritmo de aproximação assintótica com fator $\frac{2-k}{1-k/3}$ para a variação com sinais do problema. Por fim, realizamos testes experimentais comparando os algoritmos propostos com outros algoritmos da literatura que fornecem uma solução válida para o problema ou que a solução foi modificada para tornar a comparação possível.

Capítulo 4

Modelos Intergênicos Rígidos

A representação de um genoma por meio de uma sequência de genes é bastante útil e amplamente utilizada em problemas de rearranjo de genomas. Entretanto, informações que não estão associadas diretamente aos genes são descartadas, o que implica em uma perda de informação. Em particular, informações referentes às regiões intergênicas, que são regiões entre cada par consecutivo de genes e nas extremidades de um genoma, não são consideradas pelos modelos que adotam uma representação clássica de um genoma. Estudos sugerem que incorporar tais estruturas aos modelos pode propiciar resultados mais realistas para a distância evolutiva entre os organismos [15, 16]. Cada região intergênica possui uma quantidade de nucleotídeos, a qual denominamos de *tamanho* da região intergênica. Neste capítulo, investigaremos as variações com e sem sinais dos seguintes problemas que consideram a informação dos genes e do tamanho das regiões intergênicas de um genoma adotando uma abordagem *não ponderada*:

- Ordenação de Permutações por Reversões Intergênicas (**Sb$_\mathbf{I}$R**)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Indel (**Sb$_\mathbf{I}$RI**)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Move (**Sb$_\mathbf{I}$RM**)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Move e Indel (**Sb$_\mathbf{I}$RMI**)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Transposição (**Sb$_\mathbf{I}$RT**)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Indel (**Sb$_\mathbf{I}$RTI**)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Move (**Sb$_\mathbf{I}$RTM**)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição, Move e Indel (**Sb$_\mathbf{I}$RTMI**)

Além disso, investigaremos as variações com e sem sinais dos seguintes problemas considerando uma abordagem *ponderada*:

- Ordenação de Permutações por Operações Intergênicas Ponderadas de Reversão e Indel (**Sb$_{\mathbf{WI}}$RI**)
- Ordenação de Permutações por Operações Intergênicas Ponderadas de Reversão e Transposição (**Sb$_{\mathbf{WI}}$RT**)
- Ordenação de Permutações por Operações Intergênicas Ponderadas de Reversão, Transposição e Indel (**Sb$_{\mathbf{WI}}$RTI**)

Neste capítulo, iremos referenciar aos eventos de rearranjo de reversão intergênica, transposição intergênica, move intergênico e indel intergênico simplesmente por reversão, transposição, move e indel, respectivamente. Além disso, referiremos a um breakpoint intergênico simplesmente por breakpoint. Dada uma sequência de eventos de rearranjo S, denotamos por $|S|$ o tamanho da sequência S, ou seja, a quantidade de eventos em S.

Dada uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e adotando um cenário não ponderado, a *distância* entre $(\pi, \breve{\pi})$ e $(\iota, \breve{\iota})$, denotada por $di_{\mathcal{M}}(\mathcal{I})$, é o tamanho da menor sequência de eventos de rearranjo S, tal que todo evento de S pertence ao modelo $\mathcal{M}$ e $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$. Os modelos de rearranjo em um cenário não ponderado considerados neste capítulo são identificados pelas siglas apresentadas na Tabela 4.1.

Tabela 4.1: Siglas dos modelos de rearranjo considerados para instâncias intergênicas rígidas em um cenário não ponderado.

Problema	**Sigla do Modelo**	**Conjunto de Eventos de Rearranjo**
Sb$_{\mathbf{I}}$R	**R**	$\{\rho\}$
Sb$_{\mathbf{I}}$RI	**RI**	$\{\rho, \delta\}$
Sb$_{\mathbf{I}}$RM	**RM**	$\{\rho, \mu\}$
Sb$_{\mathbf{I}}$RMI	**RMI**	$\{\rho, \mu, \delta\}$
Sb$_{\mathbf{I}}$RT	**RT**	$\{\rho, \tau\}$
Sb$_{\mathbf{I}}$RTI	**RTI**	$\{\rho, \tau, \delta\}$
Sb$_{\mathbf{I}}$RTM	**RTM**	$\{\rho, \tau, \mu\}$
Sb$_{\mathbf{I}}$RTMI	**RTMI**	$\{\rho, \tau, \mu, \delta\}$

Dada uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e adotando um cenário ponderado, a *distância ponderada* entre $(\pi, \breve{\pi})$ e $(\iota, \breve{\iota})$, denotada por $dwi_{\mathcal{M}}(\mathcal{I})$, é o custo da sequência de eventos de rearranjo S com menor custo total, tal que todo evento de S pertence ao modelo $\mathcal{M}$ e $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$. O custo total de uma sequência de eventos de rearranjo $S = (\gamma_1, \gamma_2, \ldots, \gamma_k)$, onde cada evento pertence ao modelo $\mathcal{M}$, é dado por $\sum_{\gamma_i \in S} c(\gamma_i)$, tal que $c(\gamma_i)$ representa o custo associado ao tipo do evento γ_i por uma função de custo. Os modelos de rearranjo em um cenário ponderado considerados neste capítulo são identificados pelas siglas apresentadas na Tabela 4.2.

Quando for adotado um modelo de rearranjo composto exclusivamente por eventos de rearranjo conservativos, será assumido que a instância intergênica rígida para o problema será sempre balanceada.

Tabela 4.2: Siglas dos modelos de rearranjo considerados para instâncias intergênicas rígidas em um cenário ponderado.

Problema	Sigla do Modelo	Conjunto de Eventos de Rearranjo
$\mathbf{Sb_{WI}RI}$	**RI**	$\{\rho, \delta\}$
$\mathbf{Sb_{WI}RT}$	**RT**	$\{\rho, \tau\}$
$\mathbf{Sb_{WI}RTI}$	**RTI**	$\{\rho, \tau, \delta\}$

Parte dos resultados que serão apresentados neste capítulo foram publicados nas revistas *Journal of Computational Biology* [22] e *Algorithms for Molecular Biology* [23] em 2020 e 2021, respectivamente.

4.1 Abordagem não Ponderada

Nesta seção, apresentaremos os resultados para as variações com e sem sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$ em um cenário não ponderado.

4.1.1 Limitantes Inferiores

Nesta seção, apresentaremos limitantes inferiores para as variações com e sem sinais dos problemas investigados neste capítulo em um cenário não ponderado.

Em instâncias intergênicas rígidas com e sem sinais utilizaremos os conceitos de breakpoint intergênicos tipo dois e um, respectivamente (Seção 2.4.2). Os eventos de rearranjo de reversão, transposição, move e indel afetam, respectivamente, as seguintes quantidades de regiões intergênicas: duas, três, duas e uma. No melhor cenário, cada uma das regiões intergênicas afetadas faz parte de um breakpoint removido após o evento de rearranjo ser aplicado. Com isso, obtemos os seguintes lemas.

Lema 4.1.1. *Seja $\mathcal{I}$ uma instância intergênica rígida sem sinais. Para qualquer reversão ρ temos que $\Delta ib_1(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 4.1.2. *Seja $\mathcal{I}$ uma instância intergênica rígida sem sinais. Para qualquer transposição τ temos que $\Delta ib_1(\mathcal{I}, S = (\tau)) \geq -3$.*

Lema 4.1.3. *Seja $\mathcal{I}$ uma instância intergênica rígida sem sinais. Para qualquer move μ temos que $\Delta ib_1(\mathcal{I}, S = (\mu)) \geq -2$.*

Lema 4.1.4. *Seja $\mathcal{I}$ uma instância intergênica rígida sem sinais. Para qualquer indel δ temos que $\Delta ib_1(\mathcal{I}, S = (\delta)) \geq -1$.*

Lema 4.1.5. *Seja $\mathcal{I}$ uma instância intergênica rígida com sinais. Para qualquer reversão ρ temos que $\Delta ib_2(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 4.1.6. *Seja $\mathcal{I}$ uma instância intergênica rígida com sinais. Para qualquer transposição τ temos que $\Delta ib_2(\mathcal{I}, S = (\tau)) \geq -3$.*

Lema 4.1.7. *Seja $\mathcal{I}$ uma instância intergênica rígida com sinais. Para qualquer move μ temos que $\Delta ib_2(\mathcal{I}, S = (\mu)) \geq -2$.*

Lema 4.1.8. *Seja $\mathcal{I}$ uma instância intergênica rígida com sinais. Para qualquer indel δ temos que $\Delta ib_2(\mathcal{I}, S = (\delta)) \geq -1$.*

Teorema 4.1.9. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Temos que:*

$$
\begin{aligned}
di_{\boldsymbol{R}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{2},\\
di_{\boldsymbol{RI}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{2},\\
di_{\boldsymbol{RM}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{2},\\
di_{\boldsymbol{RMI}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{2},\\
di_{\boldsymbol{RT}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{3},\\
di_{\boldsymbol{RTI}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{3},\\
di_{\boldsymbol{RTM}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{3} \text{ e}\\
di_{\boldsymbol{RTMI}}(\mathcal{I}) &\geq \tfrac{ib_1(\mathcal{I})}{3}.
\end{aligned}
$$

Demonstração. Pela Observação 2.4.2, para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ é necessário remover os $ib_1(\mathcal{I})$ breakpoints tipo um de $\mathcal{I}$. Dessa forma, obtemos um limitante inferior para cada um dos modelos através da divisão de $ib_1(\mathcal{I})$ pela maior quantidade de breakpoints tipo um que podem ser removidos por um evento permitido no modelo de rearranjo. Os lemas 4.1.1, 4.1.2, 4.1.3 e 4.1.4 mostram a quantidade máxima de breakpoints tipo um que podem ser removidos de uma instância intergênica rígida sem sinais pelos eventos de reversão, transposição, move e indel, respectivamente. Logo, o teorema segue. □

Teorema 4.1.10. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Se $\mathcal{I}$ for balanceada, então temos que $di_{\boldsymbol{RTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Caso contrário, temos que $di_{\boldsymbol{RTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})+2}{3}$.*

Demonstração. Pela Observação 2.4.2, para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ é necessário remover os $ib_1(\mathcal{I})$ breakpoints tipo um de $\mathcal{I}$. Note que se $\mathcal{I}$ for balanceada, então podemos aplicar o Teorema 4.1.9. Caso contrário, sabemos que para ser possível transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, pelo menos um indel deve ser utilizado. Pelo Lema 4.1.4, temos que, no máximo, um breakpoint tipo um pode ser removido utilizando uma operação de indel. No melhor caso, após aplicar apenas um indel, $\mathcal{I}$ torna-se uma instância balanceada e um breakpoint tipo um é removido. Assim, restam $ib_1(\mathcal{I}) - 1$ breakpoints para serem removidos de $\mathcal{I}$. Considerando os eventos de reversão, transposição e indel, três breakpoints tipo um, no máximo, podem ser removidos por operação (lemas 4.1.1, 4.1.2 e 4.1.4). Logo, pelo menos $\frac{ib_1(\mathcal{I})-1}{3}$ eventos de reversão, transposição ou indel são necessários para remover o restante dos breakpoints tipo um. Dessa forma, pelo menos $\frac{ib_1(\mathcal{I})-1}{3} + 1 = \frac{ib_1(\mathcal{I})+2}{3}$ eventos de reversão, transposição e indel são necessários para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, e o teorema segue. □

Teorema 4.1.11. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Temos que:*

$$
\begin{array}{lcl}
di_{\boldsymbol{R}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{2}, \\
di_{\boldsymbol{RI}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{2}, \\
di_{\boldsymbol{RM}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{2}, \\
di_{\boldsymbol{RMI}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{2}, \\
di_{\boldsymbol{RT}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{3}, \\
di_{\boldsymbol{RTI}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{3}, \\
di_{\boldsymbol{RTM}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{3} \text{ e} \\
di_{\boldsymbol{RTMI}}(\mathcal{I}) & \geq & \frac{ib_2(\mathcal{I})}{3}.
\end{array}
$$

Demonstração. A prova é similar à descrita no Teorema 4.1.9, mas considerando os lemas 4.1.5, 4.1.6, 4.1.7 e 4.1.8. □

Teorema 4.1.12. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Se $\mathcal{I}$ for balanceada, então temos que $di_{\boldsymbol{RTI}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$. Caso contrário, temos que $di_{\boldsymbol{RTI}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})+2}{3}$.*

Demonstração. Pela Observação 2.4.2, para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ é necessário remover os $ib_2(\mathcal{I})$ breakpoints tipo dois de $\mathcal{I}$. Note que se $\mathcal{I}$ for balanceada, então podemos aplicar o Teorema 4.1.11. Caso contrário, sabemos que para ser possível transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, pelo menos um indel deve ser utilizado. Pelo Lema 4.1.8, temos que, no máximo, um breakpoint tipo dois pode ser removido utilizando uma operação de indel. No melhor caso, após aplicar apenas um indel, $\mathcal{I}$ torna-se uma instância balanceada e um breakpoint tipo dois é removido. Assim, restam $ib_2(\mathcal{I}) - 1$ breakpoints para serem removidos de $\mathcal{I}$. Considerando os eventos de reversão, transposição e indel, três breakpoints tipo um, no máximo, podem ser removidos por operação (lemas 4.1.5, 4.1.6 e 4.1.8). Logo, pelo menos $\frac{ib_2(\mathcal{I})-1}{3}$ eventos de reversão, transposição ou indel são necessários para remover o restante dos breakpoints tipo dois. Dessa forma, pelo menos $\frac{ib_2(\mathcal{I})-1}{3} + 1 = \frac{ib_2(\mathcal{I})+2}{3}$ eventos de reversão, transposição e indel são necessários para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, e o teorema segue. □

Considerando o grafo de ciclos ponderado rígido (Seção 2.6.2) criado a partir de uma instância intergênica rígida com sinais, o evento de reversão afeta duas arestas pretas do grafo e pode aumentar tanto o número de ciclos como também o número de ciclos balanceados. O evento de transposição afeta três arestas pretas do grafo e pode aumentar tanto o número de ciclos como o número de ciclos balanceados. O evento de move afeta duas arestas pretas do grafo e pode aumentar somente o número de ciclos balanceados no grafo. O evento de indel afeta apenas uma aresta preta do grafo e pode aumentar somente o número de ciclos balanceados. Dessa forma, dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, temos que $\Delta c(G(\mathcal{I}), S = (\rho)) \in \{1, 0, -1\}$ e $\Delta c_b(G(\mathcal{I}), S = (\rho)) \in \{1, 0, -1\}$ para qualquer reversão ρ [11, 57]. De maneira similar, temos que $\Delta c(G(\mathcal{I}), S = (\tau)) \in \{2, 0, -2\}$ e $\Delta c_b(G(\mathcal{I}), S = (\tau)) \in \{2, 1, 0, -1, -2\}$ para qualquer transposição τ [12, 59], $\Delta c(G(\mathcal{I}), S = (\mu)) = 0$ e $\Delta c_b(G(\mathcal{I}), S = (\mu)) \in \{2, 1, 0, -1, -2\}$ para qualquer move μ [59], e $\Delta c(G(\mathcal{I}), S = (\delta)) = 0$ e $\Delta c_b(G(\mathcal{I}), S = (\delta)) \in \{1, 0, -1\}$ para qualquer indel δ [57]. Com isso, obtemos os seguintes limitantes inferiores.

Teorema 4.1.13. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Temos que:*

$$\begin{array}{lcl} di_{\boldsymbol{RM}}(\mathcal{I}) & \geq & n+1-\frac{c(G(\mathcal{I}))+c_b(G(\mathcal{I}))}{2} \ e \\ di_{\boldsymbol{RMI}}(\mathcal{I}) & \geq & n+1-\frac{c(G(\mathcal{I}))+c_b(G(\mathcal{I}))}{2}. \end{array}$$

Demonstração. Note que, para atingir o genoma alvo, é necessário aumentar tanto o número de ciclos quanto o de ciclos balanceados em $G(\mathcal{I})$ para $n+1$ (Observação 2.6.3). Dados os eventos de reversão, move e indel, temos que $c(G(\mathcal{I}))+c_b(G(\mathcal{I}))$ pode aumentar, no máximo, em duas unidades. Logo, o teorema segue. □

Teorema 4.1.14. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Temos que:*

$$\begin{array}{lcl} di_{\boldsymbol{RTI}}(\mathcal{I}) & \geq & \frac{n+1-c_b(G(\mathcal{I}))}{2} \ e \\ di_{\boldsymbol{RTMI}}(\mathcal{I}) & \geq & \frac{n+1-c_b(G(\mathcal{I}))}{2}. \end{array}$$

Demonstração. Note que, para atingir o genoma alvo, é necessário aumentar o número de ciclos balanceados em $G(\mathcal{I})$ para $n+1$ (Observação 2.6.3). Dados os eventos de reversão, transposição, move e indel, temos que $c_b(G(\mathcal{I}))$ pode aumentar, no máximo, em duas unidades. Logo, o teorema segue. □

4.1.2 Análise de Complexidade

Nesta seção, realizaremos uma análise de complexidade considerando as variações com e sem sinais dos problemas investigados neste capítulo em um cenário não ponderado. Os problemas citados e investigados nessa seção referem-se a suas respectivas versões de decisão.

Inicialmente, descrevemos a versão de decisão de alguns problemas, sendo eles:

- A variação sem sinais do problema de Ordenação de Permutações por Reversões (**SbR**)
- As variações com e sem sinais do problema de Ordenação de Permutações por Reversões e Transposições (**SbRT**)
- O problema 3-Partição (**3-PART**).

Esses três problemas pertencem à classe NP-difícil [30, 46, 55].

(**SbR**) (Versão de Decisão)
Entrada: Uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ e um número natural d.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho\}$, capaz de transformar π em ι, tal que $|S| \leq d$?

(**SbRT**) (Versão de Decisão)
Entrada: Uma instância clássica com ou sem sinais $\mathcal{I} = (\pi, \iota)$ e um número natural d.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau\}$, capaz de transformar π em ι, tal que $|S| \leq d$?

(**3-PART**) (Versão de Decisão)
Entrada: Um conjunto de números inteiros positivos $A = \{a_1, a_2, \ldots, a_{3n}\}$, tal que $\sum_{i=1}^{3n} a_i = Bn$ e $B \in \mathbb{Z}^+$. Além disso, $\frac{B}{4} < a_i < \frac{B}{2}$, com $1 \leq i \leq 3n$.
Pergunta: Existe uma partição do conjunto A em triplas $A_1, A_2, \ldots, A_n$, tal que $\sum_{a_i \in A_j} a_i = B$ para cada tripla A_j, com $1 \leq j \leq n$?

A seguir, descrevemos a versão de decisão das variações com e sem sinais dos problemas que investigaremos neste capítulo em um cenário não ponderado.

$\mathbf{Sb_IR}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRI}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \delta\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRM}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \mu\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRMI}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \mu, \delta\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRT}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRTI}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau, \delta\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRTM}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau, \mu\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRTMI}$ (Versão de Decisão)
Entrada: Um número natural t e uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$.
Pergunta: Existe uma sequência de eventos de rearranjo S, com base no modelo de rearranjo $\mathcal{M} = \{\rho, \tau, \mu, \delta\}$, capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

A seguir mostramos que todas as variações sem sinais dos problemas investigados nesse capítulo, em um cenário não ponderado, pertencem à classe NP-difícil.

Teorema 4.1.15. *Os problemas **Sb_IR**, **Sb_IRI**, **Sb_IRM** e **Sb_IRMI** em instâncias intergênicas rígidas sem sinais pertencem à classe NP-difícil.*

Demonstração. Dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ e um valor d para a versão de decisão do problema **SbR**, criaremos uma instância intergênica rígida sem sinais $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ e um valor t para a versão de decisão do problema **$\mathbf{Sb_IR}$**, **$\mathbf{Sb_IRI}$**, **$\mathbf{Sb_IRM}$** ou **$\mathbf{Sb_IRMI}$** da seguinte maneira: (i) $\pi' = \pi$, (ii) $\iota' = \iota$, (iii) $\breve{\pi}' = \breve{\iota}' = (0, 0, \ldots, 0)$ e (iv) $t = d$. Agora mostramos que a instância $(\mathcal{I}, d)$ do problema **SbR** é sim se, e somente se, a instância $(\mathcal{I}', t)$ do problema **$\mathbf{Sb_IR}$**, **$\mathbf{Sb_IRI}$**, **$\mathbf{Sb_IRM}$** ou **$\mathbf{Sb_IRMI}$** é sim.

($\Rightarrow$) Suponha que exista uma sequência S com d reversões, tal que $\pi \cdot S = \iota$. Considere a sequência S' criada a partir da sequência S mapeando cada reversão $\rho^{(i,j)}$ em uma reversão intergênica $\rho^{(i,j)}_{(0,0)}$. Note que $(\pi, \breve{\pi}) \cdot S' = (\iota, \breve{\iota})$ e $|S'| = t = d$, uma vez que o tamanho de todas as regiões intergênicas no genoma de origem e alvo é zero.

($\Leftarrow$) Agora suponha que exista uma sequência S' com t eventos de rearranjo tal que $(\pi, \breve{\pi}) \cdot S' = (\iota, \breve{\iota})$. Primeiramente, mostraremos que se a sequência S' não for composta exclusivamente por reversões intergênicas, então ela pode ser modificada para possuir tal característica mantendo a resposta positiva para a instância $(\mathcal{I}', t)$. Seja S' uma sequência que gera uma resposta positiva para a instância $(\mathcal{I}', t)$ do problema **$\mathbf{Sb_IR}$**, **$\mathbf{Sb_IRI}$**, **$\mathbf{Sb_IRM}$** ou **$\mathbf{Sb_IRMI}$** e não é composta exclusivamente por reversões intergênicas. Neste caso, criaremos uma sequência S'' composta exclusivamente por reversões intergênicas, tal que $|S''| < |S'|$ e $(\pi, \breve{\pi}) \cdot S'' = (\iota, \breve{\iota})$. Para cada reversão intergênica $\rho^{(i,j)}_{(x,y)}$ de S' adicione

em S'' a reversão intergênica $\rho^{(i,j)}_{(0,0)}$. Note que os eventos de move e indel não afetam a ordem dos genes. Além disso, pela construção de $\mathcal{I}'$, temos que $\breve{\pi}' = \breve{\iota}'$. Logo, $(\pi, \breve{\pi}) \cdot S'' = (\iota, \breve{\iota})$. Por fim, substitua S' por S''. Agora, com S' composta exclusivamente por reversões intergênicas, considere a sequência S criada a partir da sequência S' mapeando cada reversão intergênica $\rho^{(i,j)}_{(x,y)}$ em uma reversão $\rho^{(i,j)}$. É importante notar que, como a sequência S' é composta exclusivamente por reversões intergênicas, isso implica que nenhum nucleotídeo é inserido no genoma de origem. Logo, para cada reversão intergênica $\rho^{(i,j)}_{(x,y)}$ de S', temos que $x = y = 0$ e isso faz com que seja possível realizar tal mapeamento. Dessa forma, temos que $\pi \cdot S = \iota$ e $|S| \leq t = d$. Logo, o teorema segue. □

Teorema 4.1.16. *Os problemas* ***Sb$_I$RT****,* ***Sb$_I$RTI****,* ***Sb$_I$RTM*** *e* ***Sb$_I$RTMI*** *em instâncias intergênicas rígidas sem sinais pertencem à classe NP-difícil.*

Demonstração. A prova é similar à descrita no Teorema 4.1.15, mas utilizando uma redução da versão de decisão da variação sem sinais do problema **SbRT** e considerando que a sequência S' para a instância $(\mathcal{I}', t)$ do problema **Sb$_\mathbf{I}$RT**, **Sb$_\mathbf{I}$RTI**, **Sb$_\mathbf{I}$RTM** ou **Sb$_\mathbf{I}$RTMI** é composta por reversões intergênicas e transposições intergênicas ao invés de ser composta exclusivamente por reversões intergênicas. □

Os problemas **Sb$_\mathbf{I}$R**, **Sb$_\mathbf{I}$RT** e **Sb$_\mathbf{I}$RTM** em instâncias intergênicas rígidas com sinais pertencem à classe NP-difícil [57, 59]. A seguir, mostramos que a variação com sinais dos problemas **Sb$_\mathbf{I}$RM**, **Sb$_\mathbf{I}$RMI**, **Sb$_\mathbf{I}$RTI** e **Sb$_\mathbf{I}$RTMI** também pertencem à classe NP-difícil. Para a variação com sinais dos problemas **Sb$_\mathbf{I}$RM** e **Sb$_\mathbf{I}$RMI** iremos realizar uma redução do problema **3-PART**, enquanto que para a variação com sinais dos problemas **Sb$_\mathbf{I}$RTI** e **Sb$_\mathbf{I}$RTMI** utilizaremos um redução da variação com sinais do problema **SbRT**.

Teorema 4.1.17. *Os problemas* ***Sb$_I$RM*** *e* ***Sb$_I$RMI*** *em instâncias intergênicas rígidas com sinais pertencem à classe NP-difícil.*

Demonstração. Dada uma instância $A = \{a_1, a_2, \ldots, a_{3n}\}$ do problema **3-PART**, criamos uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ do problema **Sb$_\mathbf{I}$RM** ou **Sb$_\mathbf{I}$RMI** da seguinte forma:

i. $\pi = \iota = (+0\ +1\ +2\ \ldots\ +4n)$.

ii. Atribua o valor $\breve{\pi}_i = a_i$, para $1 \leq i \leq 3n$, e o valor $\breve{\pi}_j = 0$, para $3n + 1 \leq j \leq 4n$.

iii. Atribua o valor $\breve{\iota}_i = 0$, para $1 \leq i \leq 3n$, e o valor $\breve{\iota}_j = B$, para $3n + 1 \leq j \leq 4n$.

iv. $t = 3n$.

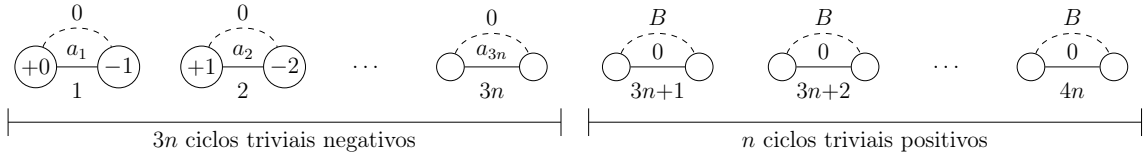
O Exemplo 4.1.1 ilustra o processo de criação de uma instância $\mathcal{I}$ do problema **Sb$_\mathbf{I}$RM** ou **Sb$_\mathbf{I}$RMI** com base na instância A do problema **3-PART**. Note que o grafo de ciclos ponderado rígido $G(\mathcal{I})$ é composto exclusivamente por ciclos triviais.

Exemplo 4.1.1.

$$A = \{a_1, a_2, \ldots, a_{3n}\}$$

$$(\pi, \breve{\pi}) = ((+0\ +1\ +2\ \ldots\ +4n), (a_1, a_2, \ldots, a_{3n},\ 0,\ 0, \ldots,\ 0))$$
$$(\iota, \breve{\iota}) = ((+0\ +1\ +2\ \ldots\ +4n), (\ 0,\ 0, \ldots,\ \ 0,\ B, B, \ldots, B))$$
$$\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$$

$G(\mathcal{I})$



Agora, mostramos que a instância A do **3-PART** é sim se, e somente se, $di_{\mathbf{RM}}(\mathcal{I}) \leq t = 3n$ e $di_{\mathbf{RMI}}(\mathcal{I}) \leq t = 3n$.

($\Rightarrow$) Suponha que seja possível particionar a instância $A = \{a_1, a_2, \ldots, a_{3n}\}$ do problema **3-PART** em n triplas, tal que $\sum_{a_i \in A_j} a_i = B$ para cada tripla A_j, com $1 \leq j \leq n$. Então, é possível transformar o genoma de origem $(\pi, \breve{\pi})$ no genoma alvo $(\iota, \breve{\iota})$ usando $3n$ operações de move. Para cada tripla A_j, com $1 \leq j \leq n$, aplique as seguintes operações de move $\mu_{(a_i)}^{(i,3n+j)}$, para $a_i \in A_j$. Repita esse procedimento em todas as n triplas, produzindo uma sequência de $3n$ operações de move que resulta em $c(G(\mathcal{I})) = c_b(G(\mathcal{I})) = 4n$. Assim, $di_{\mathbf{RM}}(\mathcal{I}) \leq 3n$ e $di_{\mathbf{RMI}}(\mathcal{I}) \leq 3n$.

($\Leftarrow$) Agora suponha que $di_{\mathbf{RM}}(\mathcal{I}) \leq 3n$ e $di_{\mathbf{RMI}}(\mathcal{I}) \leq 3n$. Observe que, para atingir o genoma alvo desejado, os ciclos triviais com arestas pretas rotuladas com o valor 0 até $3n$ devem formar ciclos triviais com peso zero em suas arestas pretas.

Note que $G(\mathcal{I})$ tem $4n$ ciclos triviais, onde: os primeiros $3n$ ciclos triviais (da esquerda para a direita usando a representação padrão de $G(\mathcal{I})$) têm peso positivo em suas arestas pretas; e os últimos n ciclos triviais têm peso zero em suas arestas pretas.

Dizemos que um ciclo $C = (c^1)$ é *alpha* se C for trivial e tiver peso zero em suas duas arestas (preta e cinza). Denotamos por $c_\alpha(G(\mathcal{I}))$ o número de ciclos alpha em $G(\mathcal{I})$. Observe que qualquer sequência de operações de rearranjo que transforme o genoma de origem no genoma alvo deve, necessariamente, afetar os primeiros $3n$ ciclos triviais para alterar o peso em suas arestas pretas para zero. Consequentemente, ao alcançar o genoma alvo temos que $c_\alpha(G(\mathcal{I})) = 3n$.

Dada uma sequência S de operações, denotamos por:

$$\Delta c_\alpha(G(\mathcal{I}), S) = \frac{c_\alpha(G(\mathcal{I}')) - c_\alpha(G(\mathcal{I}))}{|S|},$$

tal que $\mathcal{I}' = ((\pi, \breve{\pi}) \cdot S, (\iota, \breve{\iota}))$, a variação média no número de ciclos alpha gerados pela sequência S.

Um move $\mu_{(x)}^{(i,j)}$ atua em até dois ciclos. Se o move μ atua em duas arestas pretas do mesmo ciclo, então o peso desse ciclo permanece inalterado ($\Delta c_\alpha(G(\mathcal{I}), S = (\mu)) = 0$). Quando atua em dois ciclos, temos os seguintes casos:

- Se nenhum dos ciclos for trivial, então $\Delta c_\alpha(G(\mathcal{I}), S = (\mu)) = 0$, dado que a operação de move não altera o tamanho dos ciclos.

- Se pelo menos um dos ciclos for trivial, então o move pode transferir o peso em sua aresta preta para outro ciclo e, na melhor das hipóteses, $\Delta c_\alpha(G(\mathcal{I}), S = (\mu)) \leq 1$.

- Se ambos os ciclos são alpha, então $\Delta c_\alpha(G(\mathcal{I}), S = (\mu)) = 0$, dado que, neste caso, as arestas pretas de ambos os ciclos têm peso zero. Consequentemente, o peso da aresta preta dos ciclos permanece inalterado.

Uma reversão $\rho^{(i,j)}_{(x,y)}$ pode criar até dois ciclos alpha ($\Delta c_\alpha(G(\mathcal{I}), S = (\rho)) \leq 2$). No entanto, isso ocorre apenas no caso particular em que uma reversão divide um ciclo de tamanho dois onde ambas as arestas pretas têm peso zero. Dado que uma operação de move não altera o tamanho dos ciclos, para obter um ciclo de tamanho dois com tal característica, temos que uma reversão ρ que une dois ciclos deve ser aplicada previamente. Observe que qualquer reversão que une ciclos não cria ciclos alpha ($\Delta c_\alpha(G(\mathcal{I}), S = (\rho)) \leq 0$), pois esta operação resulta em um ciclo não trivial. Para obter um ciclo de tamanho dois onde ambas as arestas pretas tenham peso zero, temos três possibilidades:

- O primeiro caso consiste em uma reversão ρ_1, que une dois ciclos alpha (diminuindo o número de ciclos alpha em duas unidades), seguida por uma reversão ρ_2, que divide esse ciclo de tamanho dois gerado em dois ciclos alpha novamente. Observe que neste caso a reversão ρ_2 desfaz ρ_1, então $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1, \rho_2)) = 0$, uma vez que $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1)) = -2$ e $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_2)) = 2$.

- No segundo caso, temos uma reversão ρ_1 que une dois ciclos triviais onde ambas as arestas pretas têm peso maior que zero. Pela construção do ciclo de tamanho dois, temos que pelo menos uma de suas arestas pretas tem peso maior que zero. Para remover o peso nas arestas pretas do ciclo de tamanho dois, pelo menos um move μ_1 deve ser aplicado. Observe que o move μ_1 não cria ciclos alpha ($\Delta c_\alpha(G(\mathcal{I}), s = (\mu_1)) \leq 0$), pois transfere o peso de uma aresta preta de um ciclo de tamanho dois para outro ciclo. Por fim, é aplicada uma reversão ρ_2, que divide o ciclo de tamanho dois, gerando dois ciclos alpha. Observe que $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1, \mu_1, \rho_2)) \leq \frac{2}{3}$, uma vez que $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1)) = 0$, $\Delta c_\alpha(G(\mathcal{I}), S = (\mu_1)) \leq 0$ e $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_2)) = 2$.

- A última possibilidade é quando uma reversão ρ_1 une um ciclo trivial cuja aresta preta tem peso maior que zero com um ciclo alpha, diminuindo o número de ciclos alpha em uma unidade. Semelhante ao caso anterior, pelo menos uma das arestas pretas no ciclo de tamanho dois tem peso maior que zero e, pelo menos uma operação de move μ_1 deve ser aplicada. No final, uma reversão ρ_2 é aplicada no ciclo de tamanho dois, gerando dois ciclos alpha. Esse caso resulta em $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1, \mu_1, \rho_2)) \leq \frac{1}{3}$, uma vez que $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1)) = -1$, $\Delta c_\alpha(G(\mathcal{I}), S = (\mu_1)) \leq 0$ e $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_2)) = 2$.

Considerando o cenário em que uma reversão cria apenas um ciclo alpha, também temos o fato de que uma reversão que une ciclos deve ser aplicada previamente. Assim,

temos que $\Delta c_\alpha(G(\mathcal{I}), S = (\rho_1, \rho_2)) \leq \frac{1}{2}$. Isto implica que se qualquer reversão for usada em uma sequência S que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, então $|S| > 3n$.

Para indels, notamos que depois que removemos o peso de uma resta preta, ela pode aumentar o número de ciclos alpha em uma unidade. No entanto, após qualquer indel que remove peso de arestas pretas, pelo menos um indel que insere peso é necessário para balancear a instância $\mathcal{I}$ novamente e, como esse indel não pode aumentar o número de ciclos alpha, temos como resultado mais de $3n$ operações em uma sequência S que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

Nesse caso, para a instância $\mathcal{I}$ do problema $\mathbf{Sb_IRM}$ ou $\mathbf{Sb_IRMI}$, temos que uma sequência S, tal que $|S| \leq 3n$ e $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$, deve ser composta exclusivamente por moves.

As $3n$ operações de move transferirão todo o peso das $3n$ arestas pretas com rótulos de 1 até $3n$ para as arestas pretas com rótulos de $3n + 1$ até $4n$, criando um ciclo alpha cada. As arestas pretas com rótulos de $3n + 1$ até $4n$ devem receber um peso total que corresponda à soma dos pesos de exatamente três arestas pretas com rótulos 1 até $3n$, pois $\frac{B}{2} > a_i > \frac{B}{4}$. Ao final do processo, basta rastrear o peso que foi transferido para criar cada ciclo alpha para construir uma solução para a instância A do **3-PART**. □

Teorema 4.1.18. *Os problemas* $\boldsymbol{Sb_IRTI}$ *e* $\boldsymbol{Sb_IRTMI}$ *em instâncias intergênicas rígidas com sinais pertencem à classe NP-difícil.*

Demonstração. A prova é similar à descrita no Teorema 4.1.15, mas utilizando uma redução da versão de decisão da variação com sinais do problema **SbRT** e considerando que a sequência S' para a instância $(\mathcal{I}', t)$ do problema $\mathbf{Sb_IRTI}$ ou $\mathbf{Sb_IRTMI}$ é composta por reversões intergênicas e transposições intergênicas ao invés de ser composta exclusivamente por reversões intergênicas. □

4.1.3 Instâncias Intergênicas Rígidas sem Sinais

Nesta seção, apresentaremos algoritmos para os problemas em um cenário não ponderado e em instâncias intergênicas rígidas sem sinais.

Inicialmente iremos apresentar lemas utilizados por múltiplos algoritmos. O lema seguinte faz uso do conceito de um par conectado de breakpoints (Definição 2.4.9) para mostrar uma característica importante presente em instâncias intergênicas rígidas balanceadas sem sinais e também em instâncias intergênicas rígidas sem sinais em que o total de nucleotídeos no genoma de origem é maior do que o total de nucleotídeos no genoma alvo.

Lema 4.1.19. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida sem sinais tal que* $\sum_{i=1}^{n+1} \breve{\pi}_i \geq \sum_{i=1}^{n+1} \breve{\iota}_i$ *e* $ib_1(\mathcal{I}) > 1$. *É sempre possível encontrar um par conectado de breakpoints.*

Demonstração. Suponha por contradição que exista uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, tal que $\sum_{i=1}^{n+1} \breve{\pi}_i \geq \sum_{i=1}^{n+1} \breve{\iota}_i$, $ib_1(\mathcal{I}) > 1$, e que não exista nenhum par conectado de breakpoints em $\mathcal{I}$. Note que $\mathcal{I}$ não deve possuir nenhum breakpoint sobrecarregado, uma vez que, pela característica de um breakpoint sobrecarregado, ele

está conectado com qualquer outro breakpoint. Além disso, $\mathcal{I}$ deve possuir, pelo menos, um breakpoint suave. Note que, se $\mathcal{I}$ possuir apenas breakpoints subcarregados, implica que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que contradiz a suposição inicial de que $\sum_{i=1}^{n+1} \breve{\pi}_i \geq \sum_{i=1}^{n+1} \breve{\iota}_i$. Como não existe nenhum par conectado de breakpoints em $\mathcal{I}$, isso implica que as regiões intergênicas dos breakpoints suaves não possuem nucleotídeos suficiente para removê-los sem torná-los subcarregados. Dessa forma, temos que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que novamente contradiz a suposição inicial de que $\sum_{i=1}^{n+1} \breve{\pi}_i \geq \sum_{i=1}^{n+1} \breve{\iota}_i$. Logo, o lema segue. □

Lema 4.1.20. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais com apenas um breakpoint tipo um (π_i, π_{i+1}). Então (π_i, π_{i+1}) é um breakpoint forte.*

Demonstração. Suponha por contradição que $\mathcal{I}$ é uma instância intergênica rígida sem sinais com apenas um breakpoint tipo um (π_i, π_{i+1}), tal que (π_i, π_{i+1}) não é um breakpoint forte. Por definição, temos que $|\pi_{i+1} - \pi_i| \neq 1$. Seja π_j o elemento em π com valor $\pi_i + 1$. Note que $j \neq i + 1$, uma vez que $|\pi_{i+1} - \pi_i| \neq 1$. Consequentemente, (π_{j-1}, π_j) ou (π_j, π_{j+1}) forma um breakpoint tipo um, o que contradiz a suposição de $\mathcal{I}$ possuir apenas um breakpoint tipo um. □

Lema 4.1.21. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Temos que $ib_1(\mathcal{I}) \neq 1$.*

Demonstração. Como $\mathcal{I}$ é uma instância intergênica rígida balanceada, temos que a seguinte condição é verdadeira: $\sum_{i=1}^{n+1} \breve{\pi}_i = \sum_{i=1}^{n+1} \breve{\iota}_i$. Agora vamos mostrar que não existe tal instância em que $ib_1(\mathcal{I}) = 1$. Suponha por contradição que exista uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ com $ib_1(\mathcal{I}) = 1$. Como $ib_1(\mathcal{I}) = 1$, então o breakpoint tipo um (π_i, π_{i+1}), obrigatoriamente, deve ser forte (Lema 4.1.20). Caso contrário, temos que $ib_1(\mathcal{I}) > 1$. Logo, temos que $\breve{\pi}_{i+1} \neq \breve{\iota}_{i+1}$, o que implica que $\sum_{i=1}^{n+1} \breve{\pi}_i \neq \sum_{i=1}^{n+1} \breve{\iota}_i$ e contradiz a suposição inicial de que $\mathcal{I}$ é balanceada. □

Note que, pelo Lema 4.1.21, podemos concluir que não existe uma instância intergênica rígida balanceada sem sinais com apenas um breakpoint tipo um.

Reversão

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_IR}$.

Lema 4.1.22. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) um par conectado de breakpoints. Então é possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando, no máximo, duas reversões.*

Demonstração. Sem perda de generalidade, assuma que $i < j$. Como os breakpoints (π_i, π_{i+1}) e (π_j, π_{j+1}) estão conectados, por definição, um dos seguintes casos deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Neste caso, precisamos

aplicar apenas uma reversão $\rho^{(i+1,j)}_{(x,y)}$ para posicionar o elemento π_j no lado direito do elemento π_i ou posicionar o elemento π_{i+1} no lado esquerdo do elemento π_{j+1}. Como $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, então sempre é possível atribuir valores para os parâmetros x e y de forma que o tamanho da região intergênica entre os elementos consecutivos, posicionados pela reversão, tenha o mesmo tamanho que a região intergênica entre os mesmos elementos no genoma alvo (Figura 4.1(a)).

ii. O par de elementos (π_i, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Inicialmente, iremos provar que deve existir um breakpoint tipo um (π_k, π_{k+1}), tal que $k < i$ ou $k > j$. Suponha por contradição que não exista um breakpoint tipo um (π_k, π_{k+1}), tal que $k < i$ ou $k > j$. Isso implica que os segmentos $(\pi_0, \ldots, \pi_i)$ e $(\pi_{j+1}, \ldots, \pi_{n+1})$ são compostos por elementos consecutivos, ou seja, não existe breakpoints tipo um entre os elementos de ambos os segmentos. Sabemos também que π_i e π_{j+1} são elementos consecutivos em ι. Entretanto, se ambas as afirmações forem verdadeiras, então os valores dos elementos no segmento $(\pi_{i+1}, \ldots, \pi_j)$ não estão presentes em ι, o que contradiz a construção da instância $\mathcal{I}$, pois π e ι são permutações que compartilham o mesmo conjunto de valores. Após identificar o breakpoint tipo um (π_k, π_{k+1}), temos duas possibilidades: se $k < i$, aplicamos a reversão $\rho^{(k+1,i)}_{(0,\breve{\pi}_{i+1})}$, que não gera nenhum novo breakpoint, e obtemos o caso (i) (Figura 4.1(b)); se $k > j$, aplicamos a reversão $\rho^{(j+1,k)}_{(0,\breve{\pi}_{k+1})}$, que também não gera nenhum novo breakpoint, e obtemos o caso (i) (Figura 4.1(c)).

iii. O par de elementos (π_{i+1}, π_j) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Inicialmente, vamos provar que deve existir um breakpoint tipo um (π_k, π_{k+1}), tal que $i < k < j$. Suponha por contradição que não exista um breakpoint tipo um (π_k, π_{k+1}), tal que $i < k < j$. Isso implica que o segmento $(\pi_{i+1}, \pi_{i+2}, \ldots, \pi_j)$ é composto por pelo menos três elementos consecutivos, ou seja, não existe breakpoints tipo um entre os elementos do segmento. Caso contrário, (π_{i+1}, π_j) seria uma adjacência intergênica. Sabemos também que π_{i+1} e π_j são elementos consecutivos em ι. Entretanto, se ambas as afirmações forem verdadeiras, então $|\pi_j - \pi_{i+1}| > 1$, o que contradiz a suposição de que π_{i+1} e π_j são elementos consecutivos em ι. Após identificar o breakpoint tipo um (π_k, π_{k+1}), aplicamos a reversão $\rho^{(i+1,k)}_{(0,\breve{\pi}_{k+1})}$, que não gera nenhum novo breakpoint, e obtemos o caso (i) (Figura 4.1(d)).

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Inicialmente, aplicamos a reversão $\rho^{(i+1,j)}_{(0,\breve{\pi}_{j+1})}$, que não modifica o tamanho das regiões intergênicas $\breve{\pi}_{i+1}$ e $\breve{\pi}_{j+1}$, e obtemos o caso (i) (Figura 4.1(e)).

Note que o caso (i) aplica apenas uma reversão e remove, pelo menos, um breakpoint tipo um. Os casos (ii), (iii) e (iv) aplicam uma reversão que não remove nenhum breakpoint

tipo um, mas garante que nenhum novo breakpoint seja gerado e que o caso (i) poderá ser aplicado em seguida. No pior caso, duas reversões são aplicadas e, pelo menos, um breakpoint tipo um é removido de $\mathcal{I}$. Logo, o lema segue. □

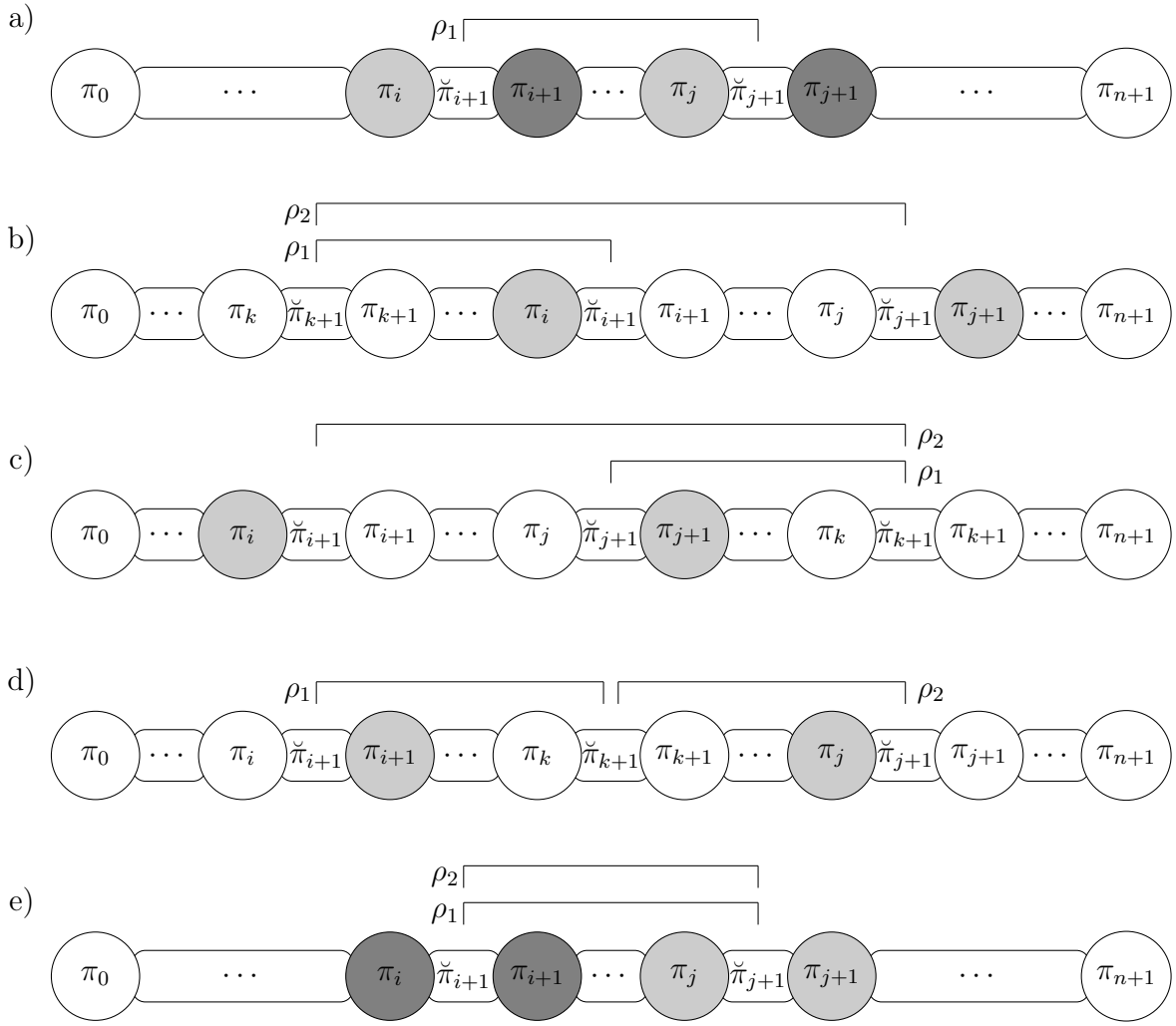


Figura 4.1: Possibilidades que podem surgir quando existe um par de breakpoints conectados e reversões que podem ser aplicadas para remover, pelo menos, um breakpoint tipo um. O par de elementos consecutivos no genoma alvo está representado por tons de cinza.

Considere o Algoritmo 4 para a variação sem sinais do problema $\mathbf{Sb_IR}$.

Lema 4.1.23. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 4 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ reversões.*

Demonstração. No Algoritmo 4, temos que enquanto $ib_1(\mathcal{I})$ for maior que um, ou seja, $(\pi, \breve{\pi})$ for diferente de $(\iota, \breve{\iota})$ (pela Observação 2.4.2 e Lema 4.1.21), o seguinte procedimento é aplicado: pelos lemas 4.1.19 e 4.1.22, sempre podemos encontrar um par conectado de breakpoints e remover, pelo menos, um breakpoint tipo um após aplicar duas reversões, no máximo. A cada iteração do algoritmo, pelo menos um breakpoint tipo um é removido. Dessa forma, o genoma alvo eventualmente será alcançado. No pior caso, cada breakpoint

Algoritmo 4: Um algoritmo de aproximação para o problema $\mathbf{Sb_IR}$.

Entrada: Uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto ib_1(I) > 1 faça
       ▷ Lema 4.1.19
3      (π_i, π_{i+1}), (π_j, π_{j+1}) ← encontre um par de breakpoints conectados
       ▷ Lema 4.1.22
4      se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (i) então
5          S' ← (ρ_1)
6      senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (ii) então
7          S' ← (ρ_1, ρ_2)
8      senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (iii) então
9          S' ← (ρ_1, ρ_2)
10     senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (iv) então
11         S' ← (ρ_1, ρ_2)
12     S ← S + S'
13     I ← ((π, π̆) · S', (ι, ῐ))
14 retorne S
```

tipo um é removido utilizando duas reversões, e $2ib_1(\mathcal{I})$ reversões são utilizadas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Logo, o lema segue. □

Note que o Algoritmo 4 pode ser analisado considerando os seguintes pontos:

- Encontrar o par conectado de breakpoints, que pode ser feito em tempo linear com o auxílio da permutação inversa de π.
- Aplicação dos casos do Lema 4.1.22, que no pior caso também pode levar um tempo linear se for necessário encontrar o breakpoint tipo um (π_k, π_{k+1}) nos casos (ii) e (iii).

Como esse processo é repetido n vezes, no máximo, então o tempo de execução do Algoritmo 4 é $\mathcal{O}(n^2)$.

Teorema 4.1.24. *O Algoritmo 4 é uma 4-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_IR}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.1.23, o Algoritmo 4 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ reversões. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{R}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

Reversão e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_IRI}$.

Lema 4.1.25. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida desbalanceada sem sinais tal que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$. É sempre possível aplicar um indel δ de forma que $\Delta ib_1(\mathcal{I}, S = (\delta)) \leq 0$ e $\mathcal{I}$ torna-se uma instância intergênica rígida balanceada.*

Demonstração. Como $\mathcal{I}$ é desbalanceada, então $ib_1(\mathcal{I}) > 0$. Seja (π_i, π_{i+1}) um breakpoint tipo um de $\mathcal{I}$. Aplique o indel $\delta_{(x)}^{(i+1)}$, tal que $x = \sum_{i=1}^{n+1} \breve{\iota}_i - \sum_{i=1}^{n+1} \breve{\pi}_i$. Note que o indel insere a quantidade necessária de nucleotídeos na região intergênica $\breve{\pi}_{i+1}$ para tornar $\mathcal{I}$ uma instância balanceada. No pior caso, (π_i, π_{i+1}) continua sendo um breakpoint tipo um e o lema segue. □

Lema 4.1.26. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida desbalanceada sem sinais tal que $ib_1(\mathcal{I}) = 1$. É sempre possível aplicar um indel δ de forma que $\Delta ib_1(\mathcal{I}, S = (\delta)) = -1$.*

Demonstração. Seja (π_i, π_{i+1}) o único breakpoint tipo um de $\mathcal{I}$. Como (π_i, π_{i+1}) é único breakpoint tipo um de $\mathcal{I}$, então ele deve ser um breakpoint forte, obrigatoriamente (Lema 4.1.20). Aplique o indel $\delta_{(x)}^{(i+1)}$, tal que $x = \breve{\iota}_{i+1} - \breve{\pi}_{i+1}$. Note que o indel insere ou remove a quantidade necessária de nucleotídeos na região intergênica $\breve{\pi}_{i+1}$ para remover o breakpoint (π_i, π_{i+1}) caso ele seja subcarregado ou sobrecarregado, respectivamente. O breakpoint (π_i, π_{i+1}) é removido após a aplicação do evento de indel e o lema segue. □

Considere o Algoritmo 5 para a variação sem sinais do problema $\mathbf{Sb_IRI}$.

Lema 4.1.27. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. O Algoritmo 5 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão e indel.*

Demonstração. Podemos analisar o Algoritmo 5 considerando três cenários:

- $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$: nesse cenário o Algoritmo 5 aplica um indel (linhas 2-5) que pode não remover nenhum breakpoint tipo um, mas transforma $\mathcal{I}$ em uma instância balanceada. Caso ainda existam breakpoints em $\mathcal{I}$, então o laço de repetição (linhas 6-17) remove, por iteração, pelo menos um breakpoint tipo um utilizando, no máximo, duas reversões. Esse processo se repete até que todos os breakpoints tipo um de $\mathcal{I}$ sejam removidos. Como todos os breakpoints tipo um são removidos, então $(\pi, \breve{\pi})$ é transformada em $(\iota, \breve{\iota})$. Note que, se o indel aplicado inicialmente não remover nenhum breakpoint tipo um, então pelo menos uma reversão é aplicada em seguida. Além disso, pelo Lema 4.1.21, podemos deduzir que a última reversão que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ deve, obrigatoriamente, remover dois breakpoints tipo um. Isso implica que, no máximo, $2ib_1(\mathcal{I})$ reversões e indels são utilizadas pelo Algoritmo 5 para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.
- $\sum_{i=1}^{n+1} \breve{\pi}_i = \sum_{i=1}^{n+1} \breve{\iota}_i$: para esse cenário o Algoritmo 5 comporta-se exatamente como o Algoritmo 4, que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ reversões.

Algoritmo 5: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRI}$.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

1 Seja $S \leftarrow ()$
▷ Lema 4.1.25
2 **se** $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$ **então**
3 $\quad S' \leftarrow (\delta_1)$
4 $\quad S \leftarrow S + S'$
5 $\quad \mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
6 **enquanto** $ib_1(\mathcal{I}) > 1$ **faça**
$\quad$ ▷ Lema 4.1.19
7 $\quad (\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
$\quad$ ▷ Lema 4.1.22
8 $\quad$ **se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (i) **então**
9 $\quad\quad S' \leftarrow (\rho_1)$
10 $\quad$ **senão se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (ii) **então**
11 $\quad\quad S' \leftarrow (\rho_1, \rho_2)$
12 $\quad$ **senão se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iii) **então**
13 $\quad\quad S' \leftarrow (\rho_1, \rho_2)$
14 $\quad$ **senão se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iv) **então**
15 $\quad\quad S' \leftarrow (\rho_1, \rho_2)$
16 $\quad S \leftarrow S + S'$
17 $\quad \mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
18 ▷ Lema 4.1.26
19 **se** $ib_1(\mathcal{I}) = 1$ **então**
20 $\quad S' \leftarrow (\delta_1)$
21 $\quad S \leftarrow S + S'$
22 $\quad \mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
23 **retorne** S

- $\sum_{i=1}^{n+1} \breve{\pi}_i > \sum_{i=1}^{n+1} \breve{\iota}_i$: nesse último cenário enquanto $ib_1(\mathcal{I})$ for maior que um, o Algoritmo 5 aplica, no máximo, duas reversões a cada iteração do laço de repetição (linhas 6-17) que removem, pelo menos, um breakpoint tipo um. Por fim, um indel é aplicado (linhas 19-22) removendo o último breakpoint tipo um e transformando $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

Nos três cenários, o Algoritmo 5 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ reversões e indels. Logo, o lema segue. □

Note que o Algoritmo 5 difere do Algoritmo 4 pelos trechos responsáveis por aplicar uma operação de indel (linhas 2-5 e 19-22). Ambos os trechos podem ser realizados em tempo linear. Dessa forma, o tempo de execução do Algoritmo 5 também é $\mathcal{O}(n^2)$.

Teorema 4.1.28. *O Algoritmo 5 é uma 4-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_IRI}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.1.27, o Algoritmo 5 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$

eventos de reversão e indel. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

Reversão e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_IRM}$.

Lema 4.1.29. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) um par conectado de breakpoints. É possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando, no máximo, duas reversões ou um move.*

Demonstração. Note que a aplicação do Lema 4.1.22 já é suficiente para provar este lema. Entretanto, iremos melhorar o caso (iv) para utilizarmos apenas um evento de move ao invés de duas reversões. Sem perda de generalidade, assuma que $i < j$. Como o par de breakpoints (π_i, π_{i+1}) e (π_j, π_{j+1}) está conectado, por definição, um dos seguintes casos deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma reversão como descrito no caso (i) do Lema 4.1.22.

ii. O par de elementos (π_i, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma sequência de duas reversões como descrito no caso (ii) do Lema 4.1.22.

iii. O par de elementos (π_{i+1}, π_j) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma sequência de duas reversões como descrito no caso (iii) do Lema 4.1.22.

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Note que, pela definição, (π_i, π_{i+1}) ou (π_j, π_{j+1}) é um breakpoint forte. Se (π_i, π_{i+1}) for um breakpoint forte, então ele pode ser sobrecarregado ou subcarregado. Caso ele seja sobrecarregado, então aplicamos um move $\mu_{(x)}^{(i+1,j+1)}$, tal que $x = \breve{\pi}_{i+1} - \breve{\iota}_{max(\pi_i,\pi_{i+1})}$. Caso contrário, aplicamos um move $\mu_{(x)}^{(j+1,i+1)}$, tal que $x = \breve{\iota}_{max(\pi_i,\pi_{i+1})} - \breve{\pi}_{i+1}$. De maneira similar, se (π_j, π_{j+1}) for um breakpoint forte, então ele pode ser sobrecarregado ou subcarregado. Caso ele seja sobrecarregado, então aplicamos um move $\mu_{(x)}^{(j+1,i+1)}$, tal que $x = \breve{\pi}_{j+1} - \breve{\iota}_{max(\pi_j,\pi_{j+1})}$. Caso contrário, aplicamos um move $\mu_{(x)}^{(i+1,j+1)}$, tal que $x = \breve{\iota}_{max(\pi_j,\pi_{j+1})} - \breve{\pi}_{j+1}$. Em ambos os cenários, pelo menos um breakpoint tipo um é removido após a aplicação do move (Figura 4.2).

Note que o caso (i) aplica apenas uma reversão e remove, pelo menos, um breakpoint tipo um. Os casos (ii) e (iii) aplicam inicialmente uma reversão que não remove nenhum breakpoint tipo um, mas garante que nenhum novo breakpoint é gerado e que o caso (i) poderá ser aplicado em seguida. Por fim, o caso (iv) remove, pelo menos, um breakpoint tipo um utilizando um move. No pior caso, duas reversões são aplicadas e, pelo menos, um breakpoint tipo um é removido de $\mathcal{I}$. Logo, o lema segue. □

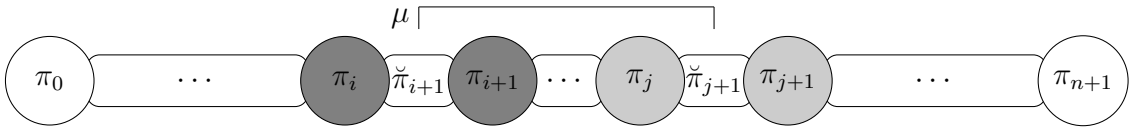


Figura 4.2: Exemplo de remoção de pelo menos um breakpoint tipo um após a aplicação de um move μ.

Considere o Algoritmo 6 para a variação sem sinais do problema $\mathbf{Sb_IRM}$.

Algoritmo 6: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRM}$.

Entrada: Uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões e moves S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
enquanto $ib_1(\mathcal{I}) > 1$ **faça**
  ▷ `Lema 4.1.19`
  $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  ▷ `Lema 4.1.29`
  se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (i) **então**
    $S' \leftarrow (\rho_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iv) **então**
    $S' \leftarrow (\mu_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (ii) **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iii) **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorne S

Lema 4.1.30. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 6 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão e move.*

Demonstração. No Algoritmo 6, temos que enquanto $ib_1(\mathcal{I})$ for maior que um, ou seja, $(\pi, \breve{\pi})$ for diferente de $(\iota, \breve{\iota})$ (pela Observação 2.4.2 e Lema 4.1.21), o seguinte procedimento é aplicado: pelos lemas 4.1.19 e 4.1.29, sempre podemos encontrar um par conectado de breakpoints e remover, pelo menos, um breakpoint tipo um após aplicar, no máximo, duas reversões ou um move. A cada iteração do algoritmo, pelo menos um breakpoint

tipo um é removido. Dessa forma, o genoma alvo eventualmente será alcançado. No pior caso, cada breakpoint tipo um é removido utilizando dois eventos de rearranjo. Logo, uma sequência com, no máximo, $2ib_1(\mathcal{I})$ reversões e moves é utilizada para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e o lema segue. □

Note que o Algoritmo 6 difere do Algoritmo 4 apenas pelo caso (iv) de um par conectado de breakpoints, que pode ser realizado em tempo constante. Logo, o tempo de execução do Algoritmo 6 também é $\mathcal{O}(n^2)$.

Teorema 4.1.31. *O Algoritmo 6 é uma 4-aproximação para a variação sem sinais do problema* ***Sb$_\mathbf{I}$RM****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.1.30, o Algoritmo 6 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão e move. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RM}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

Reversão, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema **Sb$_\mathbf{I}$RMI** (Algoritmo 7).

Lema 4.1.32. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. O Algoritmo 7 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão, move e indel.*

Demonstração. A prova é similar à descrita no Lema 4.1.27. □

Note que o Algoritmo 7, em comparação com o Algoritmo 5, difere apenas pelo caso (iv) de um par conectado de breakpoints, que pode ser realizado em tempo constante. Logo, o tempo de execução do Algoritmo 7 também é $\mathcal{O}(n^2)$.

Teorema 4.1.33. *O Algoritmo 7 é uma 4-aproximação para a variação sem sinais do problema* ***Sb$_\mathbf{I}$RMI****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.1.32, o Algoritmo 7 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão, move e indel. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RMI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

Reversão e Transposição

Nesta seção, apresentaremos algoritmos para a variação sem sinais do problema **Sb$_\mathbf{I}$RT** com fatores de aproximação 6, 4.5 e 4.

Lema 4.1.34. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) um par conectado de breakpoints. É possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando, no máximo, duas reversões ou uma transposição.*

Algoritmo 7: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRMI}$.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, moves e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
▷ Lema 4.1.25
se $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
enquanto $ib_1(\mathcal{I}) > 1$ **faça**
  ▷ Lema 4.1.19
  $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  ▷ Lema 4.1.29
  se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (i) **então**
    $S' \leftarrow (\rho_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iv) **então**
    $S' \leftarrow (\mu_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (ii) **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iii) **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
▷ Lema 4.1.26
se $ib_1(\mathcal{I}) = 1$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorne S

Demonstração. Note que a aplicação do Lema 4.1.22 já é suficiente para provar este lema. Entretanto, iremos melhorar os casos (ii) e (iii) para utilizarmos apenas um evento de transposição ao invés de duas reversões. Sem perda de generalidade, assuma que $i < j$. Como os breakpoints (π_i, π_{i+1}) e (π_j, π_{j+1}) estão conectados, por definição, um dos seguintes casos deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma reversão como descrito no caso (i) do Lema 4.1.22.

ii. O par de elementos (π_i, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Nesse caso, sabemos que deve existir um breakpoint tipo um (π_k, π_{k+1}), tal que $k < i$ ou $k > j$ (caso (ii), Lema 4.1.22). Se $k < i$, aplicamos uma transposição $\tau_{(x,y,z)}^{(k+1,i+1,j+1)}$ para posicionar o elemento π_i no

lado esquerdo do elemento π_{j+1} (Figura 4.3(a)). Se $k > j$, aplicamos uma transposição $\tau_{(x,y,z)}^{(i+1,j+1,k+1)}$ para posicionar o elemento π_{j+1} no lado direito do elemento π_i (Figura 4.3(b)). Em ambos os cenários, temos que $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$. Logo, os parâmetros x, y e z sempre podem ser escolhidos de forma que, ao posicionar lado a lado o par de elementos (π_i, π_{j+1}), o tamanho da região entre eles seja igual no genoma de origem e alvo.

iii. O par de elementos (π_{i+1}, π_j) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Nesse caso, sabemos que deve existir um breakpoint tipo um (π_k, π_{k+1}), tal que $i < k < j$ (caso (iii), Lema 4.1.22). Após identificar o breakpoint tipo um (π_k, π_{k+1}), aplicamos a transposição $\tau_{(x,y,z)}^{(i+1,k+1,j+1)}$ para posicionar o elemento π_j no lado esquerdo do elemento π_{i+1} (Figura 4.3(c)). Como $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, então os parâmetros x, y e z sempre podem ser escolhidos de forma que, ao posicionar lado a lado o par de elementos (π_{i+1}, π_j), o tamanho da região entre eles seja igual no genoma de origem e alvo.

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma sequência de duas reversões como descrito no caso (iv) do Lema 4.1.22.

Note que o caso (i) aplica apenas uma reversão que remove pelo menos um breakpoint tipo um. Os casos (ii) e (iii) aplicam apenas uma transposição que remove pelo menos um breakpoint tipo um. Por fim, o caso (iv) aplica duas reversões e remove pelo menos um breakpoint tipo um. No pior caso, duas reversões são aplicadas e, pelo menos, um breakpoint tipo um é removido de $\mathcal{I}$. Logo, o lema segue. □

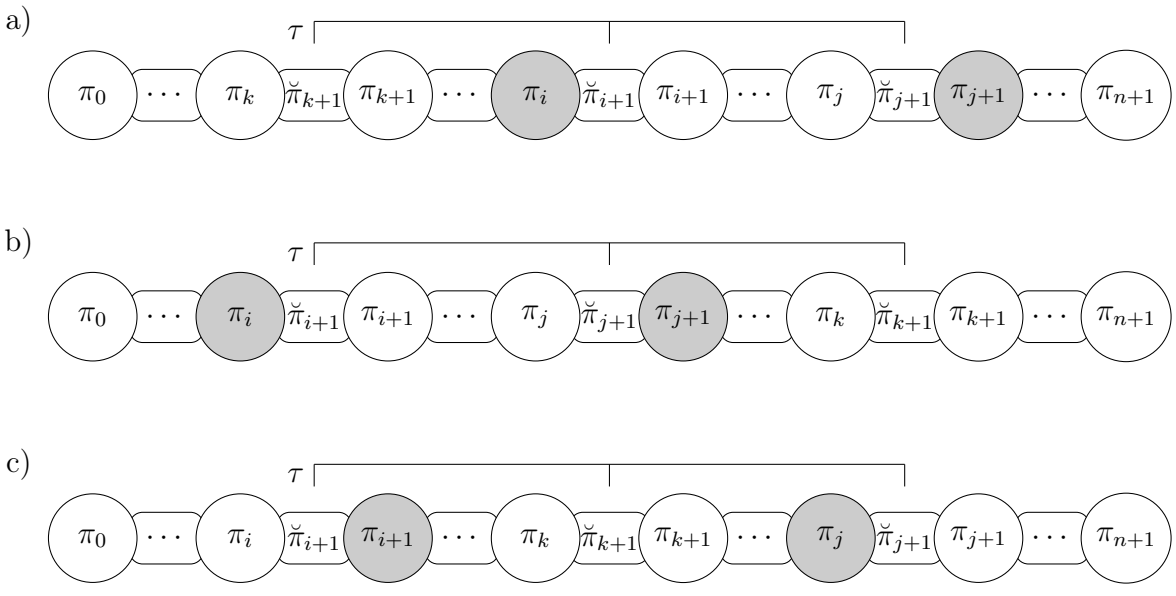


Figura 4.3: Exemplo de remoção de, pelo menos, um breakpoint tipo um após a aplicação de uma transposição τ.

Considere o Algoritmo 8 para a variação sem sinais do problema $\mathbf{Sb_IRT}$.

Algoritmo 8: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRT}$.

Entrada: Uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões e transposições S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto ib_1(I) > 1 faça
       ▷ Lema 4.1.19
3      (π_i, π_{i+1}), (π_j, π_{j+1}) ← encontre um par de breakpoints conectados
       ▷ Lema 4.1.34
4      se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (i) então
5          S' ← (ρ_1)
6      senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (ii) então
7          S' ← (τ_1)
8      senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (iii) então
9          S' ← (τ_1)
10     senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso (iv) então
11         S' ← (ρ_1, ρ_2)
12     S ← S + S'
13     I ← ((π, π̆) · S', (ι, ῐ))
14 retorne S
```

Lema 4.1.35. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 8 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão e transposição.*

Demonstração. No Algoritmo 8, temos que enquanto $ib_1(\mathcal{I})$ for maior que um, ou seja, $(\pi, \breve{\pi})$ for diferente de $(\iota, \breve{\iota})$ (pela Observação 2.4.2 e Lema 4.1.21), o seguinte procedimento é aplicado: pelos lemas 4.1.19 e 4.1.34, sempre podemos encontrar um par conectado de breakpoints e remover pelo menos um breakpoint tipo um após aplicar, no máximo, duas reversões ou uma transposição. A cada iteração do algoritmo, pelo menos um breakpoint tipo um é removido. Dessa forma, o genoma alvo eventualmente será alcançado. No pior caso, cada breakpoint tipo um é removido utilizando dois eventos de rearranjo. Logo, uma sequência com, no máximo, $2ib_1(\mathcal{I})$ reversões e transposições é utilizada para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e o lema segue. □

Note que o Algoritmo 8, em comparação com o Algoritmo 4, difere apenas pelos casos (ii) e (iii) de um par conectado de breakpoints, que também podem ser realizados em tempo linear. Logo, o tempo de execução do Algoritmo 8 também é $\mathcal{O}(n^2)$.

Teorema 4.1.36. *O Algoritmo 8 é uma 6-aproximação para a variação sem sinais do problema $\mathbf{Sb_IRT}$.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.1.35, o Algoritmo 8 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão e transposição. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

A seguir, apresentaremos lemas que serão utilizados para obtermos um algoritmo de aproximação com fator 4.5 para a variação sem sinais do problema $\mathbf{Sb_IRT}$.

Lema 4.1.37. *Seja $(\pi, \breve{\pi})$ uma representação intergênica rígida e sejam duas transposições consecutivas no formato $\tau^{(i,j,k)}_{(\varphi_i,\varphi_j,\varphi_k)}$ e $\tau^{(i,i+k-j,k)}_{(\varphi'_i,\varphi'_{i+k-j},\varphi'_k)}$. É possível realizar qualquer redistribuição de nucleotídeos nas regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$ após aplicas consecutivamente as transposições $\tau^{(i,j,k)}_{(\varphi_i,\varphi_j,\varphi_k)}$ e $\tau^{(i,i+k-j,k)}_{(\varphi'_i,\varphi'_{i+k-j},\varphi'_k)}$ em $(\pi, \breve{\pi})$.*

Demonstração. Temos que mostrar que sempre é possível encontrar valores para as triplas $(\varphi_i, \varphi_j, \varphi_k)$ e $(\varphi'_i, \varphi'_{i+k-j}, \varphi'_k)$ para qualquer redistribuição de nucleotídeos nas regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$.

Como usamos apenas transposições, sabemos que $\breve{\pi}_i + \breve{\pi}_j + \breve{\pi}_k = \breve{\pi}''_i + \breve{\pi}''_j + \breve{\pi}''_k$, onde $\breve{\pi}''_i$, $\breve{\pi}''_j$ e $\breve{\pi}''_k$ representam os tamanhos das regiões intergênicas após a aplicação das duas transposições consecutivas.

Com base no fluxo de nucleotídeos entre as regiões intergênicas, realizaremos uma redução para uma instância I_{MF} do problema de fluxo máximo, e mostraremos que sempre é possível encontrar uma solução para essa instância que satisfaça as restrições de redistribuição das regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$. Uma instância do problema de fluxo máximo é dada em um grafo onde cada aresta possui uma capacidade máxima de fluxo que pode ser transferido entre o par de vértices conectados. O objetivo consiste em determinar o fluxo máximo que pode ser enviado de um vértice de origem até um vértice de destino no grafo.

A Figura 4.4 (esquerda) mostra o fluxo que os nucleotídeos nas regiões intergênicas podem seguir ao aplicar as duas transposições consecutivas (declaradas no enunciado). Note que cada região intergênica pode enviar nucleotídeos para duas regiões intergênicas distintas. Assim, podemos criar um grafo, com vértices numerados de 1 até 9, onde cada vértice corresponde a uma região intergênica e onde existe um arco (f, g) com capacidade ilimitada se a região intergênica f puder enviar nucleotídeos para a região intergênica g.

Por fim, para obter a instância I_{MF} do problema de fluxo máximo, adicionaremos os vértices 0 (origem) e 10 (destino), juntamente com os seguintes arcos: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(7, 10)$, $(8, 10)$ e $(9, 10)$ com suas respectivas capacidades: $a = \breve{\pi}_i$, $b = \breve{\pi}_j$, $c = \breve{\pi}_k$, $x = \breve{\pi}''_i$, $y = \breve{\pi}''_j$ e $z = \breve{\pi}''_k$. Todos os outros arcos têm capacidade infinita atribuída. A Figura 4.4 (direita) mostra a instância I_{MF} obtida do problema de fluxo máximo.

Se analisarmos a Figura 4.4 (direita), podemos ver que o fluxo máximo da instância I_{MF} é limitado a $\max\{(a+b+c), (x+y+z)\}$, mas sabemos que $(a+b+c) = (x+y+z) = F$. Observe também que, se houver uma redistribuição das regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$, isso significa que a instância I_{MF} tem uma solução onde o máximo fluxo é F. Por outro lado, podemos ver que, se a instância I_{MF} tiver uma solução com fluxo máximo de F e todas as variáveis da solução forem inteiras, isso significa que é possível redistribuir as regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$.

Agora, mostraremos que a instância I_{MF} sempre tem uma solução com fluxo máximo F, onde todas as variáveis da solução são inteiras. Vamos provar este resultado fornecendo uma solução para a instância I_{MF}, que é obtida em três etapas.

A etapa 1 consiste em remover os vértices 8 e 9 de I_{MF} (Figura 4.5 (esquerda)), e resolver a instância usando Programação Linear (PL) para obter uma possível solução

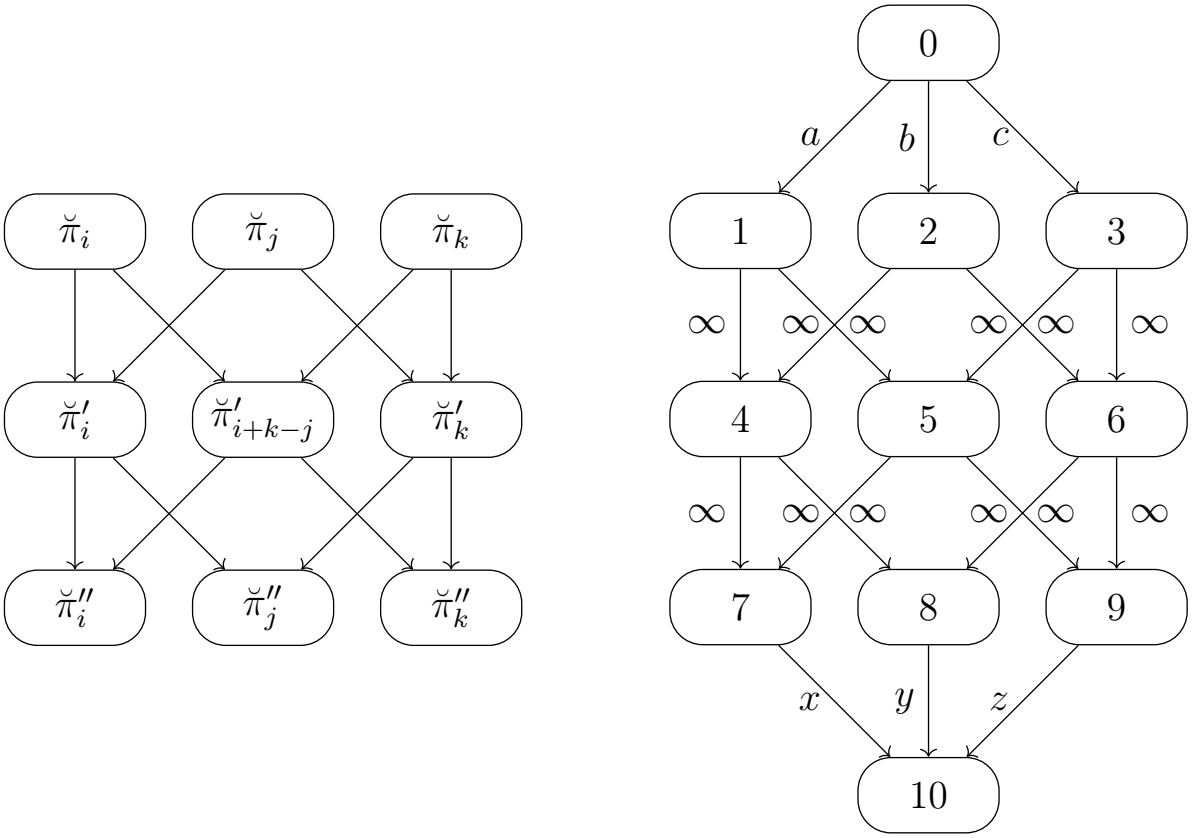


Figura 4.4: No lado esquerdo, uma representação do fluxo de nucleotídeos entre as regiões intergênicas. No lado direito, a instância I_{MF} do problema de fluxo máximo com os vértices de origem (0) e destino (10).

fracionária. Observe que o fluxo máximo para essa etapa é menor ou igual a x, pois $(a + b + c) \geq x$ e x é o valor do corte mínimo para a instância. Além disso, existe uma solução que atinge exatamente x (por exemplo, envie até a pelo caminho $(1, 4, 7)$; envie até b pelo caminho $(2, 4, 7)$; envie até c pelo caminho $(3, 5, 7)$). Seja X' a matriz da solução, na qual $X'_{i,j}$ representa o fluxo que vai do vértice i ao vértice j na solução. Sabemos que $(X'_{0,1} + X'_{0,2} + X'_{0,3}) = x$ e $x \in \mathbb{N}$. Se $\{X'_{0,1}, X'_{0,2}, X'_{0,3}\} \not\subset \mathbb{N}$, podemos obter valores inteiros para as variáveis $X'_{0,1}$, $X'_{0,2}$ e $X'_{0,3}$ redistribuindo a parte fracionária entre os arcos, isso é possível porque sabemos que $(a + b + c) \geq x$. Como todos os caminhos que partem dos vértices 1, 2 e 3 e chegam ao vértice 7 têm capacidade ilimitada, podemos obter uma solução inteira para as variáveis restantes. Após esse processo, obtemos uma solução inteira X' para a etapa 1.

A etapa 2 consiste em remover os vértices 7 e 9 de I_{MF}, e atualizar a capacidade dos arcos $(0, 1)$, $(0, 2)$ e $(0, 3)$ para $a' = a - X'_{0,1}$, $b' = b - X'_{0,2}$ e $c' = c - X'_{0,3}$, respectivamente (Figura 4.5 (centro)). Em outras palavras, levamos em consideração, para os arcos $(0, 1)$, $(0, 2)$ e $(0, 3)$, as capacidades que já foram utilizadas na etapa 1. Observe que $a' + b' + c' = a + b + c - x$, mas também sabemos que $a + b + c = x + y + z$, assim $a' + b' + c' = a + b + c - x = y + z \geq y$. Observe que o fluxo máximo para essa etapa é menor ou igual a y, pois $a' + b' + c' \geq y$ e y é o valor do corte mínimo para a instância. Além disso, existe uma solução que atinge exatamente y (por exemplo, envie até a' pelo caminho $(1, 4, 8)$;

envie até b' pelo caminho $(2, 4, 8)$; envie até c' pelo caminho $(3, 6, 8)$). Similarmente ao processo realizado na etapa 1, resolvemos o problema para obter uma solução X'' onde o fluxo máximo é y e todas as variáveis são inteiras.

A etapa 3 consiste em remover os vértices 7 e 8 de I_{MF}, e atualizar a capacidade dos arcos $(0, 1)$, $(0, 2)$ e $(0, 3)$ para $a'' = a' - X''_{0,1}$, $b'' = b' - X''_{0,2}$ e $c'' = c' - X''_{0,3}$, respectivamente (Figura 4.5 (direita)). Em outras palavras, levamos em consideração, para os arcos $(0, 1)$, $(0, 2)$ e $(0, 3)$, as capacidades que já foram utilizadas nas etapas 1 e 2. Observe que $a'' + b'' + c'' = a + b + c - x - y$, mas também sabemos que $a + b + c = x + y + z$, portanto $a'' + b'' + c'' = a + b + c - x - y = z$. Observe que o fluxo máximo para essa etapa é igual a z, pois $a'' + b'' + c'' = z$, e existe uma solução que atinge exatamente z (por exemplo, envie a'' pelo caminho $(1, 5, 9)$; envie b'' pelo caminho $(2, 6, 9)$; envie c'' pelo caminho $(3, 6, 9)$). Da mesma forma que o processo realizado na etapa 1, resolvemos o problema para obter uma solução X''' onde o fluxo máximo é z e todas as variáveis são números inteiros.

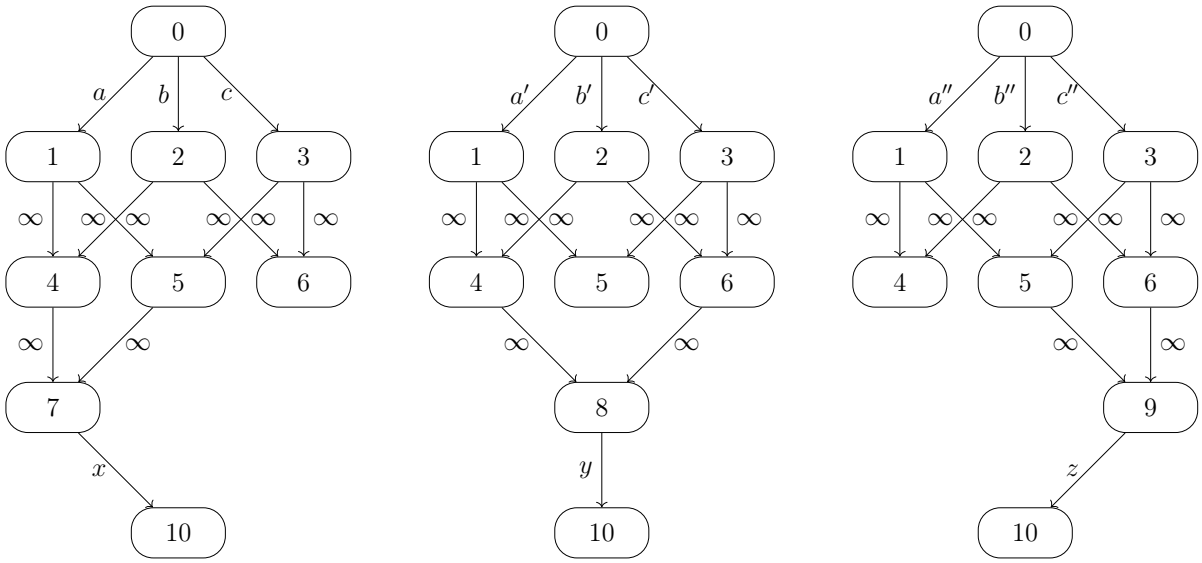


Figura 4.5: Divisão da instância I_{MF} do problema de fluxo máximo em três etapas.

A solução final X consiste na soma de todas as capacidades utilizadas pelas soluções nas etapas 1, 2 e 3, ou seja, $\forall\ 1 \leq i, j \leq 10$, $X_{i,j} = X'_{i,j} + X''_{i,j} + X'''_{i,j}$. Observe que a solução X não viola nenhuma restrição de capacidade, todas as variáveis são inteiras e o fluxo máximo é $F = a + b + c = x + y + z$.

□

Em resumo, o Lema 4.1.37 nos permite, com duas transposições consecutivas, redistribuir o tamanho de três regiões intergênicas mantendo os genes na mesma ordem e orientação.

Lema 4.1.38. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais com pelo menos dois breakpoints sobrecarregados. Existe uma sequência de duas transposições que remove pelo menos dois breakpoints tipo um de $\mathcal{I}$.*

Demonstração. Sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) dois breakpoints sobrecarregados de $\mathcal{I}$. Agora, observe que deve existir um terceiro breakpoint tipo um (π_k, π_{k+1}) em $\mathcal{I}$. Caso contrário, $\mathcal{I}$ seria uma instância desbalanceada. Pelo Lema 4.1.37, sabemos que é possível realizar qualquer redistribuição de nucleotídeos em três regiões intergênicas utilizando duas transposições consecutivas. Dessa forma, podemos realizar a redistribuição do tamanho das regiões intergênicas $\breve{\pi}_{i+1}$, $\breve{\pi}_{j+1}$ e $\breve{\pi}_{k+1}$ para $\breve{\iota}_{\max(\pi_i,\pi_{i+1})}$, $\breve{\iota}_{\max(\pi_j,\pi_{j+1})}$ e $\breve{\pi}_{k+1} + (\breve{\pi}_{i+1} - \breve{\iota}_{\max(\pi_i,\pi_{i+1})}) + (\breve{\pi}_{j+1} - \breve{\iota}_{\max(\pi_j,\pi_{j+1})})$, respectivamente. Nesse caso, o excesso de nucleotídeos nos breakpoints sobrecarregados é transferido para o breakpoint (π_k, π_{k+1}). Como resultado, pelo menos dois breakpoints tipo um são removidos após a aplicação de duas transposições, e o lema segue. □

Lema 4.1.39. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais com apenas um breakpoint sobrecarregado (π_i, π_{i+1}) e com pelo menos um breakpoint subcarregado (π_j, π_{j+1}) tal que $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_x + \breve{\iota}_y$, onde $x = \max(\pi_i, \pi_{i+1})$ e $y = \max(\pi_j, \pi_{j+1})$. Existe uma sequência de duas transposições ou duas reversões que remove pelo menos dois breakpoints tipo um de $\mathcal{I}$.*

Demonstração. Note que o excesso de nucleotídeos na região intergênica $\breve{\pi}_{i+1}$ é maior ou igual à quantidade de nucleotídeos que falta na região intergênica $\breve{\pi}_{j+1}$. Caso $ib_1(\mathcal{I}) \geq 3$, utilizaremos, para selecionar o terceiro breakpoint tipo um (π_k, π_{k+1}) $(k \notin \{i, j\})$, a seguinte ordem de prioridade: breakpoint suave, breakpoint sobrecarregado e breakpoint subcarregado. Note que o terceiro breakpoint tipo um deve ser de um dos tipos da lista de prioridades, obrigatoriamente. Em seguida, aplicamos o mesmo processo descrito no Lema 4.1.38. Caso contrário ($ib_1(\mathcal{I}) < 3$), isso implica que o excesso de nucleotídeos na região intergênica $\breve{\pi}_{i+1}$ é justamente a quantidade de nucleotídeos que falta na região intergênica $\breve{\pi}_{j+1}$. Logo, podemos aplicar as duas reversões descritas no caso *iv* do Lema 4.1.22, e o lema segue. □

Lema 4.1.40. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais com apenas um breakpoint sobrecarregado (π_i, π_{i+1}) e sem nenhum breakpoint subcarregado (π_j, π_{j+1}) tal que $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_x + \breve{\iota}_y$, onde $x = \max(\pi_i, \pi_{i+1})$ e $y = \max(\pi_j, \pi_{j+1})$. Existe uma sequência de duas reversões que remove o breakpoint sobrecarregado (π_i, π_{i+1}) de $\mathcal{I}$ e não gera nenhum outro.*

Demonstração. Note que um breakpoint sobrecarregado sempre vai estar conectado com qualquer outro breakpoint tipo um. Além disso, um segundo breakpoint tipo um (π_k, π_{k+1}) deve existir (subcarregado ou suave). Dessa forma, pelo caso (iv) do Lema 4.1.22, temos que os breakpoints (π_i, π_{i+1}) e (π_k, π_{k+1}) estão conectados e o breakpoint sobrecarregado (π_i, π_{i+1}) é removido por uma sequência de duas reversões. Como não existe nenhum breakpoint subcarregado (π_j, π_{j+1}) em $\mathcal{I}$, tal que $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_{\max(\pi_i,\pi_{i+1})} + \breve{\iota}_{\max(\pi_j,\pi_{j+1})}$, isso implica que a aplicação das duas reversões não gera breakpoints sobrecarregados, e o lema segue. □

Lema 4.1.41. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais, sem breakpoints sobrecarregados e com $ib_1(\mathcal{I}) > 0$. Existe em $\mathcal{I}$ pelo menos um par suavemente conectado de breakpoints.*

Demonstração. Dada uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, sem breakpoints sobrecarregados e com $ib_1(\mathcal{I}) > 0$. Suponha por contradição que em $\mathcal{I}$ não existe um par suavemente conectado de breakpoints. Como $\mathcal{I}$ não possui breakpoints sobrecarregados, devem existir pelo menos dois breakpoints suaves. Caso contrário, $\mathcal{I}$ teria apenas breakpoints subcarregados e isso implicaria que $\mathcal{I}$ é uma instância desbalanceada, ou seja, $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que não é possível, dado que $\mathcal{I}$ é uma instância intergênica rígida balanceada. Entretanto, como estamos supondo que não existe em $\mathcal{I}$ um par suavemente conectado de breakpoints, então os nucleotídeos presentes nas regiões intergênicas dos breakpoints suaves não são suficientes para removê-los sem torná-los breakpoints subcarregados. Logo, temos que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que é uma contradição por $\mathcal{I}$ ser uma instância intergênica rígida balanceada. □

Lema 4.1.42. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) um par suavemente conectado de breakpoints. É possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando, no máximo, uma reversão ou uma transposição.*

Demonstração. O Lema 4.1.34 apresenta os quatro casos que abrangem todas as possibilidades a partir de um par conectado de breakpoints. Em particular, os casos (i), (ii) e (iii) são os únicos em que é possível que ambos os breakpoints tipo um sejam suaves. Nos três casos, apenas uma reversão ou uma transposição é utilizada para remover pelo menos um breakpoint tipo um de $\mathcal{I}$. Logo, o lema segue. □

Considere o Algoritmo 9 para a variação sem sinais do problema $\mathbf{Sb_IRT}$.

Lema 4.1.43. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 9 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{3ib_1(\mathcal{I})}{2}$ eventos de reversão e transposição.*

Demonstração. Podemos realizar a análise de cada iteração do Algoritmo 9 considerando duas fases:

- A remoção de breakpoints sobrecarregados: Caso existam dois ou mais breakpoints sobrecarregados em $\mathcal{I}$, duas transposições são aplicadas removendo dois breakpoints sobrecarregados (linhas 3-4). Caso exista apenas um breakpoint sobrecarregado em $\mathcal{I}$, então é verificado se existe um breakpoint subcarregado de forma que o excesso de nucleotídeos na região intergênica do breakpoint sobrecarregado seja suficiente para remover o breakpoint subcarregado. Caso exista este breakpoint subcarregado, então duas transposições ou duas reversões são aplicadas removendo tanto o breakpoint sobrecarregado como o subcarregado (linhas 6-7). Caso contrário, o breakpoint sobrecarregado é removido com duas reversões, sem gerar nenhum breakpoint sobrecarregado (linhas 8-9).

- A remoção de breakpoints suaves: Se o algoritmo chegou até esse ponto (linha 11), isso significa que não existe nenhum breakpoint sobrecarregado em $\mathcal{I}$ e deve existir pelo menos um par suavemente conectado de breakpoints. Dado um par suavemente conectado de breakpoints, então é possível remover um breakpoint tipo um utilizando, no máximo, uma reversão ou uma transposição.

Algoritmo 9: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRT}$.

Entrada: Uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões e transposições S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
enquanto $ib_1(\mathcal{I}) > 1$ **faça**
  se *existir pelo menos dois breakpoints sobrecarregados em* $\mathcal{I}$ **então**
    ▷ `Lema 4.1.38`
    $S' \leftarrow (\tau_1, \tau_2)$
  senão se *existir apenas um breakpoint sobrecarregado* (π_i, π_{i+1}) *em* $\mathcal{I}$ **então**
    se *existir um breakpoint subcarregado* (π_j, π_{j+1}) *em* $\mathcal{I}$*, tal que* $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_x + \breve{\iota}_y$*, onde* $x = \max(\pi_i, \pi_{i+1})$ *e* $y = \max(\pi_j, \pi_{j+1})$ **então**
      ▷ `Lema 4.1.39`
      $S' \leftarrow (\tau_1, \tau_2)$ ou (ρ_1, ρ_2)
    senão
      ▷ `Lema 4.1.40`
      $S' \leftarrow (\rho_1, \rho_2)$
  senão
    ▷ `Lemas 4.1.41 e 4.1.42`
    $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par suavemente conectado de breakpoints
    se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (i) **então**
      $S' \leftarrow (\rho_1)$
    senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (ii) **então**
      $S' \leftarrow (\tau_1)$
    senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iii) **então**
      $S' \leftarrow (\tau_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorne S

Note que, a cada iteração do Algoritmo 9, pelo menos um breakpoint tipo um é removido. Dessa forma, o genoma alvo eventualmente será alcançado. Além disso, observe que, no pior caso, pelo menos um breakpoint tipo um é removido por duas reversões na fase de remoção de breakpoints sobrecarregados e pelo menos um breakpoint tipo um é removido por uma reversão ou uma transposição na fase de remoção de breakpoints suaves. Entretanto, se o pior caso da fase de remoção de breakpoints sobrecarregados ocorrer, sabemos que: (i) o genoma alvo ainda não foi alcançado, ou seja, $(\pi, \breve{\pi})$ é diferente de $(\iota, \breve{\iota})$; (ii) $\mathcal{I}$ não possui mais nenhum breakpoint sobrecarregado. Com essas duas constatações temos que o pior caso da fase de remoção de breakpoints sobrecarregados é, obrigatoriamente, seguido por uma fase de remoção de breakpoints suaves. Logo, no pior caso, temos que pelo menos dois breakpoints tipo um são removidos após a aplicação de, no máximo, três eventos de reversão e transposição. Como inicialmente $\mathcal{I}$ possui $ib_1(\mathcal{I})$ breakpoints tipo um, então no máximo $\frac{3ib_1(\mathcal{I})}{2}$ eventos de reversão e transposição são utilizados pelo Algoritmo 9 para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, e o lema segue. □

Note que as fases de remoção de breakpoints sobrecarregados e suaves podem ser realizadas em tempo linear. Como a quantidade máxima de breakpoints tipo um em uma instância é $n+1$ e o algoritmo, a cada iteração, remove pelo menos um breakpoint tipo um, então o tempo de execução do Algoritmo 9 é $\mathcal{O}(n^2)$.

Teorema 4.1.44. *O Algoritmo 9 é uma* 4.5*-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_IRT}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.1.43, o Algoritmo 9 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{3ib_1(\mathcal{I})}{2}$ eventos de reversão e transposição. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

A seguir, apresentaremos lemas que serão utilizados para obtermos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_IRT}$.

Lema 4.1.45. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida balanceada sem sinais. Se* $ib_1(\mathcal{I}) > 0$ *e não existir nenhum par suavemente conectado de breakpoints em* $\mathcal{I}$*, então deve existir pelo menos um breakpoint sobrecarregado em* $\mathcal{I}$.

Demonstração. Suponha por contradição que não exista um breakpoint sobrecarregado em $\mathcal{I}$. Note que $\mathcal{I}$ não pode ter apenas breakpoints subcarregados, pois isso implica que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que contradiz o fato de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. Nesse caso, devem existir pelo menos dois breakpoints suaves. Entretanto, como não existe em $\mathcal{I}$ um par suavemente conectado de breakpoints, isso significa que os nucleotídeos presentes nas regiões intergênicas dos breakpoints suaves não são suficientes para removê-los sem torná-los em breakpoints subcarregados. Logo, temos que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que contradiz o fato de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. □

Lema 4.1.46. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida balanceada sem sinais. Se* $\mathcal{I}$ *possui apenas um breakpoint sobrecarregado* (π_i, π_{i+1})*, pelo menos um breakpoint subcarregado* (π_j, π_{j+1}) *e nenhum par suavemente conectado de breakpoints, então* $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_x + \breve{\iota}_y$*, onde* $x = \max(\pi_i, \pi_{i+1})$ *e* $y = \max(\pi_j, \pi_{j+1})$.

Demonstração. Suponha por contradição que $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} < \breve{\iota}_x + \breve{\iota}_y$. Como não existe nenhum par suavemente conectado de breakpoints em $\mathcal{I}$, temos que $\mathcal{I}$ não possui breakpoints suaves ou a quantidade de nucleotídeos presente em suas regiões intergênicas é insuficiente para removê-los. Em ambos os casos, ao mover o excesso de nucleotídeos da região intergênica $\breve{\pi}_{i+1}$ do breakpoint sobrecarregado (π_i, π_{i+1}) para a região intergênica $\breve{\pi}_{j+1}$ do breakpoint subcarregado (π_j, π_{j+1}), temos que $\mathcal{I}$ ainda permanece com, pelo menos, um breakpoint subcarregado e possivelmente breakpoints suaves que não estão suavemente conectados. Logo, temos que $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, o que contradiz o fato de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. □

Lema 4.1.47. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida balanceada sem sinais. Se* $\mathcal{I}$ *possui apenas um breakpoint sobrecarregado* (π_i, π_{i+1})*, pelo menos um breakpoint subcarregado* (π_j, π_{j+1}) *e nenhum par suavemente conectado de breakpoints, então existe uma sequência de duas transposições ou duas reversões que remove, pelo menos, dois breakpoints tipo um de* $\mathcal{I}$.

Demonstração. Seja $x = \max(\pi_i, \pi_{i+1})$ e $y = \max(\pi_j, \pi_{j+1})$, pelo Lema 4.1.46, sabemos que $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_x + \breve{\iota}_y$. Logo, podemos aplicar o Lema 4.1.39, e o lema segue. □

Note que a sequência de transposições aplicadas pelo Lema 4.1.47 pode até gerar um breakpoint sobrecarregado. Entretanto, se isso ocorrer implica que a instância $\mathcal{I}$ não possui breakpoints suaves. Esse fato é decorrente da lista de prioridades para a seleção do terceiro breakpoint no Lema 4.1.39, uma vez que adicionar ou remover nucleotídeos em uma região intergênica de um breakpoint suave não o transforma em um breakpoint forte.

Observação 4.1.1. Nenhum breakpoint super forte pode ser removido por uma operação de reversão ou transposição resultante do Lema 4.1.42.

Lema 4.1.48. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Se $\mathcal{I}$ não possui um par suavemente conectado de breakpoints, então é possível, após aplicar uma reversão ou uma transposição, criar um breakpoint subcarregado mantendo $\mathcal{I}$ sem um par suavemente conectado de breakpoints ou criar um breakpoint super forte subcarregado.*

Demonstração. Se houver pelo menos uma strip suave decrescente em $\mathcal{I}$, então existe um par de breakpoints suaves (π_i, π_{i+1}) e (π_j, π_{j+1}), com $i < j$, tal que (π_i, π_j) ou (π_{i+1}, π_{j+1}) são consecutivos em ι [50]. Se (π_i, π_j) são consecutivos em ι, então aplicamos uma reversão $\rho^{(i+1,j)}_{(\breve{\pi}_{i+1}, \breve{\pi}_{j+1})}$. Caso contrário, (π_{i+1}, π_{j+1}) são consecutivos em ι e aplicamos uma reversão $\rho^{(i+1,j)}_{(0,0)}$. Observe que, em ambos os casos, todos os nucleotídeos são movidos para o breakpoint forte subcarregado criado, o que garante que a instância permaneça sem um par suavemente conectado de breakpoints. Se não houver uma strip decrescente em $\mathcal{I}$, sempre é possível encontrar três breakpoints suaves (π_i, π_{i+1}), (π_j, π_{j+1}) e (π_k, π_{k+1}), de modo que uma transposição $\tau^{(i+1,j+1,k+1)}_{(0,0,0)}$ cria um breakpoint forte subcarregado e nenhum breakpoint forte é removido [71]. Além disso, como a instância possui apenas strips suaves crescentes, temos a garantia de que o breakpoint forte subcarregado criado (unindo duas strips suaves crescentes) é um breakpoint super forte subcarregado, e o lema segue. □

Lema 4.1.49. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais com apenas um breakpoint sobrecarregado, sem nenhum breakpoint subcarregado e sem nenhum par suavemente conectado de breakpoints. Existe uma sequência com no máximo três operações que remove pelo menos dois breakpoints tipo um ou uma sequência com no máximo quatro operações que remove pelo menos três breakpoints tipo um.*

Demonstração. Inicialmente, podemos notar que $ib_1(\mathcal{I}) \geq 3$, uma vez que é impossível criar uma instância intergênica rígida balanceada com apenas um breakpoint sobrecarregado e um breakpoint suave. Aplicando o Lema 4.1.48, temos duas possibilidades: (i) um breakpoint subcarregado é criado, mantendo $\mathcal{I}$ sem um par suavemente conectado de breakpoints e, assim, podemos aplicar o Lema 4.1.47 (resultando em uma sequência com três operações que remove, pelo menos, dois breakpoints tipo um); (ii) um breakpoint super forte subcarregado é criado. Nesse caso, se $\mathcal{I}$ continuar sem nenhum par suavemente conectado de breakpoints, então podemos aplicar o Lema 4.1.47 (resultando também em uma sequência com três operações que remove, pelo menos, dois breakpoints tipo um).

Caso contrário, se $\mathcal{I}$ tiver um par suavemente conectado do breakpoints, o Lema 4.1.42 pode ser aplicado. Pela Observação 4.1.1, nenhum breakpoint super forte pode ser removido por uma operação de reversão ou transposição resultante do Lema 4.1.42. Logo, um dos seguintes casos pode ocorrer:

- Um novo breakpoint sobrecarregado é criado e podemos aplicar o Lema 4.1.38 (resultando em uma sequência com quatro operações que remove, pelo menos, três breakpoints tipo um).
- Um par suavemente conectado de breakpoints é criado e o Lema 4.1.42 é aplicado (resultando em uma sequência com três operações que remove, pelo menos, dois breakpoints tipo um).
- Não existe nenhum par suavemente conectado de breakpoints em $\mathcal{I}$ e o Lema 4.1.47 é aplicado (resultando em uma sequência com quatro operações que remove, pelo menos, três breakpoints tipo um).

□

Observação 4.1.2. Note que, se apenas dois breakpoints tipo um forem removidos pelo Lema 4.1.49, então isso implica que o genoma alvo ainda não foi alcançado, ou seja, $(\pi, \breve{\pi})$ é diferente de $(\iota, \breve{\iota})$.

Agora considere o Algoritmo 10, que consiste em quatro casos dependendo do número de breakpoints sobrecarregados ou da existência de um par suavemente conectado de breakpoints.

Lema 4.1.50. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 10 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{4ib_1(\mathcal{I})}{3}$ eventos de reversão e transposição.*

Demonstração. O Algoritmo 10 pode ser analisado considerando os seguintes casos:

- $\mathcal{I}$ tem pelo menos dois breakpoints sobrecarregados (linhas 3-4).
- $\mathcal{I}$ tem pelo menos um par suavemente conectado de breakpoints (linhas 5-6).
- $\mathcal{I}$ tem apenas um breakpoint sobrecarregado, pelo menos um breakpoint subcarregado e nenhum par suavemente conectado de breakpoints (linhas 8-9).
- $\mathcal{I}$ tem apenas um breakpoint sobrecarregado, nenhum breakpoint subcarregado e nenhum par suavemente conectado de breakpoints (linhas 10-11).

Note que, a cada iteração do Algoritmo 10, pelo menos um dos quatro casos deve obrigatoriamente ser aplicado e, pelo menos, um breakpoint tipo um é removido. Dessa forma, o genoma alvo eventualmente será alcançado e o algoritmo para. Observe que, se o algoritmo atingir os casos 3 ou 4, há exatamente um breakpoint sobrecarregado em $\mathcal{I}$ (Lema 4.1.45) e nenhum par suavemente conectado de breakpoints. Caso contrário, o caso 1 ou 2 seria aplicado.

Algoritmo 10: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRT}$.

Entrada: Uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões e transposições S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto ib_1(I) > 1 faça
3      se existir pelo menos dois breakpoints sobrecarregados em I então
           ▷ Lema 4.1.38
4          Seja S' uma sequência com duas transposições que remove, pelo menos, dois
           breakpoints tipo um de I
5      senão se existir um par suavemente conectado de breakpoints em I então
           ▷ Lema 4.1.42
6          Seja S' uma sequência com uma reversão ou uma transposição que remove,
           pelo menos, um breakpoint tipo um de I
7      senão
           ▷ Existe exatamente um breakpoint sobrecarregado (Lema 4.1.45)
8          se existir pelo menos um breakpoint subcarregado em I então
               ▷ Lema 4.1.47
9              Seja S' uma sequência com duas transposições ou duas reversões que
               remove, pelo menos, dois breakpoints tipo um de I
10         senão
               ▷ Lema 4.1.49
11             Seja S' uma sequência com, no máximo, três operações que remove, pelo
               menos, dois breakpoints tipo um de I ou uma sequência com, no máximo,
               quatro operações que remove, pelo menos, três breakpoints tipo um de I
12
13     S ← S + S'
14     I ← ((π, π̆) · S', (ι, ῐ))
15 retorne S
```

Os casos 1, 2 e 3 removem, em média, um breakpoint tipo um por operação. No pior cenário do caso 4 (onde dois breakpoints tipo um são removidos com três operações), temos, pela Observação 4.1.2, que $(\pi, \breve{\pi}) \neq (\iota, \breve{\iota})$, e o caso 1, 2 ou 3 será aplicado posteriormente, com todos eles garantindo uma sequência de operações que remove, em média, um breakpoint tipo um por operação. Assim, em média, cada breakpoint tipo um é removido usando, no máximo, $\frac{4}{3}$ operações. Logo, o lema segue. □

Note que cada caso do Algoritmo 10 é realizado em tempo linear utilizando as estruturas auxiliares de lista de breakpoints e a permutação inversa de π (ou seja, uma permutação que indica a posição de cada elemento i em π). Como $ib_1(\mathcal{I}) \leq n + 1$, o tempo de execução do Algoritmo 10 é $\mathcal{O}(n^2)$.

Teorema 4.1.51. *O Algoritmo 10 é uma 4-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_IRT}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.1.50, o Algoritmo 10 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{4ib_1(\mathcal{I})}{3}$ eventos de reversão e transposição. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

Reversão, Transposição e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_IRTI}$.

Lema 4.1.52. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e seja S uma sequência de reversões, transposições e indels que transforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$. É possível construir uma sequência S' que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| = |S'|$.*

Demonstração. Criaremos a sequência S' com base nas operações da sequência S. Para cada evento de rearranjo β em S, mantendo a ordem relativa, utilize o seguinte mapeamento:

- Se β for uma reversão $\rho^{(i,j)}_{(x,y)}$, então adicione em S' a reversão $\rho^{(i,j)}_{(x,x')}$.
- Se β for uma transposição $\tau^{(i,j,k)}_{(x,y,z)}$, então adicione em S' a transposição $\rho^{(i,i+k-j,k)}_{(x,z,y)}$.
- Se β for um indel $\delta^{(i)}_{(x)}$, então adicione em S' o indel $\delta^{(i)}_{(-x)}$.

Por fim, inverta a sequência S'. Note que cada operação adicionada na sequência S' desfaz a mudança realizada por sua respectiva operação β da sequência S. Além disso, ao inverter a sequência S' temos que a ordem com que as operações são aplicadas também é invertida. Logo, $(\pi, \breve{\pi}) \cdot S' = (\iota, \breve{\iota})$ e $|S| = |S'|$, e o lema segue. □

A Figura 4.6 mostra um exemplo de uma sequência $S' = (\tau^{(3,6,7)}_{(1,1,0)}, \rho^{(1,5)}_{(0,1)}, \delta^{(0)}_{(+5)})$ sendo construída a partir de uma instância $\mathcal{I} = (((0\ 5\ 4\ 2\ 1\ 6\ 3\ 7), (3,3,1,2,3,3,0)), ((0\ 1\ 2\ 3\ 4\ 5\ 6\ 7), (6,2,0,3,3,5,1)))$ e uma sequência $S = (\delta^{(0)}_{(-5)}, \rho^{(1,5)}_{(0,3)}, \tau^{(3,4,7)}_{(1,0,1)})$ que transforma o genoma alvo no genoma de origem.

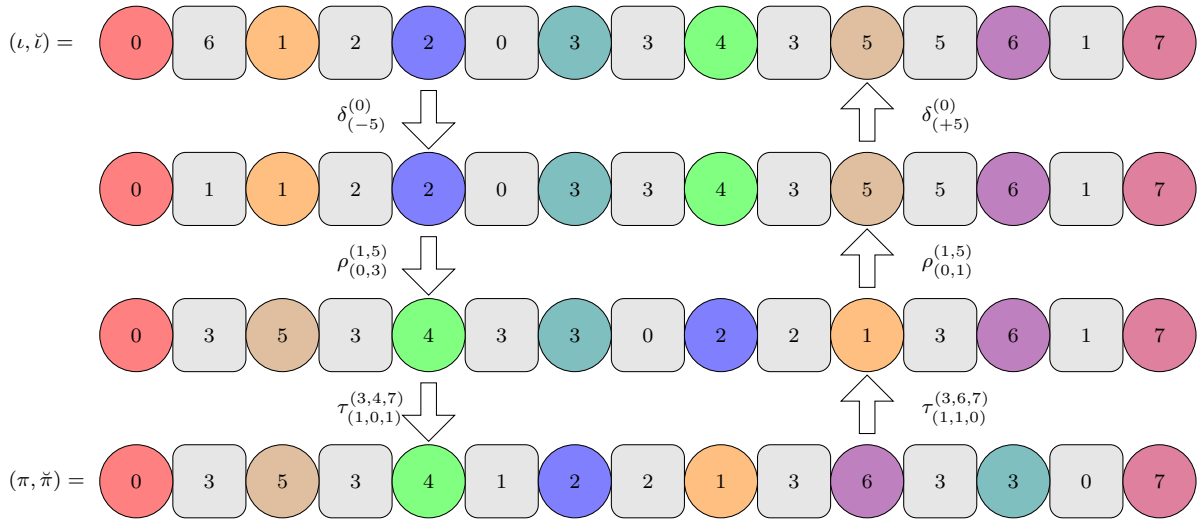


Figura 4.6: Exemplo de construção de uma sequência S' que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ a partir de uma sequência S que transforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$.

Observação 4.1.3. Note que, se temos uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um algoritmo $\mathcal{A}$ que fornece uma sequência de eventos que transforma

$(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, então podemos utilizar o algoritmo $\mathcal{A}$ para obter uma sequência de eventos que transforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$. Para isso, basta criarmos uma instância intergênica rígida sem sinais $\mathcal{I}' = ((\pi^{-1} \circ \iota, \breve{\iota}), (\pi^{-1} \circ \pi, \breve{\pi}))$, onde $\alpha \circ \sigma$ representa a operação de composição entre as permutações α e σ. A sequência fornecida pelo algoritmo $\mathcal{A}$ para a instância $\mathcal{I}'$ também transforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$. Além disso, temos que $ib_1(\mathcal{I}) = ib_1(\mathcal{I}')$, uma vez que esse processo apenas realiza um novo mapeamento para os rótulos dos genes de forma que, no genoma alvo, eles sigam o padrão definido para a permutação identidade.

A composição entre as permutações $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ e $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ resulta em uma nova permutação $\alpha \circ \sigma = (\alpha_{\sigma_1}, \alpha_{\sigma_2}, \ldots, \alpha_{\sigma_n})$. Além disso, $\sigma \circ \sigma^{-1} = \sigma^{-1} \circ \sigma = \iota$. Em outras palavras, ao criar a instância $\mathcal{I}'$ estamos mapeando os genes do genoma de origem (elementos da permutação π) com os valores padrão da permutação identidade e alterando os valores dos genes do genoma alvo (elementos da permutação ι) para refletir que os genes iguais no genoma de origem e alvo possuam o mesmo valor associado.

Considere o Algoritmo 11 para a variação sem sinais do problema **Sb$_\mathbf{I}$RTI**.

Algoritmo 11: Um algoritmo de aproximação para o problema **Sb$_\mathbf{I}$RTI**.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, transposições e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

se $\mathcal{I}$ *for uma instância balanceada* **então**
  Seja S uma sequência de operações fornecida pelo Algoritmo 10 para a instância $\mathcal{I}$
  retorne S
senão
  se $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$ **então**
    ▷ `Lema 4.1.25`
    Seja δ_1 um indel que torna $\mathcal{I}$ uma instância balanceada
    $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot \delta_1, (\iota, \breve{\iota}))$
    Seja S uma sequência de operações fornecida pelo Algoritmo 10 para a instância $\mathcal{I}$
    retorne $(\delta_1) + S$
  senão
    $\mathcal{I}' \leftarrow ((\pi^{-1} \circ \iota, \breve{\iota}), (\pi^{-1} \circ \pi, \breve{\pi}))$
    ▷ `Lema 4.1.25`
    Seja δ_1 um indel que torna $\mathcal{I}'$ uma instância balanceada
    $\mathcal{I}' \leftarrow ((\pi^{-1} \circ \iota, \breve{\iota}) \cdot \delta_1, (\pi^{-1} \circ \pi, \breve{\pi}))$
    Seja S uma sequência de operações fornecida pelo Algoritmo 10 para a instância $\mathcal{I}'$
    $S \leftarrow (\delta_1) + S$
    ▷ `Lema 4.1.52`
    Seja S' uma sequência, criada a partir de S e $\mathcal{I}$, que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$
    retorne S'

Lema 4.1.53. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. O Algoritmo 11 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Caso $\mathcal{I}$ seja uma instância intergênica rígida*

balanceada, então são utilizados, no máximo, $\frac{4ib_1(\mathcal{I})}{3}$ *eventos de reversão e transposição. Caso contrário, são utilizados, no máximo,* $\frac{4ib_1(\mathcal{I})}{3} + 1$ *eventos de reversão, transposição e indel.*

Demonstração. O Algoritmo 11 pode ser analisado considerando dois cenários. Caso $\mathcal{I}$ seja uma instância intergênica rígida balanceada, então o Algoritmo 10 é aplicado e, pelo Lema 4.1.50, a sequência de eventos fornecida pelo Algoritmo 10 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{4b_1(\mathcal{I})}{3}$ eventos de reversão e transposição. Caso contrário, $\mathcal{I}$ é desbalanceada e, nesse cenário, temos duas possibilidades:

- $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$: Nesse caso, pelo Lema 4.1.25, existe um indel que torna $\mathcal{I}$ uma instância intergênica rígida balanceada e, em seguida, o Algoritmo 10 pode ser aplicado. Como resultado, no máximo, $\frac{4ib_1(\mathcal{I})}{3} + 1$ eventos de reversão, transposição e indel são utilizados para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.
- $\sum_{i=1}^{n+1} \breve{\pi}_i > \sum_{i=1}^{n+1} \breve{\iota}_i$: Nesse caso, pela Observação 4.1.3, uma instância auxiliar $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ é criada, onde $\breve{\pi}' = \breve{\iota}$, $\breve{\iota}' = \breve{\pi}$, $\sum_{i=1}^{n+1} \breve{\pi}'_i < \sum_{i=1}^{n+1} \breve{\iota}'_i$ e $ib_1(\mathcal{I}) = ib_1(\mathcal{I}')$. Pelo Lema 4.1.25, existe um indel que torna $\mathcal{I}'$ uma instância intergênica rígida balanceada e, em seguida, o Algoritmo 10 pode ser aplicado, resultando em uma sequência S com, no máximo, $\frac{4ib_1(\mathcal{I}')}{3} + 1$ eventos de reversão, transposição e indel que transforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$. Pelo Lema 4.1.52, podemos criar uma sequência S', com o mesmo tamanho de S, e que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

Em ambos os cenários, $(\pi, \breve{\pi})$ é transformado em $(\iota, \breve{\iota})$ e a quantidade de eventos utilizados para tal não ultrapassa o limite estabelecido, e o lema segue. □

Note que o Algoritmo 11 não possui laços de repetição e a sub-rotina com maior gasto computacional de tempo é decorrente do uso do Algoritmo 10 ($\mathcal{O}(n^2)$), sendo que as demais operações podem ser realizadas em tempo linear. Logo, o tempo de execução do Algoritmo 11 também é $\mathcal{O}(n^2)$.

Teorema 4.1.54. *O Algoritmo 11 é uma 4-aproximação para a variação sem sinais do problema* ***Sb$_I$RTI****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.1.53, o Algoritmo 10 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Além disso, caso $\mathcal{I}$ seja uma instância intergênica rígida balanceada, então são utilizados, no máximo, $\frac{4ib_1(\mathcal{I})}{3}$ eventos de reversão e transposição. Caso contrário, são utilizados, no máximo, $\frac{4ib_1(\mathcal{I})}{3} + 1$ eventos de reversão, transposição e indel. Pelo Teorema 4.1.10, temos o seguinte limitante inferior: (i) caso $\mathcal{I}$ seja uma instância intergênica rígida balanceada, $di_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$; (ii) caso contrário, $di_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})+2}{3}$. Se $\mathcal{I}$ for balanceada, temos que $\frac{\frac{4ib_1(\mathcal{I})}{3}}{\frac{ib_1(\mathcal{I})}{3}} = 4$. Se $\mathcal{I}$ for desbalanceada, temos que $\frac{\frac{4ib_1(\mathcal{I})}{3}+1}{\frac{ib_1(\mathcal{I})+2}{3}} = \frac{\frac{4ib_1(\mathcal{I})+3}{3}}{\frac{ib_1(\mathcal{I})+2}{3}} = \frac{4ib_1(\mathcal{I})+3}{ib_1(\mathcal{I})+2}$. Entretanto, como $\frac{4ib_1(\mathcal{I})+3}{ib_1(\mathcal{I})+2} < \frac{4(ib_1(\mathcal{I})+2)}{ib_1(\mathcal{I})+2} = 4$, o teorema segue. □

Reversão, Transposição e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação sem sinais do problema $\mathbf{Sb_IRTM}$.

Lema 4.1.55. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) um par conectado de breakpoints. É possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando apenas um evento de reversão, transposição ou move.*

Demonstração. Sem perda de generalidade, assuma que $i < j$. Como os breakpoints (π_i, π_{i+1}) e (π_j, π_{j+1}) estão conectados, por definição, um dos seguintes casos deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma reversão como descrito no caso (i) do Lema 4.1.22.

ii. O par de elementos (π_i, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma transposição como descrito no caso (ii) do Lema 4.1.34.

iii. O par de elementos (π_{i+1}, π_j) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma transposição como descrito no caso (iii) do Lema 4.1.34.

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique um move como descrito no caso (iv) do Lema 4.1.29.

Note que os casos (i), (ii), (iii) e (iv) removem pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando apenas um evento de reversão, transposição ou move. Logo, o lema segue. □

Considere o Algoritmo 12 para a variação sem sinais do problema $\mathbf{Sb_IRTM}$.

Lema 4.1.56. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 12 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $ib_1(\mathcal{I})$ eventos de reversão, transposição e move.*

Demonstração. No Algoritmo 12, temos que enquanto $ib_1(\mathcal{I})$ for maior que um, ou seja, $(\pi, \breve{\pi})$ for diferente de $(\iota, \breve{\iota})$ (pela Observação 2.4.2 e Lema 4.1.21), o seguinte procedimento é aplicado: pelos lemas 4.1.19 e 4.1.55, sempre podemos encontrar um par conectado de breakpoints e remover, pelo menos, um breakpoint tipo um após aplicar uma operação de reversão, transposição ou move. A cada iteração do algoritmo, pelo menos um breakpoint tipo um é removido. Dessa forma, o genoma alvo eventualmente será alcançado. No pior caso, cada breakpoint tipo um é removido utilizando um evento de rearranjo. Logo, $ib_1(\mathcal{I})$ operações de reversão, transposição ou move são utilizadas, no máximo, para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, e o lema segue. □

Algoritmo 12: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTM}$.

Entrada: Uma instância intergênica rígida balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, transposições e moves S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
enquanto $ib_1(\mathcal{I}) > 1$ **faça**
  ▷ `Lema 4.1.19`
  $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  ▷ `Lema 4.1.55`
  se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (i) **então**
    $S' \leftarrow (\rho_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (ii) **então**
    $S' \leftarrow (\tau_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iii) **então**
    $S' \leftarrow (\tau_1)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iv) **então**
    $S' \leftarrow (\mu_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorne S

Note que, no pior caso, cada iteração do Algoritmo 12 pode levar um tempo linear para ser aplicada. Como pelo menos um breakpoint tipo um é removido por iteração e $ib_1(\mathcal{I}) \leq n + 1$, então o tempo de execução do Algoritmo 12 é $\mathcal{O}(n^2)$.

Teorema 4.1.57. *O Algoritmo 12 é uma 3-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_IRTM}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.1.56, o Algoritmo 12 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $ib_1(\mathcal{I})$ eventos de reversão, transposição e move. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RTM}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

Reversão, Transposição, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação sem sinais do problema $\mathbf{Sb_IRTMI}$ (Algoritmo 13).

Lema 4.1.58. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida sem sinais. O Algoritmo 13 transforma* $(\pi, \breve{\pi})$ *em* $(\iota, \breve{\iota})$ *utilizando, no máximo,* $ib_1(\mathcal{I})$ *eventos de reversão, transposição, move e indel.*

Demonstração. A prova é similar à descrita no Lema 4.1.27. □

Podemos notar que Algoritmo 13, em comparação com o Algoritmo 12, possui adicionalmente apenas sub-rotinas que podem ser realizadas em tempo linear. Logo, o tempo de execução do Algoritmo 13 é $\mathcal{O}(n^2)$.

Algoritmo 13: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTMI}$.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, transposições, moves e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
▷ `Lema 4.1.25`
se $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
enquanto $ib_1(\mathcal{I}) > 1$ **faça**
  ▷ `Lema 4.1.19`
  (π_i, π_{i+1}), $(\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  ▷ `Lema 4.1.55`
  se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso* (i) **então**
    $S' \leftarrow (\rho_1)$
  senão se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso* (ii) **então**
    $S' \leftarrow (\tau_1)$
  senão se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso* (iii) **então**
    $S' \leftarrow (\tau_1)$
  senão se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso* (iv) **então**
    $S' \leftarrow (\mu_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
▷ `Lema 4.1.26`
se $ib_1(\mathcal{I}) = 1$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorne S

Teorema 4.1.59. *O Algoritmo 13 é uma 3-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_IRTMI}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.1.58, o Algoritmo 13 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $ib_1(\mathcal{I})$ eventos de reversão, transposição, move e indel. Pelo Teorema 4.1.9, temos o seguinte limitante inferior: $di_{\mathbf{RTMI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

Resultados Experimentais

Nesta seção, apresentaremos os resultados práticos dos algoritmos desenvolvidos para a variação sem sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$.

Nós criamos uma base de dados para cada problema e utilizamos os identificadores $U_{\mathbf{Sb_IR}}$, $U_{\mathbf{Sb_IRI}}$, $U_{\mathbf{Sb_IRM}}$, $U_{\mathbf{Sb_IRMI}}$, $U_{\mathbf{Sb_IRT}}$, $U_{\mathbf{Sb_IRTI}}$, $U_{\mathbf{Sb_IRTM}}$ e $U_{\mathbf{Sb_IRTMI}}$ para a base de

dados dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$, respectivamente. Cada base de dados é dividida em cinco grupos. Cada grupo possui 1000 instâncias do mesmo tamanho, sendo que o tamanho de uma instância é a quantidade de genes do genoma de origem e alvo. Além disso, cada grupo é identificado pelo tamanho das instâncias contidas nele. Os identificadores do grupos de cada base de dados são 100, 200, 300, 400 e 500. A quantidade de operações utilizadas para gerar cada instância é baseada em uma porcentagem do identificador de cada grupo e difere entre as bases de dados. A Tabela 4.3 mostra, para cada base de dados, a porcentagem adotada por operação ao se criar uma instância.

Tabela 4.3: Porcentagem de operações aplicadas para gerar cada instância intergênica rígida sem sinais.

Base de Dados	Revesões	Transposições	Moves	Indels
$U_{\mathbf{Sb_IR}}$	50%	0%	0%	0%
$U_{\mathbf{Sb_IRI}}$	40%	0%	0%	10%
$U_{\mathbf{Sb_IRM}}$	40%	0%	10%	0%
$U_{\mathbf{Sb_IRMI}}$	40%	0%	5%	5%
$U_{\mathbf{Sb_IRT}}$	25%	25%	0%	0%
$U_{\mathbf{Sb_IRTI}}$	20%	20%	0%	10%
$U_{\mathbf{Sb_IRTM}}$	20%	20%	10%	0%
$U_{\mathbf{Sb_IRTMI}}$	20%	20%	5%	5%

Cada instância é gerada da seguinte forma: Seja $\mathcal{T} = (\iota, \breve{\iota})$ uma representação intergênica rígida sem sinais de um genoma alvo, de forma que o tamanho de cada região intergênica $\breve{\iota}_i$ foi escolhido de maneira aleatória no intervalo [0..100]. Em seguida, criamos a representação do genoma de origem $\mathcal{S}$ realizando uma cópia de $\mathcal{T}$. Com base na disponibilidade de operações de reversão, transposição, move e indel determinada para cada base de dados e grupo, uma operação σ é escolhida de maneira aleatória e aplicada em $\mathcal{S}$ ($\mathcal{S} = \mathcal{S} \cdot \sigma$). Os parâmetros de cada operação também são escolhidos de forma aleatória dentro do limite de valores válidos. O valor do parâmetro x de um indel $\delta_{(x)}^{(i)}$ aplicado em uma região intergênica $\breve{\pi}_i$ é escolhido dentro do intervalo $[-\breve{\pi}_i..\breve{\pi}_i]$, também de maneira aleatória. Quando não houver operações disponíveis para serem aplicadas, então temos a instância $\mathcal{I}$, composta pela dupla de representação intergênica rígida sem sinais $(\mathcal{S}, \mathcal{T})$. Esse processo repete-se até que cada grupo possua um total de 1000 instâncias. Em todos os casos, consideramos sorteios aleatórios com uma distribuição uniforme.

Note que, para cada base de dados, a soma das porcentagens adotadas considerando cada operação é igual a 50%. Logo, toda instância de um grupo foi gerada utilizando um total de operações que corresponde a metade do identificador do grupo, sendo que a quantidade de operações por tipo depende da base de dados que é considerada.

A seguir, apresentamos os resultados dos algoritmos propostos para a variação sem sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$. A coluna OP indica o total de operações utilizadas para gerar cada instância de um grupo. As colunas Distância e Aproximação indicam a quantidade de operações e o fator de aproximação para uma solução fornecida por um algoritmo.

A Tabela 4.4 mostra os resultados do Algoritmo 4 utilizando a base de dados $U_{\mathbf{Sb_IR}}$.

Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

Tabela 4.4: Resultados do Algoritmo 4 utilizando a base de dados $U_{\mathbf{Sb_IR}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	68	83.94	106	2.15	2.61	3.18
200	100	144	168.79	193	2.33	2.65	3.04
300	150	216	253.66	287	2.27	2.65	3.05
400	200	301	337.98	381	2.41	2.66	2.95
500	250	380	423.19	469	2.45	2.66	2.87

A partir da Tabela 4.4, é possível notar que a aproximação média fornecida pelo Algoritmo 4 para cada grupo tende a ser bem estável, com uma variação de apenas 0.05. O maior fator de aproximação registrado foi de 3.18, que pode ser observado no grupo 100. Na prática, observa-se que os fatores de aproximação fornecidos pelo algoritmo na base de dados foram significativamente menores do que o limite teórico provado para o algoritmo.

A Tabela 4.5 mostra os resultados do Algoritmo 5 utilizando a base de dados $U_{\mathbf{Sb_IRI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

Tabela 4.5: Resultados do Algoritmo 5 utilizando a base de dados $U_{\mathbf{Sb_IRI}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	65	82.99	104	2.24	2.75	3.29
200	100	139	164.76	194	2.39	2.75	3.12
300	150	216	247.57	285	2.51	2.77	3.12
400	200	287	329.26	369	2.50	2.76	3.03
500	250	374	412.41	457	2.50	2.77	3.02

Pela Tabela 4.5, podemos perceber que o Algoritmo 5, em comparação com o Algoritmo 4, apesar de apresentar uma distância média menor em todos os grupos, forneceu uma aproximação média superior para todos os grupos analisados. Uma possível explicação para esse comportamento é devido ao fato de um evento de indel criar, no máximo, um breakpoint durante o processo de criação de cada instância, enquanto uma reversão pode criar até dois breakpoints. O Algoritmo 5 também apresenta pouca variação na aproximação média entre os diferentes grupos e, na prática, os fatores de aproximação observados pelo algoritmo na base de dados também foram significativamente menores do que o limite teórico provado para o algoritmo.

A Tabela 4.6 mostra os resultados do Algoritmo 6 utilizando a base de dados $U_{\mathbf{Sb_IRM}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

Podemos perceber pela Tabela 4.6 que o Algoritmo 6 apresenta um comportamento similar aos algoritmos 4 e 5, mas com uma variação na aproximação média levemente

Tabela 4.6: Resultados do Algoritmo 6 utilizando a base de dados $U_{\mathbf{Sb_IRM}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	68	86.95	109	2.22	2.71	3.16
200	100	150	176.10	202	2.34	2.78	3.11
300	150	231	266.10	306	2.50	2.81	3.10
400	200	312	355.62	397	2.56	2.82	3.03
500	250	403	445.95	493	2.62	2.83	3.04

maior entre os grupos. A maior e a menor aproximação observada foi, respectivamente, 3.16 e 2.22, com ambos os registros ocorrendo no grupo 100.

A Tabela 4.7 mostra os resultados do Algoritmo 7 utilizando a base de dados $U_{\mathbf{Sb_IRMI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

Tabela 4.7: Resultados do Algoritmo 7 utilizando a base de dados $U_{\mathbf{Sb_IRMI}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	63	84.21	103	2.18	2.71	3.22
200	100	144	170.58	196	2.45	2.76	3.10
300	150	218	256.80	292	2.44	2.78	3.16
400	200	303	342.47	384	2.53	2.79	3.05
500	250	383	430.03	468	2.53	2.80	3.01

Na Tabela 4.7 podemos observar que a aproximação média do Algoritmo 7 variou de 2.71 até 2.80. A menor e a maior aproximação observada foi, respectivamente, 2.18 e 3.22, com ambos os registros ocorrendo no grupo 100. Logo, o grupo 100 foi o que apresentou maior variação entre os valores mínimo e máximo do fator de aproximação, com a variação sendo de 1.04. O grupo que apresentou a menor variação entre as aproximações mínima e máxima foi o grupo 500, com o valor de 0.48. Na prática, o Algoritmo 7 apresentou resultados melhores do que o limite teórico provado para o mesmo.

A Tabela 4.8 mostra os resultados dos algoritmos 8, 9 e 10 utilizando a base de dados $U_{\mathbf{Sb_IRT}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

A partir da Tabela 4.8, podemos perceber que os três algoritmos apresentaram resultados bem similares considerando a distância e aproximação média, mesmo possuindo diferentes limites teóricos de aproximação. Inicialmente, podemos perceber que o Algoritmo 8, mesmo possuindo um limite teórico de aproximação superior, forneceu uma distância média ligeiramente menor, comparada à fornecida pelo Algoritmo 9, em todos os grupos. Note que a diferença foi extremamente pequena, sendo que a diferença absoluta para cada grupo foi inferior a 0.75. O algoritmo que apresentou o melhor resultado, considerando as métricas de distância média e aproximação média, foi o Algoritmo 10, sendo que a maior aproximação observada foi de 3.04, registrada no grupo 100, e para os demais grupos a aproximação foi menor ou igual a 3.0. É importante mencionar o bom desempe-

Tabela 4.8: Resultados dos algoritmos 8, 9 e 10 utilizando a base de dados $U_{\mathbf{Sb_IRT}}$.

Algoritmo 8							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	62	72.24	84	2.79	2.96	3.16
200	100	125	143.11	159	2.85	2.97	3.08
300	150	196	214.05	231	2.88	2.97	3.05
400	200	262	284.96	304	2.90	2.97	3.04
500	250	334	355.82	380	2.91	2.97	3.02

Algoritmo 9							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	62	72.39	83	2.79	2.96	3.20
200	100	126	143.57	159	2.85	2.97	3.11
300	150	194	214.53	231	2.90	2.98	3.09
400	200	265	285.54	306	2.90	2.98	3.06
500	250	335	356.56	380	2.92	2.98	3.05

Algoritmo 10							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	59	71.18	80	2.76	2.91	3.04
200	100	126	142.12	156	2.85	2.94	3.00
300	150	194	213.11	230	2.88	2.96	3.00
400	200	260	283.95	305	2.89	2.96	3.00
500	250	334	354.92	375	2.91	2.96	3.00

nho prático dos três algoritmos apresentando fatores de aproximação significativamente menores do que os limites teóricos provados para cada um deles.

A Tabela 4.9 mostra os resultados do Algoritmo 11 utilizando a base de dados $U_{\mathbf{Sb_IRTI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.10.

Tabela 4.9: Resultados do Algoritmo 11 utilizando a base de dados $U_{\mathbf{Sb_IRTI}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	57	67.13	77	2.65	2.86	2.96
200	100	118	133.46	147	2.82	2.92	2.97
300	150	184	199.48	218	2.85	2.94	2.98
400	200	249	265.46	285	2.89	2.95	2.98
500	250	308	331.91	356	2.90	2.95	2.99

Pela Tabela 4.9, é possível notar que houve pouca variação entre a aproximação média fornecida pelo Algoritmo 11 para os grupos. Além disso, considerando a variação entre

a menor e a maior aproximação observada em cada grupo, temos que o grupo 100 foi o que apresentou a maior variação, com o valor de 0.31, enquanto os grupos 400 e 500 apresentaram a menor variação, com o valor de 0.09. Por fim, a aproximação máxima observada em todos os grupos foi menor que 3.00.

A Tabela 4.10 mostra os resultados do Algoritmo 12 utilizando a base de dados $U_{\mathbf{Sb_IRTM}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

Tabela 4.10: Resultados do Algoritmo 12 utilizando a base de dados $U_{\mathbf{Sb_IRTM}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	58	68.77	78	2.76	2.89	2.96
200	100	122	137.88	153	2.80	2.93	2.98
300	150	186	207.02	226	2.88	2.95	2.98
400	200	255	276.14	296	2.88	2.96	2.98
500	250	322	344.79	365	2.90	2.96	2.99

Na Tabela 4.10, podemos notar que a aproximação média fornecida pelo Algoritmo 12, para cada grupo, apresentou um valor próximo tanto da aproximação mínima como da aproximação máxima. Além disso, considerando todos os grupos, a menor e a maior aproximação foram 2.76 e 2.99, respectivamente. A menor aproximação foi registrada no grupo 100, enquanto a maior aproximação foi registrada no grupo 500.

A Tabela 4.11 mostra os resultados do Algoritmo 13 utilizando a base de dados $U_{\mathbf{Sb_IRTMI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.9.

Tabela 4.11: Resultados do Algoritmo 13 utilizando a base de dados $U_{\mathbf{Sb_IRTMI}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	57	68.25	79	2.75	2.93	3.00
200	100	122	136.20	155	2.84	2.95	3.00
300	150	184	203.75	224	2.89	2.96	3.00
400	200	249	271.46	292	2.91	2.96	3.00
500	250	319	338.86	362	2.91	2.97	3.00

A partir da Tabela 4.11 notamos que em todos os grupos o Algoritmo 13 apresentou, para pelo menos uma instância, uma aproximação (computada com base no limitante inferior) que atingiu o limite teórico provado para o mesmo (coluna aproximação máxima). Além disso, a aproximação média tende a aumentar conforme o tamanho da instância cresce, mas a variação entre os grupos foi de apenas 0.04.

De maneira geral, todos os algoritmos apresentaram um bom desempenho na prática. Vale ressaltar o desempenho dos algoritmos 8 e 9 para a variação sem sinais do problema **Sb$_\mathbf{I}$RT**, que mesmo possuindo um limite teórico para a aproximação mais elevado, apresentaram resultados compatíveis com os resultados do melhor algoritmo conhecido para o problema até o momento (Algoritmo 10).

Heurística Gulosa

Com base em uma estratégia gulosa, nós propomos duas heurísticas que podem ser incorporadas aos algoritmos desenvolvidos para a variação sem sinais dos problemas investigados neste capítulo. As heurísticas consistem em verificar se existe, em uma instância intergênica sem sinais $\mathcal{I}$, uma operação de reversão, transposição ou move que remove dois ou mais breakpoints tipo um e aplicá-la. O Algoritmo 14 mostra os passos da heurística I, que devem ser executados seguindo uma ordem de prioridade, e considera apenas os eventos de reversão e transposição.

Algoritmo 14: Heurística Gulosa I.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um modelo de rearranjo $\mathcal{M}$

Saída: Uma sequência vazia ou com apenas um evento de reversão ou transposição

$S \leftarrow ()$
se $\tau \in \mathcal{M}$ *e existe uma transposição* τ *que remove três breakpoints tipo um* **então**
  $S \leftarrow (\tau_1)$
se $\tau \in \mathcal{M}$ *e existe uma transposição* τ *que remove dois breakpoints tipo um* **então**
  $S \leftarrow (\tau_1)$
se $\rho \in \mathcal{M}$ *e existe uma reversão* ρ *que remove dois breakpoints tipo um* **então**
  $S \leftarrow (\rho_1)$
retorne S

O Algoritmo 15 mostra os passos da heurística II, que é uma extensão da heurística I incluindo o evento de move.

Algoritmo 15: Heurística Gulosa II.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um modelo de rearranjo $\mathcal{M}$

Saída: Uma sequência vazia ou com apenas um evento de reversão, transposição ou move

$S \leftarrow ()$
se $\tau \in \mathcal{M}$ *e existe uma transposição* τ *que remove três breakpoints tipo um* **então**
  $S \leftarrow (\tau_1)$
se $\tau \in \mathcal{M}$ *e existe uma transposição* τ *que remove dois breakpoints tipo um* **então**
  $S \leftarrow (\tau_1)$
se $\rho \in \mathcal{M}$ *e existe uma reversão* ρ *que remove dois breakpoints tipo um* **então**
  $S \leftarrow (\rho_1)$
se $\mu \in \mathcal{M}$ *e existe um move* μ *que remove dois breakpoints tipo um* **então**
  $S \leftarrow (\mu_1)$
retorne S

Os algoritmos propostos para a variação sem sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$ devem, a cada iteração, verificar se a heurística I ou II fornece uma sequência não vazia. Caso a resposta seja positiva, então o algoritmo deve incluir a operação fornecida pela heurística na sua sequência

de eventos e atualizar a instância intergênica sem sinais $\mathcal{I}$. Após isso, uma nova iteração deve ser realizada pelo algoritmo. Caso contrário, o algoritmo segue aplicando normalmente os demais passos previstos.

Note que caso a heurística I ou II forneça alguma operação, então sabemos que pelo menos dois breakpoints tipo um de $\mathcal{I}$ são removidos. Logo, a utilização de uma das heurísticas pelos algoritmos indicados não altera o fator de aproximação teórico garantido pelos mesmos. Entretanto, existe uma diferença entre o tempo de execução da heurística I e II. Note que os passos que aplicam uma reversão ou uma transposição podem ser realizados em tempo linear, com auxílio da permutação inversa de π. Contudo, o passo que aplica um move, no pior caso, possui um tempo de execução de $\mathcal{O}(n \log n)$. Note que esse passo utiliza apenas breakpoints subcarregados e sobrecarregados, verificando se existe um par com um breakpoint sobrecarregado e um breakpoint subcarregado, tal que o excesso no tamanho da região intergênica do breakpoint sobrecarregado é exatamente a quantidade que falta na região intergênica do breakpoint subcarregado. Uma forma de realizar esse passo é ordenando os breakpoints sobrecarregados pelo excesso no tamanho da região intergênica e, para cada breakpoint subcarregado, realizar uma busca binária verificando se é possível encontrar um par com tal característica.

Dessa forma, a utilização da heurística I (Algoritmo 14) não altera o tempo de execução dos algoritmos propostos para a variação sem sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$, mantendo-os em $\mathcal{O}(n^2)$. Entretanto, o uso da heurística II (Algoritmo 15) afeta o tempo de execução dos algoritmos para a variação sem sinais dos problemas $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$, alterando-os para $\mathcal{O}(n^2 \log n)$.

4.1.4 Instâncias Intergênicas Rígidas com Sinais

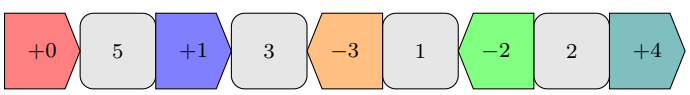
Nesta seção, apresentaremos algoritmos para os problemas investigados neste capítulo em um cenário não ponderado e em instâncias intergênicas rígidas com sinais. Inicialmente iremos apresentar algumas definições e lemas que serão utilizados por múltiplos algoritmos.

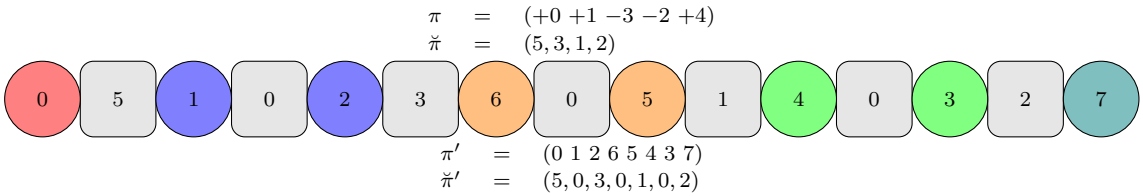
A seguir, descrevemos a transformação de *duplicação*, que pode ser aplicada em uma representação intergênica rígida com sinais. Caso a representação esteja na forma estendida, os elementos π_0 e π_{n+1} são ignorados, a transformação é aplicada e a nova representação resultante é então estendida.

Definição 4.1.1. Dada uma representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi})$ com n genes, a *duplicação* $\mathcal{D}(\mathcal{R})$ cria uma representação intergênica rígida sem sinais $\mathcal{R}' = (\pi', \breve{\pi}')$ com $2n$ genes, de forma que cada elemento $\pi_i \in \pi$ é mapeado em dois elementos de π' (π'_{2i-1} e π'_{2i}). Caso $\pi_i > 0$, então $\pi'_{2i-1} = 2\pi_i - 1$ e $\pi'_{2i} = 2\pi_i$. Caso contrário, $\pi'_{2i-1} = |2\pi_i|$ e $\pi'_{2i} = |2\pi_i| - 1$. Além disso, $\breve{\pi}'_{2i-1} = \breve{\pi}_i$, para $i \in [1..n+1]$, e $\breve{\pi}'_{2j} = 0$, para $j \in [1..n]$.

O Exemplo 4.1.2 mostra a transformação de duplicação sendo aplicada em uma representação intergênica rígida com sinais $\mathcal{R} = ((+0\ +1\ -3\ -2\ +4), (5, 3, 1, 2))$.

Exemplo 4.1.2.





Seja $\mathcal{F}$ uma função que recebe uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, com n genes, e cria uma instância intergênica rígida sem sinais $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$, com $2n$ genes, da seguinte forma: (i) $(\pi', \breve{\pi}') = \mathcal{D}((\pi, \breve{\pi}))$ e (ii) $(\iota', \breve{\iota}') = \mathcal{D}((\iota, \breve{\iota}))$.

Lema 4.1.60. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ com n genes, a função $\mathcal{F}$ cria uma instância intergênica rígida sem sinais $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ com $2n$ genes, tal que $ib_1(\mathcal{I}') = ib_2(\mathcal{I})$.*

Demonstração. Perceba que cada breakpoint tipo dois em $\mathcal{I}$ é mapeado pela função $\mathcal{F}$ em um breakpoint tipo um em $\mathcal{I}'$. Além disso, cada elemento $\pi_i \in \pi$, que é mapeado em dois elementos de π' (π'_{2i-1} e π'_{2i}), sempre gera uma adjacência intergênica em $\mathcal{I}'$, pois $|\pi'_{2i-1} - \pi'_{2i}| = 1$ e a região intergênica entre π'_{2i-1} e π'_{2i} tem tamanho zero tanto no genoma de origem como no genoma alvo. Por fim, se um par (π_i, π_{i+1}) não é um breakpoint tipo dois em $\mathcal{I}$, então uma adjacência intergênica (π'_{2i}, π'_{2i+1}) é criada em $\mathcal{I}'$. Logo, $ib_1(\mathcal{I}') = ib_2(\mathcal{I})$ e o lema segue. □

Lema 4.1.61. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida sem sinais tal que $\mathcal{I}' = \mathcal{F}(\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota})))$. Seja S' uma sequência de eventos de reversão, transposição, move e indel que transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ de forma que nenhum evento de S' afeta as adjacências intergênicas de $\mathcal{I}'$. Existe uma função $\mathcal{G}$ que cria, a partir de S', uma sequência de eventos S que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e $|S| = |S'|$.*

Demonstração. Seja $\mathcal{G}$ uma função que cria uma sequência S com base nas operações da sequência S'. Para cada evento de rearranjo β em S', mantendo a ordem relativa, o seguinte mapeamento é utilizado:

- Se β for uma reversão $\rho^{(i,j)}_{(x,y)}$, então adicione em S a reversão $\rho^{(\frac{i+1}{2}, \frac{j}{2})}_{(x,y)}$.
- Se β for uma transposição $\tau^{(i,j,k)}_{(x,y,z)}$, então adicione em S a transposição $\rho^{(\frac{i+1}{2}, \frac{j+1}{2}, \frac{k+1}{2})}_{(x,y,z)}$.
- Se β for um move $\mu^{(i,j)}_{(x)}$, então adicione em S o move $\mu^{(\frac{i+1}{2}, \frac{j+1}{2})}_{(x)}$.
- Se β for um indel $\delta^{(i)}_{(x)}$, então adicione em S o indel $\delta^{(\frac{i+1}{2})}_{(x)}$.

Como as adjacências intergênicas de $\mathcal{I}'$ não são afetadas por qualquer evento de S', então isso significa que nenhuma região $\breve{\pi}'_{2j}$, para $j \in [1..n]$, foi afetada. Sabendo disso, temos a garantia de que nenhum evento de S' separa os pares de elemento π'_{2i-1} e π'_{2i}, para $i \in [1..n]$. Note que o mapeamento utilizado pela função $\mathcal{G}$ considera a mudança na posição dos elementos causada pela transformação de duplicação. Dessa forma, a sequência S fornecida pela função $\mathcal{G}$ é capaz de transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e $|S| = |S'|$. Logo, o lema segue. □

Note que as funções $\mathcal{F}$ e $\mathcal{G}$ podem ser computadas em tempo linear.

Reversão

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema $\mathbf{Sb_IR}$ (Algoritmo 16).

Algoritmo 16: Um algoritmo de aproximação para o problema $\mathbf{Sb_IR}$.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$
▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 4 para a instância $\mathcal{I}'$
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Note que o tempo de execução do Algoritmo 16 é $\mathcal{O}(n^2)$ (tempo de execução do Algoritmo 4), uma vez que as funções $\mathcal{F}$ e $\mathcal{G}$ são executas em tempo linear.

Teorema 4.1.62. *O Algoritmo 16 é uma 4-aproximação para a variação com sinais do problema* $\mathbf{Sb_IR}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. Note que o Algoritmo 4 fornece uma sequência de reversões que afetam apenas breakpoints tipo um. Logo, nenhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.1.23, Algoritmo 4 utiliza, no máximo, $2ib_1(\mathcal{I}')$ reversões para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$. Pelo Lema 4.1.61, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'| \leq 2ib_1(\mathcal{I}')$. Pelo Lema 4.1.60, temos que $ib_1(\mathcal{I}') = ib_2(\mathcal{I})$. Logo, $|S| \leq 2ib_2(\mathcal{I})$. Pelo Teorema 4.1.11, temos o seguinte limitante inferior: $di_{\mathbf{R}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{2}$. Logo, o teorema segue. □

Reversão e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema $\mathbf{Sb_IRI}$ (Algoritmo 17).

Algoritmo 17: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRI}$.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão e indel S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$
▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e indel fornecida pelo Algoritmo 5 para a instância $\mathcal{I}'$
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Teorema 4.1.63. *O Algoritmo 17 é uma 4-aproximação para a variação com sinais do problema* ***Sb$_I$RI****.*

Demonstração. A prova é similar à descrita no Teorema 4.1.62, mas considerando que o Algoritmo 5 utiliza, no máximo, $2ib_1(\mathcal{I}')$ eventos de reversão e indel para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ (Lema 4.1.27). □

Reversão e Move

Nesta seção, apresentaremos algoritmos de aproximação com fatores 4 e 2 para a variação com sinais do problema **Sb$_I$RM**. Agora, considere o Algoritmo 18.

Algoritmo 18: Um algoritmo de aproximação para o problema **Sb$_I$RM**.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão e move S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$
▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e move fornecida pelo Algoritmo 6 para a instância $\mathcal{I}'$
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Teorema 4.1.64. *O Algoritmo 18 é uma 4-aproximação para a variação com sinais do problema* ***Sb$_I$RM****.*

Demonstração. A prova é similar à descrita no Teorema 4.1.62, mas considerando que o Algoritmo 6 utiliza, no máximo, $2ib_1(\mathcal{I}')$ eventos de reversão e move para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ (Lema 4.1.30). □

A seguir, utilizaremos a estrutura de grafo de ciclos ponderado rígido e apresentaremos lemas usados para obtenção de um algoritmo que garante um fator de aproximação 2 para a variação com sinais do problema **Sb$_I$RM**.

Lema 4.1.65. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais tal que em $G(\mathcal{I})$ existe um ciclo negativo. Existe em $G(\mathcal{I})$ pelo menos um ciclo positivo.*

Demonstração. Suponha por contradição que $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ é uma instância intergênica rígida balanceada com sinais, tal que em $G(\mathcal{I})$ existe um ciclo negativo e não existe nenhum ciclo positivo. Como em $G(\mathcal{I})$ existe, pelo menos, um ciclo negativo e os ciclos restantes só podem ser negativos ou balanceados, temos que $\sum_{i=1}^{n+1} \breve{\pi}_i > \sum_{i=1}^{n+1} \breve{\iota}_i$, o que contradiz a suposição de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. □

Lema 4.1.66. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ existe um ciclo trivial negativo $C = (c^1)$ e pelo menos um outro ciclo $D = (d^1, \ldots, d^k)$ desbalanceado. É possível aumentar o número de ciclos balanceados de $G(\mathcal{I})$ em pelo menos uma unidade após aplicar uma operação de move.*

Demonstração. Neste caso, basta transferir o peso extra da aresta preta c^1 para a primeira aresta preta do ciclo D utilizando uma operação de move. Como C é um ciclo trivial, após aplicação da operação de move ele se torna balanceado, e o lema segue. □

Lema 4.1.67. *[Lema 5.1 de Oliveira et al. [57]] Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ existe pelo menos um ciclo divergente negativo ou balanceado C. É possível aumentar o número de ciclos de $G(\mathcal{I})$ em uma unidade e o número de ciclos balanceados de $G(\mathcal{I})$ em uma unidade, após aplicar uma operação de reversão.*

Lema 4.1.68. *[Lema 5.2 de Oliveira et al. [57]] Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que todos os ciclos não triviais negativos ou balanceados de $G(\mathcal{I})$ são convergentes. É possível aumentar o número de ciclos de $G(\mathcal{I})$ em uma unidade e o número de ciclos balanceados de $G(\mathcal{I})$ em uma unidade, após aplicar uma sequência de duas reversões.*

Agora considere o Algoritmo 19 e observe que ele possui três passos: (i) operações de move aplicadas a ciclos negativos triviais (lemas 4.1.65 e 4.1.66); (ii) reversões aplicadas em ciclos divergentes negativos ou balanceados (Lema 4.1.67); (iii) duas reversões consecutivas aplicadas em ciclos convergentes negativos ou balanceados (Lema 4.1.68).

Algoritmo 19: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRM}$.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões e moves S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto c(G(I)) < n + 1 ou c_b(G(I)) < n + 1 faça
3      se existir um ciclo trivial negativo em G(I) então
           ▷ Lemas 4.1.65 e 4.1.66
4          S' ← (μ_1)
5      senão se existir um ciclo divergente negativo ou balanceado em G(I) então
           ▷ Lema 4.1.67
6          S' ← (ρ_1)
7      senão
           ▷ Lema 4.1.68
8          S' ← (ρ_1, ρ_2)
9      S ← S + S'
10     I ← ((π, π̆) · S', (ι, ῐ))
11 retorne S
```

Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e uma sequência de eventos de rearranjo S, seja $\Delta_{c+c_b}(\mathcal{I}, S) = \frac{\Delta c(G(\mathcal{I}),S)+\Delta c_b(G(\mathcal{I}),S)}{|S|}$ o número médio de ciclos e de ciclos balanceados criados por cada operação de S. Note que nas sequências geradas pelos passos (i), (ii) e (iii) temos que $\Delta_{c+c_b}(\mathcal{I}, S) \geq 1$.

Lema 4.1.69. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. O Algoritmo 19 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2(n + 1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$ eventos de reversão e move.*

Demonstração. Pela Observação 2.6.3, sabemos que se $c(G(\mathcal{I})) = n+1$ e $c_b(G(\mathcal{I})) = n+1$, então o genoma alvo foi alcançado. Note que, enquanto essa condição não for satisfeita, o Algoritmo 19 garante que todo ciclo trivial negativo seja transformado em balanceado pelos lemas 4.1.65 e 4.1.66. Se não existir nenhum ciclo trivial negativo, qualquer ciclo divergente negativo ou balanceado é dividido em dois ciclos com a garantia de que pelo menos um deles seja balanceado (Lema 4.1.67). Se nenhuma dessas duas situações anteriores ocorrer, $G(\mathcal{I})$ pode ter ciclos triviais positivos ou balanceados, ciclos divergentes positivos e ciclos convergentes (positivos, negativos ou balanceados). Assim, todos os ciclos não triviais negativos e balanceados em $G(\mathcal{I})$ são convergentes e o Lema 4.1.68 pode ser aplicado. Observe que o Algoritmo 19, em cada iteração, sempre executa um dos passos (i), (ii) ou (iii). Além disso, cada passo aumenta, em pelo menos uma unidade, o número de ciclos ou, em pelo menos uma unidade, o número de ciclos balanceados. Esse processo se repete até $G(\mathcal{I})$ possuir $n+1$ ciclos e $n+1$ ciclos balanceados que, consequentemente transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e o algoritmo encerra sua execução. Como $\Delta_{c+c_b}(\mathcal{I}, S) \geq 1$ para as sequências geradas pelos passos (i), (ii) e (iii), no máximo, $2(n+1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$ eventos de reversão e move são utilizados. Assim, o lema segue. □

Perceba que os passos (i) e (ii) do Algoritmo 19 podem ser realizados em tempo linear. Entretanto, o tempo de execução do passo (iii) é $\mathcal{O}(n^2)$, uma vez que é necessário verificar o cruzamento dos ciclos para determinar a sequência de duas reversões. Como o laço da linha 2 repete-se, no máximo, n vezes, então o tempo de execução do Algoritmo 19 é $\mathcal{O}(n^3)$.

Teorema 4.1.70. *O Algoritmo 19 é uma 2-aproximação para a variação com sinais do problema* ***Sb$_\mathbf{I}$RM****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. Pelo Lema 4.1.69, o Algoritmo 19 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2(n+1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$ eventos de reversão e move. Pelo Teorema 4.1.13, temos o seguinte limitante inferior: $di_{\mathbf{RM}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I}))+c_b(G(\mathcal{I}))}{2}$. Assim, o teorema segue. □

Reversão, Move e Indel

Nesta seção, apresentaremos algoritmos de aproximação com fatores 4 e 2 para a variação com sinais do problema **Sb$_\mathbf{I}$RMI**. Agora, considere o Algoritmo 20.

Teorema 4.1.71. *O Algoritmo 20 é uma 4-aproximação para a variação com sinais do problema* ***Sb$_\mathbf{I}$RMI****.*

Demonstração. A prova é similar à descrita no Teorema 4.1.62, mas considerando que o Algoritmo 7 utiliza, no máximo, $2ib_1(\mathcal{I}')$ eventos de reversão, move e indel para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ (Lema 4.1.32). □

A seguir, utilizaremos a estrutura de grafo de ciclos ponderado rígido e apresentaremos lemas usados para obtenção de um algoritmo que garante um fator de aproximação 2 para a variação com sinais do problema **Sb$_\mathbf{I}$RMI**.

Algoritmo 20: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRMI}$.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão, move e indel S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, move e indel fornecida pelo Algoritmo 7 para a instância $\mathcal{I}'$
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Lema 4.1.72. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ existe um ciclo trivial negativo $C = (c^1)$. É possível aumentar o número de ciclos balanceados de $G(\mathcal{I})$ em uma unidade, após aplicar uma operação de indel.*

Demonstração. Note que o ciclo C é trivial, então ele possui apenas uma aresta preta e uma aresta cinza. Neste caso, basta aplicar um indel na aresta preta c^1 do ciclo C removendo $|\sum_{e'_i \in E_c(C)} w_c(e'_i) - \sum_{e_i \in E_p(C)} w_p(e_i)|$ nucleotídeos. Dessa forma, o ciclo C é transformado em balanceado, e o lema segue. □

Lema 4.1.73. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ existe um ciclo positivo C. É possível aumentar o número de ciclos balanceados de $G(\mathcal{I})$ em uma unidade, após aplicar uma operação de indel.*

Demonstração. Neste caso, basta aplicar um indel na primeira aresta preta do ciclo C inserindo $\sum_{e'_i \in E_c(C)} w_c(e'_i) - \sum_{e_i \in E_p(C)} w_p(e_i)$ nucleotídeos. Dessa forma, o ciclo C é transformado em balanceado, e o lema segue. □

Agora, considere o Algoritmo 21 e observe que ele possui quatro passos: (i) operações de move ou indel aplicadas em ciclos negativos triviais (lemas 4.1.66 e 4.1.72); (ii) reversões aplicadas em ciclos divergentes negativos ou balanceados (Lema 4.1.67); (iii) duas reversões consecutivas aplicadas em ciclos convergentes negativos ou balanceados (Lema 4.1.68); (iv) indels aplicados em ciclos positivos (Lema 4.1.73).

Note que nas sequências geradas pelos passos (i), (ii), (iii) e (iv) do Algoritmo 21, também temos que $\Delta_{c+c_b}(\mathcal{I}, S) \geq 1$.

Lema 4.1.74. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. O Algoritmo 21 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2(n+1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$ eventos de reversão, move e indel.*

Demonstração. Pela Observação 2.6.3, sabemos que se $c(G(\mathcal{I})) = n+1$ e $c_b(G(\mathcal{I})) = n+1$, então o genoma alvo foi alcançado. Note que, enquanto essa condição não for satisfeita, o Algoritmo 21 garante que todo ciclo trivial negativo seja transformado em balanceado pelos lemas 4.1.66 e 4.1.72. Se não existir nenhum ciclo trivial negativo, qualquer ciclo divergente negativo ou balanceado é dividido em dois ciclos com a garantia de que pelo menos um deles seja balanceado (Lema 4.1.67). Se nenhuma dessas duas

Algoritmo 21: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRMI}$.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, moves e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto c(G(I)) < n + 1 ou c_b(G(I)) < n + 1 faça
3      se existir um ciclo trivial negativo C em G(I) então
4          se existir um ciclo desbalanceado D em G(I), tal que D ≠ C, então
               ▷ Lema 4.1.66
5              S' ← (μ_1)
6          senão
               ▷ Lema 4.1.72
7              S' ← (δ_1)
8
9      senão se existir um ciclo divergente negativo ou balanceado em G(I) então
           ▷ Lema 4.1.67
10         S' ← (ρ_1)
11     senão se existir um ciclo convergente negativo ou balanceado em G(I) então
           ▷ Lema 4.1.68
12         S' ← (ρ_1, ρ_2)
13     senão
           ▷ Lema 4.1.73
14         S' ← (δ_1)
15     S ← S + S'
16     I ← ((π, π̆) · S', (ι, ῐ))
17 retorne S
```

situações anteriores ocorrer, $G(\mathcal{I})$ pode ter ciclos triviais positivos ou balanceados, ciclos divergentes positivos e ciclos convergentes (positivos, negativos ou balanceados). Se existir pelo menos um ciclo convergente negativo ou balanceado, então o Lema 4.1.68 pode ser aplicado. Caso o passo (iii) não seja aplicado, isso implica que deve existir pelo menos um ciclo positivo em $G(\mathcal{I})$ e o passo (iv) pode ser aplicado, aumentando o número de ciclos balanceados em uma unidade (Lema 4.1.73). Observe que o Algoritmo 21, em cada iteração, sempre executa um dos passos (i), (ii), (iii) ou (iv). Além disso, cada passo aumenta, em pelo menos uma unidade, o número de ciclos ou, em pelo menos uma unidade, o número de ciclos balanceados. Esse processo se repete até $G(\mathcal{I})$ possuir $n+1$ ciclos e $n+1$ ciclos balanceados que, consequentemente transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e o algoritmo encerra sua execução. Como $\Delta_{c+c_b}(\mathcal{I}, S) \geq 1$ para as sequências geradas pelos passos (i), (ii), (iii) e (iv), no máximo, $2(n+1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$ eventos de reversão, move e indel são utilizados. Assim, o lema segue. □

Note que o Algoritmo 21, em comparação com o Algoritmo 19, difere apenas pelos passos (i) e (iv), que podem ser realizados em tempo linear. Logo, o tempo de execução do Algoritmo 21 também é $\mathcal{O}(n^3)$.

Teorema 4.1.75. *O Algoritmo 21 é uma 2-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_IRMI}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Pelo Lema 4.1.74, o Algoritmo 21 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2(n+1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$ eventos de reversão, move e indel. Pelo Teorema 4.1.13, temos o seguinte limitante inferior: $di_{\mathbf{RMI}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I}))+c_b(G(\mathcal{I}))}{2}$. Assim, o teorema segue. □

Reversão e Transposição

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema **Sb$_\mathbf{I}$RT** (Algoritmo 22).

Algoritmo 22: Um algoritmo de aproximação para o problema **Sb$_\mathbf{I}$RT**.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão e transposição S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e transposição fornecida pelo Algoritmo 10 para a instância $\mathcal{I}'$
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Teorema 4.1.76. *O Algoritmo 22 é uma 4-aproximação para a variação com sinais do problema* ***Sb$_I$RT****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. Note que o Algoritmo 10 fornece uma sequência de reversões e transposições que afetam apenas breakpoints tipo um. Logo, nenhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.1.50, o Algoritmo 10 utiliza, no máximo, $\frac{4ib_1(\mathcal{I}')}{3}$ eventos de reversão e transposição para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$. Pelo Lema 4.1.61, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'| \leq \frac{4ib_1(\mathcal{I}')}{3}$. Pelo Lema 4.1.60, temos que $ib_1(\mathcal{I}') = ib_2(\mathcal{I})$. Logo, $|S| \leq \frac{4ib_2(\mathcal{I})}{3}$. Pelo Teorema 4.1.11, temos o seguinte limitante inferior: $di_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$. Logo, o teorema segue. □

Reversão, Transposição e Indel

Nesta seção, apresentaremos dois algoritmos de aproximação com fatores 4 e 3 para a variação com sinais do problema **Sb$_\mathbf{I}$RTI**. Considere o Algoritmo 23.

Teorema 4.1.77. *O Algoritmo 23 é uma 4-aproximação para a variação com sinais do problema* ***Sb$_I$RTI****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Note que o Algoritmo 11 fornece uma sequência de reversões, transposições e indels que afetam apenas breakpoints tipo um. Logo, nenhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além

Algoritmo 23: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTI}$.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão, transposição e indel S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$
▷ Lema 4.1.60
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição e indel fornecida pelo Algoritmo 11 para a instância $\mathcal{I}'$
▷ Lema 4.1.61
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

disso, pelo Lema 4.1.53, o Algoritmo 11 utiliza, no máximo, $\frac{4ib_1(\mathcal{I}')}{3}$ ou $\frac{4ib_1(\mathcal{I}')}{3} + 1$ eventos de reversão, transposição e indel para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$, se $\mathcal{I}'$ for balanceada ou desbalanceada, respectivamente. Pelo Lema 4.1.61, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'|$. Pelo Lema 4.1.60, temos que $ib_1(\mathcal{I}') = ib_2(\mathcal{I})$. Logo, caso $\mathcal{I}$ seja balanceada temos que $|S| \leq \frac{4ib_2(\mathcal{I})}{3}$. Caso contrário, temos que $|S| \leq \frac{4ib_2(\mathcal{I})}{3} + 1$. Pelo Teorema 4.1.12, temos o seguinte limitante inferior: (i) se $\mathcal{I}$ for balanceada, então $di_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$; (ii) caso contrário, $di_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})+2}{3}$. Se $\mathcal{I}$ for balanceada, temos que $\frac{\frac{4ib_2(\mathcal{I})}{3}}{\frac{ib_2(\mathcal{I})}{3}} = 4$. Se $\mathcal{I}$ for desbalanceada, temos que $\frac{\frac{4ib_2(\mathcal{I})}{3}+1}{\frac{ib_2(\mathcal{I})+2}{3}} = \frac{\frac{4ib_2(\mathcal{I})+3}{3}}{\frac{ib_2(\mathcal{I})+2}{3}} = \frac{4ib_2(\mathcal{I})+3}{ib_2(\mathcal{I})+2}$. Entretanto, como $\frac{4ib_2(\mathcal{I})+3}{ib_2(\mathcal{I})+2} < \frac{4(ib_2(\mathcal{I})+2)}{ib_2(\mathcal{I})+2} = 4$, o teorema segue. □

A seguir, utilizaremos a estrutura de grafo de ciclos ponderado rígido e apresentaremos lemas usados para obtenção de um algoritmo de aproximação com um fator 3 para a variação com sinais do problema $\mathbf{Sb_IRTI}$.

Lema 4.1.78. *[Lema 4.3 de Oliveira et al. [59]] Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ existe pelo menos um ciclo orientado negativo C. É é possível aumentar o número de ciclos balanceados de $G(\mathcal{I})$ em uma unidade, após aplicar uma operação de transposição.*

Lema 4.1.79. *[Lema 4.6 de Oliveira et al. [59]] Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ existe pelo menos um ciclo orientado balanceado C. É possível aumentar o número de ciclos balanceados de $G(\mathcal{I})$ em duas unidades, após aplicar três operações de transposição.*

Lema 4.1.80. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais tal que em $G(\mathcal{I})$ todos os ciclos não triviais são não orientados balanceados ou negativos. É possível aumentar o número de ciclos balanceados de $G(\mathcal{I})$ em duas unidades, após aplicar três operações de reversão.*

Demonstração. Seja C um ciclo não orientado balanceado ou negativo em $G(\mathcal{I})$. Como C é um ciclo não orientado, ele possui pelo menos um *open gate*. Como todos os *open gates* em $G(\mathcal{I})$ são fechados [11], então existe um ciclo não trivial D que se cruza com C. Como todos os ciclos não triviais em $G(\mathcal{I})$ são não orientados balanceados ou negativos, então D

é um ciclo não orientado balanceado ou negativo. Neste caso, basta aplicar uma reversão nas arestas pretas de D que estão imediatamente antes e depois da aresta cinza que fecha o *open gate* de C. Como resultado, C passa a ser um ciclo divergente e o Lema 4.1.67 pode ser aplicado, aumentando o número de ciclos balanceados em uma unidade. Entretanto, a segunda reversão faz com que o ciclo D se torne divergente e o Lema 4.1.67 pode ser aplicado novamente, aumentando o número de ciclos balanceados em mais uma unidade. No fim, o número de ciclos balanceados de $G(\mathcal{I})$ aumenta em duas unidades, após aplicar três operações de reversão e o lema segue. □

Agora considere o Algoritmo 24. Note que, em cada passo, é tratado um determinado tipo de ciclo visando aumentar o número de ciclos balanceados em $G(\mathcal{I})$.

Algoritmo 24: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTI}$.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, transposições e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto c_b(G(I)) < n + 1 faça
3      se existir um ciclo positivo em G(I) então
           ▷ Lema 4.1.73
4          S' ← (δ_1)
5      senão se existir um ciclo trivial negativo em G(I) então
           ▷ Lema 4.1.72
6          S' ← (δ_1)
7      senão se existir um ciclo divergente em G(I) então
           ▷ Lema 4.1.67
8          S' ← (ρ_1)
9      senão se existir um ciclo orientado C em G(I) então
10         se C for negativo então
               ▷ Lema 4.1.78
11             S' ← (τ_1)
12         senão
               ▷ Lema 4.1.79
13             S' ← (τ_1, τ_2, τ_3)
14
15     senão
           ▷ Lema 4.1.80
16         S' ← (ρ_1, ρ_2, ρ_3)
17     S ← S + S'
18     I ← ((π, π̆) · S', (ι, ῐ))
19 retorne S
```

Lema 4.1.81. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. O Algoritmo 24 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{3(n+1-c_b(G(\mathcal{I})))}{2}$ eventos de reversão, transposição e indel.*

Demonstração. Pela Observação 2.6.3, sabemos que se $c_b(G(\mathcal{I})) = n + 1$, então o genoma

alvo foi alcançado. Note que, enquanto essa condição não for satisfeita, o Algoritmo 24 aplicará um dos seguintes passos:

i. Se houver algum ciclo positivo em $G(\mathcal{I})$, então aplicando um indel o ciclo torna-se balanceado (Lema 4.1.73).

ii. Se existir um ciclo trivial negativo em $G(\mathcal{I})$, então aplicando um indel o ciclo torna-se balanceado (Lema 4.1.72).

iii. Note que neste ponto do algoritmo todos os ciclos de $G(\mathcal{I})$ são negativos ou balanceados. Se houver algum ciclo divergente (negativo ou balanceado) em $G(\mathcal{I})$, então aplicando uma reversão é possível aumentar o número de ciclos balanceados em uma unidade (Lema 4.1.67).

iv. Neste ponto do algoritmo os ciclos não triviais em $G(\mathcal{I})$ são, obrigatoriamente, convergentes. Caso exista um ciclo orientado negativo, então com uma transposição é possível aumentar o número de ciclos balanceados em uma unidade (Lema 4.1.78). Caso exista um ciclo orientado balanceado, então com uma sequência de três transposições é possível aumentar o número de ciclos balanceados em duas unidades (Lema 4.1.79).

v. Neste ponto do algoritmo os ciclos não triviais em $G(\mathcal{I})$ são, obrigatoriamente, convergentes não orientados balanceados ou negativos. Por fim, com uma sequência de três reversões é possível aumentar o número de ciclos balanceados em duas unidades (Lema 4.1.80).

Note que nos cinco passos o número de ciclos balanceados aumenta em pelo menos uma unidade. Logo, o genoma alvo eventualmente será alcançado e o algoritmo termina a execução. Além disso, no pior caso, três operações são utilizadas para aumentar o número de ciclos em duas unidades. Dessa forma, temos que, no máximo, $\frac{3(n+1-c_b(G(\mathcal{I})))}{2}$ eventos de reversão, transposição e indel são utilizados para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, e o lema segue. □

Note que o Algoritmo 24 é uma extensão do algoritmo apresentado por Oliveira *et al.* [59] para o problema $\mathbf{Sb_IRT}$ em instâncias intergênicas balanceadas com sinais. Por esse motivo, no pior caso, o tempo de execução do Algoritmo 24 também é $\mathcal{O}(n^4)$.

Teorema 4.1.82. *O Algoritmo 24 é uma 3-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_IRTI}$.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Pelo Lema 4.1.81, temos que o Algoritmo 24 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{3(n+1-c_b(G(\mathcal{I})))}{2}$ eventos de reversão, transposição e indel. Pelo Teorema 4.1.14, temos o seguinte limitante inferior: $di_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{n+1-c_b(G(\mathcal{I}))}{2}$. Logo, o teorema segue. □

Algoritmo 25: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTM}$.

Entrada: Uma instância intergênica rígida balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão, transposição e move S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

▷ Lema 4.1.60

1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição e move fornecida pelo Algoritmo 12 para a instância $\mathcal{I}'$

▷ Lema 4.1.61

3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Reversão, Transposição e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação com sinais do problema $\mathbf{Sb_IRTM}$ (Algoritmo 25).

Teorema 4.1.83. *O Algoritmo 25 é uma 3-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_IRTM}$.

Demonstração. A prova é similar à descrita no Teorema 4.1.76, mas considerando que o Algoritmo 12 utiliza, no máximo, $ib_1(\mathcal{I}')$ eventos de reversão, transposição e move para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ (Lema 4.1.56). □

Reversão, Transposição, Move e Indel

Nesta seção, apresentaremos dois algoritmos de aproximação com fator 3 para a variação com sinais do problema $\mathbf{Sb_IRTMI}$. O primeiro é baseado no conceito de breakpoints intergênicos e o segundo é baseado na estrutura de grafo de ciclos ponderado rígido. Considere o Algoritmo 26.

Algoritmo 26: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTMI}$.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão, transposição, move e indel S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

▷ Lema 4.1.60

1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição, move e indel fornecida pelo Algoritmo 13 para a instância $\mathcal{I}'$

▷ Lema 4.1.61

3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Teorema 4.1.84. *O Algoritmo 26 é uma 3-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_IRTMI}$.

Demonstração. A prova é similar à descrita no Teorema 4.1.76, mas considerando que o Algoritmo 13 utiliza, no máximo, $ib_1(\mathcal{I}')$ eventos de reversão, transposição, move e indel para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ (Lema 4.1.58). □

A seguir, utilizaremos a estrutura de grafo de ciclos ponderado rígido e apresentaremos outro algoritmo de aproximação com um fator 3. Considere o Algoritmo 27, que é similar ao Algoritmo 24, mas inclui o evento de move como uma possibilidade no segundo caso.

Algoritmo 27: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTMI}$.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões, transposições, moves e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
enquanto $c_b(G(\mathcal{I})) < n + 1$ **faça**
  se *existir um ciclo positivo em* $G(\mathcal{I})$ **então**
    ▷ Lema 4.1.73
    $S' \leftarrow (\delta_1)$
  senão se *existir um ciclo trivial negativo em* $G(\mathcal{I})$ **então**
    se *existir um outro ciclo desbalanceado em* $G(\mathcal{I})$ **então**
      ▷ Lema 4.1.66
      $S' \leftarrow (\mu_1)$
    senão
      ▷ Lema 4.1.72
      $S' \leftarrow (\delta_1)$
  senão se *existir um ciclo divergente em* $G(\mathcal{I})$ **então**
    ▷ Lema 4.1.67
    $S' \leftarrow (\rho_1)$
  senão se *existir um ciclo orientado* C *em* $G(\mathcal{I})$ **então**
    se C *for negativo* **então**
      ▷ Lema 4.1.78
      $S' \leftarrow (\tau_1)$
    senão
      ▷ Lema 4.1.79
      $S' \leftarrow (\tau_1, \tau_2, \tau_3)$
  senão
    ▷ Lema 4.1.80
    $S' \leftarrow (\rho_1, \rho_2, \rho_3)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorne S

Lema 4.1.85. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida com sinais, o Algoritmo 27 transforma* $(\pi, \breve{\pi})$ *em* $(\iota, \breve{\iota})$ *utilizando, no máximo,* $\frac{3(n+1-c_b(G(\mathcal{I})))}{2}$ *eventos de reversão, transposição, move e indel.*

Demonstração. A prova é similar à descrita no Lema 4.1.81 considerando que no segundo caso do algoritmo, um ciclo trivial negativo pode ser transformado em balanceado através de um evento de move ou indel. □

Note que o Algoritmo 27, no pior caso, tem o tempo de execução equiparável ao Algoritmo 24, que é $\mathcal{O}(n^4)$.

Teorema 4.1.86. *O Algoritmo 27 é uma 3-aproximação para a variação com sinais do problema* ***Sb$_I$RTMI****.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Pelo Lema 4.1.85, temos que o Algoritmo 27 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $\frac{3(n+1-c_b(G(\mathcal{I})))}{2}$ eventos de reversão, transposição, move e indel. Pelo Teorema 4.1.14, temos o seguinte limitante inferior: $di_{\mathbf{RTMI}}(\mathcal{I}) \geq \frac{n+1-c_b(G(\mathcal{I}))}{2}$. Logo, o teorema segue. □

Resultados Experimentais

Nesta seção, apresentaremos os resultados práticos dos algoritmos propostos para a variação com sinais dos problemas **Sb$_I$R**, **Sb$_I$RI**, **Sb$_I$RM**, **Sb$_I$RMI**, **Sb$_I$RT**, **Sb$_I$RTI**, **Sb$_I$RTM** e **Sb$_I$RTMI**.

Criamos também uma base de dados para cada problema e utilizamos os identificadores $S_{\mathbf{Sb_IR}}$, $S_{\mathbf{Sb_IRI}}$, $S_{\mathbf{Sb_IRM}}$, $S_{\mathbf{Sb_IRMI}}$, $S_{\mathbf{Sb_IRT}}$, $S_{\mathbf{Sb_IRTI}}$, $S_{\mathbf{Sb_IRTM}}$ e $S_{\mathbf{Sb_IRTMI}}$ para a base de dados dos problemas **Sb$_I$R**, **Sb$_I$RI**, **Sb$_I$RM**, **Sb$_I$RMI**, **Sb$_I$RT**, **Sb$_I$RTI**, **Sb$_I$RTM** e **Sb$_I$RTMI**, respectivamente. O processo de criação das bases de dados foi similar ao descrito na Seção 4.1.3, diferindo apenas pelo fato de que cada instância foi criada a partir de um par de representações intergênicas rígidas com sinais. Logo, ao aplicar um evento de reversão, os genes do segmento afetado também têm suas orientações invertidas.

Além das bases de dados criadas por nós, utilizamos outras duas apresentadas por Oliveira *et al.* [59]. As bases foram geradas da seguinte forma: Inicialmente, utilizando uma representação intergênica com sinais, foram gerados 100 genomas alvos $(\iota, \breve{\iota})$, com 100 genes cada, sendo cada valor de $\breve{\iota}_i$, com $1 \leq i \leq 101$, escolhido aleatoriamente e de maneira uniforme no intervalo [0..100]. Depois disso, a partir de cada genoma alvo $(\iota, \breve{\iota})$, foram gerados 100 genomas de origem $(\pi, \breve{\pi})$ aplicando:

- DB$_{\mathbf{Sb_IRT}}$: OP operações aleatórias de reversões e transposições (sendo 50% de cada) em cada genoma alvo $(\iota, \breve{\iota})$.
- DB$_{\mathbf{Sb_IRTM}}$: OP operações aleatórias de reversões, transposições e moves (sendo 50% de reversões, 40% de transposições e 10% de moves) em cada genoma alvo $(\iota, \breve{\iota})$.

Os parâmetros de cada operação aplicada foram gerados aleatoriamente considerando a faixa de valores válidos. O valor de OP variou de 10 até 100, em intervalos de 10. Para cada valor de OP, foi gerado um grupo com 10.000 instâncias. As bases de dados DB$_{\mathbf{Sb_IRT}}$ e DB$_{\mathbf{Sb_IRTM}}$ têm um total de 100.000 instâncias cada.

A seguir, apresentamos os resultados dos algoritmos propostos para a variação com sinais dos problemas **Sb$_I$R**, **Sb$_I$RI**, **Sb$_I$RM**, **Sb$_I$RMI**, **Sb$_I$RT**, **Sb$_I$RTI**, **Sb$_I$RTM** e **Sb$_I$RTMI**. A coluna OP indica o total de operações utilizadas para gerar cada instância

de um grupo. As colunas Distância e Aproximação indicam, respectivamente, a quantidade de operações e o fator de aproximação para uma solução fornecida por um dado algoritmo.

A Tabela 4.12 mostra os resultados do Algoritmo 16 utilizando a base de dados $S_{\mathbf{Sb_IR}}$. O fator de aproximação foi computado utilizando o limitante inferior apresentado no Teorema 4.1.11.

Tabela 4.12: Resultados do Algoritmo 16 utilizando a base de dados $S_{\mathbf{Sb_IR}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	72	91.49	114	2.31	2.85	3.35
200	100	150	184.58	210	2.46	2.89	3.18
300	150	233	278.13	308	2.60	2.91	3.17
400	200	328	371.25	415	2.68	2.92	3.18
500	250	407	464.12	515	2.67	2.92	3.19

Pela Tabela 4.12, podemos perceber que, em média, a distância fornecida pelo Algoritmo 16 é um valor próximo de 90% do tamanho das instâncias. Além disso, o fator mínimo de aproximação obtido foi de 2.31 no grupo 100. Já o fator de aproximação máximo obtido foi de 3.35, também no grupo 100. Podemos notar ainda que o fator de aproximação médio tende a aumentar conforme o tamanho da instância também aumenta. Entretanto, a variação entre os fatores de aproximação mínimo e máximo tende a diminuir conforme o tamanho da instância aumenta. Além disso, o fator de aproximação observado na prática foi significativamente menor do que o fator teórico provado para o algoritmo. Vale ressaltar que existe para a variação com sinais do problema $\mathbf{Sb_IR}$ um algoritmo com fator de aproximação 2 [57] baseado na estrutura de grafo de ciclos. O Algoritmo 16 garante um fator de aproximação 4, mas utiliza o conceito de breakpoint.

A Tabela 4.13 mostra os resultados do Algoritmo 17 utilizando a base de dados $S_{\mathbf{Sb_IRI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11.

Tabela 4.13: Resultados do Algoritmo 17 utilizando a base de dados $S_{\mathbf{Sb_IRI}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	68	88.94	109	2.36	2.94	3.46
200	100	143	178.26	204	2.60	2.98	3.32
300	150	234	267.55	303	2.66	2.99	3.25
400	200	311	356.04	394	2.72	2.99	3.22
500	250	397	445.76	495	2.76	3.00	3.24

Pela Tabela 4.13, podemos notar que o Algoritmo 17 apresentou um desempenho similar ao Algoritmo 16. É importante mencionar que o fator de aproximação médio foi levemente maior em todos os grupos. Entretanto, a distância média fornecida pelo Algoritmo 17, em todos os grupos, foi menor.

A Tabela 4.14 mostra os resultados dos algoritmos 18 e 19 utilizando a base de dados $S_{\mathbf{Sb_IRM}}$. Os fatores de aproximação do Algoritmo 18 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11. Já os fatores de aproximação do Algoritmo 19 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.13.

Tabela 4.14: Resultados dos algoritmos 18 e 19 utilizando a base de dados $S_{\mathbf{Sb_IRM}}$.

Algoritmo 18							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	68	93.081	113	2.37	2.91	3.35
200	100	159	189.21	231	2.50	2.98	3.38
300	150	250	286.05	324	2.71	3.02	3.27
400	200	344	382.30	426	2.77	3.03	3.30
500	250	421	478.94	531	2.82	3.04	3.25

Algoritmo 19							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	46	53.41	61	1.03	1.13	1.22
200	100	97	107.62	117	1.09	1.14	1.22
300	150	150	161.69	174	1.09	1.14	1.18
400	200	201	215.68	229	1.10	1.14	1.19
500	250	256	269.92	284	1.11	1.14	1.18

É possível notar, pelos dados da Tabela 4.14, que o fator de aproximação médio fornecido pelo Algoritmo 18, baseado no conceito de breakpoints, tende a aumentar conforme o tamanho da instância cresce. Além disso, ele foi superior a 3.0 nos grupos 300, 400 e 500. O menor fator de aproximação foi de 2.37, observado no grupo 100. Já o maior fator de aproximação foi de 3.38, observado no grupo 200. Considerando a distância média, em cada grupo, temos um valor superior a 90% do tamanho das instâncias. Considerando o Algoritmo 19, que é baseado na estrutura de grafo de ciclos ponderado rígido, podemos notar que ele tende a fornecer soluções melhores para a variação com sinais do problema $\mathbf{Sb_IRM}$, em comparação com o Algoritmo 18. Podemos constatar essa tendência pela coluna de aproximação média, que para o grupo 100 apresentou um valor de 1.13 e, para os demais grupos, de 1.14. Além disso, pela coluna de distância média, é possível notar que a quantidade de operações utilizadas nas soluções fornecidas pelo Algoritmo 19 foi próxima da quantidade de operações utilizadas para gerar as instâncias em cada grupo (coluna OP).

A Tabela 4.15 mostra os resultados dos algoritmos 20 e 21 utilizando a base de dados $S_{\mathbf{Sb_IRMI}}$. Os fatores de aproximação do Algoritmo 20 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11. Já os fatores de aproximação do Algoritmo 21 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.13.

Pelos resultados mostrados na Tabela 4.15, podemos notar um comportamento similar ao observado nos algoritmos da Tabela 4.14, com o Algoritmo 21 (baseado na estrutura de grafo de ciclos ponderado rígido) tendendo a fornecer melhores soluções para a varia-

Tabela 4.15: Resultados dos algoritmos 20 e 21 utilizando a base de dados $S_{\mathbf{Sb_IRMI}}$.

Algoritmo 20							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	69	90.98	112	2.34	2.93	3.43
200	100	148	183.65	212	2.55	2.97	3.33
300	150	237	276.73	320	2.63	3.00	3.24
400	200	327	369.42	411	2.73	3.01	3.21
500	250	415	462.58	520	2.77	3.02	3.25

Algoritmo 21							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	46	51.50	58	1.04	1.13	1.22
200	100	94	102.84	110	1.07	1.12	1.19
300	150	143	154.12	163	1.09	1.12	1.17
400	200	192	205.23	216	1.08	1.12	1.15
500	250	246	256.71	268	1.09	1.12	1.15

ção com sinais do problema **Sb$_\mathbf{I}$RMI**, quando comparado ao Algoritmo 20 (baseado no conceito de breakpoints). É importante notar que, em ambos os algoritmos e em todos os grupos, houve uma redução no valor da distância média em comparação com os resultados dos algoritmos da Tabela 4.14.

A Tabela 4.16 mostra os resultados do Algoritmo 22 utilizando a base de dados $S_{\mathbf{Sb_IRT}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11.

Tabela 4.16: Resultados do Algoritmo 22 utilizando a base de dados $S_{\mathbf{Sb_IRT}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	57	71.03	81	2.75	2.91	3.00
200	100	127	141.76	156	2.82	2.94	3.00
300	150	196	212.60	230	2.86	2.95	3.00
400	200	259	283.29	301	2.89	2.95	3.00
500	250	332	354.02	379	2.88	2.96	3.00

Pela Tabela 4.16, podemos notar que o fator de aproximação máximo observado pelo Algoritmo 22 foi de 3.0 e ocorreu em todos os grupos. Além disso, há uma tendência crescente do fator de aproximação conforme o aumento do tamanho das instâncias. Entretanto, a variação no fator de aproximação médio foi pequena, com o menor valor sendo de 2.91 (observado no grupo 100) e o maior valor sendo de 2.96 (observado no grupo 500).

Existe, para a variação com sinais do problema **Sb$_\mathbf{I}$RT**, um algoritmo com fator de aproximação 3 (baseado na estrutura de grafo de ciclos ponderado rígido), que iremos chamar de 3**Sb$_\mathbf{I}$RT**. Como os fatores de aproximação dos algoritmos 22 e 3**Sb$_\mathbf{I}$RT** estão próximos, nós realizamos um comparativo entre eles utilizando a base de dados DB$_{\mathbf{Sb_IRT}}$.

O Algoritmo 22 foi executado adotando a heurística I (Algoritmo 14). A Tabela 4.17 mostra os resultados dos algoritmos 22 e 3$\mathbf{Sb_IRT}$ utilizando a base de dados $DB_{\mathbf{Sb_IRT}}$. Na referida tabela, os valores presentes nas colunas M e ME indicam, para cada grupo, a porcentagem de soluções fornecidas pelo Algoritmo 22 com tamanho estritamente menor (coluna M) e com tamanho menor ou igual (coluna ME), quando comparadas às soluções fornecidas pelo algoritmo 3$\mathbf{Sb_IRT}$. Os fatores de aproximação de ambos os algoritmos foram computados utilizando o limitante inferior baseado na estrutura de grafo de ciclos ponderado rígido [59, Teorema 3.8].

Tabela 4.17: Comparação entre os algoritmos 22 e 3$\mathbf{Sb_IRT}$ utilizando a base de dados $DB_{\mathbf{Sb_IRT}}$.

OP	Algoritmo 22		3$\mathbf{Sb_IRT}$		M	ME
	Distância Média	Aproximação Média	Distância Média	Aproximação Média		
10	12.19	1.63	12.60	1.68	51.37%	68.70%
20	25.60	1.71	26.37	1.77	54.72%	66.48%
30	36.91	1.66	40.23	1.81	78.60%	85.53%
40	46.10	1.60	53.29	1.85	95.84%	97.62%
50	53.21	1.56	63.78	1.87	99.46%	99.74%
60	58.97	1.55	71.88	1.89	99.81%	99.92%
70	63.29	1.55	77.83	1.90	99.97%	99.98%
80	66.73	1.55	82.45	1.91	99.95%	99.99%
90	69.48	1.55	85.98	1.92	99.97%	99.99%
100	71.64	1.56	88.78	1.93	99.99%	99.99%

A partir dos dados da Tabela 4.17, é possível observar que o Algoritmo 22, em todos os grupos, foi capaz de fornecer melhores resultados considerando as métricas de aproximação média e distância média. Além disso, considerando os grupos com OP maior que 20, o algoritmo forneceu melhores soluções em mais de 75% das instâncias (coluna M). Considerando os grupos com OP maior que 30, o Algoritmo 22 forneceu soluções de tamanho melhor ou igual (coluna ME) em mais de 97% das instâncias. É importante observar que, à medida que o valor de OP aumenta, a diferença absoluta entre a distância média fornecida pelos algoritmos 22 e 3$\mathbf{Sb_IRT}$ também aumenta significativamente. Quando OP é maior que 50, a diferença absoluta entre as distâncias médias é superior a 10, o que indica que o Algoritmo 22 tende a oferecer melhores soluções em cenários onde foi utilizado um número maior de operações.

A Tabela 4.18 mostra os resultados dos algoritmos 23 e 24 utilizando a base de dados $S_{\mathbf{Sb_IRTI}}$. Os fatores de aproximação do Algoritmo 23 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.12. Já os fatores de aproximação do Algoritmo 24 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.14.

Pela Tabela 4.18, podemos perceber que o Algoritmo 23 apresentou um comportamento similar ao Algoritmo 22 na Tabela 4.16. Um ponto de diferença foi que o fator de aproximação máximo obtido pelo Algoritmo 23 foi de 2.99, observado no grupo 500. Além disso, a aproximação média foi levemente menor em todos os grupos. O Algoritmo 24 foi o que apresentou o melhor desempenho, fornecendo fator de aproximação médio pra-

Tabela 4.18: Resultados dos algoritmos 23 e 24 utilizando a base de dados $S_{\mathbf{Sb_IRTI}}$.

Algoritmo 23							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	58	67.00	78	2.68	2.86	2.96
200	100	118	133.06	147	2.74	2.91	2.97
300	150	180	198.95	218	2.82	2.93	2.98
400	200	249	264.78	284	2.84	2.94	2.98
500	250	306	331.04	354	2.87	2.94	2.99

Algoritmo 24							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	55	63.41	71	1.96	2.01	2.13
200	100	113	126.13	136	1.96	2.00	2.08
300	150	173	188.59	202	1.98	2.00	2.06
400	200	236	251.23	265	1.99	2.00	2.04
500	250	294	313.94	329	1.99	2.00	2.03

ticamente estável em todos os grupos, variando entre 2.00 e 2.01. Além disso, o fator de aproximação máximo atingido foi de 2.13, no grupo 100.

A Tabela 4.19 mostra os resultados do Algoritmo 25 utilizando a base de dados $S_{\mathbf{Sb_IRTM}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11.

Tabela 4.19: Resultados do Algoritmo 25 utilizando a base de dados $S_{\mathbf{Sb_IRTM}}$.

-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	57	68.58	78	2.73	2.88	2.96
200	100	120	137.53	152	2.82	2.93	2.98
300	150	183	206.42	224	2.85	2.94	2.98
400	200	253	275.25	297	2.87	2.95	2.98
500	250	322	343.59	363	2.88	2.95	2.99

A partir da Tabela 4.19, é possível notar que as métricas de mínimo, média e máximo para a aproximação fornecida pelo Algoritmo 25 tendem a ser estáveis nos diferentes grupos. A maior variação foi observada na aproximação mínima, com o menor valor sendo 2.73 no grupo 100 e o maior valor sendo 2.88 no grupo 500. A menor variação foi observada na aproximação máxima, com o menor valor sendo 2.96 também no grupo 100 e o maior valor sendo 2.99 também no grupo 500.

Existe, para a variação com sinais do problema **Sb$_\mathbf{I}$RTM**, um algoritmo com fator de aproximação 2.5 (baseado na estrutura de grafo de ciclos ponderado rígido), que iremos chamar de 2.5**Sb$_\mathbf{I}$RTM**. Como os fatores de aproximação dos algoritmos 25 e 2.5**Sb$_\mathbf{I}$RTM** estão próximos, nós realizamos um comparativo entre eles utilizando a base de dados $\mathrm{DB}_{\mathbf{Sb_IRTM}}$. O Algoritmo 25 foi executado adotando a heurística I (Algoritmo 14). A

Tabela 4.20 mostra os resultados dos algoritmos 25 e 2.5$\mathbf{Sb_IRTM}$ utilizando a base de dados $DB_{\mathbf{Sb_IRTM}}$. Na referida tabela, os valores presentes nas colunas M e ME indicam, para cada grupo, a porcentagem de soluções fornecidas pelo Algoritmo 25 com tamanho estritamente menor (coluna M) e com tamanho menor ou igual (coluna ME), quando comparadas às soluções fornecidas pelo algoritmo 2.5$\mathbf{Sb_IRTM}$. Os fatores de aproximação de ambos os algoritmos foram computados utilizando o limitante inferior baseado na estrutura de grafo de ciclos ponderado rígido [59, Teorema 7.6].

Tabela 4.20: Comparação entre os algoritmos 25 e 2.5$\mathbf{Sb_IRT}$ utilizando a base de dados $DB_{\mathbf{Sb_IRTM}}$.

OP	Algoritmo 25		2.5$\mathbf{Sb_IRTM}$		M	ME
	Distância Média	Aproximação Média	Distância Média	Aproximação Média		
10	12.25	1.64	11.79	1.57	32.30%	53.18%
20	25.57	1.72	24.97	1.68	34.77%	49.58%
30	36.69	1.67	38.21	1.74	63.31%	73.75%
40	45.61	1.62	50.47	1.79	89.60%	93.69%
50	52.65	1.58	60.46	1.81	97.69%	98.75%
60	58.19	1.56	68.17	1.83	99.45%	99.78%
70	62.54	1.55	74.06	1.84	99.75%	99.90%
80	65.96	1.55	78.68	1.85	99.93%	99.96%
90	68.67	1.55	82.09	1.85	99.92%	99.97%
100	70.82	1.55	84.85	1.86	99.97%	100.00%

Pelos dados da Tabela 4.20, podemos notar que o algoritmo 2.5$\mathbf{Sb_IRTM}$, quando comparado ao Algoritmo 25, apresentou um resultado ligeiramente melhor em relação à aproximação média e distância média nos grupos com $OP = 10$ e $OP = 20$. Considerando esses dois grupos ($OP = 10$ e $OP = 20$), a diferença absoluta entre a distância média fornecida pelos algoritmos foi menor que 0.61. Além disso, pela coluna M, podemos notar que, nos referidos grupos o Algoritmo 25 forneceu melhores soluções em 32.30% e 34.77% das instâncias, respectivamente. Isso mostra que o Algoritmo 25 pode atuar de forma complementar ao algoritmo 2.5$\mathbf{Sb_IRTM}$, mesmo nos casos em que ambos fornecem resultados semelhantes. Como melhores estimativas tendem a resultar em análises aprimoradas, selecionar o melhor resultado entre cada algoritmo é uma boa alternativa para auxiliar nessa tarefa. Em relação aos grupos onde OP é maior que 20, o Algoritmo 25 forneceu melhores resultados, considerando a aproximação média e distância média. Além disso, nos mesmos grupos, o Algoritmo 25 forneceu soluções de tamanho menor ou equivalente (coluna ME) em mais de 73% das instâncias.

A Tabela 4.21 mostra os resultados dos algoritmos 26 e 27 utilizando a base de dados $S_{\mathbf{Sb_IRTMI}}$. Os fatores de aproximação do Algoritmo 26 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11. Já os fatores de aproximação do Algoritmo 27 foram computados utilizando o limitante inferior apresentado no Teorema 4.1.14.

Pela Tabela 4.21, é possível notar que para o Algoritmo 26, em pelo menos uma instância de cada grupo, foi observado um fator de aproximação (computado com base no limitante inferior) que atingiu o limite teórico, sendo este igual a 3 (coluna aproximação

Tabela 4.21: Resultados dos algoritmos 26 e 27 utilizando a base de dados $S_{\mathbf{Sb_IRTMI}}$.

Algoritmo 26							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	57	68.01	77	2.73	2.92	3.00
200	100	122	135.70	153	2.81	2.94	3.00
300	150	181	203.11	225	2.86	2.95	3.00
400	200	250	270.60	291	2.88	2.95	3.00
500	250	317	337.80	360	2.90	2.96	3.00

Algoritmo 27							
-	-	Distância			Aproximação		
Grupo	OP	Mínimo	Média	Máximo	Mínimo	Média	Máximo
100	50	55	65.31	74	1.91	2.00	2.15
200	100	117	130.16	142	1.96	2.00	2.07
300	150	180	194.70	208	1.97	2.00	2.05
400	200	241	259.25	276	1.97	2.00	2.03
500	250	309	323.66	340	1.98	2.00	2.03

máxima). Considerando todos os grupos, a aproximação média foi superior a 2.91 e inferior a 2.97. Levando-se em conta a distância fornecida pelo Algoritmo 26, temos uma variação relativamente pequena, uma vez que a distância máxima subtraída da distância mínima observada em cada grupo foi menor do que 45. O algoritmo 27 foi o que apresentou o melhor desempenho, com uma aproximação média de 2.00 em todos os grupos. Além disso, a variação entre a menor (1.91) e a maior (2.15) aproximação observada foi de apenas 0.24, com ambos os registros ocorrendo no grupo 100.

De maneira geral, todos os algoritmos apresentaram um bom desempenho na prática. O único algoritmo em que foi possível observar uma aproximação, computada com base no limitante inferior, que atingiu o limite teórico foi o Algoritmo 26. Vale ressaltar que isso não significa que o fator de aproximação teórico provado para o algoritmo é justo. Uma aproximação é justa quando encontramos uma instância cuja a razão entre a solução fornecida pelo algoritmo e uma solução ótima é igual, ou tão próxima quanto se queira, do fator teórico de aproximação garantido pelo algoritmo. Note que, até o momento, computar o valor ótimo para a distância das instâncias que foram utilizadas é uma tarefa impraticável. Por esse motivo, utilizamos os limitantes inferiores para computar a informação referente ao fator de aproximação obtido em cada instância pelos algoritmos.

Resultados com Genomas Reais

Nesta seção, apresentaremos os resultados utilizando genomas reais de cianobactérias.

Para demonstrar a aplicabilidade dos algoritmos propostos, utilizamos dados reais de 97 genomas de cianobactérias do Cyanorak 2.1 [45], um sistema para visualização e curadoria de genomas de picocianobactérias marinhas e salobras. A quantidade de genes por genoma variou de 1834 até 4391, sendo que a porcentagem da quantidade de genes replicados em relação à quantidade total de genes, na média, foi menor que 5%.

Realizamos uma etapa de pré-processamento para garantir que os dados se ajustassem às restrições do modelo, que é dividida em duas fases:

- Mapear a sequência de genes e o tamanho das regiões intergênicas em uma representação intergênica rígida com sinais $(\pi, \breve{\pi})$: Para cada genoma, mapeamos a primeira ocorrência dos genes em uma permutação π e calculamos o tamanho das regiões intergênicas para obter $\breve{\pi}$.

- Emparelhamento: Para cada par de genomas, realizamos um emparelhamento para que os genes e os blocos conservados compartilhados por ambos os genomas fossem mantidos, enquanto os genes restantes fossem removidos através de um processo que simula uma sequência de deleções. Por fim, se a instância resultante não for balanceada, aplicamos apenas um indel para torná-la balanceada inserindo nucleotídeos do genoma com menos nucleotídeos no total.

Para realizarmos um comparativo, utilizamos o Algoritmo 25 (com a heurística I, Algoritmo 14), proposto para o problema $\mathbf{Sb_IRTM}$, que leva em consideração tanto a estrutura dos genes como o tamanho das regiões intergênica, e o Algoritmo 2**SbRT** [71], que é uma 2-aproximação para a variação com sinais do problema de Ordenação de Permutações por Reversões e Transposições (**SbRT**). Os problemas $\mathbf{Sb_IRTM}$ e **SbRT** possuem modelos de rerranjo semelhantes, diferenciando apenas pelo evento de move. Entretanto, como mencionado previamente, o evento de move atua como uma generalização do evento de transposição em um contexto intergênico, onde um dos segmentos afetados é uma região intergênica. Por esse motivo, o Algoritmo 25 foi utilizado no comparativo.

Executamos os algoritmos para cada instância resultante do emparelhamento, mas note que o algoritmo 2**SbRT** considera apenas a estrutura dos genes. Logo, o algoritmo 2**SbRT** recebeu como entrada apenas (π, ι) de cada instância resultante do emparelhamento. O número de eventos de rearranjo do genoma para cada emparelhamento foi calculado pelo total de deleções usadas na etapa de pré-processamento acrescido do tamanho da sequência de reversões, transposições e moves fornecida pelos algoritmos. Para cada algoritmo, esses números foram armazenados em uma matriz de distâncias.

Por fim, construímos duas árvores filogenéticas com base nas matrizes de distâncias calculadas a partir dos algoritmos e usando o método de Reconstrução de Ordem Circular [53]. Para analisar as características topológicas das árvores filogenéticas, realizamos uma comparação com a árvore filogenética apresentada por Laurence *et al.* [45] usando uma ferramenta, baseada nas subárvores de concordância máxima (MAST), para determinar a congruência topológica entre duas árvores filogenéticas [70]. A Tabela 4.22 mostra os resultados obtidos. A coluna MAST indica o número de folhas nas subárvores de concordância máxima. A coluna I_{cong} mostra o índice de congruência topológica, computado através do MAST obtido normalizado pela média esperada do MAST de árvores aleatórias. A coluna P-value mostra a probabilidade, sob a suposição de que a hipótese nula é verdadeira, de se obter um resultado similar ou superior ao observado. Note que indicativos para bons resultados são: (i) possuir um MAST com o maior número de folhas; (ii) apresentar um valor maior que um para I_{cong}; e (iii) obter um valor de P-value o mais próximo de zero possível.

Tabela 4.22: Análise das características topológicas das árvores filogenéticas, geradas pelos resultados dos algoritmos 25 e 2**SbRT**, comparadas com a árvore filogenética apresentada por Laurence *et al.* [45].

	MAST	I_{cong}	P-value
2**SbRT**	46 folhas	3.17	6.76e-22
Algoritmo 25	51 folhas	3.52	2.77e-25

A Tabela 4.22 indica que ambas as árvores filogenéticas têm uma alta concordância com as árvores filogenéticas apresentadas por Laurence *et al.* [45], com a árvore filogenética obtida através do Algoritmo 25 fornecendo um MAST com mais folhas e, consequentemente, um melhor valor para I_{cong} e P-value. É importante mencionar que o objetivo deste experimento usando genomas reais é demonstrar a aplicabilidade do nosso algoritmo, que considera as informações referentes aos genes e o tamanho das regiões intergênicas, comparado com um modelo similar que considera apenas a ordem e orientação dos genes. Nós utilizamos o mesmo estágio de pré-processamento de dados e método de reconstrução para fornecer uma comparação justa. No entanto, os resultados podem diferir especialmente considerando genomas com características diferentes e o método de reconstrução adotado. A Figura 4.7 mostra a árvore filogenética construída usando o método de Reconstrução de Ordem Circular [53] com a matriz de distâncias do Algoritmo 25.

Pela Figura 4.7, observamos que a abordagem separou os organismos considerando as espécies e realizou bons agrupamentos[1]. Vale ressaltar que a árvore foi baseada exclusivamente em informações de eventos de rearranjo.

[1]Figura criada usando o pacote `treeio` da linguagem R [72])

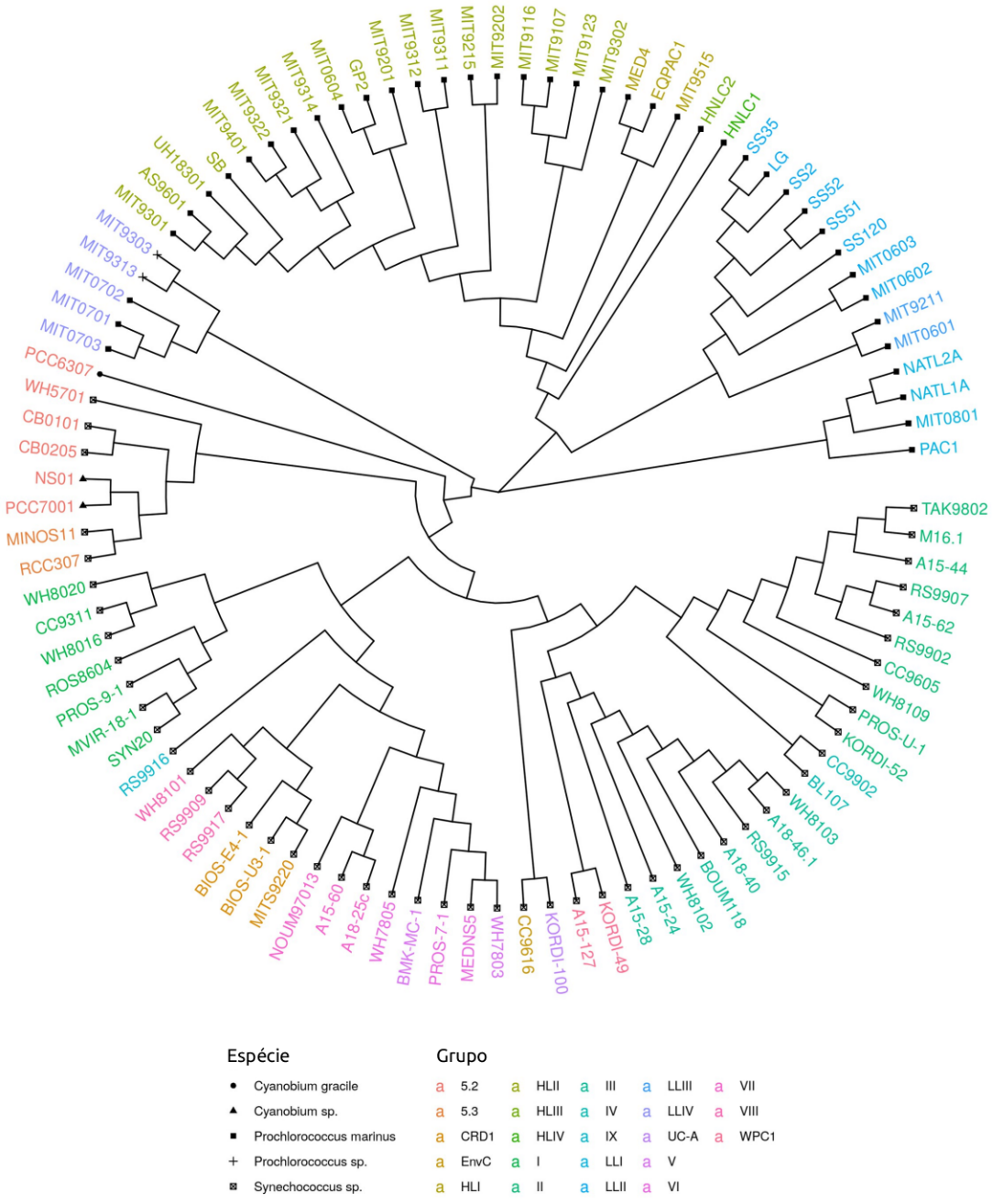


Figura 4.7: Árvore filogenética baseada em rearranjos de genomas usando o Algoritmo 25 (com a estratégia gulosa da heurística I) e 97 genomas do sistema Cyanorak 2.1.

4.2 Abordagem Ponderada

Nesta seção, apresentaremos os resultados considerando diferentes funções de custo para as variações com e sem sinais dos problemas **Sb$_{\mathbf{WI}}$RI**, **Sb$_{\mathbf{WI}}$RTI** e **Sb$_{\mathbf{WI}}$RT** em um cenário ponderado.

A primeira função de custo, que chamamos de W_1, considera que os eventos de reversão e transposição possuem custo igual a 1, enquanto um indel tem um custo equivalente à quantidade de nucleotídeos inseridos ou removidos da região intergênica. Essa função de custo foi estudada para investigar limitantes para o modelo considerando o caso em que o peso do evento de indel é variável e está diretamente ligado com o quanto a região intergênica é afetada. A seguir, descrevemos formalmente a função de custo W_1:

$$W_1(\beta) = \begin{cases} |x|, \textit{para } \beta = \delta_{(x)}^{(i)} \\ 1, \textit{para } \beta = \rho_{(x,y)}^{(i,j)} \\ 1, \textit{para } \beta = \tau_{(x,y,z)}^{(i,j,k)} \end{cases}$$

Dada uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, denotamos por $IR(\mathcal{I}) = |\sum_{\breve{\pi}_i \in \breve{\pi}} \breve{\pi}_i - \sum_{\breve{\iota}_i \in \breve{\iota}} \breve{\iota}_i|$ a quantidade de nucleotídeos que precisam ser removidos ou inseridos no genoma de origem para que $\mathcal{I}$ seja uma instância balanceada. Note que reversões e transposições podem transferir nucleotídeos entre regiões intergênicas, mas não removem ou inserem nucleotídeos. Logo, toda instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ para os problemas **Sb$_{\mathbf{WI}}$RI** e **Sb$_{\mathbf{WI}}$RTI** induz um custo mínimo de $IR(\mathcal{I})$, o que resulta nos seguintes limitantes inferiores.

Teorema 4.2.1. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Com base na função de custo W_1, temos que:*

$dwi_{\boldsymbol{RI}}(\mathcal{I}) \geq \max(IR(\mathcal{I}), \frac{ib_1(\mathcal{I})}{2})$ *e*

$dwi_{\boldsymbol{RTI}}(\mathcal{I}) \geq \max(IR(\mathcal{I}), \frac{ib_1(\mathcal{I})}{3})$.

Demonstração. Diretamente pelo Teorema 4.1.9 e pelo custo mínimo de $IR(\mathcal{I})$ discutido anteriormente. □

Teorema 4.2.2. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Com base na função de custo W_1, temos que:*

$dwi_{\boldsymbol{RI}}(\mathcal{I}) \geq \max(IR(\mathcal{I}), \frac{ib_2(\mathcal{I})}{2})$ *e*

$dwi_{\boldsymbol{RTI}}(\mathcal{I}) \geq \max(IR(\mathcal{I}), \frac{ib_2(\mathcal{I})}{3})$.

Demonstração. Diretamente pelo Teorema 4.1.11 e pelo custo mínimo de $IR(\mathcal{I})$ discutido anteriormente. □

A seguir, mostramos que, com base na função de custo W_1, é possível garantir aproximações para os problemas **Sb$_{\mathbf{WI}}$RI** e **Sb$_{\mathbf{WI}}$RTI** a partir de alguns dos algoritmos apresentados para o cenário não ponderado.

Teorema 4.2.3. *O Algoritmo 5 é uma 5-aproximação para a variação sem sinais do problema **Sb$_{\boldsymbol{WI}}$RI** considerando a função de custo W_1.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Note que o Algoritmo 5 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando, no máximo, $2ib_1(\mathcal{I})$ eventos de reversão e indel (Lema 4.1.27). Entretanto, caso $\mathcal{I}$ seja uma instância desbalanceada, então o Algoritmo 5 aplica, no máximo, um indel com custo $IR(\mathcal{I})$. Caso contrário, apenas reversões são utilizadas. Logo, no pior caso, uma sequência de eventos de rearranjo fornecida pelo Algoritmo 5 tem um custo total de $IR(\mathcal{I}) + 2ib_1(\mathcal{I})$. Agora, consideramos duas possibilidades:

- $IR(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$: Neste caso, $2IR(\mathcal{I}) \geq ib_1(\mathcal{I})$ e o custo máximo da sequência de eventos de rearranjo fornecida pelo Algoritmo 5 é $5IR(\mathcal{I})$.
- $IR(\mathcal{I}) < \frac{ib_1(\mathcal{I})}{2}$: Neste caso, o custo máximo da sequência de eventos de rearranjo fornecida pelo Algoritmo 5 é $\frac{5ib_1(\mathcal{I})}{2}$.

Em ambos os casos, o custo máximo da sequência de eventos de rearranjo fornecida pelo Algoritmo 5 dividido pelo valor do limitante inferior, apresentado no Teorema 4.2.1, é igual a 5 e o teorema segue. □

Teorema 4.2.4. *O Algoritmo 17 é uma 5-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_{WI}RI}$ *considerando a função de custo* W_1.

Demonstração. A prova é similar à descrita no Teorema 4.2.3, mas considerando o Algoritmo 17, que utiliza o Algoritmo 5 em parte de sua execução, e o limitante inferior apresentado no Teorema 4.2.2. □

Teorema 4.2.5. *O Algoritmo 11 é uma 5-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_{WI}RTI}$ *considerando a função de custo* W_1.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.1.53, o Algoritmo 11 utiliza, no máximo, $\frac{4ib_1(\mathcal{I})}{3}$ ou $\frac{4ib_1(\mathcal{I})}{3} + 1$ eventos de reversão, transposição e indel para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, se $\mathcal{I}$ for balanceada ou desbalanceada, respectivamente. Entretanto, caso $\mathcal{I}$ seja uma instância desbalanceada, então Algoritmo 11 aplica, no máximo, um indel com custo $IR(\mathcal{I})$. Caso contrário, apenas reversões e transposições são utilizadas. Logo, no pior caso, uma sequência de eventos de rearranjo fornecida pelo Algoritmo 11 tem um custo total de $IR(\mathcal{I}) + \frac{4ib_1(\mathcal{I})}{3}$. Agora, consideramos duas possibilidades:

- $IR(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$: Neste caso, $3IR(\mathcal{I}) \geq ib_1(\mathcal{I})$ e o custo máximo da sequência de eventos de rearranjo fornecida pelo Algoritmo 11 é $IR(\mathcal{I}) + \frac{12IR(\mathcal{I})}{3} = 5IR(\mathcal{I})$.
- $IR(\mathcal{I}) < \frac{ib_1(\mathcal{I})}{3}$: Neste caso, o custo máximo da sequência de eventos de rearranjo fornecida pelo Algoritmo 11 é $\frac{ib_1(\mathcal{I})}{3} + \frac{4ib_1(\mathcal{I})}{3} = \frac{5ib_1(\mathcal{I})}{3}$.

Em ambos os casos, o custo máximo da sequência de eventos de rearranjo fornecida pelo Algoritmo 11 dividido pelo valor do limitante inferior, apresentado no Teorema 4.2.1, é igual a 5 e o teorema segue. □

Teorema 4.2.6. *O Algoritmo 23 é uma 5-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_{WI}RI}$ *considerando a função de custo* W_1.

Demonstração. A prova é similar à descrita no Teorema 4.2.5, mas considerando o Algoritmo 23, que utiliza o Algoritmo 11 em parte de sua execução, e o limitante inferior apresentado no Teorema 4.2.2. □

A segunda função de custo, chamada de W_2, é baseada no fato de que um indel afeta apenas uma região intergênica do genoma e não altera a ordem dos genes. Dessa forma, consideramos que o evento de indel possui a metade do custo de um evento de reversão ou transposição. A seguir, descrevemos formalmente a função de custo W_2:

$$W_2(\beta) = \begin{cases} \frac{1}{2}, \textit{para } \beta = \delta_{(x)}^{(i)} \\ 1, \textit{para } \beta = \rho_{(x,y)}^{(i,j)} \\ 1, \textit{para } \beta = \tau_{(x,y,z)}^{(i,j,k)} \end{cases}$$

Com base nos custos da função W_2, obtemos para as variações com e sem sinais do problema $\mathbf{Sb_{WI}RTI}$, os seguintes limitantes inferiores.

Teorema 4.2.7. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Com base na função de custo W_2, temos que $dwi_{\boldsymbol{RTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$.*

Demonstração. Pelos lemas 4.1.1, 4.1.2 e 4.1.4, sabemos que, por operação, os eventos de reversão, transposição e indel removem, respectivamente, no máximo 2, 3 e 1 breakpoint tipo um. Com base no melhor caso de cada evento de rearranjo e na função de custo W_2, temos que:

$$dwi_{\mathbf{RTI}}(\mathcal{I}) \geq \min\left(\frac{ib_1(\mathcal{I})}{2} \times 1, \frac{ib_1(\mathcal{I})}{3} \times 1, ib_1(\mathcal{I}) \times \frac{1}{2}\right) = \frac{ib_1(\mathcal{I})}{3},$$

e o teorema segue. □

Teorema 4.2.8. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Com base na função de custo W_2, temos que $dwi_{\boldsymbol{RTI}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$.*

Demonstração. A prova é similar à descrita no Teorema 4.2.7, mas considerando os lemas 4.1.5, 4.1.6 e 4.1.8. □

Lema 4.2.9. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) um par conectado de breakpoints. É possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando no máximo uma reversão, uma transposição ou dois indels.*

Demonstração. Sem perda de generalidade, assuma que $i < j$. Como os breakpoints (π_i, π_{i+1}) e (π_j, π_{j+1}) estão conectados, por definição, uma das seguintes possibilidades deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma reversão como descrito no caso (i) do Lema 4.1.22.

ii. O par de elementos (π_i, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma transposição como descrito no caso (ii) do Lema 4.1.34.

iii. O par de elementos (π_{i+1}, π_j) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Aplique uma transposição como descrito no caso (iii) do Lema 4.1.34.

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não forma uma adjacência intergênica, sendo os elementos consecutivos em ι e $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} \geq \breve{\iota}_k$, onde $\breve{\iota}_k$ é o tamanho da região intergênica entre o par de elementos consecutivos em ι. Neste caso, (π_i, π_{i+1}) ou (π_j, π_{j+1}) deve ser um breakpoint forte. Caso (π_i, π_{i+1}) seja um breakpoint forte, então ele pode ser subcarregado ou sobrecarregado. Caso (π_i, π_{i+1}) seja subcarregado, então aplique a sequência de dois indels $(\delta^{i+1}_{(x)}, \delta^{j+1}_{(-x)})$, com $x = |\breve{\pi}_{i+1} - \breve{\iota}_{\max(\pi_i, \pi_{i+1})}|$. Caso contrário, aplique a sequência de dois indels $(\delta^{i+1}_{(-x)}, \delta^{j+1}_{(x)})$, com $x = |\breve{\pi}_{i+1} - \breve{\iota}_{\max(\pi_i, \pi_{i+1})}|$. Caso (π_j, π_{j+1}) seja um breakpoint forte, basta replicar a sequência de indels considerando a mudança no breakpoint. Note que essa sequência sempre pode ser aplicada, pois os dois breakpoints estão conectados. Além disso, o breakpoint tipo um é removido sem afetar quantidade total de nucleotídeos de $\mathcal{I}$.

Note que o caso (i) aplica apenas um reversão e remove, pelo menos, um breakpoint tipo um. Os casos (ii) e (iii) aplicam apenas uma transposição e removem, pelo menos, um breakpoint tipo um. Por fim, o caso (iv) remove, pelo menos, um breakpoint tipo um após aplicar dois indels. Logo, o lema segue. □

Considere o Algoritmo 28 para a variação sem sinais do problema **Sb$_{\mathbf{WI}}$RTI** considerando a função de custo W_2.

Lema 4.2.10. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. O Algoritmo 28, considerando a função de custo W_2, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão, transposição e indel S, com um custo total de, no máximo, $ib_1(\mathcal{I})$.*

Demonstração. Podemos analisar o Algoritmo 28 considerando três cenários:

- $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$: neste cenário o Algoritmo 28 aplica um indel (linhas 2-5) que pode não remover nenhum breakpoint tipo um, mas torna $\mathcal{I}$ uma instância balanceada. Caso ainda existam breakpoints em $\mathcal{I}$, então o laço de repetição (linhas 6-17) remove, por iteração, pelo menos um breakpoint tipo um utilizando, no máximo, uma reversão, uma transposição ou dois indels (Lema 4.2.9). Esse processo repete-se até que todos os breakpoints tipo um de $\mathcal{I}$ sejam removidos. Como todos os breakpoints tipo um são removidos, então $(\pi, \breve{\pi})$ é transformado em $(\iota, \breve{\iota})$. Note que se o indel aplicado inicialmente não remover nenhum breakpoint tipo um, então pelo menos uma reversão, uma transposição ou dois indels são aplicados em seguida. Além disso, pelo Lema 4.1.21, podemos deduzir que caso a última operação que

Algoritmo 28: Um algoritmo de aproximação para o problema $\mathbf{Sb_{WI}RTI}$.

Entrada: Uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de reversões transposições e indels S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

1 Seja $S \leftarrow ()$
▷ Lema 4.1.25
2 **se** $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$ **então**
3 $S' \leftarrow (\delta_1)$
4 $S \leftarrow S + S'$
5 $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
6 **enquanto** $ib_1(\mathcal{I}) > 1$ **faça**
▷ Lema 4.1.19
7 $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
▷ Lema 4.2.9
8 **se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (i) **então**
9 $S' \leftarrow (\rho_1)$
10 **senão se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (ii) **então**
11 $S' \leftarrow (\tau_1)$
12 **senão se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iii) **então**
13 $S' \leftarrow (\tau_1)$
14 **senão se** $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso* (iv) **então**
15 $S' \leftarrow (\delta_1, \delta_2)$
16 $S \leftarrow S + S'$
17 $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
18 ▷ Lema 4.1.26
19 **se** $ib(\mathcal{I}) = 1$ **então**
20 $S' \leftarrow (\delta_1)$
21 $S \leftarrow S + S'$
22 $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
23 **retorne** S

transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ seja uma reversão ou uma transposição, então ela deve, obrigatoriamente, remover pelo menos dois breakpoints tipo um. Caso essa operação seja um indel, então as duas últimas operações são indels e ambas removem um breakpoint tipo um. Isso implica que o custo total da sequência fornecida pelo Algoritmo 28 é de, no máximo, $ib_1(\mathcal{I})$.

- $\sum_{i=1}^{n+1} \breve{\pi}_i = \sum_{i=1}^{n+1} \breve{\iota}_i$: para esse cenário o Algoritmo 28 executará o laço de repetição (linhas 6-17) que remove, por iteração, pelo menos um breakpoint tipo um utilizando, no máximo, uma reversão, uma transposição ou dois indels (Lema 4.2.9). Logo, o custo total da sequência fornecida pelo Algoritmo 28 é de, no máximo, $ib_1(\mathcal{I})$.

- $\sum_{i=1}^{n+1} \breve{\pi}_i > \sum_{i=1}^{n+1} \breve{\iota}_i$: neste último cenário, enquanto $ib_1(\mathcal{I})$ for maior que um, o Algoritmo 28 aplica, no máximo, uma reversão, uma transposição ou dois indels em cada iteração do laço de repetição (linhas 6-17) que removem pelo menos um breakpoint tipo um. Por fim, um indel é aplicado (linhas 19-22) transformando

$(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. O custo para remover cada breakpoint nesse caso é de, no máximo, um. Logo, o custo total da sequência fornecida pelo Algoritmo 28 é de, no máximo, $ib_1(\mathcal{I})$.

Nos três cenários o Algoritmo 28 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de reversões, transposições e indels com um custo total de, no máximo, $ib_1(\mathcal{I})$, e o lema segue. □

Note que o tempo de execução do Algoritmo 28 é $\mathcal{O}(n^2)$, uma vez que aplicar cada uma das operações requer um tempo linear (considerando encontrar um par conectado de breakpoints) e o processo pode repetir-se por até $ib_1(\mathcal{I}) \leq n + 1$ vezes.

Teorema 4.2.11. *O Algoritmo 28 é uma 3-aproximação para a variação sem sinais do problema* ***Sb$_{WI}$RTI*** *considerando a função de custo W_2.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.2.10, o Algoritmo 28 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão, transposição e indel S, com um custo total de, no máximo, $ib_1(\mathcal{I})$. Pelo limitante inferior, apresentado no Teorema 4.2.7, temos que: $dwi_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

Considere o Algoritmo 29 para a variação com sinais do problema **Sb$_{WI}$RTI** considerando a função de custo W_2.

Algoritmo 29: Um algoritmo de aproximação para o problema **Sb$_{WI}$RTI**.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão, transposição e indel S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição e indel fornecida pelo Algoritmo 28 para a instância $\mathcal{I}'$
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

Lema 4.2.12. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. O Algoritmo 28, considerando a função de custo W_2, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão, transposição e indel S, com um custo total de, no máximo, $ib_2(\mathcal{I})$.*

Demonstração. Note que o Algoritmo 28 fornece uma sequência de reversões, transposições e indels que afetam apenas breakpoints tipo um. Logo, nenhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.2.10, o Algoritmo 28 utiliza uma sequência de eventos de reversão, transposição e indel, com um custo total de, no máximo, $ib_1(\mathcal{I}')$. Pelo Lema 4.1.61, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'|$. Além disso, S e $|S'|$ possuem a mesma quantidade de operações por tipo. Pelo Lema 4.1.60, temos que $ib_1(\mathcal{I}') = ib_2(\mathcal{I})$.

Logo, o custo total da sequência de eventos de reversão, transposição e indel fornecida pelo Algoritmo 29 é de, no máximo, $ib_2(\mathcal{I})$. □

O Algoritmo 29 também possui um tempo de execução de $\mathcal{O}(n^2)$, uma vez que as funções $\mathcal{F}$ e $\mathcal{G}$ são executadas em tempo linear e o Algoritmo 28, no pior caso, requer um tempo quadrático.

Teorema 4.2.13. *O Algoritmo 29 é uma 3-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_{WI}RTI}$ *considerando a função de custo* W_2.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida com sinais. Pelo Lema 4.2.12, o Algoritmo 29 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão, transposição e indel, com um custo total de, no máximo, $ib_2(\mathcal{I})$. Pelo limitante inferior, apresentado no Teorema 4.2.8, temos que: $dwi_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$. Logo, o teorema segue. □

A terceira função de custo, chamada de W_3, é baseada na proporção de regiões intergênicas afetadas por cada evento de rearranjo [4]. A seguir, descrevemos formalmente a função de custo W_3:

$$W_3(\beta) = \begin{cases} 1, \textit{para } \beta = \delta_{(x)}^{(i)} \\ 2, \textit{para } \beta = \rho_{(x,y)}^{(i,j)} \\ 3, \textit{para } \beta = \tau_{(x,y,z)}^{(i,j,k)} \end{cases}$$

Com base nos custos da função W_3, obtemos para as variações com e sem sinais do problema $\mathbf{Sb_{WI}RTI}$, os seguintes limitantes inferiores.

Teorema 4.2.14. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida sem sinais. Com base na função de custo* W_3, *temos que* $dwi_{\boldsymbol{RTI}}(\mathcal{I}) \geq ib_1(\mathcal{I})$.

Demonstração. Pelos lemas 4.1.1, 4.1.2 e 4.1.4, sabemos que, por operação, os eventos de reversão, transposição e indel removem, respectivamente, no máximo 2, 3 e 1 breakpoint tipo um. Com base no melhor caso de cada evento de rearranjo e na função de custo W_3, temos que:

$$dwi_{\mathbf{RTI}}(\mathcal{I}) \geq \min\left(\frac{ib_1(\mathcal{I})}{2} \times 2, \frac{ib_1(\mathcal{I})}{3} \times 3, ib_1(\mathcal{I}) \times 1\right) = ib_1(\mathcal{I}),$$

e o teorema segue. □

Teorema 4.2.15. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida com sinais. Com base na função de custo* W_3, *temos que* $dwi_{\boldsymbol{RTI}}(\mathcal{I}) \geq ib_2(\mathcal{I})$.

Demonstração. A prova é similar à descrita no Teorema 4.2.14, mas considerando os lemas 4.1.5, 4.1.6 e 4.1.8. □

Lema 4.2.16. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida sem sinais. o Algoritmo 28, considerando a função de custo* W_3, *transforma* $(\pi, \breve{\pi})$ *em* $(\iota, \breve{\iota})$ *utilizando uma sequência de eventos de reversão, transposição e indel* S, *com um custo total de, no máximo,* $3ib_1(\mathcal{I})$.

Demonstração. A prova é similar à descrita no Lema 4.2.10 considerando a função de custo W_3. □

Teorema 4.2.17. *O Algoritmo 28 é uma 3-aproximação para a variação sem sinais do problema* $\boldsymbol{Sb_{WI}RTI}$ *considerando a função de custo* W_3.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida sem sinais. Pelo Lema 4.2.16, o Algoritmo 28 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão, transposição e indel S, com um custo total de, no máximo, $3ib_1(\mathcal{I})$. Pelo limitante inferior, apresentado no Teorema 4.2.14, temos que: $dwi_{\mathbf{RTI}}(\mathcal{I}) \geq ib_1(\mathcal{I})$. Logo, o teorema segue. □

Lema 4.2.18. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida com sinais. O Algoritmo 29, considerando a função de custo* W_3*, transforma* $(\pi, \breve{\pi})$ *em* $(\iota, \breve{\iota})$ *utilizando uma sequência de eventos de reversão, transposição e indel* S*, com um custo total de, no máximo,* $3ib_2(\mathcal{I})$.

Demonstração. A prova é similar à descrita no Lema 4.2.12 considerando a função de custo W_3. □

Teorema 4.2.19. *O Algoritmo 29 é uma 3-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_{WI}RTI}$ *considerando a função de custo* W_3.

Demonstração. Pelo Lema 4.2.18, o Algoritmo 29 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão, transposição e indel S, com um custo total de, no máximo, $3ib_2(\mathcal{I})$. Pelo limitante inferior, apresentado no Teorema 4.2.15, temos que: $dwi_{\mathbf{RTI}}(\mathcal{I}) \geq ib_2(\mathcal{I})$. Logo, o teorema segue. □

A seguir, apresentamos duas funções de custo para o problema $\mathbf{Sb_{WI}RT}$ que são amplamente utilizadas na literatura em modelos compostos pelos eventos conservativos de reversão e transposição. A primeira função de custo, chamada de W_4, adota os custos 1 e 2 para os eventos de reversão e transposição, respectivamente [42]. A segunda função de custo, chamada de W_5, adota os custos 2 e 3 para os eventos de reversão e transposição, respectivamente [56]. A seguir, descrevemos formalmente as funções de custo W_4 e W_5:

$$W_4(\beta) = \begin{cases} 1, \textit{para } \beta = \rho^{(i,j)}_{(x,y)} \\ 2, \textit{para } \beta = \tau^{(i,j,k)}_{(x,y,z)} \end{cases}$$

$$W_5(\beta) = \begin{cases} 2, \textit{para } \beta = \rho^{(i,j)}_{(x,y)} \\ 3, \textit{para } \beta = \tau^{(i,j,k)}_{(x,y,z)} \end{cases}$$

Com base nos custos da funções W_4, obtemos para as variações com e sem sinais do problema $\mathbf{Sb_{WI}RT}$, os seguintes limitantes inferiores.

Teorema 4.2.20. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida balanceada sem sinais. Com base na função de custo* W_4*, temos que* $dwi_{\boldsymbol{RT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$.

Demonstração. Pelos lemas 4.1.1, 4.1.2, sabemos que, por operação, os eventos de reversão e transposição removem, respectivamente, no máximo, 2 e 3 breakpoints tipo um. Com base no melhor caso de cada evento de rearranjo e na função de custo W_4, temos que:

$$dwi_{\mathbf{RT}}(\mathcal{I}) \geq \min\left(\frac{ib_1(\mathcal{I})}{2} \times 1, \frac{ib_1(\mathcal{I})}{3} \times 2\right) = \frac{ib_1(\mathcal{I})}{2},$$

e o teorema segue. □

Teorema 4.2.21. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. Com base na função de custo W_4, temos que $dwi_{\boldsymbol{RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{2}$.*

Demonstração. A prova é similar à descrita no Teorema 4.2.20, mas considerando os lemas 4.1.5 e 4.1.6. □

Lema 4.2.22. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 9, considerando a função de custo W_4, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição S, com um custo total de, no máximo, $2ib_1(\mathcal{I})$.*

Demonstração. Pelo Lema 4.1.43, sabemos que o Algoritmo 9 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Para obtermos o custo total máximo de uma sequência de reversões e transposições fornecida pelo Algoritmo 9 vamos considerar suas duas fases: (i) remoção de breakpoints sobrecarregados: nessa fase cada breakpoint tipo um é removido com um custo médio de 2, no pior caso; (ii) remoção de breakpoints suaves: no pior caso, cada breakpoint tipo um é removido por uma transposição, que tem custo 2. Dessa forma, o custo total máximo de uma sequência de reversões e transposições fornecida pelo Algoritmo 9 é $2ib_1(\mathcal{I})$, e o lema segue. □

Teorema 4.2.23. *O Algoritmo 9 é uma 4-aproximação para a variação sem sinais do problema $\boldsymbol{Sb_{WI}RT}$ considerando a função de custo W_4.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.2.22, o Algoritmo 9, com base na função de custo W_4, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição com um custo total de, no máximo, $2ib_1(\mathcal{I})$. Pelo limitante inferior, apresentado no Teorema 4.2.20, temos que: $dwi_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{2}$. Logo, o teorema segue. □

Considere o Algoritmo 30 para a variação com sinais do problema $\mathbf{Sb_{WI}RT}$ considerando a função de custo W_4.

Lema 4.2.24. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. O Algoritmo 30, considerando a função de custo W_4, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição S, com um custo total de, no máximo, $2ib_2(\mathcal{I})$.*

Demonstração. Note que o Algoritmo 9 fornece uma sequência de reversões e transposições que afetam apenas breakpoints tipo um. Logo, nenhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.2.22, o Algoritmo 9 utiliza uma sequência de eventos

Algoritmo 30: Um algoritmo de aproximação para o problema $\mathbf{Sb_{WI}RT}$.

Entrada: Uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$
Saída: Uma sequência de eventos de reversão e transposição S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

▷ `Lema 4.1.60`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e transposição fornecida pelo Algoritmo 9
▷ `Lema 4.1.61`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorne** S

de reversão e transposição com um custo total de, no máximo, $2ib_1(\mathcal{I}')$ para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$. Pelo Lema 4.1.61, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'|$. Além disso, S e $|S'|$ possuem a mesma quantidade de operações por tipo. Pelo Lema 4.1.60, temos que $2ib_1(\mathcal{I}') = 2ib_2(\mathcal{I})$. Logo, o custo total da sequência de eventos de reversão e transposição fornecida pelo Algoritmo 30 é de, no máximo, $2ib_2(\mathcal{I})$. □

Teorema 4.2.25. *O Algoritmo 30 é uma 4-aproximação para a variação com sinais do problema* $\boldsymbol{Sb_{WI}RT}$ *considerando a função de custo* W_4.

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. Pelo Lema 4.2.24, o Algoritmo 30, com base na função de custo W_4, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição com um custo total de, no máximo, $2ib_2(\mathcal{I})$. Pelo limitante inferior, apresentado no Teorema 4.2.21, temos que $dwi_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{2}$. Logo, o teorema segue. □

Com base nos custos da funções W_5, obtemos para as variações com e sem sinais do problema $\mathbf{Sb_{WI}RT}$, os seguintes limitantes inferiores.

Teorema 4.2.26. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida balanceada sem sinais. Com base na função de custo* W_5*, temos que* $dwi_{\boldsymbol{RT}}(\mathcal{I}) \geq ib_1(\mathcal{I})$.

Demonstração. Pelos lemas 4.1.1 e 4.1.2, sabemos que, por operação, os eventos de reversão e transposição removem, respectivamente, no máximo 2e 3 breakpoints tipo um. Com base no melhor caso de cada evento de rearranjo e na função de custo W_5, temos que:

$$dwi_{\mathbf{RT}}(\mathcal{I}) \geq \min\left(\frac{ib_1(\mathcal{I})}{2} \times 2, \frac{ib_1(\mathcal{I})}{3} \times 3\right) = ib_1(\mathcal{I}),$$

e o teorema segue. □

Teorema 4.2.27. *Seja* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *uma instância intergênica rígida balanceada com sinais. Com base na função de custo* W_5*, temos que* $dwi_{\boldsymbol{RT}}(\mathcal{I}) \geq ib_2(\mathcal{I})$.

Demonstração. A prova é similar à descrita no Teorema 4.2.26, mas considerando os lemas 4.1.5 e 4.1.6. □

Lema 4.2.28. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. O Algoritmo 9, considerando a função de custo W_5, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição S, com um custo total de, no máximo, $\frac{7ib_1(\mathcal{I})}{2}$.*

Demonstração. Pelo Lema 4.1.43, sabemos que o Algoritmo 9 transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Para obtermos o custo total máximo de uma sequência de reversões e transposições fornecida pelo Algoritmo 9 vamos considerar suas duas fases: (i) remoção de breakpoints sobrecarregados: nessa fase cada breakpoint tipo um é removido com um custo médio de 4, no pior caso; (ii) remoção de breakpoints suaves: no pior caso, cada breakpoint tipo um é removido por uma transposição, que tem custo 3. Entretanto, se ocorrer o pior caso da fase de remoção de breakpoints sobrecarregados, então a fase de remoção de breakpoints suaves será executada em seguida. Dessa forma, no pior caso, dois breakpoints tipo um são removidos com um custo de $4 + 3 = 7$. Logo, o custo total máximo de uma sequência de reversões e transposições fornecida pelo Algoritmo 9 é $\frac{ib_1(\mathcal{I})}{2} \times 7 = \frac{7ib_1(\mathcal{I})}{2}$, e o lema segue. □

Teorema 4.2.29. *O Algoritmo 9 é uma 3.5-aproximação para a variação sem sinais do problema $\boldsymbol{Sb_{WI}RT}$ considerando a função de custo W_5.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada sem sinais. Pelo Lema 4.2.28, o Algoritmo 9, com base na função de custo W_5, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição com um custo total de, no máximo, $\frac{7ib_1(\mathcal{I})}{2}$. Pelo limitante inferior, apresentado no Teorema 4.2.26, temos que: $dwi_{\mathbf{RT}}(\mathcal{I}) \geq ib_1(\mathcal{I})$. Logo, o teorema segue. □

Lema 4.2.30. *Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. O Algoritmo 30, considerando a função de custo W_5, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição S, com um custo total de, no máximo, $\frac{7ib_2(\mathcal{I})}{2}$.*

Demonstração. A prova é similar à descrita no Lema 4.2.24 considerando a função de custo W_5. □

Teorema 4.2.31. *O Algoritmo 30 é uma 3.5-aproximação para a variação com sinais do problema $\boldsymbol{Sb_{WI}RT}$ considerando a função de custo W_5.*

Demonstração. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida balanceada com sinais. Pelo Lema 4.2.30, o Algoritmo 9, com base na função de custo W_5, transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando uma sequência de eventos de reversão e transposição com um custo total de, no máximo, $\frac{7ib_2(\mathcal{I})}{2}$. Pelo limitante inferior, apresentado no Teorema 4.2.27, temos que: $dwi_{\mathbf{RT}}(\mathcal{I}) \geq ib_2(\mathcal{I})$. Logo, o teorema segue. □

4.3 Conclusões

Neste capítulo, investigamos as variações com e sem sinais de oito problemas considerando instâncias intergênicas rígidas em um cenário não ponderado e os eventos de rearranjo de

reversão, transposição, move e indel. Para todas as variações dos problemas em que a complexidade ainda era desconhecida, apresentamos uma prova de NP-dificuldade, com exceção da variação com sinais do problema de Ordenação de Permutações por Operações Intergênicas de Reversão e Indel ($\mathbf{Sb_IRI}$). Para todas as variações investigadas, apresentamos pelo menos um algoritmo de aproximação com fator constante baseado no conceito de breakpoint intergênico. Além disso, apresentamos algoritmos com fatores de aproximação melhores baseados na estrutura de grafo de ciclos ponderado rígido.

Realizamos testes experimentais com os algoritmos propostos para verificar o desempenho prático dos mesmos. Além disso, utilizamos dados de 97 genomas reais e construímos uma árvore filogenética com base exclusivamente na matriz de distâncias fornecida por um dos nossos algoritmos. Comparamos essa árvore filogenética com outra presente na literatura, construída com os mesmos 97 genomas, e o resultado apontou que existe uma alta concordância entre elas.

Em um cenário ponderado, investigamos as variações com e sem sinais de três problemas considerando instâncias intergênicas rígidas e os eventos de rearranjo de reversão, transposição e indel. Consideramos diferentes funções de custo e apresentamos algoritmos de aproximação com um fator constante para todas as variações investigadas.

Capítulo 5

Modelos Intergênicos Flexíveis

Neste capítulo, investigaremos problemas que consideram, além da ordem dos genes e do tamanho das regiões intergênicas, um certo grau de flexibilidade em relação ao tamanho das regiões intergênicas no genoma alvo desejado. Nesse contexto, consideraremos os eventos de reversão intergênica, transposição intergênica, move intergênico e indel intergênico, e investigaremos as variações com e sem sinais dos seguintes problemas:

- Ordenação de Permutações por Reversões Intergênicas com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}R}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Indel com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RI}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Move com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RM}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Move e Indel com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RMI}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Transposição com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RT}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Indel com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RTI}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Move com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RTM}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição, Move e Indel com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}RTMI}$)

Além disso, investigaremos as variações sem sinais dos seguintes problemas:

- Ordenação de Permutações por Transposições Intergênicas com Regiões Intergênicas Flexíveis ($\mathbf{Sb_{FI}T}$)

- Ordenação de Permutações por Operações Intergênicas de Transposição e Move com Regiões Intergênicas Flexíveis (**Sb$_{\mathbf{FI}}$TM**)

Note que nesses dois últimos problemas, tanto o evento de transposição intergênica como o evento de move intergênico não alteram a orientação dos genes. Por esse motivo, apenas a variação sem sinais será investigada.

Neste capítulo, iremos referenciar aos eventos de rearranjo de reversão intergênica, transposição intergênica, move intergênico e indel intergênico simplesmente por reversão, transposição, move e indel, respectivamente. Além disso, referiremos a um breakpoint intergênico simplesmente por breakpoint. Dada uma sequência de eventos de rearranjo S, denotamos por $|S|$ o tamanho da sequência S, ou seja, a quantidade de eventos em S.

Dada uma instância intergênica flexível com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, a *distância flexível* entre $(\pi, \breve{\pi})$ e $(\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max})$, denotada por $dfi_{\mathcal{M}}(\mathcal{I})$, é o tamanho da menor sequência de eventos de rearranjo S, tal que todo evento de S pertence ao modelo $\mathcal{M}$ e $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\pi}')$, onde $\breve{\iota}_i^{\min} \leq \breve{\pi}'_i \leq \breve{\iota}_i^{\max}$ para $i \in [1..n+1]$. Os modelos de rearranjo considerados neste capítulo são identificados pelas siglas apresentadas na Tabela 5.1.

Tabela 5.1: Siglas dos modelos de rearranjo considerados para instâncias intergênicas flexíveis.

Problema	**Sigla do Modelo**	**Conjunto de Eventos de Rearranjo**
Sb$_{\mathbf{FI}}$R	**R**	$\{\rho\}$
Sb$_{\mathbf{FI}}$RI	**RI**	$\{\rho, \delta\}$
Sb$_{\mathbf{FI}}$RM	**RM**	$\{\rho, \mu\}$
Sb$_{\mathbf{FI}}$RMI	**RMI**	$\{\rho, \mu, \delta\}$
Sb$_{\mathbf{FI}}$T	**T**	$\{\tau\}$
Sb$_{\mathbf{FI}}$TM	**TM**	$\{\tau, \mu\}$
Sb$_{\mathbf{FI}}$RT	**RT**	$\{\rho, \tau\}$
Sb$_{\mathbf{FI}}$RTI	**RTI**	$\{\rho, \tau, \delta\}$
Sb$_{\mathbf{FI}}$RTM	**RTM**	$\{\rho, \tau, \mu\}$
Sb$_{\mathbf{FI}}$RTMI	**RTMI**	$\{\rho, \tau, \mu, \delta\}$

Dada uma instância intergênica flexível com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, utilizaremos as expressões *atingir o genoma alvo* ou *alcançar o genoma alvo* quando π é transformado em ι e para toda região intergênica $\breve{\pi}_i$ em $\breve{\pi}$, temos que $\breve{\iota}_i^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_i^{\max}$.

Parte dos resultados que serão apresentados neste capítulo foram aceitos para publicação na revista *IEEE/ACM Transactions on Computational Biology and Bioinformatics* [20] em 2022.

5.1 Análise de Complexidade

Nesta seção, realizaremos uma análise de complexidade dos problemas que consideram um grau de flexibilidade em relação ao tamanho das regiões intergênicas no genoma alvo desejado.

Note que todos os problemas investigados neste capítulo generalizam suas respectivas versões que consideram um valor estrito (rígido) para o tamanho das regiões intergênicas

desejadas no genoma alvo. Isso pode ser facilmente constatado por meio de uma redução. No Capítulo 4 foi mostrado que a variação sem sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$ pertencem à classe NP-difícil. Adicionalmente, temos que a variação sem sinais dos problemas $\mathbf{Sb_IT}$ e $\mathbf{Sb_ITM}$ também pertencem à classe NP-difícil [59]. Dessa forma, temos o seguinte lema.

Lema 5.1.1. *Os problemas* $\boldsymbol{Sb_{FI}R}$, $\boldsymbol{Sb_{FI}RI}$, $\boldsymbol{Sb_{FI}RM}$, $\boldsymbol{Sb_{FI}RMI}$, $\boldsymbol{Sb_{FI}RT}$, $\boldsymbol{Sb_{FI}RTI}$, $\boldsymbol{Sb_{FI}RTM}$, $\boldsymbol{Sb_{FI}RTMI}$, $\boldsymbol{Sb_{FI}T}$ *e* $\boldsymbol{Sb_{FI}TM}$, *em instâncias intergênicas flexíveis sem sinais, pertencem à classe NP-difícil.*

Demonstração. Sejam $\mathcal{P}_f$ e $\mathcal{P}_r$ problemas com base no mesmo modelo de rearranjo $\mathcal{M}$, mas com $\mathcal{P}_f$ e $\mathcal{P}_r$ possuindo, respectivamente, uma característica de flexibilidade e de rigidez em relação ao tamanho das regiões intergênicas no genoma alvo. Seja $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ uma instância intergênica rígida para o problema $\mathcal{P}_r$, então podemos criar uma instância intergênica flexível $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'^{\min}, \breve{\iota}'^{\max}))$ para o problema $\mathcal{P}_f$ da seguinte forma:

- $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$
- $\iota' = \iota$
- $\breve{\iota}'^{\min} = \breve{\iota}'^{\max} = \breve{\iota}$

Note que pela construção da instância $\mathcal{I}'$ e pelo fato de que $\mathcal{P}_f$ e $\mathcal{P}_r$ adotam o mesmo modelo de rearranjo $\mathcal{M}$, temos que $dfi_{\mathcal{M}}(\mathcal{I}') = di_{\mathcal{M}}(\mathcal{I})$. □

Além disso, o Capítulo 4 apresenta a informação de que a variação com sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$ também pertencem à classe NP-difícil. Com isso, obtemos o seguinte lema.

Lema 5.1.2. *Os problemas* $\boldsymbol{Sb_{FI}R}$, $\boldsymbol{Sb_{FI}RM}$, $\boldsymbol{Sb_{FI}RMI}$, $\boldsymbol{Sb_{FI}RT}$, $\boldsymbol{Sb_{FI}RTI}$, $\boldsymbol{Sb_{FI}RTM}$ *e* $\boldsymbol{Sb_{FI}RTMI}$, *em instâncias intergênicas flexíveis com sinais, pertencem à classe NP-difícil.*

Demonstração. A prova é similar à descrita no Lema 5.1.1. □

É importante lembrar que, semelhante à variação com sinais do problema $\mathbf{Sb_IRI}$, a variação com sinais do problema $\mathbf{Sb_{FI}RI}$ também permanece com a complexidade desconhecida.

5.2 Limitantes Inferiores

Nesta seção, apresentaremos limitantes inferiores para as variações com e sem sinais dos problemas investigados neste capítulo.

Para a variação sem sinais dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RI}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RMI}$, $\mathbf{Sb_{FI}RT}$, $\mathbf{Sb_{FI}RTI}$, $\mathbf{Sb_{FI}RTM}$ e $\mathbf{Sb_{FI}RTMI}$ utilizaremos o conceito de região intergênica flexível (Seção 2.5). Note que os eventos de rearranjo de reversão, transposição, move e indel afetam, respectivamente, as seguintes quantidades de regiões intergênicas: duas,

três, duas e uma. No melhor cenário, cada uma das regiões intergênicas afetadas pode ser instável ou auxiliar, e são removidas após a aplicação do evento de rearranjo. Com isso, obtemos os lemas descritos na sequência.

Lema 5.2.1. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. Para qualquer reversão ρ temos que $\Delta ir_i(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 5.2.2. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. Para qualquer transposição τ temos que $\Delta ir_i(\mathcal{I}, S = (\tau)) \geq -3$.*

Lema 5.2.3. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. Para qualquer move μ temos que $\Delta ir_i(\mathcal{I}, S = (\mu)) \geq -2$.*

Lema 5.2.4. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. Para qualquer indel δ temos que $\Delta ir_i(\mathcal{I}, S = (\delta)) \geq -1$.*

Além disso, considerando uma instância intergênica flexível balanceada sem sinais e com base em um modelo composto exclusivamente por eventos conservativos, temos os seguintes lemas:

Lema 5.2.5. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. Para qualquer reversão ρ temos que $\Delta ir_i(\mathcal{I}, S = (\rho)) + \Delta ir_a(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 5.2.6. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. Para qualquer transposição τ temos que $\Delta ir_i(\mathcal{I}, S = (\tau)) + \Delta ir_a(\mathcal{I}, S = (\tau)) \geq -3$.*

Lema 5.2.7. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. Para qualquer move μ temos que $\Delta ir_i(\mathcal{I}, S = (\mu)) + \Delta ir_a(\mathcal{I}, S = (\mu)) \geq -2$.*

Com base na quantidade máxima de regiões intergênicas instáveis e auxiliares que cada evento pode remover de uma instância intergênica flexível, obtemos os limitantes inferiores descritos a seguir.

Teorema 5.2.8. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. Temos que:*

$$\begin{aligned} dfi_{\boldsymbol{RI}}(\mathcal{I}) &\geq \tfrac{ir_i(\mathcal{I})}{2}, \\ dfi_{\boldsymbol{RMI}}(\mathcal{I}) &\geq \tfrac{ir_i(\mathcal{I})}{2}, \\ dfi_{\boldsymbol{RTI}}(\mathcal{I}) &\geq \tfrac{ir_i(\mathcal{I})}{3} \text{ e} \\ dfi_{\boldsymbol{RTMI}}(\mathcal{I}) &\geq \tfrac{ir_i(\mathcal{I})}{3}. \end{aligned}$$

Demonstração. Pela Observação 2.5.1, temos que todas as regiões intergênicas instáveis devem ser removidas para que o genoma alvo seja alcançado. Pelos lemas 5.2.1, 5.2.2, 5.2.3 e 5.2.4, temos que os eventos de reversão, transposição, move e indel podem remover, no máximo, 2, 3, 2 e 1 região intergênica instável, respectivamente. Como a instância $\mathcal{I}$ possui $ir_i(\mathcal{I})$ regiões intergênicas instáveis e considerando o máximo de regiões intergênicas instáveis que podem ser removidas por cada evento nos modelos de rearranjo **RI**, **RMI**, **RTI** e **RTMI**, o teorema segue. □

Teorema 5.2.9. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. Temos que:*

$$\begin{array}{lcl} dfi_{\boldsymbol{R}}(\mathcal{I}) & \geq & \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{2}, \\ dfi_{\boldsymbol{RM}}(\mathcal{I}) & \geq & \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{2}, \\ dfi_{\boldsymbol{RT}}(\mathcal{I}) & \geq & \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{3} \ e \\ dfi_{\boldsymbol{RTM}}(\mathcal{I}) & \geq & \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{3}. \end{array}$$

Demonstração. Note que todos os problemas listados possuem um modelo de rearranjo composto exclusivamente por eventos conservativos. Pela Observação 2.5.2, temos que todas as regiões intergênicas instáveis e auxiliares devem ser removidas para que o genoma alvo seja alcançado. Pelos lemas 5.2.5, 5.2.6 e 5.2.7, temos que a variação no número de regiões intergênicas instáveis acrescida da variação no número de regiões intergênicas auxiliares após aplicar um evento de reversão, transposição e move é maior ou igual a -2, -3 e -2, respectivamente. Como a instância $\mathcal{I}$ possui $ir_i(\mathcal{I}) + ir_a(\mathcal{I})$ regiões intergênicas instáveis ou auxiliares, considerando o máximo de regiões intergênicas instáveis ou auxiliares que podem ser removidas por cada evento nos modelos de rearranjo **R**, **RM**, **RT** e **RTM**, o teorema segue. □

A seguir, com base na estrutura de grafo de ciclos ponderado flexível (Seção 2.6.3), apresentamos limitantes inferiores para a variação sem sinais dos problemas $\mathbf{Sb_{FI}T}$ e $\mathbf{Sb_{FI}TM}$, e para a variação com sinais dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RI}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RMI}$, $\mathbf{Sb_{FI}RT}$ e $\mathbf{Sb_{FI}RTM}$.

Note que os eventos de reversão e transposição afetam, respectivamente, duas e três arestas pretas do grafo de ciclos ponderado flexível e podem aumentar tanto o número de ciclos como o número de ciclos estáveis e definitivos. O evento de move afeta duas arestas pretas do grafo, mas pode aumentar somente o número de ciclos estáveis e definitivos. Já o evento de indel afeta apenas uma aresta preta do grafo e pode aumentar apenas o número de ciclos estáveis.

Dessa forma, dada uma instância intergênica flexível $\mathcal{I}$, considerando a variação no número de ciclos (Δc), a variação no número ciclos estáveis (Δc_e) e a variação no número ciclos definitivos (Δc_d), temos que para qualquer reversão ρ, $\Delta c(G(\mathcal{I}), S = (\rho)) \in \{1, 0, -1\}$, $\Delta c_e(G(\mathcal{I}), S = (\rho)) \in \{1, 0, -1\}$ e $\Delta c_d(G(\mathcal{I}), S = (\rho)) \in \{1, 0, -1\}$. Para qualquer transposição τ, temos que $\Delta c(G(\mathcal{I}), S = (\tau)) \in \{2, 0, -2\}$, $\Delta c_e(G(\mathcal{I}), S = (\tau)) \in \{2, 1, 0, -1, -2\}$ e $\Delta c_d(G(\mathcal{I}), S = (\tau)) \in \{2, 1, 0, -1, -2\}$. Para qualquer move μ, temos que $\Delta c(G(\mathcal{I}), S = (\mu)) = 0$, $\Delta c_e(G(\mathcal{I}), S = (\mu)) \in \{2, 1, 0, -1, -2\}$ e $\Delta c_d(G(\mathcal{I}), S = (\mu)) \in \{2, 1, 0, -1, -2\}$. Por fim, para qualquer indel δ, temos que $\Delta c(G(\mathcal{I}), S = (\delta)) = 0$ e $\Delta c_e(G(\mathcal{I}), S = (\delta)) \in \{1, 0, -1\}$. Com isso, obtemos os seguintes limitantes inferiores:

Teorema 5.2.10. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. Temos que:*

$$\begin{array}{lcl} dfi_{\boldsymbol{T}}(\mathcal{I}) & \geq & \frac{n+1-c_d(G(\mathcal{I}))}{2} \ e \\ dfi_{\boldsymbol{TM}}(\mathcal{I}) & \geq & \frac{n+1-c_d(G(\mathcal{I}))}{2}. \end{array}$$

Demonstração. Pela Observação 2.6.5, temos que, para atingir o genoma alvo, é necessário que $c(G(\mathcal{I})) = c_d(G(\mathcal{I})) = n + 1$. Como $c_d(G(\mathcal{I})) \leq c(G(\mathcal{I}))$, então se fizermos com que $G(\mathcal{I})$ possua $n + 1$ ciclos definitivos temos, garantidamente, que $c(G(\mathcal{I})) = c_d(G(\mathcal{I})) =$

$n+1$. Tanto o evento de transposição como o de move podem aumentar o número de ciclos definitivos, no máximo, em duas unidades. Logo, são necessárias pelo menos $\frac{n+1-c_d(G(\mathcal{I}))}{2}$ operações de transposição ou move para atingir o genoma alvo, e o teorema segue. □

Teorema 5.2.11. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. Temos que: $dfi_{\mathbf{R}}(\mathcal{I}) \geq n + 1 - c_d(G(\mathcal{I}))$.*

Demonstração. A prova é similar à apresentada no Teorema 5.2.10 considerando que o evento de reversão pode aumentar o número de ciclos definitivos, no máximo, em uma unidade. □

Teorema 5.2.12. *Seja $\mathcal{I}$ uma instância intergênica flexível com sinais. Temos que:*

$$\begin{aligned} dfi_{\mathbf{RI}}(\mathcal{I}) &\geq n + 1 - c_e(G(\mathcal{I})), \\ dfi_{\mathbf{RTI}}(\mathcal{I}) &\geq \tfrac{n+1-c_e(G(\mathcal{I}))}{2} \text{ e} \\ dfi_{\mathbf{RTMI}}(\mathcal{I}) &\geq \tfrac{n+1-c_e(G(\mathcal{I}))}{2}. \end{aligned}$$

Demonstração. Pela Observação 2.6.4, temos que, para atingir o genoma alvo, é necessário que $c(G(\mathcal{I})) = c_e(G(\mathcal{I})) = n + 1$. Como $c_e(G(\mathcal{I})) \leq c(G(\mathcal{I}))$, então se fizermos com que $G(\mathcal{I})$ possua $n+1$ ciclos estáveis temos, garantidamente, que $c(G(\mathcal{I})) = c_e(G(\mathcal{I})) = n+1$. Tanto o evento de reversão como o de indel podem aumentar o número de ciclos estáveis, no máximo, em uma unidade. Os eventos de transposição e move podem aumentar o número de ciclos estáveis, no máximo, em duas unidade. Logo, são necessárias pelo menos $n + 1 - c_e(G(\mathcal{I}))$ operações de reversão ou indel para atingir o genoma alvo no problema $\mathbf{Sb_{FI}RI}$. Para os problemas $\mathbf{Sb_{FI}RTI}$ e $\mathbf{Sb_{FI}RTMI}$ são necessárias pelo menos $\frac{n+1-c_e(G(\mathcal{I}))}{2}$ operações para atingir o genoma alvo, e o teorema segue. □

Teorema 5.2.13. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. Temos que: $dfi_{\mathbf{RM}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I}))+c_d(G(\mathcal{I}))}{2}$.*

Demonstração. Pela Observação 2.6.5, temos que, para atingir o genoma alvo, é necessário que $c(G(\mathcal{I})) = c_d(G(\mathcal{I})) = n+1$. Logo, temos que aumentar a quantidade de ciclos e ciclos definitivos em $n + 1 - c(G(\mathcal{I}))$ e $n + 1 - c_d(G(\mathcal{I}))$ unidades, respectivamente. Totalizando a quantidade de ciclos e ciclos definitivos que precisam ser criados temos o seguinte valor: $2(n + 1) - (c(G(\mathcal{I})) + c_b(G(\mathcal{I})))$. Considerando os eventos de reversão e move, temos que para qualquer evento $\gamma \in \{\rho, \mu\}$ é verdade que $\Delta c(G(\mathcal{I}), S = (\gamma)) + \Delta c_d(G(\mathcal{I}), S = (\gamma)) \leq 2$. Dessa forma, são necessárias pelo menos $\frac{2(n+1)-(c(G(\mathcal{I}))+c_d(G(\mathcal{I})))}{2} = n + 1 - \frac{c(G(\mathcal{I}))+c_d(G(\mathcal{I}))}{2}$ operações de reversão ou move para atingir o genoma alvo, e o teorema segue. □

Teorema 5.2.14. *Seja $\mathcal{I}$ uma instância intergênica flexível com sinais. Temos que: $dfi_{\mathbf{RMI}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I}))+c_e(G(\mathcal{I}))}{2}$.*

Demonstração. A prova é similar à apresentada no Teorema 5.2.13 considerando que, para qualquer evento $\gamma \in \{\rho, \mu\,\delta\}$, é verdade que $\Delta c(G(\mathcal{I}), S = (\gamma)) + \Delta c_e(G(\mathcal{I}), S = (\gamma)) \leq 2$. □

Teorema 5.2.15. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. Temos que:*

$$\begin{aligned} dfi_{\mathbf{RT}}(\mathcal{I}) &\geq \tfrac{n+1-c_d(G(\mathcal{I}))}{2} \text{ e} \\ dfi_{\mathbf{RTM}}(\mathcal{I}) &\geq \tfrac{n+1-c_d(G(\mathcal{I}))}{2}. \end{aligned}$$

Demonstração. A prova é similar à apresentada no Teorema 5.2.10 incluindo a consideração de que o evento de reversão pode aumentar o número de ciclos definitivos, no máximo, em uma unidade. □

5.3 Algoritmos de Aproximação

Nesta seção, apresentaremos algoritmos de aproximação para as variações com e sem sinais dos problemas investigados neste capítulo. Inicialmente, apresentaremos algumas funções de redução que criam uma instância intergênica rígida a partir de uma instância intergênica flexível. Os algoritmos de aproximação apresentados nesta seção usam, como sub-rotinas, os algoritmos para instâncias com regiões rígidas apresentados no Capítulo 4.

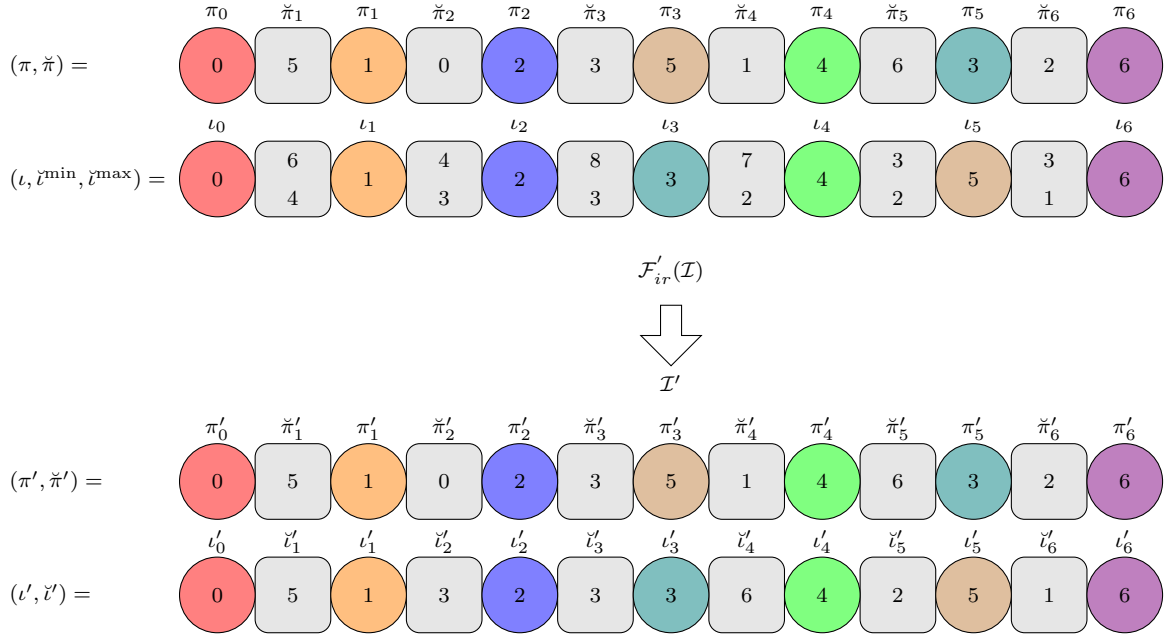
Dada uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, a função $\mathcal{F}'_{ir}$ cria uma instância intergênica rígida sem sinais $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ da seguinte forma:

- $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$
- $\iota' = \iota$
- Inicialmente, atribua em $\breve{\iota}'_i$ o valor $\breve{\iota}_i^{\min}$, para $i \in [1..(n+1)]$. Em seguida, para cada região intergênica estável $\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})$, atribua o valor $\breve{\pi}_i$ em $\breve{\iota}'_k$, onde $k = \max(\pi_{i-1}, \pi_i)$.

Denotamos por $\mathcal{F}'_{ir}(\mathcal{I})$ a instância intergênica rígida sem sinais criada pela função $\mathcal{F}'_{ir}$ a partir de uma instância intergênica flexível sem sinais $\mathcal{I}$. Perceba que a função $\mathcal{F}'_{ir}$ apenas define valores estritos para os tamanhos das regiões intergênicas no genoma alvo $\breve{\iota}'$ da instância intergênica rígida $\mathcal{I}'$, uma vez que $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$ e $\iota' = \iota$. Além disso, a função $\mathcal{F}'_{ir}$ pode ser executada em tempo linear.

O Exemplo 5.3.1 apresenta uma instância intergênica rígida sem sinais $\mathcal{I}' = ((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 0, 3, 1, 6, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (5, 3, 3, 6, 2, 1))$ criada pela função $\mathcal{F}'_{ir}$ a partir de uma instância intergênica flexível sem sinais $\mathcal{I} = ((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 0, 3, 1, 6, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 3, 3, 2, 2, 1), (6, 4, 8, 7, 3, 3)))$. Perceba que no exemplo apenas as regiões intergênicas $\breve{\pi}_1$ e $\breve{\pi}_5$ são estáveis. Por esse motivo, temos que $\breve{\iota}'_1 = 5$ e $\breve{\iota}'_4 = 6$.

Exemplo 5.3.1.



Lema 5.3.1. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida sem sinais tal que $\mathcal{I}' = \mathcal{F}'_{ir}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível sem sinais. Vale que $ir_i(\mathcal{I}) = ib_1(\mathcal{I}')$.*

Demonstração. Pela construção da função $\mathcal{F}'_{ir}$ temos que cada região intergênica estável em $\mathcal{I}$ é mapeada em uma adjacência intergênica em $\mathcal{I}'$. Além disso, cada região intergênica instável acaba tornando-se um breakpoint tipo um em $\mathcal{I}'$, seja porque os elementos antes e depois da região intergênica não são adjacentes no genoma alvo ou porque o tamanho da região intergênica é menor que o mínimo ou maior que o máximo permitido no genoma alvo. Logo, $ir_i(\mathcal{I}) = ib_1(\mathcal{I}')$ e o lema segue. □

Lema 5.3.2. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida sem sinais tal que $\mathcal{I}' = \mathcal{F}'_{ir}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível sem sinais, e seja S' uma sequência de eventos de rearranjo tal que $(\pi', \breve{\pi}') \cdot S' = (\iota', \breve{\iota}')$. Então S' é uma sequência que faz com que o genoma alvo em $\mathcal{I}$ seja atingido.*

Demonstração. Diretamente pela construção da função $\mathcal{F}'_{ir}$. Note que $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$, $\iota' = \iota$ e $\forall i \in \{1, 2, \ldots, (n+1)\} : \breve{\iota}_i^{\min} \leq \breve{\iota}'_i \leq \breve{\iota}_i^{\max}$. □

Agora considere a seguinte função de redução com base em um modelo de rearranjo composto exclusivamente por eventos de rearranjo conservativos. Dada uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, a função $\mathcal{F}''_{ir}$ cria uma instância intergênica rígida balanceada sem sinais $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ da seguinte forma:

- $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$
- $\iota' = \iota$
- Os valores de $\breve{\iota}'$ são atribuídos de acordo com o cenário da instância $\mathcal{I}$:
  - No cenário fonte, inicialmente atribua em $\breve{\iota}'_i$ o valor $\breve{\iota}_i^{\min}$, para $i \in [1..(n+1)]$. Em seguida, para cada região intergênica definitiva $\breve{\pi}_i \in \mathcal{S}_d(\mathcal{I})$, atribua o valor $\breve{\pi}_i$ em $\breve{\iota}'_k$, onde $k = \max(\pi_{i-1}, \pi_i)$. Seja α o seguinte valor:

    $$\alpha = \sum_{\breve{\pi}_i \in \mathcal{S}_a(\mathcal{I})} gap_{\min}(\breve{\pi}_i) - \sum_{\breve{\iota}_i^{\min} \in \breve{\iota}^{\min}} \breve{\iota}_i^{\min} + \sum_{\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})} (\breve{\pi}_i - gap_{\min}(\breve{\pi}_i)) + \sum_{\breve{\pi}_i \in \mathcal{S}_{i_1}(\mathcal{I})} \breve{\pi}_i.$$

    Note que, neste cenário, o valor de α é a diferença entre o excedente de nucleotídeos das regiões intergênicas auxiliares e o número mínimo de nucleotídeos que as regiões intergênicas instáveis precisam receber para tornarem-se estáveis. Perceba que esse valor deve ser menor que $gap_{\min}(\breve{\pi}_i)$, para todo $\breve{\pi}_i \in \mathcal{S}_a$, pois, caso contrário, o tamanho do conjunto $\mathcal{S}_a$ não é mínimo. Considerando que as regiões intergênicas auxiliares estão ordenadas de maneira decrescente pelo valor de $gap_{\min}$ e seja $\breve{\pi}_i$ a última região intergênica auxiliar, atribua em $\breve{\iota}'_k$, com $k = \max(\pi_{i-1}, \pi_i)$, o valor $\breve{\iota}_k^{\min} + \alpha$. Note que $\breve{\iota}_k^{\min} \leq \breve{\iota}'_k \leq \breve{\iota}_k^{\max}$, pois, por definição, $\breve{\pi}_i$ também é uma região intergênica estável, foi a última região intergênica a ser adicionada no conjunto de regiões intergênicas auxiliares e, obrigatoriamente, temos que $gap_{\min}(\breve{\pi}_i) \geq \alpha$.
  - No cenário sorvedouro, inicialmente atribua em $\breve{\iota}'_i$ o valor $\breve{\iota}_i^{\max}$, para $i \in [1..(n+1)]$. Em seguida, para cada região intergênica definitiva $\breve{\pi}_i \in \mathcal{S}_d(\mathcal{I})$, atribua o valor $\breve{\pi}_i$ em $\breve{\iota}'_k$, onde $k = \max(\pi_{i-1}, \pi_i)$. Seja α o seguinte valor:

    $$\alpha = \sum_{\breve{\pi}_i \in \mathcal{S}_a(\mathcal{I})} gap_{\max}(\breve{\pi}_i) - \sum_{\breve{\pi}_i \in \mathcal{S}_i(\mathcal{I})} \breve{\pi}_i + \sum_{\breve{\iota}_i^{\max} \in \breve{\iota}^{\max}} \breve{\iota}_i^{\max} - \sum_{\breve{\pi}_i \in \mathcal{S}_e(\mathcal{I})} (\breve{\pi}_i + gap_{\max}(\breve{\pi}_i)).$$

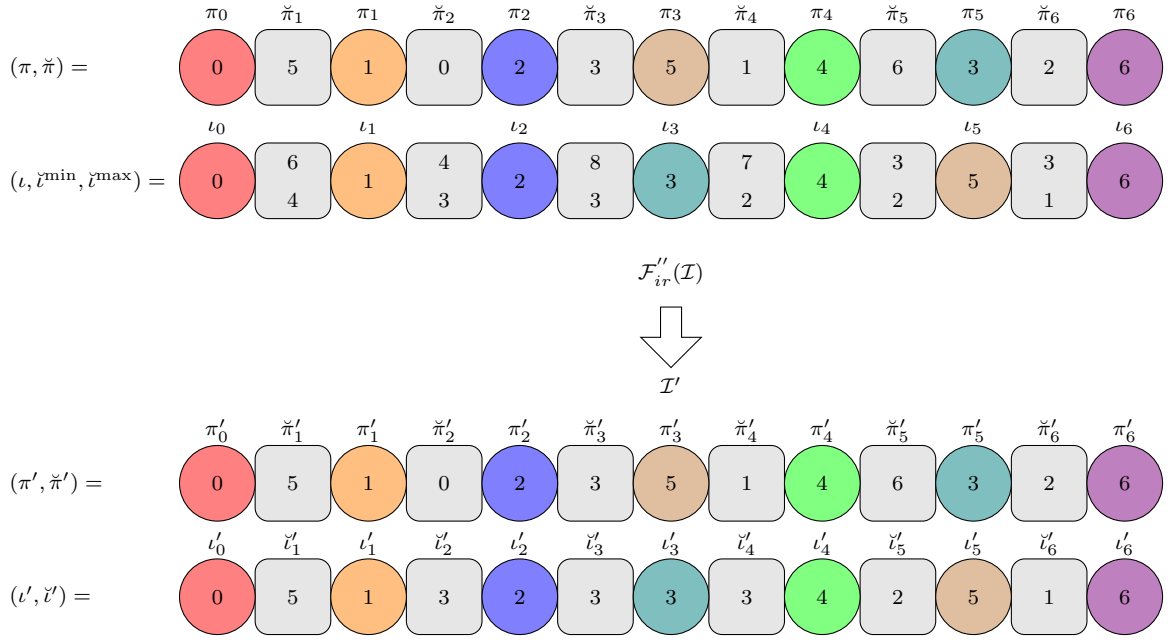
    Neste cenário, o valor de α representa o total de nucleotídeos que podem ser transferidos para as regiões intergênicas auxiliares, sem torná-las instáveis, menos a quantidade excedente total de nucleotídeos nas regiões intergênicas instáveis. Perceba que o valor de α deve ser menor que $gap_{\max}(\breve{\pi}_i)$, para todo $\breve{\pi}_i \in \mathcal{S}_a$, pois, caso contrário, o tamanho do conjunto $\mathcal{S}_a$ não é mínimo. Considerando que as regiões intergênicas auxiliares estão ordenadas de maneira decrescente pelo valor de $gap_{\max}$ e seja $\breve{\pi}_i$ a última região intergênica auxiliar, atribua em $\breve{\iota}'_k$, com $k = \max(\pi_{i-1}, \pi_i)$, o valor $\breve{\iota}_k^{\max} - \alpha$. Note que $\breve{\iota}_k^{\min} \leq \breve{\iota}'_k \leq \breve{\iota}_k^{\max}$, pois, por definição, $\breve{\pi}_i$ também é uma região intergênica estável, foi a última região intergênica a ser adicionada no conjunto de regiões intergênicas auxiliares e, obrigatoriamente, temos que $gap_{\max}(\breve{\pi}_i) \geq \alpha$.
  - No cenário de equilíbrio, temos que o conjunto de regiões intergênicas auxiliares é vazio e o total de nucleotídeos nas regiões intergênicas instáveis é suficiente para torná-las estáveis. Nesse caso, para cada região intergênica definitiva $\breve{\pi}_i \in \mathcal{S}_d(\mathcal{I})$, atribua o valor $\breve{\pi}_i$ em $\breve{\iota}'_k$, onde $k = \max(\pi_{i-1}, \pi_i)$. Para as regiões

intergênicas instáveis, sempre é possível encontrar uma lista de números inteiros não negativos que atenda aos tamanhos mínimo e máximo permitidos em cada região intergênica do genoma alvo e que considere o total de nucleotídeos disponíveis nas regiões intergênicas instáveis.

Por construção da função $\mathcal{F}_{ir}^{''}$, temos que cada região intergênica definitiva de $\mathcal{I}$ é mapeada em uma adjacência intergênica em $\mathcal{I}'$ e cada região intergênica instável e auxiliar em $\mathcal{I}$ é mapeada em um breakpoint tipo um em $\mathcal{I}'$. Além disso, pela forma como os tamanhos das regiões intergênicas são atribuídos em $\breve{\iota}'$, temos a garantia de que a instância intergênica rígida $\mathcal{I}'$ resultante é balanceada. Denotamos por $\mathcal{F}_{ir}^{''}(\mathcal{I})$ a instância intergênica rígida balanceada sem sinais criada pela função $\mathcal{F}_{ir}^{''}$ a partir de uma instância intergênica flexível balanceada sem sinais $\mathcal{I}$. A função $\mathcal{F}_{ir}^{''}$ pode ser executada em tempo $\mathcal{O}(n \log n)$, uma vez que, no pior caso, é necessário ordenar as regiões intergênicas estáveis para definir o conjunto de regiões intergênicas auxiliares.

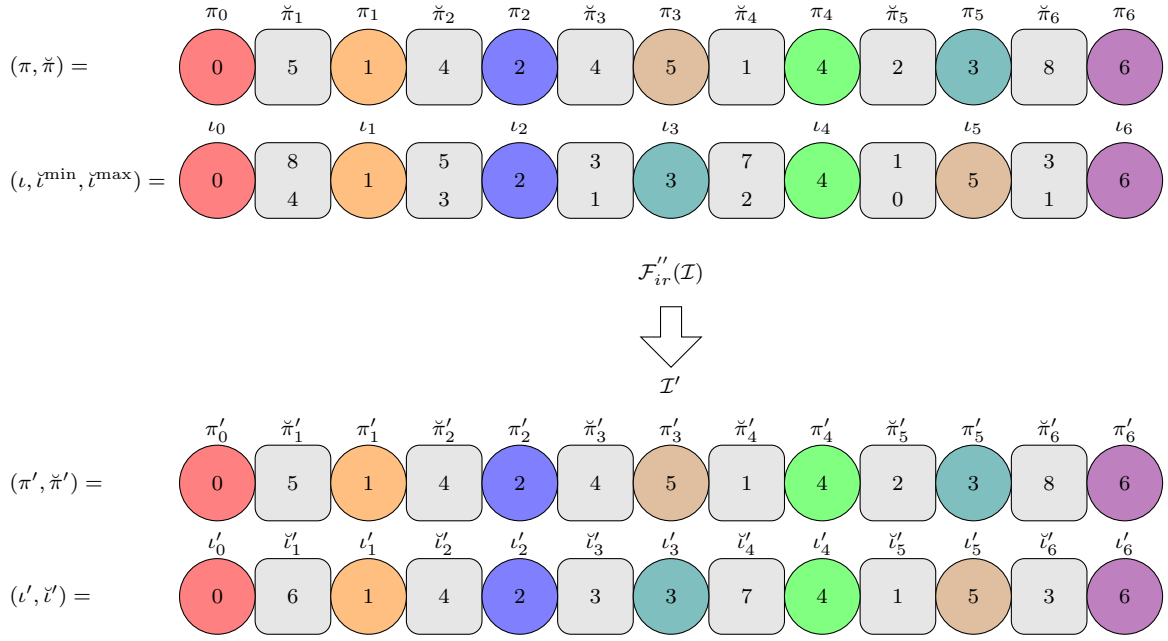
O Exemplo 5.3.2 mostra uma instância intergênica rígida balanceada sem sinais $\mathcal{I}' = ((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 0, 3, 1, 6, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (5, 3, 3, 3, 2, 1))$ criada pela função $\mathcal{F}''_{ir}$ a partir de uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 0, 3, 1, 6, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 3, 3, 2, 2, 1), (6, 4, 8, 7, 3, 3)))$. Note que a instância $\mathcal{I}$ pertence ao cenário fonte com quatro regiões intergênicas instáveis ($ir_i(\mathcal{I}) = 4$ e $\mathcal{S}_i = \{\breve{\pi}_2, \breve{\pi}_3, \breve{\pi}_4, \breve{\pi}_6\}$) e duas regiões intergênicas estáveis ($\mathcal{S}_e = \{\breve{\pi}_1, \breve{\pi}_5\}$). No exemplo, temos apenas uma região intergênica auxiliar ($ir_a(\mathcal{I}) = 1$ e $\mathcal{S}_a = \{\breve{\pi}_5\}$). Note que $gap_{\min}(\breve{\pi}_1) = 1$ e $gap_{\min}(\breve{\pi}_5) = 4$. Logo, $ir_d(\mathcal{I}) = 1$ e $\mathcal{S}_d = \{\breve{\pi}_1\}$. Além disso, perceba que $\alpha = 1$. Como $\breve{\pi}_5$ é a única (e a última) região intergênica auxiliar, temos que $\breve{\iota}'_4 = \breve{\iota}_4^{\min} + \alpha = 2 + 1 = 3$.

Exemplo 5.3.2.



O Exemplo 5.3.3 mostra uma instância intergênica rígida balanceada sem sinais $\mathcal{I}' = ((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 4, 4, 1, 2, 8)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (6, 4, 3, 7, 1, 3))$ criada pela função $\mathcal{F}''_{ir}$ a partir de uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 4, 4, 1, 2, 8)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 3, 1, 2, 0, 1), (8, 5, 3, 7, 1, 3)))$ que pertence ao cenário sorvedouro. Note que a instância $\mathcal{I}$ possui duas regiões intergênicas instáveis ($ir_i(\mathcal{I}) = 2$ e $\mathcal{S}_i = \{\breve{\pi}_3, \breve{\pi}_6\}$) e quatro regiões intergênicas estáveis ($\mathcal{S}_e = \{\breve{\pi}_1, \breve{\pi}_2, \breve{\pi}_4, \breve{\pi}_5\}$). No exemplo, temos duas regiões intergênicas auxiliares ($ir_a(\mathcal{I}) = 2$ e $\mathcal{S}_a = \{\breve{\pi}_1, \breve{\pi}_5\}$). Note que $gap_{\max}(\breve{\pi}_1) = 3$, $gap_{\max}(\breve{\pi}_2) = 1$, $gap_{\max}(\breve{\pi}_4) = 0$ e $gap_{\max}(\breve{\pi}_5) = 5$. Logo, $ir_d(\mathcal{I}) = 2$ e $\mathcal{S}_d = \{\breve{\pi}_2, \breve{\pi}_4\}$. Além disso, perceba que $\alpha = 2$. Como $\breve{\pi}_1$ é a última região intergênica auxiliar considerando uma ordenação decrescente pelo valor de $gap_{\max}$, temos que $\breve{\iota}'_1 = \breve{\iota}_1^{\max} - \alpha = 8 - 2 = 6$.

Exemplo 5.3.3.



Lema 5.3.3. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada sem sinais tal que $\mathcal{I}' = \mathcal{F}''_{ir}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível balanceada sem sinais. Vale que $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = ib_1(\mathcal{I}')$.*

Demonstração. Pela construção da função $\mathcal{F}''_1$ temos que cada região intergênica definitiva tipo $\mathcal{I}$ é mapeada em uma adjacência intergênica em $\mathcal{I}'$. Além disso, cada região intergênica instável e auxiliar acaba tornando-se um breakpoint tipo um em $\mathcal{I}'$, seja porque os elementos antes e depois da região intergênica não são adjacentes no genoma alvo ou porque o tamanho da região intergênica é menor que o mínimo ou maior que o máximo permitido no genoma alvo. Logo, $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = ib_1(\mathcal{I}')$ e o lema segue. □

Lema 5.3.4. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada sem sinais tal que $\mathcal{I}' = \mathcal{F}''_{ir}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível balanceada sem sinais, e seja S' uma sequência de eventos de rearranjo tal que $(\pi', \breve{\pi}') \cdot S' = (\iota', \breve{\iota}')$. Então S' é uma sequência que faz com que o genoma alvo em $\mathcal{I}$ seja atingido.*

Demonstração. Diretamente pela construção da função $\mathcal{F}_{ir}^{''}$. Note que $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$, $\iota' = \iota$ e $\forall i \in \{1, 2, \ldots, (n+1)\} : \breve{\iota}_i^{\min} \leq \breve{\iota}_i' \leq \breve{\iota}_i^{\max}$. □

Agora definiremos, com base no grafo de ciclos ponderado flexível, duas funções de redução que criam uma instância intergênica rígida a partir de uma instância intergênica flexível. Note que tanto o grafo de ciclos ponderado rígido como o grafo de ciclos ponderado flexível são extensões do grafo de ciclos clássico que possuem as mesmas regras para a construção dos conjuntos de vértices e arestas. A principal diferença na estrutura desses dois grafos é que nas arestas cinzas do grafo de ciclos ponderado rígido temos apenas um peso associado, que representa o tamanho estrito da região intergênica no genoma alvo que está entre os dois vértices conectados pela aresta. Já no grafo de ciclos ponderado flexível, cada aresta cinza possui dois pesos associados, sendo eles: os tamanhos mínimo e máximo permitidos no genoma alvo para a região intergênica entre os dois vértices conectados pela aresta.

Observe que, para criarmos um grafo de ciclos ponderado rígido a partir de um grafo de ciclos ponderado flexível, basta desenvolver uma forma de associar um único peso para cada aresta cinza. Consequentemente, com o grafo de ciclos ponderado rígido construído, podemos obter os valores da instância intergênica rígida que deu origem ao grafo.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, a função $\mathcal{F}_c^{'}$ cria uma instância intergênica rígida $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ da seguinte forma:

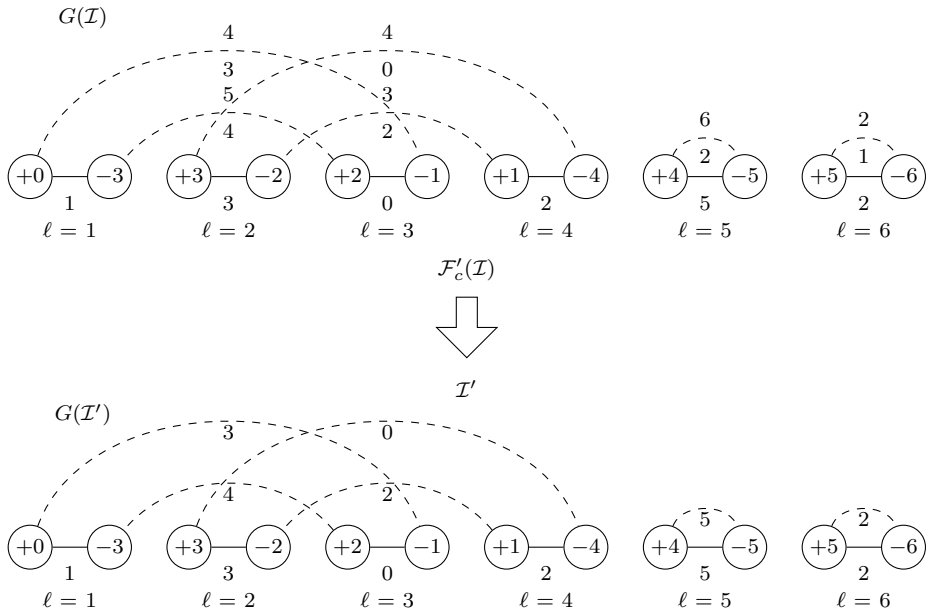
- Para cada ciclo estável $C = (c^1, \ldots, c^k)$ em $G(\mathcal{I})$, encontre uma lista de números inteiros não negativos $L = (l_1, \ldots, l_k)$, tal que $\sum_{i=1}^{k} l_i = W_p(C)$ e $\forall e_i' \in E_c(C)$: $w_c^{\min}(e_i') \leq l_i \leq w_c^{\max}(e_i')$. Note que, pela definição de um ciclo estável C, temos que $W_c^{\min}(C) \leq W_p(C) \leq W_c^{\max}(C)$, então sempre é possível encontrar uma lista L com tais características. Por fim, para cada aresta cinza e_i' de C no grafo de ciclos ponderado flexível, atribua o peso l_i na aresta cinza e_i' do grafo de ciclos ponderado rígido. Por construção, temos que cada ciclo estável em $G(\mathcal{I})$ torna-se um ciclo balanceado em $G(\mathcal{I}')$.

- Para cada ciclo instável C em $G(\mathcal{I})$, realize a seguinte atribuição de peso: para cada aresta cinza e_i' de C no grafo de ciclos ponderado flexível, atribua o peso $w_c^{\min}(e_i')$ na aresta cinza e_i' do grafo de ciclos ponderado rígido. Pela definição de um ciclo instável C, temos que $W_p(C) < W_c^{\min}(C)$ ou $W_p(C) > W_c^{\max}(C)$. Por construção, temos que cada ciclo instável em $G(\mathcal{I})$ torna-se um ciclo desbalanceado em $G(\mathcal{I}')$.

Note que os conjuntos de vértices e arestas em $G(\mathcal{I})$ e $G(\mathcal{I}')$ são os mesmos. Logo, $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$. Denotamos por $\mathcal{F}_c^{'}(\mathcal{I})$ a instância intergênica rígida criada pela função $\mathcal{F}_c^{'}$ a partir de uma instância intergênica flexível $\mathcal{I}$. Perceba que a função $\mathcal{F}_c^{'}$ apenas define o peso de cada aresta cinza no grafo de ciclos ponderado rígido. Além disso, a função $\mathcal{F}_c^{'}$ pode ser executada em tempo linear.

O Exemplo 5.3.4 mostra uma instância intergênica rígida com sinais $\mathcal{I}' = ((+0\ +3\ +2\ +1\ +4\ +5\ +6), (1, 3, 0, 2, 5, 2)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (3, 2, 4, 0, 5, 2))$ criada pela função $\mathcal{F}_c^{'}$ a partir de uma instância intergênica flexível com sinais $\mathcal{I} = (((+0\ +3\ +2\ +1\ +4\ +5\ +6), (1, 3, 0, 2, 5, 2)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (3, 2, 4, 0, 2, 1), (4, 3, 5, 4, 6, 2)))$. Note que $\mathcal{S}_i(G(\mathcal{I})) = \{C_1 = (3, 1), C_2 = (4, 2)\}$ e $\mathcal{S}_e(G(\mathcal{I})) = \{C_3 = (5), C_4 = (6)\}$. Os ciclos estáveis C_3 e C_4 são mapeados em ciclos balanceados em $\mathcal{I}'$, enquanto os ciclos instáveis C_1 e C_2 são mapeados em ciclos desbalanceados.

Exemplo 5.3.4.



Lema 5.3.5. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida tal que $\mathcal{I}' = \mathcal{F}_c^{'}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível. Vale que $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$ e $c_e(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$.*

Demonstração. Diretamente pela construção da função $\mathcal{F}_c^{'}$. □

Lema 5.3.6. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida tal que $\mathcal{I}' = \mathcal{F}_c^{'}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível, e seja S' uma sequência de eventos de rearranjo tal que $(\pi', \breve{\pi}') \cdot S' = (\iota', \breve{\iota}')$. Então S' é uma sequência que faz com que o genoma alvo em $\mathcal{I}$ seja atingido.*

Demonstração. Note que, para uma aresta cinza e'_i em $G(\mathcal{I}')$, temos um peso $w_c(e'_i)$ associado. Para a mesma aresta cinza e'_i, mas em $G(\mathcal{I})$, temos os pesos mínimo $w_c^{\min}(e'_i)$ e máximo $w_c^{\max}(e'_i)$ associados. Pela forma como os pesos foram atribuídos em cada aresta cinza e'_i de $G(\mathcal{I}')$, temos a garantia de que $w_c^{\min}(e'_i) \leq w_c(e'_i) \leq w_c^{\max}(e'_i)$. Isso implica que $\forall i \in \{1, 2, \ldots, (n+1)\} : \breve{\iota}_i^{\min} \leq \breve{\iota}'_i \leq \breve{\iota}_i^{\max}$. Além disso, o peso em cada aresta preta de $G(\mathcal{I}')$ é igual ao peso da mesma aresta preta em $G(\mathcal{I})$, com os conjuntos de vértices e arestas sendo os mesmos em $G(\mathcal{I})$ e $G(\mathcal{I}')$. Logo, $(\pi', \breve{\pi}') = (\pi, \breve{\pi})$, $\iota' = \iota$ e o lema segue. □

Agora considere a seguinte função de redução baseada em um modelo composto exclusivamente por eventos de rearranjo conservativos. Dada uma instância intergênica flexível

balanceada $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, a função $\mathcal{F}_c''$ cria uma instância intergênica rígida balanceada $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ da seguinte forma:

- Para cada ciclo definitivo $C = (c^1, \ldots, c^k)$ em $G(\mathcal{I})$, encontre uma lista de números inteiros não negativos $L = (l_1, \ldots, l_k)$, tal que $\sum_{i=1}^{k} l_i = W_p(C)$ e $\forall e'_i \in E_c(C)$: $w_c^{\min}(e'_i) \leq l_i \leq w_c^{\max}(e'_i)$. Note que um ciclo definitivo C também é um ciclo estável e temos que $W_c^{\min}(C) \leq W_p(C) \leq W_c^{\max}(C)$, então sempre é possível encontrar uma lista L com tais características. Por fim, para cada aresta cinza e'_i de C no grafo de ciclos ponderado flexível, atribua o peso l_i na aresta cinza e'_i do grafo de ciclos ponderado rígido. Por construção, temos que cada ciclo definitivo em $G(\mathcal{I})$ torna-se um ciclo balanceado em $G(\mathcal{I}')$.

- O mapeamento dos ciclos restantes em $G(\mathcal{I})$ depende do cenário em que a instância $\mathcal{I}$ se encaixa:

  - Se for o cenário fonte, então para cada ciclo instável $C \in G(\mathcal{I})$, realize a seguinte atribuição de peso: para cada aresta cinza e'_i de C, atribua o peso $w_c^{\min}(e'_i)$ na aresta cinza e'_i do grafo de ciclos ponderado rígido. Pela definição de um ciclo instável C, temos que $W_p(C) < W_c^{\min}(C)$ ou $W_p(C) > W_c^{\max}(C)$. Por construção, temos que cada ciclo instável em $G(\mathcal{I})$ torna-se um ciclo desbalanceado em $G(\mathcal{I}')$. Considerando que os ciclos auxiliares estão ordenados pelo valor de $gap_{\min}$ de maneira decrescente e que $c_a(G(\mathcal{I})) = x$, então pra os primeiros $x - 1$ ciclos auxiliares realize a seguinte atribuição de peso: para cada aresta cinza e'_i do ciclo, atribua o peso $w_c^{\min}(e'_i)$ na aresta cinza e'_i do grafo de ciclos ponderado rígido; para o último ciclo auxiliar $C = (c^1, \ldots, c^k)$, encontre uma lista de números inteiros não negativos $L = (l_1, \ldots, l_k)$, tal que $\forall e'_i \in E_c(C) : w_c^{\min}(e'_i) \leq l_i \leq w_c^{\max}(e'_i)$ e $\sum_{i=1}^{k} l_i = W_c^{\min}(C) + \alpha$, onde

$$\alpha = \sum_{C \in \mathcal{S}_a(G(\mathcal{I}))} gap_{\min}(C) + \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} gap_{\min}(C).$$

    Perceba que, neste cenário, o valor de α é a diferença entre o excedente de peso nos ciclos auxiliares (em relação ao peso total mínimo) e o peso mínimo que os ciclos instáveis precisam receber para tornarem-se estáveis. Note que C também é um ciclo estável e $gap_{\min}(C) \geq \alpha$, então sempre é possível encontrar uma lista L com tais características. Por construção, temos que cada ciclo auxiliar em $G(\mathcal{I})$ também torna-se um ciclo desbalanceado em $G(\mathcal{I}')$.

  - Se for o cenário sorvedouro, então para cada ciclo instável $C \in G(\mathcal{I})$, realize a seguinte atribuição de peso: para cada aresta cinza e'_i de C, atribua o peso $w_c^{\max}(e'_i)$ na aresta cinza e'_i do grafo de ciclos ponderado rígido. Pela definição de um ciclo instável C, temos que $W_p(C) < W_c^{\min}(C)$ ou $W_p(C) > W_c^{\max}(C)$. Por construção, temos que cada ciclo instável em $G(\mathcal{I})$ torna-se um ciclo desbalanceado em $G(\mathcal{I}')$. Considerando que os ciclos auxiliares estão ordenados pelo valor de $gap_{\max}$ de maneira decrescente e que $c_a(G(\mathcal{I})) = x$, então pra os primeiros $x - 1$ ciclos auxiliares realize a seguinte atribuição de peso: para

cada aresta cinza e'_i do ciclo no grafo de ciclos ponderado flexível, atribua o peso $w_c^{\max}(e'_i)$ na aresta cinza e'_i do grafo de ciclos ponderado rígido; para o último ciclo auxiliar $C = (c^1, \ldots, c^k)$, encontre uma lista de números inteiros não negativos $L = (l_1, \ldots, l_k)$, tal que $\forall e'_i \in E_c(C) : w_c^{\min}(e'_i) \leq l_i \leq w_c^{\max}(e'_i)$ e $\sum_{i=1}^{k} l_i = W_c^{\max}(C) - \alpha$, onde

$$\alpha = \sum_{C \in \mathcal{S}_a(G(\mathcal{I}))} gap_{\max}(C) + \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} gap_{\max}(C).$$

Neste cenário, o valor de α representa o peso total que pode ser transferido para os ciclos auxiliares, sem torná-las instáveis, menos o peso excedente dos ciclos instáveis (em relação ao peso total máximo). Note que C também é um ciclo estável e $gap_{\max}(C) \geq \alpha$, então sempre é possível encontrar uma lista L com tais características. Por construção, temos que cada ciclo auxiliar em $G(\mathcal{I})$ também torna-se um ciclo desbalanceado em $G(\mathcal{I}')$.

– Se for o cenário de equilíbrio, então o conjunto de ciclos auxiliares é vazio e a soma do peso total dos ciclos instáveis é suficiente para torná-los estáveis, ou seja,

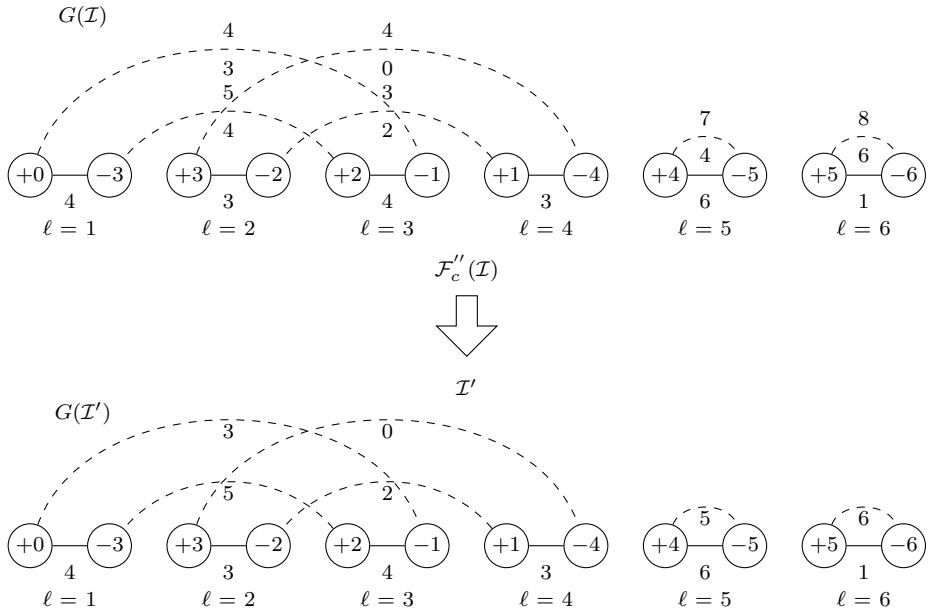
$$\sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\min}(C) \leq \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C) \leq \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\max}(C).$$

Para este caso, basta atribuir pesos nas arestas cinzas que pertencem aos ciclos instáveis em $G(\mathcal{I})$ de maneira que o peso atribuído em cada aresta cinza no grafo de ciclos ponderado rígido não viole os pesos mínimo e máximo permitidos para a mesma aresta em $G(\mathcal{I})$. Além disso, a soma dos pesos atribuídos nas arestas cinzas deve ser igual a $\sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C)$, para garantir que a instância intergênica rígida $\mathcal{I}'$ resultante seja balanceada. Uma possível forma de realizar essa tarefa é: para cada aresta e'_i pertencente a um ciclo instável em $G(\mathcal{I})$, atribua inicialmente o peso $w_c^{\min}(e'_i)$ na aresta cinza e'_i do grafo de ciclos ponderado rígido. Perceba que ainda pode ser necessário distribuir um peso de $X = \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C) - \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\min}(C)$ para essas mesmas arestas, sem violar os pesos mínimo e máximo permitidos para cada uma delas em $G(\mathcal{I})$. Note que isso é sempre possível de ser realizado, pois $\sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\min}(C) \leq \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_p(C) \leq \sum_{C \in \mathcal{S}_i(G(\mathcal{I}))} W_c^{\max}(C)$. Basta percorrer as mesmas arestas cinzas e, quando possível, incrementar o peso atribuído na aresta até que o valor de X seja igual a zero.

Note que, os conjuntos de vértices e arestas em $G(\mathcal{I})$ e $G(\mathcal{I}')$ são os mesmos. Logo, $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$. Denotamos por $\mathcal{F}_c''(\mathcal{I})$ a instância intergênica rígida balanceada criada pela função $\mathcal{F}_c''$ a partir de uma instância intergênica flexível balanceada $\mathcal{I}$. Perceba que a função $\mathcal{F}_c''$ apenas define o peso de cada aresta cinza no grafo de ciclos ponderado rígido. Além disso, a função $\mathcal{F}_c''$ pode ser executada em tempo $\mathcal{O}(n \log n)$, tendo em vista que, no pior caso, os ciclos estáveis precisam ser ordenados para definir o conjunto de ciclos auxiliares.

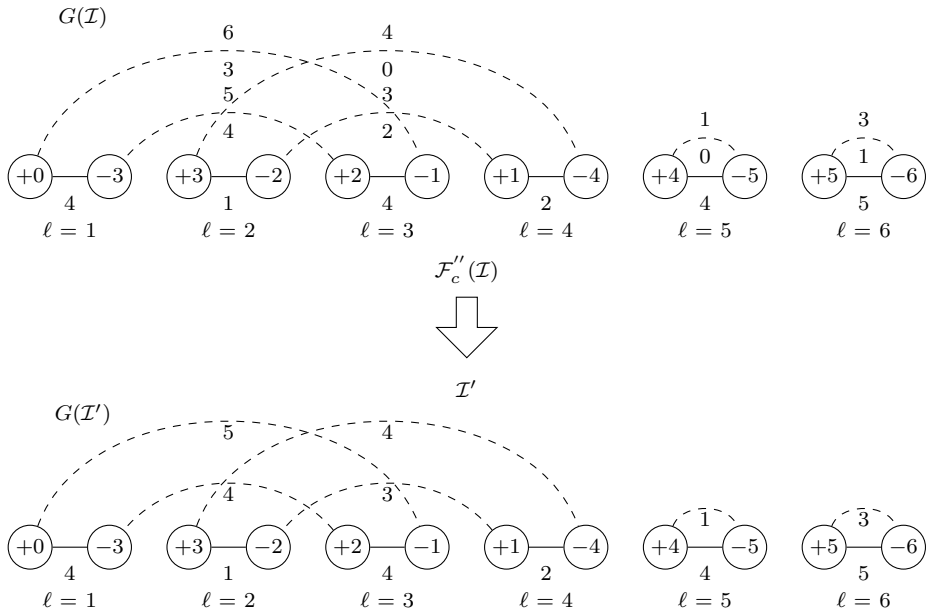
O Exemplo 5.3.5 mostra uma instância intergênica rígida balanceada sem sinais $\mathcal{I}' = ((0\ 3\ 2\ 1\ 4\ 5\ 6), (4, 3, 4, 3, 6, 1)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (3, 2, 5, 0, 5, 6))$ criada pela função $\mathcal{F}_c''$ a partir de uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((0\ 3\ 2\ 1\ 4\ 5\ 6), (4, 3, 4, 3, 6, 1)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (3, 2, 4, 0, 4, 6), (4, 3, 5, 4, 7, 8)))$. Note que a instância $\mathcal{I}$ pertence ao cenário fonte, com $\mathcal{S}_i(G(\mathcal{I})) = \{C_4 = (6)\}$ e $\mathcal{S}_e(G(\mathcal{I})) = \{C_1 = (3, 1), C_2 = (4, 2), C_3 = (5)\}$. Além disso, temos que $\mathcal{S}_a(G(\mathcal{I})) = \{C_2 = (4, 2), C_3 = (5)\}$ e $\mathcal{S}_d(G(\mathcal{I})) = \{C_1 = (3, 1)\}$. Note que os valores de $gap_{\min}(C_1)$, $gap_{\min}(C_2)$ e $gap_{\min}(C_3)$ são 1, 4 e 2, respectivamente. Além disso, temos que $\alpha = gap_{\min}(C_2) + gap_{\min}(C_3) + gap_{\min}(C_4) = 4+2-5 = 1$. Como o ciclo C_3 é o último, considerando uma ordenação decrescente pelo valor de $gap_{\min}$, e $W_c^{\min}(C_3) = 4$, temos que a soma dos pesos das arestas cinzas do ciclo C_3 em $G(\mathcal{I}')$ deve ser igual a $W_c^{\min}(C_3) + \alpha = 4 + 1 = 5$. Entretanto, C_3 é um ciclo trivial. Logo, sua única aresta cinza possui um peso 5 associado.

Exemplo 5.3.5.



O Exemplo 5.3.6 mostra uma instância intergênica rígida balanceada com sinais $\mathcal{I}' = ((+0\ +3\ +2\ +1\ +4\ +5\ +6), (4,1,4,2,4,5)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (5,3,4,4,1,3))$ criada pela função $\mathcal{F}_c^{''}$ a partir de uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = (((+0\ +3\ +2\ +1\ +4\ +5\ +6), (4,3,4,3,6,1)), ((+0\ +1\ +2\ +3\ +4\ +5\ +6), (3,2,4,0,0,1), (6,3,5,4,1,3)))$. Note que a instância $\mathcal{I}$ pertence ao cenário sorvedouro, com os conjuntos $\mathcal{S}_i(G(\mathcal{I})) = \{C_3 = (5), C_4 = (6)\}$ e $\mathcal{S}_e(G(\mathcal{I})) = \{C_1 = (3,1), C_2 = (4,2)\}$. Além disso, temos que $\mathcal{S}_a(G(\mathcal{I})) = \{C_1 = (3,1), C_2 = (4,2)\}$ e $\mathcal{S}_d(G(\mathcal{I})) = \varnothing$. Note que os valores de $gap_{\max}(C_1)$ e $gap_{\max}(C_2)$ são, respectivamente, 3 e 4. Além disso, temos que $\alpha = gap_{\max}(C_1) + gap_{\max}(C_2) + gap_{\max}(C_3) + gap_{\max}(C_4) = 3 + 4 - 3 - 2 = 2$. Como o ciclo C_1 é o último, considerando uma ordenação decrescente pelo valor de $gap_{\max}$, e $W_c^{\max}(C_1) = 11$, temos que a soma dos pesos das arestas cinzas do ciclo C_1 em $G(\mathcal{I}')$ deve ser igual a $W_c^{\max}(C_1) - \alpha = 11 - 2 = 9$. Além disso, o peso associado a cada aresta de C_1 em $G(\mathcal{I}')$ deve atender aos peso mínimo e máximo da mesma aresta em $G(\mathcal{I})$. No exemplo, temos que as arestas cinzas $(+0,-1)$ e $(+2,-3)$ possuem os pesos 5 e 4, respectivamente. Note que $w_c^{\min}((+0,-1)) \leq w_c((+0,-1)) \leq w_c^{\max}((+0,-1))$ e $w_c^{\min}((+2,-3)) \leq w_c((+2,-3)) \leq w_c^{\max}((+2,-3))$.

Exemplo 5.3.6.



Lema 5.3.7. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada tal que $\mathcal{I}' = \mathcal{F}_c^{''}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível balanceada. Vale que $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$ e $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$.*

Demonstração. Diretamente pela construção da função $\mathcal{F}_c^{''}$. □

Lema 5.3.8. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada tal que $\mathcal{I}' = \mathcal{F}_c^{'}(\mathcal{I})$, onde $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ é uma instância intergênica flexível balanceada, e seja S' uma sequência de eventos de rearranjo tal que $(\pi', \breve{\pi}') \cdot S' = (\iota', \breve{\iota}')$. Então S' é uma sequência que faz com que o genoma alvo em $\mathcal{I}$ seja atingido.*

Demonstração. A prova é similar à descrita no Lema 5.3.6. □

5.3.1 Instâncias Intergênicas Flexíveis sem Sinais

Nesta seção, apresentaremos algoritmos de aproximação para a variação sem sinais dos problemas investigados neste capítulo, com base nas funções de redução apresentadas previamente.

Reversão

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_{FI}R}$ (Algoritmo 31).

Algoritmo 31: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}R}$.

Entrada: Uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}''_{ir}(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 4 para a instância $\mathcal{I}'$

3 **retorne** S'

Teorema 5.3.9. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. O Algoritmo 31 é uma 4-aproximação para o problema $\boldsymbol{Sb_{FI}R}$.*

Demonstração. Pelo Lema 5.3.4, temos que a sequência fornecida pelo Algoritmo 4 para a instância intergênica rígida balanceada sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_IR}$ e $\mathbf{Sb_{FI}R}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}R}$. Pelo Lema 4.1.23, temos que $|S'| \leq 2ib_1(\mathcal{I}')$. Entretanto, pelo Lema 5.3.3, temos que $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq 2(ir_i(\mathcal{I}) + ir_a(\mathcal{I}))$. Pelo Teorema 5.2.9, temos o seguinte limitante inferior: $dfi_{\mathbf{R}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I}) + ir_a(\mathcal{I})}{2}$, e o teorema segue. □

Reversão e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_{FI}RI}$ (Algoritmo 32).

Teorema 5.3.10. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. O Algoritmo 32 é uma 4-aproximação para o problema $\boldsymbol{Sb_{FI}RI}$.*

Demonstração. Pelo Lema 5.3.2, temos que a sequência fornecida pelo Algoritmo 5 para a instância intergênica rígida sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_IRI}$ e $\mathbf{Sb_{FI}RI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de

Algoritmo 32: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RI}$.

Entrada: Uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}'_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e indel fornecida pelo Algoritmo 5 para a instância $\mathcal{I}'$
3 **retorne** S'

rearranjo do problema $\mathbf{Sb_{FI}RI}$. Pelo Lema 4.1.27, temos que $|S'| \leq 2ib_1(\mathcal{I}')$. Entretanto, pelo Lema 5.3.1, temos que $ir_i(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq 2ir_i(\mathcal{I})$. Pelo Teorema 5.2.8, temos o seguinte limitante inferior: $dfi_{\mathbf{RI}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})}{2}$, e o teorema segue. □

Reversão e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_{FI}RM}$ (Algoritmo 33).

Algoritmo 33: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RM}$.

Entrada: Uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão e move S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}''_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e move fornecida pelo Algoritmo 6 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.11. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. O Algoritmo 33 é uma 4-aproximação para o problema $\mathbf{Sb_{FI}RM}$.*

Demonstração. Pelo Lema 5.3.4, temos que a sequência fornecida pelo Algoritmo 6 para a instância intergênica rígida balanceada sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_{I}RM}$ e $\mathbf{Sb_{FI}RM}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RM}$. Pelo Lema 4.1.30, temos que $|S'| \leq 2ib_1(\mathcal{I}')$. Entretanto, pelo Lema 5.3.3, temos que $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq 2(ir_i(\mathcal{I}) + ir_a(\mathcal{I}))$. Pelo Teorema 5.2.9, temos o seguinte limitante inferior: $dfi_{\mathbf{RM}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{2}$, e o teorema segue. □

Reversão, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_{FI}RMI}$ (Algoritmo 34).

Algoritmo 34: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RI}$.

Entrada: Uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão, move e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}'_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, move e indel fornecida pelo Algoritmo 7 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.12. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. O Algoritmo 34 é uma 4-aproximação para o problema* $\boldsymbol{Sb_{FI}RMI}$.

Demonstração. Pelo Lema 5.3.2, temos que a sequência fornecida pelo Algoritmo 7 para a instância intergênica rígida sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_I RMI}$ e $\mathbf{Sb_{FI}RMI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RMI}$. Pelo Lema 4.1.32, temos que $|S'| \leq 2ib_1(\mathcal{I}')$. Entretanto, pelo Lema 5.3.1, temos que $ir_i(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq 2ir_i(\mathcal{I})$. Pelo Teorema 5.2.8, temos o seguinte limitante inferior: $dfi_{\mathbf{RMI}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})}{2}$, e o teorema segue. □

Reversão e Transposição

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação sem sinais do problema $\mathbf{Sb_{FI}RT}$ (Algoritmo 35).

Algoritmo 35: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RT}$.

Entrada: Uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão e transposição S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}''_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e transposição fornecida pelo Algoritmo 10 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.13. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. O Algoritmo 35 é uma 4-aproximação para o problema* $\boldsymbol{Sb_{FI}RT}$.

Demonstração. Pelo Lema 5.3.4, temos que a sequência fornecida pelo Algoritmo 10 para a instância intergênica rígida balanceada sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_I RT}$ e $\mathbf{Sb_{FI}RT}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RT}$. Pelo Lema 4.1.50, temos que $|S'| \leq \frac{4ib_1(\mathcal{I}')}{3}$. Entretanto, pelo Lema 5.3.3, temos que $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq \frac{4(ir_i(\mathcal{I}) + ir_a(\mathcal{I}))}{3}$.

Pelo Teorema 5.2.9, temos o seguinte limitante inferior: $dfi_{\mathbf{RT}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{3}$, e o teorema segue. □

Reversão, Transposição e Indel

Nesta seção, apresentaremos um algoritmo de aproximação assintótica para a variação sem sinais do problema $\mathbf{Sb_{FI}RTI}$ (Algoritmo 36).

Algoritmo 36: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RTI}$.

Entrada: Uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão, transposição e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}'_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição e indel fornecida pelo Algoritmo 11 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.14. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. O Algoritmo 36 é uma 4-aproximação assintótica para o problema $\mathbf{Sb_{FI}RTI}$.*

Demonstração. Pelo Lema 5.3.2, temos que a sequência fornecida pelo Algoritmo 11 para a instância intergênica rígida sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_{I}RTI}$ e $\mathbf{Sb_{FI}RTI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RTI}$. Pelo Lema 4.1.53, temos que $|S'| \leq \frac{4ib_1(\mathcal{I}')}{3} + 1$. Entretanto, pelo Lema 5.3.1, temos que $ir_i(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq \frac{4ir_i(\mathcal{I})}{3} + 1$. Pelo Teorema 5.2.8, temos o seguinte limitante inferior: $dfi_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})}{3}$. Com isso, temos que, no pior caso, o fator de aproximação do Algoritmo 36 é de $4dfi_{\mathbf{RTI}}(\mathcal{I}) + 1$, e o teorema segue. □

Reversão, Transposição e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação sem sinais do problema $\mathbf{Sb_{FI}RTM}$ (Algoritmo 37).

Algoritmo 37: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RTM}$.

Entrada: Uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão, transposição e move S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}''_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição e move fornecida pelo Algoritmo 12 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.15. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. O Algoritmo 37 é uma 3-aproximação para o problema* ***Sb$_{FI}$RTM****.*

Demonstração. Pelo Lema 5.3.4, temos que a sequência fornecida pelo Algoritmo 12 para a instância intergênica rígida balanceada sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas **Sb$_{\mathbf{I}}$RTM** e **Sb$_{\mathbf{FI}}$RTM** compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema **Sb$_{\mathbf{FI}}$RTM**. Pelo Lema 4.1.56, temos que $|S'| \leq ib_1(\mathcal{I}')$. Entretanto, pelo Lema 5.3.3, temos que $ir_i(\mathcal{I}) + ir_a(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq ir_i(\mathcal{I}) + ir_a(\mathcal{I})$. Pelo Teorema 5.2.9, temos o seguinte limitante inferior: $dfi_{\mathbf{RTM}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})+ir_a(\mathcal{I})}{3}$, e o teorema segue. □

Reversão, Transposição, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação sem sinais do problema **Sb$_{\mathbf{FI}}$RTMI** (Algoritmo 38).

Algoritmo 38: Um algoritmo de aproximação para o problema **Sb$_{\mathbf{FI}}$RTMI**.

Entrada: Uma instância intergênica flexível sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão, transposição, move e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}'_{ir}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição, move e indel fornecida pelo Algoritmo 13 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.16. *Seja $\mathcal{I}$ uma instância intergênica flexível sem sinais. O Algoritmo 38 é uma 3-aproximação para o problema* ***Sb$_{FI}$RTMI****.*

Demonstração. Pelo Lema 5.3.2, temos que a sequência fornecida pelo Algoritmo 13 para a instância intergênica rígida sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas **Sb$_{\mathbf{I}}$RTMI** e **Sb$_{\mathbf{FI}}$RTMI** compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema **Sb$_{\mathbf{FI}}$RTMI**. Pelo Lema 4.1.58, temos que $|S'| \leq ib_1(\mathcal{I}')$. Entretanto, pelo Lema 5.3.1, temos que $ir_i(\mathcal{I}) = ib_1(\mathcal{I}')$. Logo, $|S'| \leq ir_i(\mathcal{I})$. Pelo Teorema 5.2.8, temos o seguinte limitante inferior: $dfi_{\mathbf{RTMI}}(\mathcal{I}) \geq \frac{ir_i(\mathcal{I})}{3}$, e o teorema segue. □

Transposição

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3.5 para a variação sem sinais do problema **Sb$_{\mathbf{FI}}$T**.

Note que a versão rígida do problema **Sb$_{\mathbf{FI}}$T**, chamado de Ordenação de Permutações por Transposições Intergênicas (**Sb$_{\mathbf{I}}$T**), possui um algoritmo de aproximação com um fator de 3.5, que chamaremos de 3.5-**Sb$_{\mathbf{I}}$T**. Além disso, temos o lema a seguir.

Lema 5.3.17. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada sem sinais. O algoritmo 3.5-$\boldsymbol{Sb_I T}$ transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ utilizando uma sequência de eventos de transposição S', tal que $|S'| \leq \frac{7(n+1-c_b(G(\mathcal{I}')))}{4}$.*

Demonstração. Diretamente pelo Lema 5.1 de Oliveira *et al.* [59]. □

Agora considere o Algoritmo 39.

Algoritmo 39: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}T}$.

Entrada: Uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de transposição S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c''(\mathcal{I})$

2 Seja S' uma sequência de eventos de transposição fornecida pelo algoritmo 3.5-$\mathbf{Sb_I T}$ para a instância $\mathcal{I}'$

3 **retorne** S'

Teorema 5.3.18. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. O Algoritmo 39 é uma 3.5-aproximação para o problema $\boldsymbol{Sb_{FI}T}$.*

Demonstração. Pelo Lema 5.3.8, temos que a sequência fornecida pelo algoritmo 3.5-$\mathbf{Sb_I T}$ para a instância intergênica rígida balanceada sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_I T}$ e $\mathbf{Sb_{FI}T}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}T}$. Pelo Lema 5.3.17, temos que $|S'| \leq \frac{7(n+1-c_b(G(\mathcal{I}')))}{4}$. Entretanto, pelo Lema 5.3.7, temos que $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq \frac{7(n+1-c_d(G(\mathcal{I})))}{4}$. Pelo Teorema 5.2.10, temos o seguinte limitante inferior: $dfi_{\mathbf{T}}(\mathcal{I}) \geq \frac{n+1-c_d(G(\mathcal{I}))}{2}$, e o teorema segue. □

Transposição e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 2.5 para a variação sem sinais do problema $\mathbf{Sb_{FI}TM}$.

Note que a versão rígida do problema $\mathbf{Sb_{FI}TM}$, chamado de Ordenação de Permutações por Operações Intergênicas de Transposição e Move ($\mathbf{Sb_I TM}$), possui um algoritmo de aproximação com um fator de 2.5, que chamaremos de 2.5-$\mathbf{Sb_I TM}$. Além disso, temos o lema a seguir.

Lema 5.3.19. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada sem sinais. O algoritmo 2.5-$\boldsymbol{Sb_I TM}$ transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ utilizando uma sequência de eventos de transposição e move S', tal que $|S'| \leq \frac{5(n+1-c_b(G(\mathcal{I})))}{4}$.*

Demonstração. Diretamente pelo Lema 7.10 de Oliveira *et al.* [59]. □

Agora considere o Algoritmo 40.

Algoritmo 40: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}TM}$.

Entrada: Uma instância intergênica flexível balanceada sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de transposição e move S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c''(\mathcal{I})$

2 Seja S' uma sequência de eventos de transposição e move fornecida pelo algoritmo 2.5-$\mathbf{Sb_ITM}$ para a instância $\mathcal{I}'$

3 **retorne** S'

Teorema 5.3.20. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada sem sinais. O Algoritmo 40 é uma 2.5-aproximação para o problema $\boldsymbol{Sb_{FI}TM}$.*

Demonstração. Pelo Lema 5.3.8, temos que a sequência fornecida pelo algoritmo 2.5-$\mathbf{Sb_ITM}$ para a instância intergênica rígida balanceada sem sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada sem sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_ITM}$ e $\mathbf{Sb_{FI}TM}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}TM}$. Pelo Lema 5.3.19, temos que $|S'| \leq \frac{5(n+1-c_b(G(\mathcal{I}')))}{4}$. Entretanto, pelo Lema 5.3.7, temos que $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq \frac{5(n+1-c_d(G(\mathcal{I})))}{4}$. Pelo Teorema 5.2.10, temos o seguinte limitante inferior: $dfi_{\mathbf{TM}}(\mathcal{I}) \geq \frac{n+1-c_d(G(\mathcal{I}))}{2}$, e o teorema segue. □

Resultados Experimentais

Nesta seção, apresentaremos os resultados práticos dos algoritmos propostos para a variação sem sinais dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RI}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RMI}$, $\mathbf{Sb_{FI}RT}$, $\mathbf{Sb_{FI}RTI}$, $\mathbf{Sb_{FI}RTM}$, $\mathbf{Sb_{FI}RTMI}$, $\mathbf{Sb_{FI}T}$ e $\mathbf{Sb_{FI}TM}$.

Criamos uma base de dados para cada problema e utilizamos os identificadores $U_{\mathbf{Sb_{FI}R}}$, $U_{\mathbf{Sb_{FI}RI}}$, $U_{\mathbf{Sb_{FI}RM}}$, $U_{\mathbf{Sb_{FI}RMI}}$, $U_{\mathbf{Sb_{FI}RT}}$, $U_{\mathbf{Sb_{FI}RTI}}$, $U_{\mathbf{Sb_{FI}RTM}}$, $U_{\mathbf{Sb_{FI}RTMI}}$, $U_{\mathbf{Sb_{FI}T}}$ e $U_{\mathbf{Sb_{FI}TM}}$ para a base de dados dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RI}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RMI}$, $\mathbf{Sb_{FI}RT}$, $\mathbf{Sb_{FI}RTI}$, $\mathbf{Sb_{FI}RTM}$, $\mathbf{Sb_{FI}RTMI}$, $\mathbf{Sb_{FI}T}$ e $\mathbf{Sb_{FI}TM}$, respectivamente. Cada base de dados é dividida em cinco grupos. Cada grupo possui 1000 instâncias do tamanho 100, sendo que o tamanho de uma instância é a quantidade de genes do genoma de origem e alvo. Além disso, cada grupo é identificado pelo grau de flexibilização das instâncias contidas nele. Os identificadores dos grupos de cada base de dados são 10%, 20%, 30%, 40% e 50%. Cada instância é gerada da seguinte forma: Seja $\mathcal{S} = (\pi = (1\ 2\ \ldots\ 100), \breve{\pi})$ uma representação intergênica rígida sem sinais de um genoma de origem, de forma que o tamanho de cada região intergênica $\breve{\pi}_i$ foi escolhido de maneira aleatória no intervalo $[0..100]$. Em seguida, criamos uma representação intergênica flexível sem sinais do genoma alvo $\mathcal{T} = (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max})$ da seguinte forma: (i) $\iota = (1\ 2\ \ldots\ 100)$; (ii) Seja f o grau de flexibilização adotado no grupo, temos que $\forall \breve{\iota}_i^{\min} \in \breve{\iota}^{\min} : \breve{\iota}_i^{\min} = \lfloor \breve{\pi}_i \times (1-f) \rfloor$ e $\forall \breve{\iota}_i^{\max} \in \breve{\iota}^{\max} : \breve{\iota}_i^{\max} = \lceil \breve{\pi}_i \times (1+f) \rceil$. Com base na disponibilidade de operações de reversão, transposição, move e indel determinada para cada base de dados e grupo, uma operação σ é escolhida aleatoriamente e aplicada em $\mathcal{S}$ ($\mathcal{S} = \mathcal{S} \cdot \sigma$). Os parâmetros de cada

operação também são escolhidos de forma aleatória dentro do limite de valores válidos. O valor do parâmetro x de um indel $\delta_{(x)}^{(i)}$ aplicado em uma região intergênica $\breve{\pi}_i$ é escolhido dentro do intervalo $[-\breve{\pi}_i..\breve{\pi}_i]$, também aleatoriamente. Quando não houver operações disponíveis para serem aplicadas, então temos a instância intergênica flexível sem sinais $\mathcal{I}$, que é composta pela dupla $(\mathcal{S}, \mathcal{T})$. Esse processo repete-se até que cada grupo possua um total de 1000 instâncias. Em todos os casos, consideramos sorteios aleatórios com uma distribuição uniforme.

A quantidade de operações disponíveis para gerar cada instância difere entre as bases de dados. A Tabela 5.2 mostra, para cada base de dados, a quantidade de operações utilizada para criar cada instância.

Tabela 5.2: Quantidade de operações aplicadas para gerar cada instância intergênica flexível sem sinais.

Base de Dados	Revesões	Transposições	Moves	Indels
$U_{\mathbf{Sb_{FI}R}}$	50	0	0	0
$U_{\mathbf{Sb_{FI}RI}}$	40	0	0	10
$U_{\mathbf{Sb_{FI}RM}}$	40	0	10	0
$U_{\mathbf{Sb_{FI}RMI}}$	40	0	5	5
$U_{\mathbf{Sb_{FI}RT}}$	25	25	0	0
$U_{\mathbf{Sb_{FI}RTI}}$	20	20	0	10
$U_{\mathbf{Sb_{FI}RTM}}$	20	20	10	0
$U_{\mathbf{Sb_{FI}RTMI}}$	20	20	5	5
$U_{\mathbf{Sb_{FI}T}}$	0	50	0	0
$U_{\mathbf{Sb_{FI}TM}}$	0	40	10	0

Utilizando o conceito de regiões intergênicas e considerando todos os grupos das bases de dados $U_{\mathbf{Sb_{FI}R}}$, $U_{\mathbf{Sb_{FI}RM}}$, $U_{\mathbf{Sb_{FI}RT}}$ e $U_{\mathbf{Sb_{FI}RTM}}$, foi observado que 100% das instâncias pertencem ao cenário de equilíbrio. As instâncias foram geradas com o objetivo de ser necessária uma quantidade considerável de operações para que o genoma de origem atinja o genoma alvo. Logo, poucas regiões intergênicas estáveis tendem a ser mantidas, o que pode explicar o fato de 100% das instâncias pertencerem ao cenário de equilíbrio.

Utilizando a estrutura de grafo de ciclos ponderado flexível e considerando todos os grupos das bases de dados $U_{\mathbf{Sb_{FI}T}}$ e $U_{\mathbf{Sb_{FI}TM}}$, foi observado que 55.8%, 25.8% e 18.4% das instâncias pertencem ao cenário de equilíbrio, sorvedouro e fonte, respectivamente.

Para garantir uma proporcionalidade entre os possíveis cenários nos problemas que utilizam um modelo de rearranjo composto exclusivamente por eventos conservativos, criamos as bases de dados U_{IR} e U_C. Ambas as bases de dados possuem cinco grupos, sendo que cada grupo possui 3000 instâncias intergênicas flexíveis balanceadas sem sinais de tamanho 100 e é identificado pelo grau de flexibilização máxima das instâncias contidas nele. Os identificadores dos grupos são 10%, 20%, 30%, 40% e 50%. Utilizando o conceito de regiões intergênicas flexíveis, cada grupo da base de dados U_{IR} possui 1000 instâncias em cada cenário: equilíbrio, fonte e sorvedouro. Também para a base de dados U_C, utilizando a estrutura de grafo de ciclos ponderado flexível, cada grupo possui 1000 instâncias em cada cenário.

A geração de uma instância na base dados U_{IR} é dada da seguinte forma: Seja

$\mathcal{S} = (\pi = (1\ 2\ \ldots\ 100), \breve{\pi})$ uma representação intergênica rígida sem sinais de um genoma de origem, de forma que o tamanho de cada região intergênica $\breve{\pi}_i$ foi escolhido aleatoriamente no intervalo [0..100]. Em seguida, criamos uma representação intergênica flexível sem sinais do genoma alvo $\mathcal{T} = (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max})$ da seguinte forma: (i) $\iota = (1\ 2\ \ldots\ 100)$; (ii) Seja f o grau de flexibilização máxima adotado no grupo. Para cada valor de $i \in \{1, 2, \ldots, 101\}$, temos que l e u são porcentagens escolhidas de maneira aleatória no conjunto $\{0\%, 1\%, \ldots, f\}$ e os valores de $\breve{\iota}_i^{\min}$ e $\breve{\iota}_i^{\max}$ são atualizados para $\lfloor \breve{\pi}_i \times (1-l) \rfloor$ e $\lceil \breve{\pi}_i \times (1+u) \rceil$, respectivamente. Em seguida, 15 trocas são aplicadas em π. Uma troca muda a posição de dois elementos de π, sendo que ambas as posições são escolhidas aleatoriamente. Por fim, com base na disponibilidade de cenários para serem adicionados ao grupo, um cenário é escolhido de forma aleatória e o seguinte processamento é realizado:

- Equilíbrio: caso a instância intergênica flexível sem sinais $\mathcal{I} = (\mathcal{S}, \mathcal{T})$ pertença ao cenário de equilíbrio com base no conceito de regiões intergênicas, então $\mathcal{I}$ é adicionada ao grupo.

- Fonte: neste caso, $\lfloor \frac{\sum_{i=1}^{101} \breve{\pi}_i - \sum_{i=1}^{101} \breve{\iota}_i^{\min}}{2} \rfloor$ nucleotídeos são aleatoriamente removidos das regiões intergênicas $\breve{\pi}$. Caso a instância intergênica flexível sem sinais $\mathcal{I} = (\mathcal{S}, \mathcal{T})$ resultante pertença ao cenário fonte com base no conceito de regiões intergênicas, então $\mathcal{I}$ é adicionada ao grupo.

- Sorvedouro: neste caso, $\lfloor \frac{\sum_{i=1}^{101} \breve{\iota}_i^{\max} - \sum_{i=1}^{101} \breve{\pi}_i}{2} \rfloor$ nucleotídeos são aleatoriamente adicionados nas regiões intergênicas $\breve{\pi}$. Caso a instância intergênica flexível sem sinais $\mathcal{I} = (\mathcal{S}, \mathcal{T})$ resultante pertença ao cenário sorvedouro com base no conceito de regiões intergênicas, então $\mathcal{I}$ é adicionada ao grupo.

Este processo repete-se até que cada grupo possua 3000 instâncias. Em todos os casos, consideramos sorteios aleatórios com uma distribuição uniforme.

A criação de uma instância na base dados U_C é similar ao processo descrito anteriormente, diferenciando-se pelo fato de utilizar a estrutura de grafo de ciclos ponderado flexível para determinar o cenário de cada instância que é gerada. Além disso, caso o cenário fonte seja escolhido durante o processo de geração de uma instância, o peso $\lfloor \frac{\sum_{i=1}^{101} \breve{\pi}_i - \sum_{i=1}^{101} \breve{\iota}_i^{\min}}{2} \rfloor$ é aleatoriamente removido das arestas pretas do grafo. Caso o cenário sorvedouro seja escolhido durante o processo de geração de uma instância, o peso $\lfloor \frac{\sum_{i=1}^{101} \breve{\iota}_i^{\max} - \sum_{i=1}^{101} \breve{\pi}_i}{2} \rfloor$ é aleatoriamente adicionado nas arestas pretas do grafo.

A base de dados U_{IR} foi criada para ser utilizada pelos algoritmos da variação sem sinais dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RT}$ e $\mathbf{Sb_{FI}RTM}$. Similarmente, a base de dados U_C foi criada para ser utilizada pelos algoritmos da variação sem sinais dos problemas $\mathbf{Sb_{FI}T}$ e $\mathbf{Sb_{FI}TM}$.

A seguir, apresentamos os resultados obtidos pelos algoritmos propostos para a variação sem sinais dos problemas investigados neste capítulo. Nas tabelas apresentadas a seguir temos a informação por grupo do grau de flexibilização adotado e as métricas de distância e aproximação, sendo que para ambas temos a informação sobre o menor e o maior valor registrado, além da média dos valores do grupo. As colunas Distância e Aproximação indicam a quantidade de operações e o fator de aproximação para uma solução fornecida por um algoritmo.

A Tabela 5.3 apresenta os resultados do Algoritmo 31 utilizando as bases de dados $U_{\mathbf{Sb_{FI}R}}$ e U_{IR}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.9.

Tabela 5.3: Resultados do Algoritmo 31 utilizando as bases de dados $U_{\mathbf{Sb_{FI}R}}$ e U_{IR}.

$U_{\mathbf{Sb_{FI}R}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	69	89.14	109	2.29	2.78	3.23
20%	72	90.52	110	2.33	2.83	3.27
30%	75	92.27	119	2.44	2.89	3.31
40%	76	93.80	113	2.55	2.94	3.31
50%	74	95.03	116	2.39	2.97	3.34

U_{IR}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	44	65.97	85	2.20	2.71	3.16
20%	44	67.31	86	2.27	2.74	3.16
30%	39	68.68	89	2.19	2.77	3.25
40%	42	69.86	93	2.29	2.79	3.23
50%	40	70.65	92	2.26	2.81	3.32

Pela Tabela 5.3, é possível perceber que o Algoritmo 31 na base de dados $U_{\mathbf{Sb_{FI}R}}$ aplicou mais operações de reversão à medida que o grau de flexibilização aumentou. Este fato pode ser constatado ao verificar as métricas de distância média e aproximação média. Tal comportamento também pode ser observado na base de dados U_{IR}, mas vale ressaltar que a base de dados não foi construída com base em eventos de rearranjo e o grau de flexibilização para os tamanhos mínimo e máximo de cada região intergênica no genoma alvo não é fixo. Considerando ambas as bases de dados, as aproximações máxima e mínima registrada foram de 3.34 e 2.19, respectivamente.

A Tabela 5.4 apresenta os resultados do Algoritmo 32 utilizando a base de dados $U_{\mathbf{Sb_{FI}RI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.8.

Tabela 5.4: Resultados do Algoritmo 32 utilizando a base de dados $U_{\mathbf{Sb_{FI}RI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	67	83.44	99	2.38	2.79	3.26
20%	65	81.49	99	2.32	2.76	3.20
30%	63	79.88	99	2.39	2.72	3.21
40%	64	78.67	98	2.35	2.70	3.12
50%	61	77.40	101	2.35	2.68	3.07

Pela Tabela 5.4 é possível perceber que o Algoritmo 32 tende a utilizar menos operações na média à medida que o grau de flexibilização aumenta. Além disso, a aproximação

média também apresentou uma tendência de queda à medida que o grau de flexibilização aumenta. Podemos notar também que a distância mínima para cada grupo ficou dentro do intervalo [61..67], sendo que cada instância da base de dados foi gerada a partir de 50 operações de reversão ou indel. Entretanto, para todos os grupos, a aproximação média foi inferior a 2.80.

A Tabela 5.5 apresenta os resultados do Algoritmo 33 utilizando as bases de dados $U_{\mathbf{Sb_{FI}RM}}$ e U_{IR}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.9.

Tabela 5.5: Resultados do Algoritmo 33 utilizando as bases de dados $U_{\mathbf{Sb_{FI}RM}}$ e U_{IR}.

$U_{\mathbf{Sb_{FI}RM}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	70	88.01	109	2.22	2.82	3.22
20%	63	87.95	111	2.24	2.85	3.36
30%	66	87.37	110	2.39	2.87	3.29
40%	69	87.34	108	2.26	2.90	3.37
50%	67	86.64	110	2.43	2.91	3.35

U_{IR}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	36	57.27	80	1.83	2.36	3.12
20%	34	58.12	84	1.78	2.37	3.11
30%	34	59.05	83	1.84	2.38	3.18
40%	37	59.98	89	1.83	2.40	3.16
50%	32	60.53	90	1.84	2.41	3.24

Pelos dados da Tabela 5.5, podemos notar que, na base de dados $U_{\mathbf{Sb_{FI}RM}}$, a distância média de cada grupo obtida através do Algoritmo 33 tende a diminuir à medida que o grau de flexibilidade aumenta. Entretanto a aproximação média apresentou um comportamento oposto. A aproximação máxima registrada foi de 3.37 e ocorreu no grupo com um grau de flexibilização de 40%. Já na base de dados U_{IR}, tanto a distância média quanto a aproximação média por grupo, tendem a aumentar à medida que o grau de flexibilidade máxima também aumenta. A aproximação máxima registrada foi de 3.24 e ocorreu no grupo com um grau de flexibilização máxima de 50%.

A Tabela 5.6 apresenta os resultados do Algoritmo 34 utilizando a base de dados $U_{\mathbf{Sb_{FI}RMI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.8.

Pela Tabela 5.6, é possível observar que tanto a distância média como a aproximação média diminuem à medida que o grau de flexibilização aumenta. Além disso, a distância mínima para os grupos com 40% e 50% de flexibilização foi inferior a 60, aproximando-se da quantidade de 50 operações utilizadas para criar cada instância. A aproximação máxima registrada foi de 3.22 e ocorreu no grupo com 10% de flexibilização.

A Tabela 5.7 apresenta os resultados do Algoritmo 35 utilizando as bases de dados

Tabela 5.6: Resultados do Algoritmo 34 utilizando a base de dados $U_{\mathbf{Sb_{FI}RMI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	65	84.25	103	2.29	2.75	3.22
20%	63	81.25	100	2.19	2.69	3.14
30%	63	79.75	97	2.26	2.66	3.21
40%	57	78.23	96	2.28	2.64	3.00
50%	59	76.74	100	2.27	2.62	3.03

$U_{\mathbf{Sb_{FI}RT}}$ e U_{IR}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.9.

Tabela 5.7: Resultados do Algoritmo 35 utilizando as bases de dados $U_{\mathbf{Sb_{FI}RT}}$ e U_{IR}.

$U_{\mathbf{Sb_{FI}RT}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	64	71.63	82	2.83	2.93	3.00
20%	59	71.42	82	2.82	2.94	3.00
30%	60	71.31	80	2.83	2.94	3.00
40%	59	71.17	80	2.83	2.94	3.00
50%	60	71.27	83	2.83	2.94	3.00

U_{IR}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	33	47.24	58	2.64	2.89	3.00
20%	34	47.83	58	2.64	2.90	3.00
30%	33	48.34	59	2.71	2.90	3.00
40%	34	48.78	60	2.71	2.90	3.00
50%	32	49.00	60	2.69	2.90	3.00

Pelos dados da Tabela 5.7 podemos perceber que, em ambas as bases de dados, o Algoritmo 35 apresentou uma razão de aproximação máxima igual a 3.00. Além disso, considerando a variação entre a menor aproximação mínima e a maior aproximação máxima entre todos os grupos, obtemos os valores de 0.18 e 0.36 para as bases de dados $U_{\mathbf{Sb_{FI}RT}}$ e U_{IR}, respectivamente. A distância mínima registrada na base de dados $U_{\mathbf{Sb_{FI}RT}}$, considerando todos os grupos, foi de 59, nove a mais do que o número de operações utilizadas para gerar cada instância, sendo que a distância máxima registrada foi de 83. Também é possível observar uma estabilidade na distância média considerando todos os grupos da base $U_{\mathbf{Sb_{FI}RT}}$, onde os valores ficaram entre 71.17 e 71.63.

A Tabela 5.8 apresenta os resultados do Algoritmo 36 utilizando a base de dados $U_{\mathbf{Sb_{FI}RTI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.8.

Pelos resultados exibidos na Tabela 5.8, podemos observar que a razão de aproximação máxima obtida pelo Algoritmo 36, em todos os grupos, foi de 2.96, sendo um valor próximo

Tabela 5.8: Resultados do Algoritmo 36 utilizando a base de dados $U_{\mathbf{Sb_{FI}RTI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	58	67.40	78	2.78	2.89	2.96
20%	58	66.97	78	2.77	2.89	2.96
30%	57	66.25	77	2.78	2.89	2.96
40%	54	66.10	77	2.77	2.88	2.96
50%	54	65.59	78	2.77	2.88	2.96

ao limite teórico (3.00). É possível notar que o Algoritmo 36 apresentou uma variação pequena em relação à razão de aproximação, o que pode ser constatado observando a variação entre as aproximações mínima e máxima de cada grupo. Por fim, a distância média fornecida pelo algoritmo apresenta uma leve tendência de queda à medida que o grau de flexibilização aumenta.

A Tabela 5.9 apresenta os resultados do Algoritmo 37 utilizando as bases de dados $U_{\mathbf{Sb_{FI}RTM}}$ e U_{IR}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.9.

Tabela 5.9: Resultados do Algoritmo 37 utilizando as bases de dados $U_{\mathbf{Sb_{FI}RTM}}$ e U_{IR}.

$U_{\mathbf{Sb_{FI}RTM}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	59	68.12	77	2.79	2.91	2.96
20%	57	67.59	76	2.82	2.91	2.96
30%	58	66.92	77	2.82	2.91	2.96
40%	54	66.16	77	2.81	2.91	2.96
50%	55	65.57	75	2.82	2.91	2.96

U_{IR}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	33	46.86	57	2.67	2.87	2.95
20%	34	47.42	58	2.62	2.87	2.95
30%	33	47.97	58	2.69	2.88	2.95
40%	34	48.40	59	2.71	2.88	2.95
50%	32	48.60	60	2.71	2.88	2.95

Pela Tabela 5.9 é possível observar que, considerando os grupos de cada base de dados, a aproximação máxima do Algoritmo 37 foi de 2.96 e 2.95 nas bases de dados $U_{\mathbf{Sb_{FI}RTM}}$ e U_{IR}, respectivamente. Uma característica interessante observada, foi a variação zero, considerando a aproximação média entre os grupos da base de dados $U_{\mathbf{Sb_{FI}RTM}}$, e de 0.01, considerando os grupos da base de dados U_{IR}. Por fim, o Algoritmo 37 forneceu uma distância mínima inferior a 60 para todos os grupos da base de dados $U_{\mathbf{Sb_{FI}RTM}}$ (50 operações foram utilizadas para gerar cada instância).

A Tabela 5.10 apresenta os resultados do Algoritmo 38 utilizando a base de dados

$U_{\mathbf{Sb_{FI}RTMI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.8.

Tabela 5.10: Resultados do Algoritmo 38 utilizando a base de dados $U_{\mathbf{Sb_{FI}RTMI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	57	67.92	78	2.83	2.95	3.00
20%	58	67.55	78	2.86	2.96	3.00
30%	56	67.05	77	2.86	2.96	3.00
40%	56	66.50	77	2.86	2.96	3.00
50%	55	66.04	76	2.86	2.95	3.00

Pelos dados da Tabela 5.10 podemos ver que a razão de aproximação máxima obtida pelo Algoritmo 38, em todos os grupos, atingiu o limite teórico garantido, sendo igual a 3.00. Vale ressaltar que isso não significa que a aproximação do algoritmo é justa, uma vez que a razão de aproximação computada para cada instância foi realizada utilizando o limitante inferior (e não a distância exata). O Algoritmo 38 foi o único que, dentre os algoritmos para as variações sem sinais dos problemas investigados neste capítulo, para os experimentos práticos, apresentou uma razão de aproximação (utilizando o limitante inferior) que atingiu o limite teórico de aproximação. Por fim, é possível notar que o algoritmo apresentou pouca variação em relação às métricas de distância e de aproximação, considerando todos os grupos.

A Tabela 5.11 apresenta os resultados do Algoritmo 39 utilizando as bases de dados $U_{\mathbf{Sb_{FI}T}}$ e U_{C}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.10.

Tabela 5.11: Resultados do Algoritmo 39 utilizando as bases de dados $U_{\mathbf{Sb_{FI}T}}$ e U_{C}.

$U_{\mathbf{Sb_{FI}T}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	41	54.29	79	1.10	1.42	1.88
20%	42	53.74	75	1.15	1.42	1.89
30%	40	54.45	72	1.13	1.45	1.95
40%	42	55.20	71	1.18	1.48	1.89
50%	40	56.47	77	1.19	1.52	1.95

U_{C}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	16	35.09	52	1.13	1.70	2.27
20%	16	36.30	57	1.20	1.72	2.53
30%	18	37.60	56	1.20	1.75	2.40
40%	17	38.55	58	1.20	1.76	2.40
50%	17	39.28	60	1.20	1.78	2.36

Pela Tabela 5.11 é possível notar que o Algoritmo 39 apresentou um ótimo resultado

prático em comparação com o limite teórico garantido (3.5-aproximação). Na base de dados $U_{\mathbf{Sb_{FI}T}}$, a maior aproximação máxima foi de 1.95, registrada nos grupos 30% e 50%. Já na base de dados U_{C} a maior aproximação máxima foi de 2.53, no grupo 20%. É importante notar também que, em todos os grupos da base de dados $U_{\mathbf{Sb_{FI}T}}$, o Algoritmo 39 forneceu uma distância mínima inferior a 43. Vale ressaltar que em todas as instâncias da base de dados foram utilizadas 50 operações para gerar cada uma delas. Entretanto, algumas operações podem desfazer operações aplicadas previamente durante o processo de criação. Por esse motivo, é possível que em algumas instâncias seja necessário menos que 50 operações para atingir o genoma alvo.

A Tabela 5.12 apresenta os resultados do Algoritmo 40 utilizando as bases de dados $U_{\mathbf{Sb_{FI}TM}}$ e U_{C}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.10.

Tabela 5.12: Resultados do Algoritmo 40 utilizando a base de dados $U_{\mathbf{Sb_{FI}TM}}$ e U_{C}.

$U_{\mathbf{Sb_{FI}TM}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	38	51.13	68	1.13	1.41	1.78
20%	37	49.85	64	1.11	1.40	1.83
30%	37	50.07	67	1.09	1.43	1.88
40%	37	50.59	67	1.16	1.45	1.89
50%	37	51.57	72	1.14	1.49	1.97

U_{C}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	16	33.83	50	1.13	1.63	2.04
20%	16	35.05	55	1.20	1.66	2.13
30%	17	36.26	55	1.13	1.68	2.15
40%	17	37.20	58	1.20	1.70	2.20
50%	16	37.89	58	1.20	1.71	2.08

Pelos dados da Tabela 5.12 podemos notar que a aproximação máxima registrada do Algoritmo 40, em todos os grupos da base de dados $U_{\mathbf{Sb_{FI}TM}}$, foi inferior a 2.00. Já na base de dados U_{C}, a aproximação máxima foi de 2.20, registrada no grupo 40%. Utilizando a base de dados $U_{\mathbf{Sb_{FI}TM}}$ o Algoritmo 40 também apresentou, para todos os grupos, uma distância mínima menor do que o número de operações utilizadas para gerar cada instância, em todos os grupos esse valor foi inferior a 39.

De maneira geral, todos os algoritmos apresentaram um bom desempenho na prática, sendo que o Algoritmo 38 foi o único em que a razão de aproximação máxima atingiu o limite teórico garantido pelo mesmo.

5.3.2 Instâncias Intergênicas Flexíveis com Sinais

Nesta seção, apresentaremos algoritmos de aproximação para a variação com sinais dos problemas investigados neste capítulo, com base nas funções de redução apresentadas no início da Seção 5.3.

Reversão

Nesta seção, apresentaremos um algoritmo de aproximação com fator 2 para a variação com sinais do problema $\mathbf{Sb_{FI}R}$.

Note que a variação com sinais do problema $\mathbf{Sb_{I}R}$ possui um algoritmo de aproximação com um fator de 2 [59], que chamaremos de 2-$\mathbf{Sb_{I}R}$. Além disso, temos o lema a seguir.

Lema 5.3.21. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada com sinais. O algoritmo 2-$\boldsymbol{Sb_{I}R}$ transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ utilizando uma sequência de eventos de reversão S', tal que $|S'| \leq 2(n + 1 - c_b(G(\mathcal{I})))$.*

Demonstração. Diretamente pelo Algoritmo 1 de Oliveira *et al.* [57]. □

Agora considere o Algoritmo 41.

Algoritmo 41: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}R}$.

Entrada: Uma instância intergênica flexível balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c''(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão fornecida pelo algoritmo 2-$\mathbf{Sb_{I}R}$ para a instância $\mathcal{I}'$

3 **retorne** S'

Teorema 5.3.22. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. O Algoritmo 41 é uma 2-aproximação para o problema $\boldsymbol{Sb_{FI}R}$.*

Demonstração. Pelo Lema 5.3.8, temos que a sequência fornecida pelo algoritmo 2-$\mathbf{Sb_{I}R}$ para a instância intergênica rígida balanceada com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_{I}R}$ e $\mathbf{Sb_{FI}R}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}R}$. Pelo Lema 5.3.21, temos que $|S'| \leq 2(n + 1 - c_b(G(\mathcal{I})))$. Entretanto, pelo Lema 5.3.7, temos que $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq 2(n + 1 - c_d(G(\mathcal{I})))$. Pelo Teorema 5.2.11, temos o seguinte limitante inferior: $dfi_{\mathbf{R}}(\mathcal{I}) \geq n + 1 - c_d(G(\mathcal{I}))$, e o teorema segue. □

Reversão e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 2 para a variação com sinais do problema $\mathbf{Sb_{FI}RI}$.

Note que a variação com sinais do problema $\mathbf{Sb_{I}RI}$ possui um algoritmo de aproximação com um fator de 2 [57], que chamaremos de 2-$\mathbf{Sb_{I}RI}$. Além disso, temos o lema a seguir.

Lema 5.3.23. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida com sinais. O algoritmo 2-$\boldsymbol{Sb_{I}RI}$ transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ utilizando uma sequência de eventos de reversão e indel S', tal que $|S'| \leq 2(n + 1 - c_b(G(\mathcal{I})))$.*

Demonstração. Diretamente pelo Algoritmo 2 de Oliveira *et al.* [57]. □

Agora considere o Algoritmo 42.

Algoritmo 42: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RI}$.

Entrada: Uma instância intergênica flexível com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c'(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e indel fornecida pelo algoritmo 2-$\mathbf{Sb_{I}RI}$ para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.24. *Seja $\mathcal{I}$ uma instância intergênica flexível com sinais. O Algoritmo 42 é uma 2-aproximação para o problema $\boldsymbol{Sb_{FI}RI}$.*

Demonstração. Pelo Lema 5.3.6, temos que a sequência fornecida pelo algoritmo 2-$\mathbf{Sb_{I}RI}$ para a instância intergênica rígida com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_{I}RI}$ e $\mathbf{Sb_{FI}RI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RI}$. Pelo Lema 5.3.23, temos que $|S'| \leq 2(n + 1 - c_b(G(\mathcal{I})))$. Entretanto, pelo Lema 5.3.5, temos que $c_e(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq 2(n + 1 - c_e(G(\mathcal{I})))$. Pelo Teorema 5.2.12, temos o seguinte limitante inferior: $dfi_{\mathbf{RI}}(\mathcal{I}) \geq n + 1 - c_e(G(\mathcal{I}))$, e o teorema segue. □

Reversão e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 2 para a variação com sinais do problema $\mathbf{Sb_{FI}RM}$ (Algoritmo 43).

Teorema 5.3.25. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. O Algoritmo 43 é uma 2-aproximação para o problema $\boldsymbol{Sb_{FI}RM}$.*

Algoritmo 43: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RM}$.

Entrada: Uma instância intergênica flexível balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão e move S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c''(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão e move fornecida pelo Algoritmo 19 para a instância $\mathcal{I}'$

3 **retorne** S'

Demonstração. Pelo Lema 5.3.8, temos que a sequência fornecida pelo Algoritmo 19 para a instância intergênica rígida balanceada com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_IRM}$ e $\mathbf{Sb_{FI}RM}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RM}$. Pelo Lema 4.1.69, temos que $|S'| \leq 2(n+1) - (c(G(\mathcal{I}')) + c_b(G(\mathcal{I}')))$. Entretanto, pelo Lema 5.3.7, temos que $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$ e $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq 2(n+1) - (c(G(\mathcal{I})) + c_d(G(\mathcal{I})))$. Pelo Teorema 5.2.13, temos o seguinte limitante inferior: $dfi_{\mathbf{RM}}(\mathcal{I}) \geq n + 1 - \frac{(c(G(\mathcal{I})) + c_d(G(\mathcal{I})))}{2}$, e o teorema segue. □

Reversão, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 2 para a variação com sinais do problema $\mathbf{Sb_{FI}RMI}$ (Algoritmo 44).

Algoritmo 44: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RMI}$.

Entrada: Uma instância intergênica flexível com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão, move e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c'(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão, move e indel fornecida pelo Algoritmo 21 para a instância $\mathcal{I}'$

3 **retorne** S'

Teorema 5.3.26. *Seja $\mathcal{I}$ uma instância intergênica flexível com sinais. O Algoritmo 44 é uma 2-aproximação para o problema $\mathbf{Sb_{FI}RMI}$.*

Demonstração. Pelo Lema 5.3.6, temos que a sequência fornecida pelo Algoritmo 21 para a instância intergênica rígida com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_IRMI}$ e $\mathbf{Sb_{FI}RMI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RMI}$. Pelo Lema 4.1.74, temos que $|S'| \leq 2(n+1) - (c(G(\mathcal{I}')) + c_b(G(\mathcal{I}')))$. Entretanto, pelo Lema 5.3.5, temos que $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$ e $c_e(G(\mathcal{I})) =$

$c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq 2(n+1) - (c(G(\mathcal{I})) + c_e(G(\mathcal{I})))$. Pelo Teorema 5.2.14, temos o seguinte limitante inferior: $dfi_{\mathbf{RMI}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I}))+c_e(G(\mathcal{I}))}{2}$, e o teorema segue. □

Reversão e Transposição

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação com sinais do problema $\mathbf{Sb_{FI}RT}$.

Note que a variação com sinais do problema $\mathbf{Sb_{I}RT}$ possui um algoritmo de aproximação com um fator de 3 [59], que chamaremos de 3-$\mathbf{Sb_{I}RT}$. Além disso, temos o lema a seguir.

Lema 5.3.27. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada com sinais. O algoritmo 3-$\boldsymbol{Sb_{I}RT}$ transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ utilizando uma sequência de eventos de reversão e transposição S', tal que $|S'| \leq \frac{3(n+1-c_b(G(\mathcal{I}')))}{2}$.*

Demonstração. Diretamente pelo Lema 6.3 de Oliveira *et al.* [59]. □

Agora considere o Algoritmo 45.

Algoritmo 45: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RT}$.

Entrada: Uma instância intergênica flexível balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão e transposição S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c''(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão e transposição fornecida pelo algoritmo 3-$\mathbf{Sb_{I}RT}$ para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.28. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. O Algoritmo 45 é uma 3-aproximação para o problema $\boldsymbol{Sb_{FI}RT}$.*

Demonstração. Pelo Lema 5.3.8, temos que a sequência fornecida pelo algoritmo 3-$\mathbf{Sb_{I}RT}$ para a instância intergênica rígida balanceada com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_{I}RT}$ e $\mathbf{Sb_{FI}RT}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RT}$. Pelo Lema 5.3.27, temos que $|S'| \leq \frac{3(n+1-c_b(G(\mathcal{I}')))}{2}$. Entretanto, pelo Lema 5.3.7, temos que $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq \frac{3(n+1-c_d(G(\mathcal{I})))}{2}$. Pelo Teorema 5.2.15, temos o seguinte limitante inferior: $dfi_{\mathbf{RT}}(\mathcal{I}) \geq \frac{n+1-c_d(G(\mathcal{I}))}{2}$, e o teorema segue. □

Reversão, Transposição e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação com sinais do problema $\mathbf{Sb_{FI}RTI}$ (Algoritmo 46).

Algoritmo 46: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RTI}$.

Entrada: Uma instância intergênica flexível com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$
Saída: Uma sequência de eventos de reversão, transposição e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c'(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão, transposição e indel fornecida pelo Algoritmo 24 para a instância $\mathcal{I}'$
3 **retorne** S'

Teorema 5.3.29. *Seja $\mathcal{I}$ uma instância intergênica flexível com sinais. O Algoritmo 46 é uma 3-aproximação para o problema $\boldsymbol{Sb_{FI}RTI}$.*

Demonstração. Pelo Lema 5.3.6, temos que a sequência fornecida pelo Algoritmo 24 para a instância intergênica rígida com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_I RTI}$ e $\mathbf{Sb_{FI}RTI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RTI}$. Pelo Lema 4.1.81, temos que $|S'| \leq \frac{3(n+1-c_b(G(\mathcal{I})))}{2}$. Entretanto, pelo Lema 5.3.5, temos que $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$ e $c_e(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq \frac{3(n+1-c_e(G(\mathcal{I})))}{2}$. Pelo Teorema 5.2.12, temos o seguinte limitante inferior: $dfi_{\mathbf{RTI}}(\mathcal{I}) \geq \frac{n+1-c_e(G(\mathcal{I}))}{2}$, e o teorema segue. □

Reversão, Transposição e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 2.5 para a variação com sinais do problema $\mathbf{Sb_{FI}RTM}$.

Note que a variação com sinais do problema $\mathbf{Sb_I RTM}$ possui um algoritmo de aproximação com um fator de 2.5 [59], que chamaremos de 2.5-$\mathbf{Sb_I RTM}$. Além disso, temos o lema a seguir.

Lema 5.3.30. *Seja $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ uma instância intergênica rígida balanceada com sinais. O algoritmo 2.5-$\boldsymbol{Sb_I RTM}$ transforma $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ utilizando uma sequência de eventos de reversão, transposição e move S', tal que $|S'| \leq \frac{5(n+1-c_b(G(\mathcal{I}')))}{4}$.*

Demonstração. Diretamente pelo Lema 7.11 de Oliveira *et al.* [59]. □

Agora considere o Algoritmo 47.

Teorema 5.3.31. *Seja $\mathcal{I}$ uma instância intergênica flexível balanceada com sinais. O Algoritmo 47 é uma 2.5-aproximação para o problema $\boldsymbol{Sb_{FI}RTM}$.*

Demonstração. Pelo Lema 5.3.8, temos que a sequência fornecida pelo algoritmo 2.5-$\mathbf{Sb_I RTM}$ para a instância intergênica rígida balanceada com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível balanceada com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_I RTM}$ e $\mathbf{Sb_{FI}RTM}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RTM}$. Pelo

Algoritmo 47: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RTM}$.

Entrada: Uma instância intergênica flexível balanceada com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão, transposição e move S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c''(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão, transposição e move fornecida pelo algoritmo 2.5-$\mathbf{Sb_IRTM}$ para a instância $\mathcal{I}'$

3 **retorno** S'

Lema 5.3.30, temos que $|S'| \leq \frac{5(n+1-c_b(G(\mathcal{I}')))}{4}$. Entretanto, pelo Lema 5.3.7, temos que $c_d(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq \frac{5(n+1-c_d(G(\mathcal{I})))}{4}$. Pelo Teorema 5.2.15, temos o seguinte limitante inferior: $dfi_{\mathbf{RTM}}(\mathcal{I}) \geq \frac{n+1-c_d(G(\mathcal{I}))}{2}$, e o teorema segue. □

Reversão, Transposição, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação com sinais do problema $\mathbf{Sb_{FI}RTMI}$ (Algoritmo 48).

Algoritmo 48: Um algoritmo de aproximação para o problema $\mathbf{Sb_{FI}RTMI}$.

Entrada: Uma instância intergênica flexível com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$

Saída: Uma sequência de eventos de reversão, transposição, move e indel S, tal que $(\pi, \breve{\pi}) \cdot S$ atinge o genoma alvo de $\mathcal{I}$

1 $\mathcal{I}' = \mathcal{F}_c'(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão, transposição, move e indel fornecida pelo Algoritmo 27 para a instância $\mathcal{I}'$

3 **retorno** S'

Teorema 5.3.32. *Seja $\mathcal{I}$ uma instância intergênica flexível com sinais. O Algoritmo 48 é uma 3-aproximação para o problema $\mathbf{Sb_{FI}RTMI}$.*

Demonstração. Pelo Lema 5.3.6, temos que a sequência fornecida pelo Algoritmo 27 para a instância intergênica rígida com sinais $\mathcal{I}'$, se aplicada no genoma de origem $(\pi, \breve{\pi})$ da instância intergênica flexível com sinais $\mathcal{I}$, faz com que o genoma alvo seja alcançado. Além disso, note que os problemas $\mathbf{Sb_IRTMI}$ e $\mathbf{Sb_{FI}RTMI}$ compartilham o mesmo modelo de rearranjo. Logo, a sequência S' utiliza apenas eventos permitidos pelo modelo de rearranjo do problema $\mathbf{Sb_{FI}RTMI}$. Pelo Lema 4.1.85, temos que $|S'| \leq \frac{3(n+1-c_b(G(\mathcal{I})))}{2}$. Entretanto, pelo Lema 5.3.5, temos que $c(G(\mathcal{I})) = c(G(\mathcal{I}'))$ e $c_e(G(\mathcal{I})) = c_b(G(\mathcal{I}'))$. Logo, $|S'| \leq \frac{3(n+1-c_e(G(\mathcal{I})))}{2}$. Pelo Teorema 5.2.12, temos o seguinte limitante inferior: $dfi_{\mathbf{RTMI}}(\mathcal{I}) \geq \frac{n+1-c_e(G(\mathcal{I}))}{2}$, e o teorema segue. □

Resultados Experimentais

Nesta seção, apresentaremos os resultados práticos dos algoritmos apresentados para a variação com sinais dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RI}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RMI}$, $\mathbf{Sb_{FI}RT}$, $\mathbf{Sb_{FI}RTI}$, $\mathbf{Sb_{FI}RTM}$ e $\mathbf{Sb_{FI}RTMI}$.

Criamos também uma base de dados para cada problema e utilizamos os identificadores $S_{\mathbf{Sb_{FI}R}}$, $S_{\mathbf{Sb_{FI}RI}}$, $S_{\mathbf{Sb_{FI}RM}}$, $S_{\mathbf{Sb_{FI}RMI}}$, $S_{\mathbf{Sb_{FI}RT}}$, $S_{\mathbf{Sb_{FI}RTI}}$, $S_{\mathbf{Sb_{FI}RTM}}$ e $S_{\mathbf{Sb_{FI}RTMI}}$ para a base de dados dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RI}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RMI}$, $\mathbf{Sb_{FI}RT}$, $\mathbf{Sb_{FI}RTI}$, $\mathbf{Sb_{FI}RTM}$ e $\mathbf{Sb_{FI}RTMI}$, respectivamente. As bases de dados foram criadas de forma similar ao processo descrito na Seção 5.3.1, diferindo apenas pelo fato de que cada instância foi criada a partir das representações intergênicas rígida e flexível com sinais. Logo, ao aplicar um evento de reversão, os genes no segmento afetado também têm a orientação invertida.

Utilizando a estrutura de grafo de ciclos ponderado flexível e considerando todos os grupos das bases de dados $S_{\mathbf{Sb_{FI}R}}$, $S_{\mathbf{Sb_{FI}RM}}$, $S_{\mathbf{Sb_{FI}RT}}$ e $S_{\mathbf{Sb_{FI}RTM}}$, foi observado que 79.55%, 8.95% e 11.50% das instâncias pertencem ao cenário de equilíbrio, fonte e sorvedouro, respectivamente.

Para garantir uma proporcionalidade entre os possíveis cenários nos problemas que utilizam um modelo de rearranjo composto exclusivamente por eventos conservativos, criamos a bases de dados S_C. A base de dados possui cinco grupos, sendo que cada grupo possui 3000 instâncias intergênicas flexíveis balanceadas com sinais de tamanho 100 e é identificado pelo grau de flexibilização máxima das instâncias contidas nele. Os identificadores dos grupos são 10%, 20%, 30%, 40% e 50%. Utilizando a estrutura de grafo de ciclos ponderado flexível, cada grupo possui 1000 instâncias em cada cenário: equilíbrio, fonte e sorvedouro. O processo de criação de uma instância da base de dados S_C é semelhante ao utilizado pela base de dados U_C, descrito na Seção 5.3.1, diferenciando apenas pelo fato de que cada instância foi criada a partir das representações intergênicas rígida e flexível com sinais e que a operação de troca também altera a orientação dos elementos afetados. A base de dados S_C foi criada para ser utilizada pelos algoritmos da variação com sinais dos problemas $\mathbf{Sb_{FI}R}$, $\mathbf{Sb_{FI}RM}$, $\mathbf{Sb_{FI}RT}$ e $\mathbf{Sb_{FI}RTM}$.

A seguir, apresentamos os resultados obtidos pelos algoritmos propostos para a variação com sinais dos problemas investigados neste capítulo. Nas tabelas que serão utilizadas a seguir temos a informação, por grupo, do grau de flexibilização adotado e as métricas de distância e aproximação, sendo que para ambas as métricas temos a informação sobre o menor e o maior valor registrado, além da média dos valores do grupo. As colunas Distância e Aproximação indicam a quantidade de operações e o fator de aproximação para uma solução fornecida por um algoritmo.

A Tabela 5.13 apresenta os resultados do Algoritmo 41 utilizando as bases de dados $S_{\mathbf{Sb_{FI}R}}$ e S_{C}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.11.

Pela Tabela 5.13 é possível notar que o Algoritmo 41 apresentou um ótimo desempenho prático, sendo que, em todos os grupos de ambas as bases de dados, o algoritmo forneceu pelo menos uma solução ótima. Essa informação pode ser verificada pela métrica aproximação mínima com valor 1.00. Além disso, na base de dados $S_{\mathbf{Sb_{FI}R}}$ a aproximação média foi de 1.01 para todos os grupos e vale ressaltar que a razão de aproximação foi computada utilizando o limitante inferior. Considerando todas as instâncias das base de dados $S_{\mathbf{Sb_{FI}R}}$ e S_{C}, o Algoritmo 41 forneceu uma solução ótima para, pelo menos, 59.46% e 89.86% das instâncias, respectivamente. Na base de dados S_{C} o Algoritmo 41 apresentou valores mais elevados para a métrica aproximação máxima. Entretanto, o maior valor

Tabela 5.13: Resultados do Algoritmo 41 utilizando as bases de dados $S_{\mathbf{Sb_{FI}R}}$ e S_{C}.

$S_{\mathbf{Sb_{FI}R}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	45	49.98	55	1.00	1.01	1.10
20%	44	49.81	54	1.00	1.01	1.09
30%	43	49.73	56	1.00	1.01	1.12
40%	44	49.80	55	1.00	1.01	1.10
50%	45	49.74	54	1.00	1.01	1.09

S_{C}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	23	49.06	72	1.00	1.17	1.40
20%	24	50.72	78	1.00	1.19	1.49
30%	24	52.46	80	1.00	1.20	1.43
40%	24	53.89	82	1.00	1.21	1.46
50%	25	55.19	87	1.00	1.23	1.55

observado foi de 1.55, registrado no grupo 50%.

A Tabela 5.14 apresenta os resultados do Algoritmo 42 utilizando a base de dados $S_{\mathbf{Sb_{FI}RI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.12.

Tabela 5.14: Resultados do Algoritmo 42 utilizando a base de dados $S_{\mathbf{Sb_{FI}RI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	40	45.97	52	1.00	1.01	1.09
20%	40	44.63	50	1.00	1.01	1.11
30%	39	43.84	50	1.00	1.01	1.14
40%	38	42.98	50	1.00	1.01	1.14
50%	37	42.34	50	1.00	1.01	1.11

Pelos dados da Tabela 5.14 podemos notar que o Algoritmo 42 apresentou um comportamento similar ao Algoritmo 41 na base de dados $S_{\mathbf{Sb_{FI}R}}$, inclusive mantendo a mesma aproximação média para todos os grupos. É importante ressaltar que as bases de dados $S_{\mathbf{Sb_{FI}R}}$ e $S_{\mathbf{Sb_{FI}RI}}$ foram geradas através de um conjunto distinto de operações. Considerando todas as instâncias da base de dados $S_{\mathbf{Sb_{FI}RI}}$, o Algoritmo 42 forneceu uma solução ótima para pelo menos 62.60% das instâncias.

A Tabela 5.15 apresenta os resultados do Algoritmo 43 utilizando as bases de dados $S_{\mathbf{Sb_{FI}RM}}$ e S_{C}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.13.

Observando os resultados do Algoritmo 43, reportados na Tabela 5.15, podemos perceber que na base de dados $S_{\mathbf{Sb_{FI}RM}}$ a aproximação mínima para o grupo 10% foi de 1.02 enquanto para os demais grupos esse valor foi de 1.00. Considerando todas as instâncias

Tabela 5.15: Resultados do Algoritmo 43 utilizando as bases de dados $S_{\mathbf{Sb_{FI}RM}}$ e S_{C}.

$S_{\mathbf{Sb_{FI}RM}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	41	49.50	57	1.02	1.10	1.23
20%	40	47.48	55	1.00	1.08	1.19
30%	39	45.74	53	1.00	1.07	1.22
40%	38	44.68	53	1.00	1.05	1.18
50%	37	43.66	50	1.00	1.04	1.19

S_{C}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	23	40.14	53	1.00	1.13	1.29
20%	24	41.01	56	1.00	1.14	1.35
30%	24	41.82	57	1.00	1.15	1.32
40%	24	42.46	57	1.00	1.16	1.33
50%	25	42.94	58	1.00	1.16	1.32

da base de dados $S_{\mathbf{Sb_{FI}RM}}$, o Algoritmo 43 forneceu uma solução ótima para pelo menos 2.42% das instâncias. Entretanto, quando consideramos a base de dados S_{C}, essa porcentagem sobe para 29.97%. Considerando ambas as bases de dados e todos os grupos, a aproximação média foi inferior a 1.17, enquanto a aproximação máxima não ultrapassou 1.35.

A Tabela 5.16 apresenta os resultados do Algoritmo 44 utilizando a base de dados $S_{\mathbf{Sb_{FI}RMI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.14.

Tabela 5.16: Resultados do Algoritmo 44 utilizando a base de dados $S_{\mathbf{Sb_{FI}RMI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	39	48.31	55	1.02	1.11	1.21
20%	40	46.50	55	1.01	1.09	1.21
30%	39	45.19	53	1.00	1.07	1.20
40%	38	44.21	51	1.00	1.06	1.19
50%	37	43.50	50	1.00	1.05	1.15

Pela Tabela 5.16 é possível notar que a maior aproximação máxima observada foi de 1.21, ocorrência registrada nos grupos 10% e 20%. Inclusive, esses dois grupos foram os únicos em que a aproximação mínima foi maior que 1.00. Considerando todas as instâncias da base de dados $S_{\mathbf{Sb_{FI}RMI}}$, em pelo menos 25.00% delas o Algoritmo 44 forneceu uma solução ótima. Vale mencionar que essa quantidade pode ser maior, uma vez que tal informação é derivada da razão de aproximação obtida em cada instância da base de dados, que é computada utilizando o limitante inferior. Outro ponto interessante dos resultados é que houve uma diminuição significativa na distância média à medida que o

grau de flexibilização aumentou nos grupos.

A Tabela 5.17 apresenta os resultados do Algoritmo 45 utilizando as bases de dados $S_{\mathbf{Sb_{FI}RT}}$ e S_{C}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.15.

Tabela 5.17: Resultados do Algoritmo 45 utilizando as bases de dados $S_{\mathbf{Sb_{FI}RT}}$ e S_{C}.

$S_{\mathbf{Sb_{FI}RT}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	53	62.70	74	1.62	1.87	2.09
20%	52	62.86	76	1.64	1.88	2.12
30%	52	63.06	75	1.68	1.90	2.15
40%	53	63.58	74	1.68	1.92	2.16
50%	51	64.36	76	1.65	1.94	2.21

S_{C}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	21	39.26	54	1.53	1.91	2.20
20%	19	40.13	55	1.40	1.92	2.15
30%	22	40.98	58	1.52	1.92	2.20
40%	22	41.61	58	1.52	1.92	2.21
50%	22	42.22	60	1.59	1.93	2.30

Pelos dados da Tabela 5.17 é possível observar que a razão de aproximação média do Algoritmo 45 já é significativamente maior em comparação com a dos algoritmos apresentados anteriormente. Uma possível explicação para tal comportamento é o fato da inclusão do evento de transposição no modelo de rearranjo, impactando diretamente no limitante inferior. Entretanto, nas bases de dados $S_{\mathbf{Sb_{FI}RT}}$ e S_{C} a aproximação média para todos os grupos foi menor que 1.95 enquanto a maior razão de aproximação observada foi de 2.30, um valor significativamente menor do que a aproximação teórica garantida pelo algoritmo (3-aproximação). Considerando a variação da aproximação média entre os grupos da base de dados $S_{\mathbf{Sb_{FI}RT}}$ e S_{C}, obtemos os valores de 0.07 e 0.02, respectivamente. Essa característica é um bom indício de estabilidade do Algoritmo 45, uma vez que as bases de dados $S_{\mathbf{Sb_{FI}RT}}$ e S_{C} foram construídas de maneira totalmente diferente, mas visando abranger um cenário baseado em eventos de rearranjo e um cenário aleatório mantendo uma proporção entre os diferentes casos de redução com os quais o algoritmo pode deparar-se.

A Tabela 5.18 apresenta os resultados do Algoritmo 46 utilizando a base de dados $S_{\mathbf{Sb_{FI}RTI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.12.

Na Tabela 5.18 podemos observar que a aproximação média do Algoritmo 46, em todos os grupos, foi levemente superior em comparação com o resultado do Algoritmo 45 utilizando a base de dados $S_{\mathbf{Sb_{FI}RT}}$. O valor da aproximação média variou de 2.01 até 2.02. Entretanto, a variação entre a menor aproximação mínima e a maior aproximação máxima

Tabela 5.18: Resultados do Algoritmo 46 utilizando a base de dados $S_{\mathbf{Sb_{FI}RTI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	53	60.90	70	1.93	2.01	2.17
20%	52	59.96	68	1.93	2.02	2.14
30%	49	59.22	67	1.93	2.02	2.15
40%	49	58.73	66	1.93	2.02	2.13
50%	49	58.30	67	1.93	2.02	2.20

foi de apenas 0.27, sendo a menor aproximação mínima de 1.93, em todos os grupos, e a maior aproximação máxima de 2.20, no grupo 50%. É importante notar também que nos grupos 30%, 40% e 50% o Algoritmo 46 forneceu, para pelo menos uma instância de cada grupo, uma solução utilizando menos operações que a quantidade utilizada para gerar as instâncias da base de dados (50 operações).

A Tabela 5.19 apresenta os resultados do Algoritmo 47 utilizando as bases de dados $S_{\mathbf{Sb_{FI}RTM}}$ e S_{C}. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.15.

Tabela 5.19: Resultados do Algoritmo 47 utilizando as bases de dados $S_{\mathbf{Sb_{FI}RTM}}$ e S_{C}.

$S_{\mathbf{Sb_{FI}RTM}}$						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	52	62.60	71	1.83	1.97	2.03
20%	51	61.30	68	1.90	1.97	2.04
30%	52	60.21	69	1.89	1.97	2.09
40%	50	59.35	66	1.90	1.97	2.07
50%	49	58.87	66	1.89	1.97	2.08

S_{C}						
-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	23	40.02	52	1.75	1.96	2.08
20%	24	40.89	55	1.81	1.96	2.07
30%	24	41.70	57	1.84	1.96	2.08
40%	24	42.32	57	1.85	1.97	2.07
50%	25	42.84	59	1.85	1.97	2.07

Pelos dados da Tabela 5.19 nota-se que, na base de dados $S_{\mathbf{Sb_{FI}RTM}}$, não houve variação na aproximação média fornecida pelo Algoritmo 47 entre os grupos. Já na base de dados S_{C}, a aproximação média variou entre 1.96 e 1.97. A aproximação máxima em ambas as bases de dados foi inferior a 2.10. Nos grupos 40% e 50% da base de dados $S_{\mathbf{Sb_{FI}RTM}}$, o Algoritmo 47 forneceu, para pelo menos uma instância de cada grupo, uma solução com a mesma quantidade ou menos operações que a utilizada para gerar a instância.

A Tabela 5.20 apresenta os resultados do Algoritmo 48 utilizando a base de dados

$S_{\mathbf{Sb_{FI}RTMI}}$. A razão de aproximação obtida pelo algoritmo para cada instância foi computada utilizando o limitante inferior apresentado no Teorema 5.2.12.

Tabela 5.20: Resultados do Algoritmo 48 utilizando a base de dados $S_{\mathbf{Sb_{FI}RTMI}}$.

-	Distância			Aproximação		
Flexibilização	Mínimo	Média	Máximo	Mínimo	Média	Máximo
10%	54	62.25	70	1.94	2.01	2.13
20%	53	61.21	69	1.90	2.01	2.17
30%	52	60.28	68	1.93	2.01	2.13
40%	51	59.56	70	1.93	2.01	2.15
50%	50	59.05	66	1.96	2.01	2.13

Segundo o resultado apresentado na Tabela 5.20, é possível notar que não houve variação na aproximação média fornecida pelo Algoritmo 48 entre os grupos da base de dados $S_{\mathbf{Sb_{FI}RTMI}}$. Além disso, a variação entre a menor e a maior razão de aproximação observada foi de apenas 0.27, sendo a menor aproximação igual a 1.90, no grupo 20%, e a maior aproximação igual a 2.17, no mesmo grupo. Por fim, observa-se uma queda significativa na distância média à medida que o grau de flexibilização aumenta nos grupos.

Em geral, todos os algoritmos apresentaram um resultado prático significativamente melhor que o limitante teórico garantido para cada um deles. Vale ressaltar que, dentre os algoritmos em que o evento de transposição não pertence ao modelo de rearranjo, a maior razão de aproximação observada foi 1.35. Para os demais algoritmos, a maior razão de aproximação foi 2.30. Além disso, com base no limitante inferior, foi possível mostrar que os algoritmos 41, 42 e 43 forneceram uma solução ótima para uma quantidade significativa de instâncias. Vale mencionar que a distância flexível não é conhecida para as instâncias das bases de dados utilizadas. Dessa forma, a quantidade de instâncias em que os algoritmos forneceram uma solução ótima pode ser ainda maior.

5.4 Conclusões

Neste capítulo, estudamos uma generalização dos problemas que consideram tanto a ordem dos genes como o tamanho das regiões intergênicas. Nessa versão generalizada, adicionamos um grau de flexibilidade em relação ao tamanho das regiões intergênicas desejadas no genoma alvo. Para isso, nos modelos propostos, chamados de modelos intergênicos flexíveis, é possível especificar um intervalo de valores permitidos para o tamanho de cada região intergênica no genoma alvo.

O grau de flexibilidade possibilita agregar uma importância maior para a ordem e orientação dos genes, em comparação com o tamanho das regiões intergênicas. Considerando um intervalo de valores permitidos para o tamanho de cada região intergênica no genoma alvo, ampliou-se as possibilidades de configuração para os tamanhos das regiões intergênicas de modo que todas as restrições do modelo fossem atendidas.

Para as variações com e sem sinais dos problemas investigados neste capítulo, apresentamos algoritmos de aproximação com fator constante. Os algoritmos foram derivados de um processo de redução que permite o uso de resultados que foram apresentados para as

respectivas versões rígidas dos problemas. Por fim, realizamos testes computacionais para verificar o desempenho prático dos algoritmos em diferentes cenários de flexibilização.

Capítulo 6

Considerações Finais

Nesta tese apresentamos os resultados obtidos durante o período do doutorado. De maneira geral, os resultados aqui apresentados incorporam novas características aos problemas de rearranjo de genomas, que vão desde a adição de uma nova restrição de proporção entre a quantidade de um tipo de evento em relação ao tamanho da sequência de eventos de rearranjo em uma solução, até a inclusão de novas estruturas genéticas na representação computacional adotada nos modelos. Além disso, foram investigados problemas considerando um grau de flexibilidade nas características de um genoma alvo desejado. Para a grande maioria dos problemas que investigamos, apresentamos a prova de NP-dificuldade e desenvolvemos algoritmos de aproximação. Além disso, sempre que possível, criamos mecanismos com o intuito de melhorar os resultados práticos dos algoritmos propostos. Os resultados apresentados nessa tese geraram os seguintes artigos:

- "*A New Approach for the Reversal Distance with Indels and Moves in Intergenic Regions*", em coautoria com Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias e Zanoni Dias, foi apresentado no *19th Annual Satellite Conference of RECOMB on Comparative Genomics* (RECOMB-CG), realizada em La Jolla, USA, no mês de Maio de 2022 [25].

- "*Genome Rearrangement Distance with a Flexible Intergenic Regions Aspect*", aceito para publicação no mês de Abril de 2022 na revista *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [20]. Uma versão preliminar deste artigo foi apresentada na *7th-8th International Conference on Algorithms for Computational Biology* (AlCoB), realizada de forma virtual no mês de Novembro de 2021 [19]. Uma versão abordando instâncias sem sinais foi apresentada no *XI Latin and American Algorithms, Graphs and Optimization Symposium* (LAGOS), realizado de forma virtual no mês de Maio de 2021 [24].

- "*An Improved Approximation Algorithm for the Reversal and Transposition Distance Considering Gene order and Intergenic Sizes*", publicado no mês de Dezembro de 2021 na revista *Algorithms for Molecular Biology*, em coautoria com Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias e Zanoni Dias [23].

- "*Reversals and Transpositions Distance with Proportion Restriction*", publicado no mês de Agosto de 2021 na revista *Journal of Bioinformatics and Computational Biology*, em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [18]. Uma versão preliminar deste artigo foi apresentada no *13th Brazilian Symposium on Bioinformatics* (BSB), realizado de forma virtual no mês de Novembro de 2020 [17].

- "*Sorting by Genome Rearrangements on Both Gene Order and Intergenic Sizes*", publicado no mês de Fevereiro de 2020 na revista *Journal of Computational Biology*, em coautoria com Géraldine Jean, Guillaume Fertin, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [22]. Uma versão preliminar deste artigo foi apresentada no *15th International Symposium on Bioinformatics Research and Applications* (ISBRA), realizado em Barcelona, Espanha, no mês de Junho de 2019 [21].

Além dos artigos listados acima, que estão diretamente relacionados com essa tese, durante o doutorado contribuímos para a obtenção de outros resultados, apresentados nos seguintes artigos:

- "*A 1.375-Approximation Algorithm for Sorting by Transpositions with Faster Running Time*", em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias, apresentado no *15th Brazilian Symposium on Bioinformatic* (BSB), realizado em Búzios, Brasil, no mês de Setembro de 2022 [2].

- "*Reversal and Indel Distance with Intergenic Region Information*", aceito para publicação no mês de Outubro de 2022 na revista *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [3]. Uma versão preliminar deste artigo foi apresentada na *7th-8th International Conference on Algorithms for Computational Biology* (AlCoB), realizada de forma virtual no mês de Novembro de 2021 [1].

- "*Sorting Permutations by Intergenic Operations*", publicado no mês de Novembro de 2021 na revista *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, em coautoria com Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Ulisses Dias e Zanoni Dias [59]. Uma versão preliminar deste artigo foi apresentada na *7th-8th International Conference on Algorithms for Computational Biology* (AlCoB), realizada de forma virtual no mês de Novembro de 2021 [58].

- "*Sorting Signed Permutations by Intergenic Reversals*", publicado no mês de Novembro de 2021 na revista *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, em coautoria com Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Laurent Bulteau, Ulisses Dias e Zanoni Dias [57].

- "*Heuristics for Genome Rearrangement Distance with Replicated Genes*", publicado no mês de Novembro de 2021 na revista *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, em coautoria com Gabriel Siqueira, Ulisses Dias

e Zanoni Dias [68]. Uma versão preliminar deste artigo foi apresentada na *7th-8th International Conference on Algorithms for Computational Biology* (AlCoB), realizada de forma virtual no mês de Novembro de 2021 [67].

- "*On the Complexity of Sorting by Reversals and Transpositions Problem*", publicado no mês de Novembro de 2019 na revista *Journal of Computational Biology*, em coautoria com Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [55].

- "*Block-Interchange Distance Considering Intergenic Regions*", em coautoria com Ulisses Dias, Andre Rodrigues Oliveira e Zanoni Dias, apresentado no *12th Brazilian Symposium on Bioinformatics* (BSB), realizado em Fortaleza, Brasil, no mês de Outubro de 2019 [38].

A partir das contribuições apresentadas, é possível mencionar algumas possibilidades de trabalhos futuros: (i) considerando problemas com restrição de proporção entre operações, é possível incorporar uma representação intergênica aos modelos. Além disso, é possível investigar outras restrições de proporção considerando mais eventos de rearranjo; (ii) para os modelos que consideram a informação referente aos genes e ao tamanho das regiões intergênicas, é possível adicionar eventos de rearranjo não conservativos, que afetam tanto os genes como as regiões intergênicas; (iii) outra opção é considerar uma representação intergênica que permita múltiplas cópias de um gene; (iv) por fim, uma possível linha de investigação seria o estudo de novos algoritmos visando obter melhores resultados práticos ou teóricos para os problemas aqui investigados. Essas são algumas das possibilidades de investigação de trabalhos futuros que podem permitir uma aplicação dos resultados de maneira mais direta em genomas reais.

Referências Bibliográficas

[1] Alexsandro Oliveira Alexandrino, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversal Distance on Genomes with Different Gene Content and Intergenic Regions Information. In *Proceedings of the 8th International Conference on Algorithms for Computational Biology (AlCoB'2021)*, volume 12715, pages 121–133. Springer International Publishing, 2021.

[2] Alexsandro Oliveira Alexandrino, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. A 1.375-Approximation Algorithm for Sorting by Transpositions with Faster Running Time. In *Proceedings of the 15th Brazilian Symposium on Bioinformatics (BSB'2022)*, volume 13523 of *Lecture Notes in Computer Science*, pages 147–157, 2022.

[3] Alexsandro Oliveira Alexandrino, Klairton Lima Brito, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversal and Indel Distance with Intergenic Region Information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–13, 2022.

[4] Alexsandro Oliveira Alexandrino, Carla Negri Lintzmayer, and Zanoni Dias. Approximation Algorithms for Sorting Permutations by Fragmentation-Weighted Operations. In *Proceedings of the 5th International Conference on Algorithms for Computational Biology (AlCoB'2018)*, volume 10849, pages 53–64. Springer International Publishing, Heidelberg, Germany, 2018.

[5] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. On the Complexity of Some Variations of Sorting by Transpositions. *Journal of Universal Computer Science*, 26(9):1076–1094, 2020.

[6] 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.

[7] Martin Bader. Sorting by Reversals, Block Interchanges, Tandem Duplications, and Deletions. *BMC Bioinformatics*, 10(1):1–10, 2009.

[8] Martin Bader, Mohamed I. Abouelhoda, and Enno Ohlebusch. A Fast Algorithm for the Multiple Genome Rearrangement Problem with Weighted Reversals and Transpositions. *BMC Bioinformatics*, 9(1):1–13, 2008.

[9] Martin Bader and Enno Ohlebusch. Sorting by Weighted Reversals, Transpositions, and Inverted Transpositions. *Journal of Computational Biology*, 14(5):615–636, 2007.

[10] Vineet Bafna and Pavel A. Pevzner. Sorting Permutations by Transpositions. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA' 1995)*, pages 614–623, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

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

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

[13] Anne Bergeron. A Very Elementary Presentation of the Hannenhalli-Pevzner Theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.

[14] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002)*, volume 2461 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany, 2002.

[15] 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.

[16] Priscila Biller, Carole Knibbe, Guillaume Beslon, and Eric Tannier. Comparative Genomics on Artificial Life. In *Pursuit of the Universal*, pages 35–44. Springer International Publishing, 2016.

[17] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Sorting by Reversals and Transpositions with Proportion Restriction. In *Proceedings of the 13th Brazilian Symposium on Bioinformatics (BSB'2020)*, pages 117–128. Springer International Publishing, 2020.

[18] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversals and Transpositions Distance with Proportion Restriction. *Journal of Bioinformatics and Computational Biology*, 19(04):2150013, 2021.

[19] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Reversals Distance Considering Flexible Intergenic Regions Sizes. In *Proceedings of the 8th International Conference on Algorithms for Computational Biology (AlCoB'2021)*, pages 134–145. Springer International Publishing, 2021.

[20] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Genome Rearrangement Distance with a Flexible Intergenic Regions Aspect. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–13, 2022.

[21] 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)*, pages 28–39. Springer International Publishing, 2019.

[22] 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.

[23] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. An Improved Approximation Algorithm for the Reversal and Transposition Distance Considering Gene Order and Intergenic Sizes. *Algorithms for Molecular Biology*, 16(1):1–21, 2021.

[24] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. Reversal and Transposition Distance of Genomes Considering Flexible Intergenic Regions. In *Proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium (LAGOS'2021)*, pages 21–29. Procedia Computer Science, Elsevier, 2021.

[25] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. A New Approach for the Reversal Distance with Indels and Moves in Intergenic Regions. In *Proceedings of 19th Annual Satellite Conference of RECOMB on Comparative Genomics (RECOMB-CG 2022)*, volume 13234, pages 205–220. Springer International Publishing, 2022.

[26] 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, pages 65–75. Springer International Publishing, Heidelberg, Germany, 2018.

[27] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by Transpositions is Difficult. *SIAM Journal on Discrete Mathematics*, 26(3):1148–1180, 2012.

[28] Laurent Bulteau, Guillaume Fertin, and Eric Tannier. Genome Rearrangements with Indels in Intergenes Restrict the Scenario Space. *BMC Bioinformatics*, 17(14):426, 2016.

[29] Alberto Caprara. On the Tightness of the Alternating-Cycle Lower Bound for Sorting by Reversals. *Journal of Combinatorial Optimization*, 3(2):149–182, 1999.

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

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

[32] Xin Chen, Jie Zheng, Zheng Fu, Peng Nan, Yang Zhong, Stefano Lonardi, and Tao Jiang. Assignment of Orthologous Genes via Genome Rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315, 2005.

[33] David A. Christie. Sorting Permutations by Block-Interchanges. *Information Processing Letters*, 60(4):165–169, 1996.

[34] David A. Christie. A 3/2-Approximation Algorithm for Sorting by Reversals. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'1998)*, pages 244–252, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[35] David A. Christie and Robert W. Irving. Sorting Strings by Reversals and by Transpositions. *SIAM Journal on Discrete Mathematics*, 14(2):193–206, 2001.

[36] Ulisses Dias and Zanoni Dias. Extending Bafna-Pevzner Algorithm. In *Proceedings of the 1st International Symposium on Biocomputing (ISB'2010)*, pages 1–8, New York, NY, USA, 2010. ACM.

[37] Ulisses Dias, Gustavo R. Galvão, Carla N. Lintzmayer, and Zanoni Dias. A General Heuristic for Genome Rearrangement Problems. *Journal of Bioinformatics and Computational Biology*, 12(3):26, 2014.

[38] Ulisses Dias, Andre Rodrigues Oliveira, Klairton Lima Brito, and Zanoni Dias. Block-Interchange Distance Considering Intergenic Regions. In *Proceedings of the 12th Brazilian Symposium on Bioinformatics (BSB'2019)*, volume 11347, pages 58–69. Springer International Publishing, 2019.

[39] Nadia El-Mabrouk and David Sankoff. Analysis of Gene Order Evolution Beyond Single-Copy Genes. *Evolutionary Genomics*, pages 397–429, 2012.

[40] 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.

[41] Niklas Eriksen. *Combinatorics of Genome Rearrangements and Phylogeny*. Teknologie licentiat thesis, Kungliga Tekniska Högskolan, Stockholm, 2001.

[42] Niklas Eriksen. $(1+\epsilon)$-Approximation of Sorting by Reversals and Transpositions. *Theoretical Computer Science*, 289(1):517–529, 2002.

[43] Guillaume Fertin, Géraldine Jean, and Eric Tannier. Algorithms for Computing the Double Cut and Join Distance on both Gene Order and Intergenic Sizes. *Algorithms for Molecular Biology*, 12(1):16, 2017.

[44] Guillaume Fertin, Anthony Labarre, Irena Rusu, Éric Tannier, and Stéphane Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. The MIT Press, London, England, 2009.

[45] Laurence Garczarek, Ulysse Guyet, Hugo Doré, Gregory K Farrant, Mark Hoebeke, Loraine Brillet-Guéguen, Antoine Bisch, Mathilde Ferrieux, Jukka Siltanen, Erwan Corre, Gildas Le Corguillé, Morgane Ratin, Frances D Pitt, Martin Ostrowski, Maël Conan, Anne Siegel, Karine Labadie, Jean-Marc Aury, Patrick Wincker, David J Scanlan, and Frédéric Partensky. Cyanorak v2.1: A Scalable Information System Dedicated to the Visualization and Expert Curation of Marine and Brackish Picocyanobacteria Genomes. *Nucleic Acids Research*, 49(D1):D667–D676, 2020.

[46] 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.

[47] 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.

[48] 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.

[49] Crystal L. Kahn and Benjamin J. Raphael. Analysis of Segmental Duplications via Duplication Distance. *Bioinformatics*, 24(16):i133–i138, 2008.

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

[51] Petr Kolman and Tomasz Waleń. Reversal Distance for Strings with Duplicates: Linear Time Approximation Using Hitting Set. In *Proceedings of the International Workshop on Approximation and Online Algorithms (WAOA'2006)*, pages 279–289, 2006.

[52] Guohui Lin and Tao Jiang. A Further Improved Approximation Algorithm for Breakpoint Graph Decomposition. *Journal of Combinatorial Optimization*, 8(2):183–194, 2004.

[53] Vladimir Makarenkov and Bruno Leclerc. Tree Metrics and Their Circular Orders: Some Uses for the Reconstruction and Fitting of Phylogenetic Trees. *Mathematical Hierarchies and Biology, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 37:183–208, 1997.

[54] Aniket C. Mane, Manuel Lafond, Pedro C. Feijao, and Cedric Chauve. The Distance and Median Problems in the Single-Cut-or-Join Model with Single-Gene Duplications. *Algorithms for Molecular Biology*, 15(1):1–14, 2020.

[55] Andre Rodrigues Oliveira, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. On the Complexity of Sorting by Reversals and Transpositions Problems. *Journal of Computational Biology*, 26:1223–1229, 2019.

[56] Andre Rodrigues Oliveira, Klairton Lima Brito, Zanoni Dias, and Ulisses Dias. Sorting by Weighted Reversals and Transpositions. *Journal of Computational Biology*, 26:420–431, 2019.

[57] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Klairton Lima Brito, Laurent Bulteau, Ulisses Dias, and Zanoni Dias. Sorting Signed Permutations by Intergenic Reversals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2870–2876, 2021.

[58] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. A 3.5-Approximation Algorithm for Sorting by Intergenic Transpositions. In *Proceedings of the 7th International Conference on Algorithms for Computational Biology (AlCoB'2020)*, pages 16–28. Springer International Publishing, 2020.

[59] Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. Sorting Permutations by Intergenic Operations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2080–2093, 2021.

[60] 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(1):1–17, 2019.

[61] Andrew J. Radcliffe, Alex D. Scott, and Elizabeth L. Wilmer. Reversals and Transpositions Over Finite Alphabets. *SIAM Journal on Discrete Mathematics*, 19(1):224–244, 2005.

[62] Atif Rahman, Swakkhar Shatabda, and Masud Hasan. An Approximation Algorithm for Sorting by Reversals and Transpositions. *Journal of Discrete Algorithms*, 6(3):449–457, 2008.

[63] Dana Shapira and James A. Storer. Edit distance with move operations. *Journal of Discrete Algorithms*, 5(2):380–392, 2007.

[64] Luiz Augusto G. Silva, Luis Antonio B. Kowada, Noraí Romeu Rocco, and Maria Emília M. T. Walter. A new 1.375-approximation algorithm for sorting by transpositions. *Algorithms for Molecular Biology*, 17(1):1–17, 2022.

[65] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, and Zanoni Dias. Signed Rearrangement Distances Considering Repeated Genes and Intergenic Regions. In *Proceedings of 14th International Conference on Bioinformatics and Computational Biology (BICoB'2022)*, volume 83, pages 31–42. EasyChair, 2022.

[66] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, and Zanoni Dias. Approximation Algorithm for Rearrangement Distances Considering Repeated Genes and Intergenic Regions. *Algorithms for Molecular Biology*, 16(1):1–23, 2021.

[67] Gabriel Siqueira, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. Heuristics for Reversal Distance Between Genomes with Duplicated Genes. In *Proceedings of the 7th International Conference on Algorithms for Computational Biology (AlCoB'2020)*, pages 29–40. Springer International Publishing, 2020.

[68] Gabriel Siqueira, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. Heuristics for Genome Rearrangement Distance With Replicated Genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2094–2108, 2021.

[69] Eric Tannier, Anne Bergeron, and Marie-France Sagot. Advances on Sorting by Reversals. *Discrete Applied Mathematics*, 155(6-7):881–888, 2007.

[70] Damien M. De Vienne, Tatiana Giraud, and Olivier C. Martin. A Congruence Index for Testing Topological Similarity Between Trees. *Bioinformatics*, 23(23):3119–3124, 2007.

[71] 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.

[72] Li-Gen Wang, Tommy Tsan-Yuk Lam, Shuangbin Xu, Zehan Dai, Lang Zhou, Tingze Feng, Pingfan Guo, Casey W. Dunn, Bradley R. Jones, Tyler Bradley, Huachen Zhu, Yi Guan, Yong Jiang, and Guangchuang Yu. Treeio: An R Package for Phylogenetic Tree Input and Output with Richly Annotated and Associated Data. *Molecular Biology and Evolution*, 37(2):599–603, 2020.

[73] Eyla Willing, Simone Zaccaria, Marília DV Braga, and Jens Stoye. On the Inversion-Indel Distance. *BMC Bioinformatics*, 14:S3, 2013.

[74] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient Sorting of Genomic Permutations by Translocation, Inversion and Block Interchange. *Bioinformatics*, 21(16):3340–3346, 2005.