Universidade Estadual de Campinas
Instituto de Computação

Klairton de Lima Brito

Modelos com Proporção entre Operações e Regiões Intergênicas Rígidas e Flexíveis

CAMPINAS
1500

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 da Tese entregue à banca antes da defesa.

CAMPINAS
1500

Na versão final esta página será substituída pela ficha catalográfica.

De acordo com o padrão da CCPG: “Quando se tratar de Teses e Dissertações financiadas por agências de fomento, os beneficiados deverão fazer referência ao apoio recebido e inserir esta informação na ficha catalográfica, além do nome da agência, o número do processo pelo qual recebeu o auxílio.”
e
“caso a tese de doutorado seja feita em Cotutela, será necessário informar na ficha catalográfica o fato, a Universidade convenente, o país e o nome do orientador.”

Na versão final, esta página será substituída por outra informando a composição da banca e que a ata de defesa está arquivada pela Unicamp.

Agradecimentos

Os agradecimentos devem ocupar uma única página.

Resumo

O resumo deve ter no máximo 500 palavras e deve ocupar uma única página.

Abstract

The abstract must have at most 500 words and must fit in a single page.

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. Ao decorrer do tempo mudanças podem ocorrer nos organismos, que refletem adaptações desenvolvidas para melhor se adequar e prosperar no ambiente que estão inseridos. Em particular, 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 que podem ser pontuais ou afetar grandes trechos do genoma, que são chamadas de eventos de rearrajos de genomas. Tais eventos podem afetar o genoma modificando, inserindo ou removendo material genético [41]. Uma das formas bem aceita de estimar a proximidade de dois organismos é determinando uma sequência de eventos de rearranjos de genomas com tamanho mínimo e 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 [45, 26, 24, 52]. 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 [66, 36, 46, 51, 4], 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 (strings), 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 de 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 omitido 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 por representção 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 π qualquer em uma permutação específica na qual os elementos encontram-se ordenados de maneira crescente e com sinal positivo para o caso com sinais, essa permutação é chamada de identidade.

Quando consideramos um modelo de rearranjo composto apenas pelo evento de reversão e utilizando uma representação clássica com sinais, temos a variação com sinais do problema de Ordenação de Permutações por Reversões (**SbR**). Hannenhalli e Pevzner [45] apresentaram o primeiro algoritmo exato polinomial para o problema, sendo posteriormente simplificado por Bergeron [10]. Atualmente, temos um algoritmo com complexidade subquadrática para determinar a sequência de reversões capaz de ordenar uma permutação com sinais [63]. Entretanto, se estivermos interessados somente na distância de reversão, existe um algoritmo que executa em tempo linear [3]. Entretanto, quando consideramos uma representação clássica sem sinais, temos a variação sem sinais do problema **SbR**. Caprara [26] provou que o problema faz parte da classe de problemas NP-Difícil. Um dos primeiros algoritmos para o problema apresentou um fator de aproximação 1.75 [8]. Em seguida, Christie [30] apresentou um algoritmo com fator de aproximação 1.5. Atualmente, o melhor algoritmo para o problema apresenta um fator de aproximação 1.375 [11].

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. O problema também pertence à classe de problemas NP-Difícil, sendo a prova apresentada por Bulteau *et al.* [24]. O primeiro algoritmo para o problema foi proposto por Bafna e Pevzner [9] com um fator de aproximação 1.5. Atualmente, o melhor algoritmo para o problema apresenta um fator de aproximação 1.375 [37, 60] e heurísticas foram apresentadas por Dias e Dias [33] 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, temos o problema de Ordenação de Permutações por Block-Interchanges. Em 1996, foi apresentado um algoritmo exato polinomial para o problema [29].

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 (**SbRT**). O problema possui a variação com e sem sinais e ambas pertencem à classe de problemas NP-Difícil [52]. Os melhores algoritmos para o problema apresentam fatores de aproximação 2 [64] e $2k$ [59] (onde k é o fator de aproximação para a decomposição de ciclos [27]) para as variações com e sem sinais, respectivamente. Diversas heurísticas considerando esses problemas foram apresentadas

na literatura [34, 23].

Além dos eventos de reversão e transposição, podemos mencionar ainda o evento de rearranjo *double cut and join*(DJC) [67], que atua fragmentando 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 considerando uma representação clássica de um genoma e adotando um modelo de rearranjo composto pela combinação dos eventos de reversão, transposição e DCJ.

Tabela 1.1: Resultados conhecidos considerando uma representação clássica e diferentes modelos de rearranjo.

Representação Clássica com Sinais		
Modelo	Complexidade	Aproximação
DCJ	P [67]	Exato [67]
Reversão	P [45]	Exato [45]
Reversão e Transposição	NP-difícil [52]	2 [64]

Representação Clássica sem Sinais		
Modelo	Complexidade	Aproximação
DCJ	NP-difícil [27]	$\frac{17}{12} + \epsilon$ [27]
Reversão	NP-difícil [26]	1.375 [11]
Transposição	NP-difícil [24]	1.375 [37, 60]
Block-Interchange	P [29]	Exato [29]
Reversão e Transposição	NP-difícil [52]	$2k$ [59, 27]

Quando passamos a considerar uma representação por strings, em 2001, Christie e Irving [31] 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 (os caracteres das strings comparadas pertencem ao conjunto $\{0, 1\}$). Para isso, os autores apresentaram uma redução do problema 3-partition [44]. Em 2005, Radcliffe *et al.* [58] 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 que os autores caracterizaram 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.* [28]. Com essa relação entre os problemas, foi apresentado por Kolman e Waleń [48] 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.

A representação do genoma como uma sequência ordenada 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 [12, 13]. 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 também consideram os tamanhos das regiões intergênicas começaram a ser apresentados. Assumindo que em um genoma existe uma cópia de cada gene, então sua representação computacional pode ser dada por uma permutação (com ou sem sinais), que repesenta a ordem e a orientação dos genes, e uma lista ordenada de números naturais representando o tamanho de cada região intergência do genoma. Essa representação é chamada de intergênica.

Fertin *et al.* [40], adotando uma representação intergênica com sinais, apresentaram um modelo composto pelo evento de rearranjo DCJ, mostraram que o problema pertence à classe de problemas NP-Difícil e desenvolveram um algoritmo de aproximação com fator 4/3. Bulteau *et al.* [25] 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 ($\mathbf{Sb_IR}$). Aa variações com e sem sinais do problema pertencem à classe de problemas NP-Difícil [54, 19], sendo que os melhores algoritmos conhecidos para o problema possuem fatores de aproximação de 2 [54] e 4 [19] 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 ($\mathbf{Sb_IT}$). O problema também pertence à classe de problemas NP-Difícil [56], sendo que o melhor algoritmo conhecido para o problema possui um fator de aproximação de 3.5 [56]. Considerando um modelo composto exclusivamente pelo evento de block-interchange, temos o problema de Ordenação de Permutações por Block-Interchanges Intergênicos ($\mathbf{Sb_IBI}$). A complexidade do problema ainda permanece desconhecida e o melhor algoritmo para o problema possui um fator de aproximação 2 [35].

Quando utilizamos um modelo composto pelos eventos de reversão e transposição e adotando uma representação intergênica, temos o problema de Ordenação de Permutações por Operações Intergênicas de Reversão e Transposição ($\mathbf{Sb_IRT}$). As variações com e sem sinais do problema pertencem à classe de problemas NP-Difícil [56, 19], sendo que os melhores algoritmos conhecidos para o problema possuem fatores de aproximação de 3 [56] e 4 [20] para as variações com e sem sinais, respectivamente.

No contexto de um cenário em que é adotado a representação intergênica, Oliveira *et al.* [56] introduziram o evento de rearranjo chamado de *move*. Esse evento é similar ao evento de transposição, mas um dos segmentos afetado 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 ($\mathbf{Sb_ITM}$) e Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Move ($\mathbf{Sb_IRTM}$). Para o problema $\mathbf{Sb_ITM}$ os autores apresentaram um algoritmo de aproximação com fator 2.5 [56]. Para as variações com e sem sinais do problema $\mathbf{Sb_IRTM}$ existem algoritmos de aproximação com fatores 2.5 [56] e 3 [20], respectivamente.

Permitindo o uso dos eventos de reversão e move, temos o problema de Ordenação de Permutações por Operações Intergênicas de Reversão e Move ($\mathbf{Sb_IRM}$). A variação com

sinais desse problema pertencem à classe de problemas NP-Difícil e o melhor algoritmo para o problema possui um fator de aproximação 2 [22].

Considerando uma restrição adicional no número máximo de genes afetados pelos eventos de reversão e transposição, Oliveira *et al.* [57], 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 conhecidos considerando uma representação intergênica de um genoma.

Tabela 1.2: Resultados conhecidos considerando uma representação intergênica e diferentes modelos de rearranjo.

Representação Intergênica com Sinais		
Modelo	Complexidade	Aproximação
DCJ	NP-difícil [40]	$\frac{4}{3}$ [40]
DCJ, Inserção e Deleção	P [25]	Exato [25]
Reversão	NP-difícil [54]	2 [54]
Reversão (Super Curta)	Desconhecida	5 [57]
Reversão e Move	NP-difícil [22]	2 [22]
Reversão e Transposição	NP-difícil [56]	3 [54]
Reversão e Transposição (Super Curta)	Desconhecida	5 [57]
Reversão, Transposição e Move	NP-difícil [56]	$\frac{5}{2}$ [56]

Representação Intergênica sem Sinais		
Modelo	Complexidade	Aproximação
Reversão	NP-difícil [19]	4 [19]
Reversão (Super Curta)	Desconhecida	3 [57]
Transposição	NP-difícil [56]	$\frac{7}{2}$ [56]
Transposição (Super Curta)	Desconhecida	3 [57]
Transposição e Move	NP-difícil [56]	$\frac{5}{2}$ [56]
Block-Interchange	Desconhecida	2 [35]
Reversão e Transposição	NP-difícil [19]	4 [20]
Reversão e Transposição (Super Curta)	Desconhecida	3 [57]
Reversão, Transposição e Move	NP-difícil [20]	3 [20]

Esta tese está dividida da seguinte forma: O Capítulo 2 apresenta definições, conceitos e estruturas que serão utilizadas em três capítulos desta tese, e que são fundamentais para obtenção de resultados nos problemas que trabalhamos. O Capítulo 3 introduz um novo problema de rearranjo onde uma restrição de proporção entre operações é adicionada ao modelo. Os capítulos 4 e 5 investigam problemas em que a informação referente ao tamanho das regiões intergênicas é levada em consideração. Por fim, o Capítulo 6 apresenta as conclusões finais desta tese.

Capítulo 2

Definições

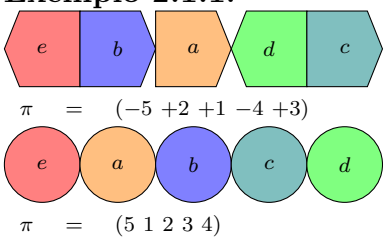
Nesse capítulo, apresentamos as formas como representamos um genoma e como os eventos de rearrajo de genomas podem afetá-los. Além disso, definimos o formato das instâncias que serão utilizadas pelos problemas investigados nos capítulos seguintes e apresentamos definições, conceitos e grafos que serão amplamente utilizados para obtenção de resultados.

2.1 Representação de Genomas

Nessa seção, apresentamos três representações de genomas que diferem nas estruturas genéticas que são 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 genome $\mathcal{G}$ seja conhecida, associamos um sinal "+" ou "−" em cada elemento π_i de π para representar a orientação de cada um dos genes de $\mathcal{G}$. Caso contrário, o sinal é simplesmente omitido. Quando representamos um genoma utilizando apenas as informações obtidas com base nas características dos genes denominamos de *representação clássica*. Além disso, denotamos por *representação clássica com sinais* quando a orientação dos genes é conhecida e *representação clássica sem sinais* caso contrário. O Exemplo 2.1.1 mostra uma representação clássica com sinais e sem sinais de genomas fictícios. Os elementos coloridos com letras no interior representam os genes, sendo que na parte supeior eles possuem orientação e na parte inferior não.

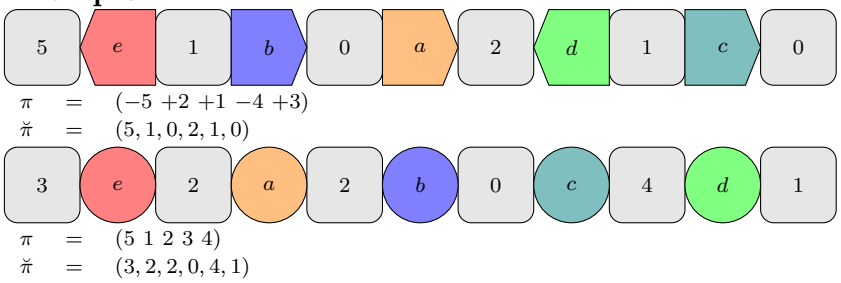
Exemplo 2.1.1.



Dado um genoma $\mathcal{G} = (\mathcal{R}_1, \mathcal{G}_1, \mathcal{R}_2, \mathcal{G}_2 \ldots, \mathcal{R}_n, \mathcal{G}_n, \mathcal{R}_{n+1})$ com n genes não repetidos $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n\}$ e $n+1$ regiões intergênicas $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{n+1}\}$, utilizamos essas duas características para representar um genoma. As regiões intergênicas estão presentes nas

extremidades do genoma e entre cada par de genes consecutivos. Cada região intergênica possui uma quantidade específica de nucleotídeos, que chamamos de *tamanho*. Dessa forma, denotamos o tamanho de uma região intergênica pela quantidade de nucleotídeos contida nela. Representamos o genoma $\mathcal{G}$ utilizando dois 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 genome $\mathcal{G}$ seja conhecida, associamos um sinal "+" ou "−" em cada elemento π_i de π para representar a orientação de cada um dos genes de $\mathcal{G}$. Caso contrário, o sinal é simplesmente omitido. O segundo elemento é uma lista de 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 $\mathcal{R}_i$ do genoma $\mathcal{G}$. Quando representamos um genoma utilizando a informação da estrutura genética dos genes e 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 *representação intergênica rígida sem sinais* caso contrário. O Exemplo 2.1.2 mostra uma representação intergênica rígida com sinais e sem sinais de genomas fictícios. Os elementos coloridos com letras no interior representam os genes, sendo que na parte supeior eles possuem orientação e na parte inferior não. Os retangulos com bordas arredondadas entre cada par de genes e nas extremidades respresentam as regiões intergênicas com o número no interior indicando seu tamanho.

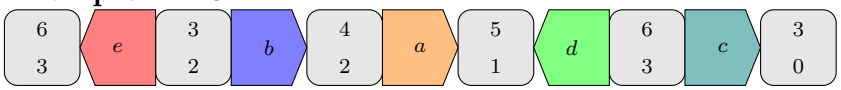
Exemplo 2.1.2.



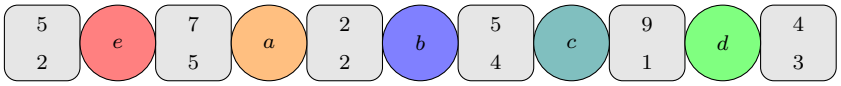
Para tornar a especificação em relação ao tamanho de cada região intergênica menos rígida, criamos uma representação denominamos de *representação intergênica flexível*. Para isso, representamos um genoma $\mathcal{G} = (\mathcal{R}_1, \mathcal{G}_1, \mathcal{R}_2, \mathcal{G}_2 \ldots, \mathcal{R}_n, \mathcal{G}_n, \mathcal{R}_{n+1})$ com n genes não repetidos $\{\mathcal{G}_1,\ \mathcal{G}_2,\ \ldots,\ \mathcal{G}_n\}$ e $n+1$ regiões intergênicas $\{\mathcal{R}_1,\ \mathcal{R}_2,\ \ldots,\ \mathcal{R}_{n+1}\}$ 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 genome $\mathcal{G}$ seja conhecida, associamos um sinal "+" ou "−" em cada elemento π_i de π para representar a orientação de cada um dos genes de $\mathcal{G}$. Caso contrário, o sinal é simplesmente omitido. Os demais elementos são duas listas de 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 $\breve{\pi}_i^{\min} \leq \mathcal{R}_i \leq \breve{\pi}_i^{\max}$, com $1 \leq i \leq n+1$. Isso faz com que o tamanho de cada região intergênica seja flexível, tornando possível especificar um intervalos 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 *representação intergênica flexível sem sinais* caso contrário. O Exemplo 2.1.3 mostra uma representação intergênica flexível com sinais e sem sinais de genomas fictícios. Os elemen-

tos coloridos com letras no interior representam os genes, sendo que na parte supeior eles possuem orientação e na parte inferior não. Os retangulos com bordas arredondadas entre cada par de genes e nas extremidades respresentam 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.



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



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

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, com $\pi_0 = 0$ e $\pi_{n+1} = n + 1$ inseridos 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, que serão apresentadas posteriormente, 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

Nessa seção, apresentamos os eventos de rearrajo considerados nessa tese e como eles podem afetar o genoma dependendo da representação que é utilizada.

Os eventos de rearranjo de genomas são classificados em eventos conservativos ou não conservativos. O 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 $\mathcal{R} \cdot \gamma$ como sendo o genoma resultante após a aplicação do evento de rearranjo γ em $\mathcal{R}$. De maneira similar, quando temos uma sequência de eventos de rerranjo $S = (\gamma_1, \gamma_2, \ldots, \gamma_k)$ 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 $\mathcal{R}$ de maneira ordenada. A seguir, mostramos como os eventos de rearranjo conservativos de reversão e transposição afetam a representação clássica de um genoma.

Definição 2.2.1. Dada uma representação clássica $\mathcal{R} = \pi$ de um genoma e sejam i e j números inteiros tal 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 $\rho^{(i,j)}$ sendo aplicada em uma representação clássica $\mathcal{R} = (\pi)$ com e sem sinais de um genoma, respectivamente.

Exemplo 2.2.1.

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

Exemplo 2.2.2.

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

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

Exemplo 2.2.3.

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

Exemplo 2.2.4.

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

Definição 2.2.2. Dada uma representação clássica $\mathcal{R} = \pi$ de um genoma e sejam i, j e k números inteiros tal 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 uma transposição $\tau^{(i,j,k)}$ sendo aplicada em uma representação clássica $\mathcal{R} = (\pi)$ de um genoma. Note que a transposição pode ser aplicada em ambas as representações clássicas, com e sem sinais.

Exemplo 2.2.5.

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

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

Exemplo 2.2.6.

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

Exemplo 2.2.7.

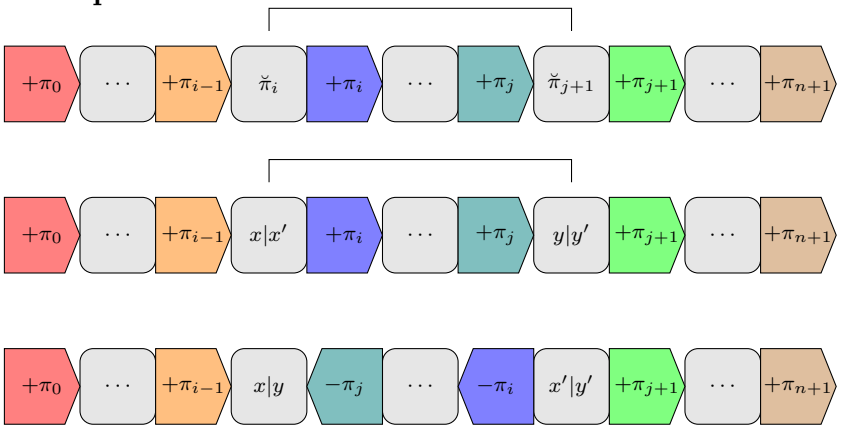
$$\begin{array}{lcl} \pi & = & (0\ 3\ 2\ 1\ \underline{5}\ \underline{4}\ 6) \\ \pi \cdot \tau^{(4,5,6)} & = & (0\ 3\ 2\ 1\ \underline{4}\ \underline{5}\ 6) \end{array}$$

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

Definição 2.2.3. Dada uma representação intergênica rígida $\mathcal{R} = (\pi, \breve{\pi})$ de um genoma e sejam i, j, x e y números inteiros tal 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 sequinte forma: $\breve{\pi}_i$ em partes com tamanho x e x', com $x' = \breve{\pi}_i - x$, e $\breve{\pi}_{j+1}$ em partes com tamanho y e y', com $y' = \breve{\pi}_{j+1} - y$. Em seguida, a sequência $(x', \pi_i, \breve{\pi}_{i+1} \dots \breve{\pi}_j, \pi_j, y)$ do genoma é invertida. Caso a representação seja com sinais, o sinal dos elementos de π_i até π_j também é invertida. 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.

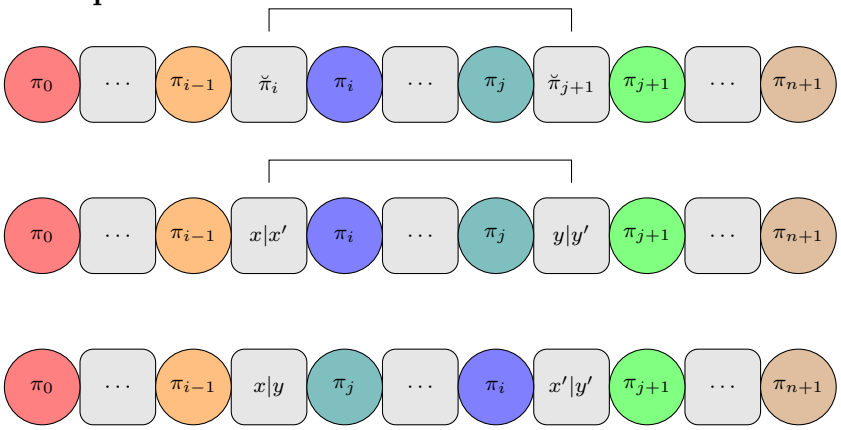
O Exemplo 2.2.8 mostra uma reversão intergênica $\rho^{(i,j)}_{(x,y)}$ genérica sendo aplicada em uma representação intergênica rígida com sinais de um genoma.

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 em uma 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 em uma 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 o tamanho alterado pelo evento, enquanto as regiões intergênicas marcadas com subscrito sofrem apenas uma troca de posição.

Exemplo 2.2.10.

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

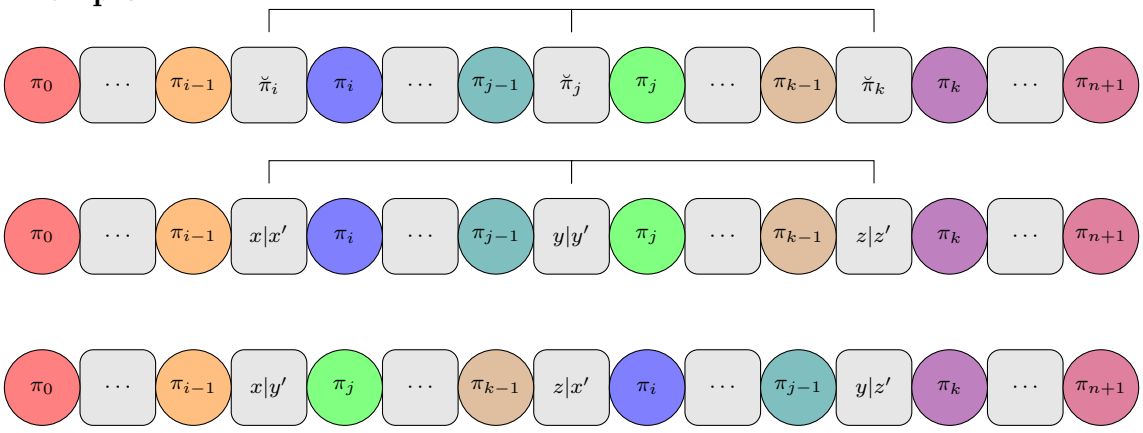
Exemplo 2.2.11.

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

Definição 2.2.4. Dada uma representação intergênica rígida $\mathcal{R} = (\pi, \breve{\pi})$ de um genoma e sejam i, j, k, x, y e z números inteiros tal 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 $\tau_{(x,y,z)}^{(i,j,k)}$ divide as regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$ da sequinte forma: $\breve{\pi}_i$ em partes com tamanho x e x', com $x' = \breve{\pi}_i - x$, $\breve{\pi}_j$ em partes com tamanho y e y', com $y' = \breve{\pi}_j - y$, e $\breve{\pi}_k$ em partes com tamanho z e z', com $z' = \breve{\pi}_k - z$. Em seguida, as sequências consecutivas $(x', \pi_i, \breve{\pi}_{i+1}, \ldots \breve{\pi}_{j-1}, \pi_{j-1}, y)$ 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 $\tau_{(x,y,z)}^{(i,j,k)}$ 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 $\tau_{(1,1,3)}^{(1,3,6)}$ sendo aplicada em uma 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 $\tau_{(0,0,1)}^{(4,5,6)}$ sendo aplicada em uma 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 o tamanho alterado pelo evento, enquanto as regiões intergênicas marcadas com subscrito sofrem apenas uma troca de posição.

Exemplo 2.2.13.

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

Exemplo 2.2.14.

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

Definição 2.2.5. Dada uma representação intergênica rígida $\mathcal{R}=(\pi,\breve{\pi})$ de um genoma e sejam i, j e x números inteiros tal que $1\leq i,j\leq n$ e $0\leq x\leq\breve{\pi}_i$. Um move intergênico $\mu_{(x)}^{(i,j)}$ 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 $\mu_{(3)}^{(2,5)}$ sendo aplicado em uma 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 indel intergênico $\mu_{(5)}^{(3,5)}$ sendo aplicado em uma 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 tamanho causado pelo evento.

Exemplo 2.2.15.

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

Exemplo 2.2.16.

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

A seguir, mostramos como o evento de rearranjo não conservativo de indel intergênico afeta a representação intergênica de um genoma.

Definição 2.2.6. Dada uma representação intergênica rígida $\mathcal{R}=(\pi,\breve{\pi})$ de um genoma e sejam i e x números inteiros tal 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 $\breve{\pi}_i$.

Note que o evento de rearranjo indel intergênico é uma forma compacta para definir os eventos de inserção e deleção utilizando a mesma notação. O Exemplo 2.2.17 mostra um indel intergênico $\delta_{(9)}^{(5)}$ sendo aplicado em uma 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 em uma 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 tamanho causado pelo evento.

Exemplo 2.2.17.

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

Exemplo 2.2.18.

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

2.3 Caracterização das Instâncias

Os problemas investigados nessa tese tem 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 realizar essa tarefa. 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 distância entre genomas buscam por a menor sequência de eventos de rearranjo $S = (\gamma_1, \gamma_2, \ldots, \gamma_k)$ pertencentes a um modelo $\mathcal{M}$ de forma que $\mathcal{R}_o \cdot S = \mathcal{R}_a$. A *distância* entre $\mathcal{R}_o$ e $\mathcal{R}_a$ no modelo $\mathcal{M}$ é o tamanho da menor sequência de eventos de rearranjo capaz de transformar $\mathcal{R}_o$ em $\mathcal{R}_a$, e é denotada por $d_{\mathcal{M}}(\mathcal{R}_o, \mathcal{R}_a)$. Os problemas de distância entre genomas assumem que cada evento de rearranjo em um modelo possui a mesma probabilidade de ocorrer em um cenário evolutivo. Entretanto, outra abordagem utiliza é associar um peso para cada tipo de evento de rearranjo pertencente a um modelo de rearranjo. Com isso, temos os problemas de distância ponderada entre genomas, que buscam por uma sequência de eventos de rearranjo $S = (\gamma_1, \gamma_2, \ldots, \gamma_k)$ pertencentes a um modelo $\mathcal{M}$ de forma que $\mathcal{R}_o \cdot S = \mathcal{R}_a$ e que o valor de $\sum_{\gamma_i \in S} p(\gamma_i)$ seja mínimo, onde $p(\gamma_i)$ representa o peso associado ao tipo do evento γ_i no modelo $\mathcal{M}$. A *distância ponderada* entre $\mathcal{R}_o$ e $\mathcal{R}_a$ no modelo $\mathcal{M}$ é o menor valor de $\sum_{\gamma_i \in S} p(\gamma_i)$ para uma sequência de eventos de rearrajo S e que $\mathcal{R}_o \cdot S = \mathcal{R}_a$, e é denotada por $dp_{\mathcal{M}}(\mathcal{R}_o, \mathcal{R}_a)$. A seguir descrevemos os tipos de instâncias que os problemas investigados posteriormente podem receber como entrada.

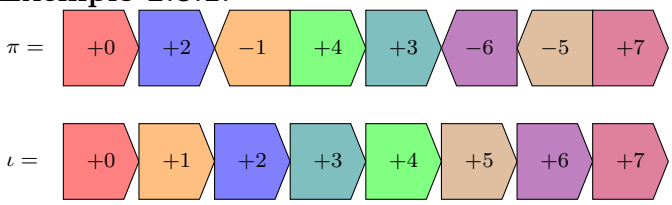
- Uma *instância clássica* é caracterizada por um par de representações clássicas de genomas (π, ι) que compartilham o mesmo conjunto de genes, sendo que ambas as representações podem ser com ou sem sinais. Por padrão, em uma instância clássica utilizaremos π e ι como sendo a representação do genoma 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 sendo a representação do genoma 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 sendo a representação do genoma de origem e alvo, respectivamente. O objetivo principal dos problemas que utilizam esse tipo de instância consiste em transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\pi}')$, tal que $\breve{\iota}_i^{\min} \leq \breve{\pi}'_i \leq \breve{\iota}_i^{\max}$, com $1 \leq i \leq n+1$.

Pelo fato de utilizarmos a representação dos genes de um genoma através de 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. O exemplo 2.3.1 mostram uma instância clássica com sinais.

Exemplo 2.3.1.



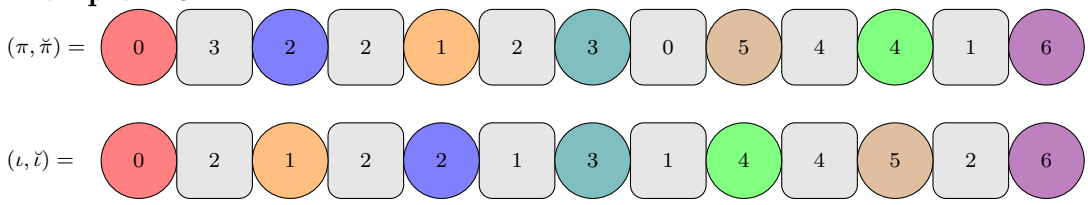
Definição 2.3.1. Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, $\mathcal{I}$ é chamada de *balanceada* se a seguinte igualdade é satisfeita:

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

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.



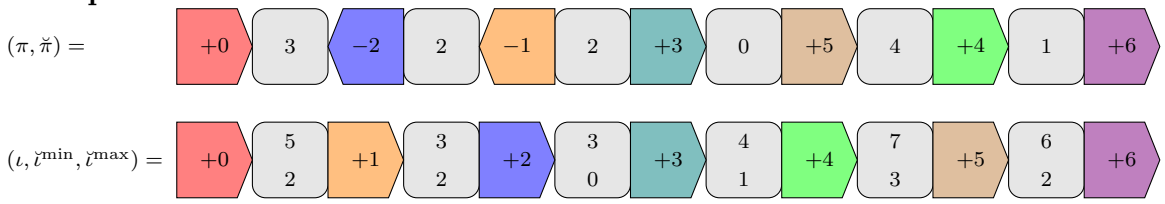
Definição 2.3.2. Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, $\mathcal{I}$ é chamada de *balanceada* se a seguinte desigualdade é satisfeita:

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

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

O exemplo 2.3.3 mostram 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 retrições 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 do genoma de origem para tornar possível transformá-lo no genoma alvo.

2.4 Breakpoints

Nessa seção, apresentamos os conceitos de breakpoints em instâncias clássicas e intergênicas rígidas. Esses conceitos são importantes para obtenção de limitantes inferiores e desenvolvimento de algoritmos para problemas que serão investigados nos capítulos seguintes.

2.4.1 Breakpoint Clássico

Nessa seção, apresentamos o conceito de breakpoint clássico.

Definição 2.4.1. 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. 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 $\pi_{i+1} - \pi_i \neq 1$.

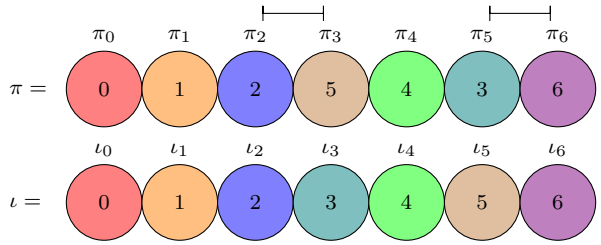
Dada uma instância clássica $\mathcal{I} = (\pi, \iota)$, o número total de breakpoints clássicos tipo um é denotado por $b_1(\mathcal{I})$. A variação no número de breakpoints clássicos tipo um após aplicar uma sequência de eventos de rearranjo S em π é denotada por $\Delta b_1(\mathcal{I}, S) = b_1(\mathcal{I}')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

Definição 2.4.3. 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 stips tenham sinal positivo. Caso contrário, a strip é definida como negativa.

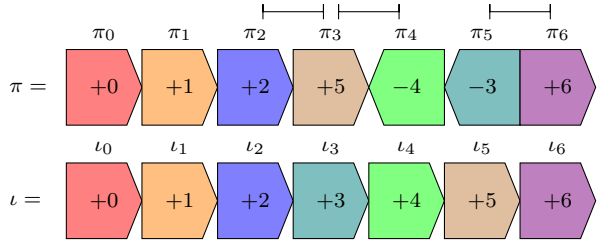
O Exemplo 2.4.1 mostra uma instância clássica sem sinais $\mathcal{I}$ = ((0 1 2 5 4 3 6), (0 1 2 3 4 5 6)). Note que a instância possui dois breakpoints clássicos tipo um ($b_1(\mathcal{I}) = 2$), sendo eles (π_2, π_3) e (π_5, π_6). Além disso, obtemos as seguintes strips da instância $\mathcal{I}$: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

Exemplo 2.4.2.



2.4.2 Breakpoint Intergênico

Nessa seção, apresentamos o conceito de breakpoint intergênico.

Definição 2.4.4. 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. 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 afeto 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 integênica entre os genes tem o mesmo tamanho no genoma origem e alvo.

Definição 2.4.6. Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e 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 no genoma de origem e alvo. Contudo, o tamanho da região intergênica entre o par de genes do genoma 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 no genoma origem e alvo, mas a região intergênica entre o par de genes do genoma origem é menor do que entre o mesmo par de genes no genoma alvo.

Definição 2.4.7. 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. 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}) são 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. 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. O par de elementos (π_i, π_{i+1}), (π_j, π_{j+1}), (π_i, π_j), (π_i, π_{j+1}), (π_{i+1}, π_j) ou (π_{i+1}, π_{j+1}) são consecutivos em ι e 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. 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. Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, *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. 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 $x = \max(|\pi_i|, |\pi_{i+1}|)$.

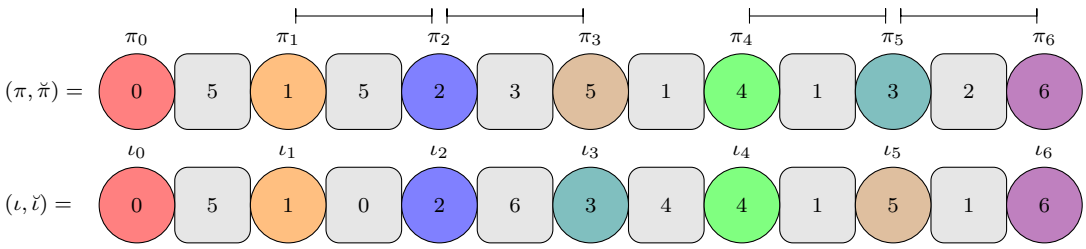
Os breakpoints intergênicos tipo um e dois são utilizados dependendo do tipo da instância intergênica rígida (com o sem sinais) e do modelo de rearranjo que é considerado, mas ambos os conceitos indicam a mesma informação: os pontos que devem ser afetados no genoma de origem para transformá-lo no genoma 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 $ib_1(\mathcal{I}) = ib_f(\mathcal{I}) + ib_s(\mathcal{I})$. A variação no número de breakpoints intergênicos tipo um após aplicar 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 $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}))$ com $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$. 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 aplicar 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 $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}))$ com $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$.

Observação 2.4.1. A única instância intergênica rígida $\mathcal{I}$ com $ib_1(\mathcal{I}) = 0$ e $ib_2(\mathcal{I}) = 0$ é $\mathcal{I} = ((\iota, \breve{\iota}), (\iota, \breve{\iota}))$.

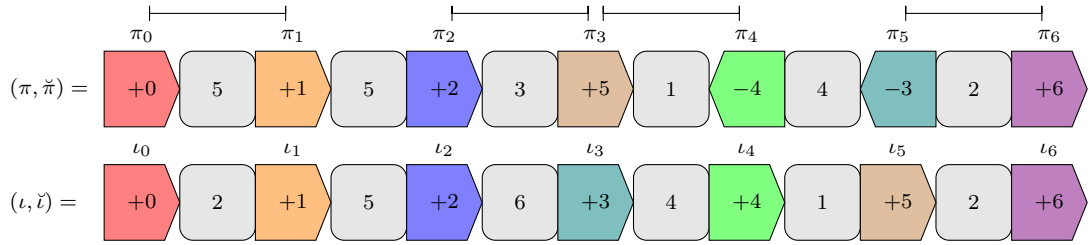
O Exemplo 2.4.3 mostra uma instância intergênica rígida sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 5, 3, 1, 1, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (5, 0, 6, 4, 1, 1)))$. Note que a instância possui quatro breakpoints intergênicos tipo um ($ib_1(\mathcal{I}) = 4$), sendo que $ib_f(\mathcal{I}) = 2$ 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

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

Definição 2.5.1. Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, uma região intergênica $\breve{\pi}_i$ é chamada de *durável* se $\breve{\iota}_k^{\min} \leq \breve{\pi}_i \leq \breve{\iota}^{\max}$, tal que $k = \max(\pi_{i-1}, \pi_i)$. Caso contrário, a região intergênica $\breve{\pi}_i$ é chamada de *temporária*.

Uma região intergênica temporária deve necessariamente ser afetada por um evento de rearranjo, seja para unir genes que são consecutivos no genoma alvo ou para alterar a quantidade de nucleotídeos na região intergênica. Os conjuntos de regiões intergênicas duráveis e temporárias são definidos como $\mathcal{D}$ e $\mathcal{T}$, respectivamente. Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, o número total de regiões intergênicas temporárias é denotado por $t(\mathcal{I})$. A variação no número de regiões intergênicas temporárias após aplicar uma sequência de eventos de rearranjo S em $(\pi, \breve{\pi})$ é denotada por $\Delta t(\mathcal{I}, S) = t(\mathcal{I}') - t(\mathcal{I})$, onde $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ com $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ e seja $\breve{\pi}_i$ uma região intergênica durável, 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 $k = \max(\pi_{i-1}, \pi_i)$. Os valores de $gap_{\min}$ e $gap_{\max}$ indicam, para cada região intergênica durável, a quantidade de nucleotídeos que podem ser removidos ou adicionados, respectivamente, ainda mantendo-a durável.

De agora em diante, as definições e conceitos que serão apresentados referem-se à instâncias intergênicas flexíveis balanceadas. Note que dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ e utilizando um modelo apenas com eventos de rearranjo conservativos, todos as regiões intergênicas temporárias precisam ser removidas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\pi}')$, tal que $\forall \breve{\pi}_i' \in \breve{\pi}', \breve{\iota}_i^{\min} \leq \breve{\pi}_i' \leq \breve{\iota}_i^{\max}$. Além disso, algumas regiões intergênicas duráveis podem ser afetadas com esse objetivo dependendo do total de nucleotídeos nas regiões intergênicas temporárias. Regiões intergênicas duráveis devem obrigatoriamente ser afetadas por algum evento de rearranjo se algum dos seguintes cenários ocorrer:

$$\texttt{(i)} \sum_{\breve{\pi}_i \in \mathcal{T}} \breve{\pi}_i < \sum_{\breve{\iota}_i^{\min} \in \breve{\iota}^{\min}} \breve{\iota}_i^{\min} - \sum_{\breve{\pi}_i \in \mathcal{D}} (\breve{\pi}_i - gap_{\min}(\breve{\pi}_i))$$

$$\texttt{(ii)} \sum_{\breve{\pi}_i \in \mathcal{T}} \breve{\pi}_i > \sum_{\breve{\iota}_i^{\max} \in \breve{\iota}^{\max}} \breve{\iota}_i^{\max} - \sum_{\breve{\pi}_i \in \mathcal{D}} (\breve{\pi}_i + gap_{\max}(\breve{\pi}_i))$$

No cenário (i), chamado de *fonte*, a quantidade de nucleotídeos nas regiões intergênicas temporárias não é suficiente para torná-las duráveis. Dessa forma, nucleotídeos das regiões intergênicas duráveis devem ser transferidos para as regiões intergênicas temporárias. No cenário (ii), chamado de *sorvedouro*, a quantidade de nucleotídeos nas regiões intergênicas temporárias excede o limite total permitido para essas regiões intergênicas. Dessa forma, nucleotídeos nucleotídeos das regiões intergênicas temporárias devem ser transferidos para as regiões intergênicas duráveis. Perceba que uma instância intergênica flexível pode não pertencer a nenhum desses cenários. Entretanto, não existe uma instância intergênica flexível que pertence aos dois cenários. Dada uma instância intergênica flexível que pertence ao cenário fonte ou sorvedouro, temos a seguinte definição.

Definição 2.5.2. Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ que pertence ao cenário fonte ou sorvedouro, uma região intergênica durável $\breve{\pi}_i$ é chamada de *auxiliar* se deve receber nucleotídeos de regiões intergênicas temporárias ou transferir nucleotídeos para regiões intergênicas temporárias.

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

$$\sum_{\breve{\pi}_i \in \mathcal{A}} gap_{\min}(\breve{\pi}_i) \geq \sum_{\breve{\iota}_i^{\min} \in \breve{\iota}^{\min}} \breve{\iota}_i^{\min} - \sum_{\breve{\pi}_i \in \mathcal{D}} (\breve{\pi}_i - gap_{\min}(\breve{\pi}_i)) - \sum_{\breve{\pi}_i \in \mathcal{T}} \breve{\pi}_i$$

Note que o conjunto $\mathcal{A}$ com tamanho mínimo pode ser facilmente obtido ordenando as regiões intergênicas duráveis em ordem decrescente por $gap_{\min}(\breve{\pi}_i)$ e rotulando-os como auxiliares até que a restrição anterior seja satisfeita. No caso do cenário sorvedouro, o conjunto de regiões intergênicas auxiliares $\mathcal{A}$ é tal que seu tamanho é mínimo e a seguinte restrição é satisfeita:

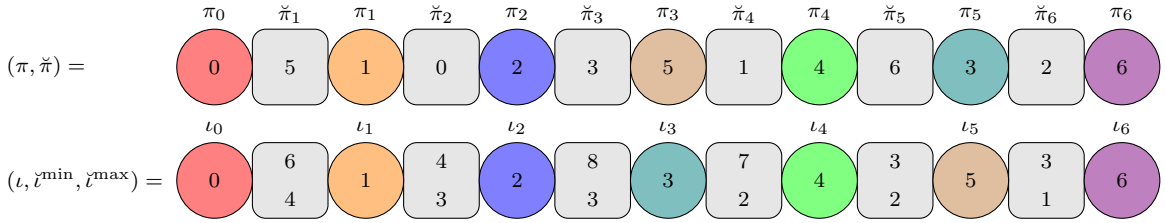
$$\sum_{\breve{\pi}_i \in \mathcal{A}} gap_{\max}(\breve{\pi}_i) \geq \sum_{\breve{\pi}_i \in \mathcal{T}} \breve{\pi}_i - \sum_{\breve{\iota}_i^{\max} \in \breve{\iota}^{\max}} \breve{\iota}_i^{\max} - \sum_{\breve{\pi}_i \in \mathcal{D}} (\breve{\pi}_i + gap_{\max}(\breve{\pi}_i))$$

Semelhante ao caso anterior, o conjunto $\mathcal{A}$ com tamanho mínimo pode ser facilmente obtido classificando as regiões intergênicas estáveis em ordem decrescente de $gap_{\max}(\breve{\pi}_i)$ e rotulando-as como auxiliares até que a restrição seja satisfeita.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, o número total de regiões intergênicas auxiliares é denotado por $a(\mathcal{I})$. A variação no número de regiões intergênicas auxiliares após aplicar uma sequência de eventos de rearranjo S em $(\pi, \breve{\pi})$ é denotada por $\Delta a(\mathcal{I}, S) = a(\mathcal{I}') - a(\mathcal{I})$, onde $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ com $(\pi', \breve{\pi}') = (\pi, \breve{\pi}) \cdot S$.

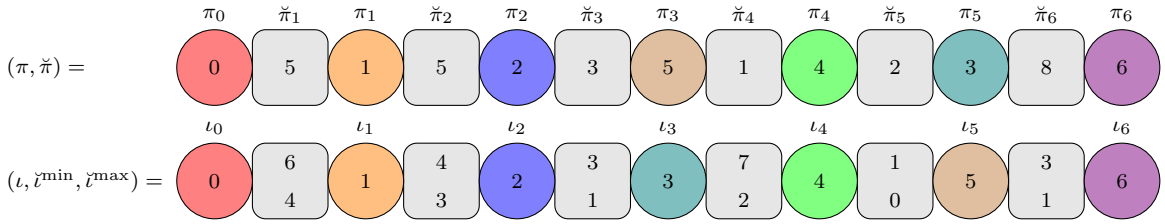
O Exemplo 2.5.1 mostra uma instância intergênica flexível sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 0, 3, 1, 6, 2)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 3, 3, 2, 2, 1), (6, 4, 8, 7, 3, 3)))$ que pertence ao cenário fonte. Note que a instância $\mathcal{I}$ possui quatro regiões intergênicas temporárias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

Exemplo 2.5.1.



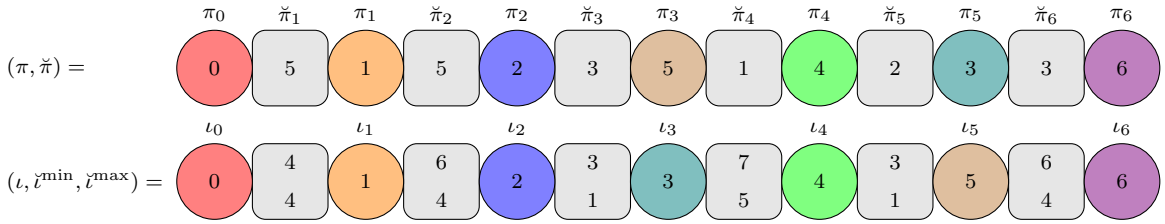
O Exemplo 2.5.2 mostra uma instância intergênica flexível sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 5, 3, 1, 2, 8)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 3, 1, 2, 0, 1), (6, 4, 3, 7, 1, 3)))$ que pertence ao cenário sorvedouro. Note que a instância $\mathcal{I}$ possui quatro regiões intergênicas temporárias ($t(\mathcal{I}) = 4$, com $\mathcal{T} = \{\breve{\pi}_2, \breve{\pi}_3, \breve{\pi}_4, \breve{\pi}_6\}$) e duas regiões intergênicas duráveis ($\mathcal{D} = \{\breve{\pi}_1, \breve{\pi}_5\}$). No exemplo, temos duas região intergênica auxiliar ($a(\mathcal{I}) = 2$, com $\mathcal{A} = \{\breve{\pi}_1, \breve{\pi}_5\}$). Note que $gap_{\max}(\breve{\pi}_1) = 1$ e $gap_{\max}(\breve{\pi}_5) = 5$.

Exemplo 2.5.2.



O Exemplo 2.5.3 mostra uma instância intergênica flexível sem sinais $\mathcal{I} = (((0\ 1\ 2\ 5\ 4\ 3\ 6), (5, 5, 3, 1, 2, 8)), ((0\ 1\ 2\ 3\ 4\ 5\ 6), (4, 4, 1, 5, 1, 4), (4, 6, 3, 7, 3, 6)))$ que não pertence ao cenário fonte ou sorvedouro. Note que por esse motivo a instância não possui regiões intergênicas auxiliares, ou seja, $a(\mathcal{I}) = 0$ e $\mathcal{A} = \varnothing$. A instância $\mathcal{I}$ possui cinco regiões intergênicas temporárias ($t(\mathcal{I}) = 5$, com $\mathcal{T} = \{\breve{\pi}_1, \breve{\pi}_3, \breve{\pi}_4, \breve{\pi}_5, \breve{\pi}_6\}$) e uma região intergênica durável ($\mathcal{D} = \{\breve{\pi}_2\}$).

Exemplo 2.5.3.



2.6 Grafo de Ciclos

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

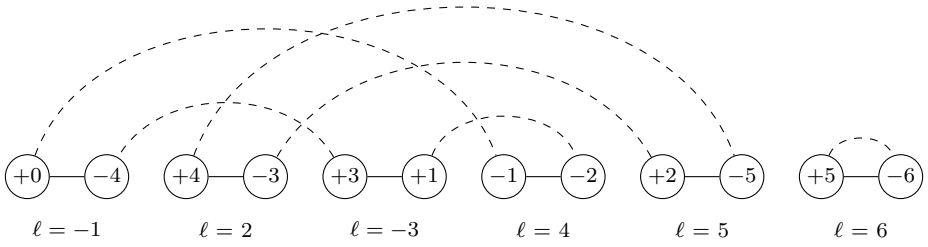
2.6.1 Grafo de Ciclos Clássico

O grafo de ciclos clássico, também chamado de grafo de breakpoints, tem seu uso bastante difundido em problemas de rearranjo de genomas que utilizam instâncias clássicas. Esse grafo evidência 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 gráfico de ciclos clássico por $G(\mathcal{I}) = (V, E, \ell)$, tal 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 $0 < i < n + 1$, 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 divido 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 que são adjacentes na permutação π, enquanto as arestas cinzas representam os elementos que são 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 e 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 a direita no grafo 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 $C \in G(\mathcal{I})$ 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. Duas arestas pretas de um ciclo $C \in G(\mathcal{I})$ são chamadas de *divergentes* se elas são percorridas em direções opostas. Caso contrário, são chamadas de *convergentes*.

Definição 2.6.2. Um ciclo $C \in G(\mathcal{I})$ é chamado de *divergente* se pelo menos uma 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. Um ciclo convergente $C = (c_1, c_2, \ldots, c_k) \in G(\mathcal{I})$ 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 $C = (c_1, c_2, \ldots, c_k)$ 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, tal que $|x_1| < |y_1|$ e que g_2 seja uma aresta cinza adjacente às arestas pretas com rótulos x_2 e y_2, tal 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 $C \in G(\mathcal{I})$ que não se cruza com nenhuma outra aresta cinza de C. Um open gate g_1 de C é fechado se outra aresta cinza (que não seja de C) cruza com g_1.

Observação 2.6.1. Todos os open gates de ciclos não tiviais em $G(\mathcal{I})$ são fechados [53].

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 aplicar a 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. O grafo de ciclos ponderado rígido incorpora na sua estrutura, através de pesos nas arestas, informações referentes ao tamanho das regiões intergênicas do genoma de origem e alvo. A seguir definimos formalmente o grafo de ciclos poderado.

Dada uma instância intergênica rígida $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, definimos o gráfico 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, 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 descrito no grafo de ciclos clássico. Além disso, todos os conceitos, definições e representações que foram apresentados no contexto de grafo de ciclos clássico também são válidas e utilizadas 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 uma 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. Um ciclo $C \in G(\mathcal{I})$ 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*.

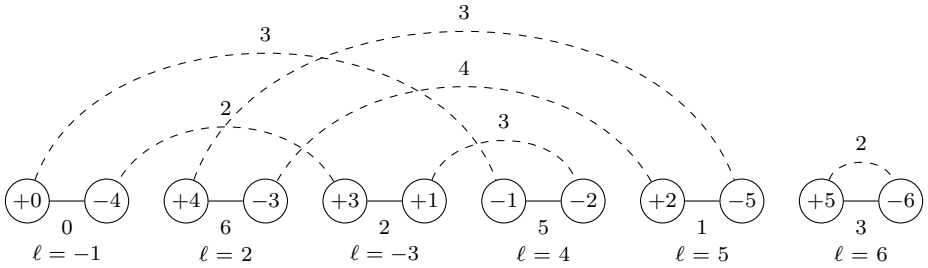
Em outras palavras, um ciclo balanceado indica que a soma dos pesos em suas aretas pretas é a mesma que a soma dos pesos em suas arestas cinzas.

Definição 2.6.5. Um ciclo desbalanceado $C \in G(\mathcal{I})$ 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 um ciclo negativo possui a soma dos pesos em suas arestas pretas maior que a soma dos pesos em suas arestas cinzas, já é um ciclo positivo acontece justamento o oposto. 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 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 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 é uma extensão do grafo de ciclos clássico. O grafo de ciclos ponderado rígido incorpora 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 poderado flexível.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, definimos o gráfico 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, 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 descrito no grafo de ciclos clássico. Além disso, todos os conceitos, definições e representações que foram apresentados no contexto de grafo de ciclos clássico também são válidas e utilizadas no grafo de ciclos ponderado flexível.

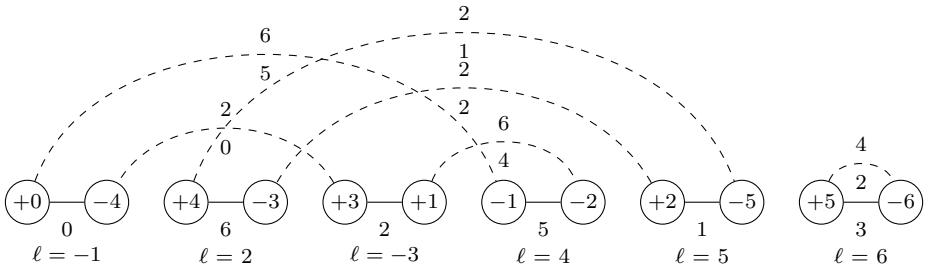
A função de peso $w_p : E_p \to \mathbb{N}_0$ associa os tamanhos das regiões intergênicas no genoma de origem com pesos nas arestas pretas do grafo. As funções de peso $w_c^{\min} : E_c \to \mathbb{N}_0$ e $w_c^{\max} : E_c \to \mathbb{N}_0$ associam, respectivamente, os tamanhos mínimos e máximos permitidos 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 $C \in G(\mathcal{I})$, denotamos por $E_p(C)$ e $E_c(C)$, respectivamente, os conjuntos de arestas pretas e cinzas que pertencem ao ciclo C. Dado um ciclo $C \in G(\mathcal{I})$, 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 a soma dos pesos mínimos e máximos em suas arestas cinzas, respectivamente.

Definição 2.6.6. Um ciclo $C \in G(\mathcal{I})$ é chamado de *verdadeiro* caso $W_g^{\min}(C) \leq W_b(C) \leq W_g^{\max}(C)$. Caso contrário, o ciclo C é chamado de *falso*.

Em outras palavras, um ciclo verdadeiro indica que o peso total é suficiente para satisfazer as restrições relativas aos pesos mínimos e máximos em cada uma de suas arestas cinzas. Definimos os conjuntos de ciclos verdadeiros e falsos em $G(\mathcal{I})$ como $\mathcal{V}$ e $\mathcal{F}$, respectivamente. Dado um ciclo $C \in G(\mathcal{I})$, denotamos por $gap_{\min}(C) = W_b(C) - W_g^{\min}(C)$ e $gap_{\max}(C) = W_g^{\max}(C) - W_b(C)$ como 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 falso, curto falso e trivial verdadeiro.

De agora em diante, as definições e conceitos que serão apresentados referem-se à instâncias intergênicas flexíveis balanceadas. Note que dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ e utilizando um modelo apenas com eventos de rearranjo conservativos, todos os ciclos falsos devem ser removidos e $G(\mathcal{I})$ deve possuir $n+1$ ciclos verdadeiros 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 verdadeiros também devem ser afetados para realizar essa tarefa. Na verdade, existem dois casos em que isso ocorre:

$$\texttt{(i)} \sum_{C \in \mathcal{F}} W_p(C) < \sum_{C \in \mathcal{F}} W_c^{\min}(C)$$

$$\texttt{(ii)} \sum_{C \in \mathcal{F}} W_p(C) > \sum_{C \in \mathcal{F}} W_c^{\max}(C)$$

No caso `(i)`, a soma do peso total de todos os ciclos falsos é menor que a soma do peso mínimo total dos mesmos ciclos, ou seja, não é possível atender todas as retrições de peso mínimo e máximo nas arestas cinzas dos ciclos falsos sem que alguns ciclos verdadeiros transfiram uma determinada quantidade de peso de suas arestas pretas para os ciclos falsos. No caso `(ii)`, a soma do peso total de todos os ciclos falsos é maior que a soma do peso máximo total dos mesmos ciclos. Nesse caso, alguns ciclos falsos precisam transferir uma determinada quantidade de peso de suas arestas pretas para alguns os ciclos verdadeiros.

Definição 2.6.7. Um ciclo verdadeiro $C \in G(\mathcal{I})$ é chamado de *ruim* se ele deve receber ou transferir peso de suas arestas pretas para outro ciclo, e é chamado de *bom* caso contrário.

Definimos os conjuntos de ciclos ruins e bons em $G(\mathcal{I})$ como $\mathcal{R}$ e $\mathcal{B}$, respectivamente. Observe que os casos `(i)` e `(ii)` podem não acorrer se $\sum_{C \in \mathcal{F}} W_c^{\min}(C) \leq \sum_{C \in \mathcal{F}} W_p(C) \leq \sum_{C \in \mathcal{F}} W_c^{\max}(C)$, em ambos os casos temos que $\mathcal{R} = \varnothing$ e $\mathcal{B} = \mathcal{V}$ (Exemplo 2.6.3). Note que se o caso `(i)` ou `(ii)` ocorrer é necessário determinar os conjuntos $\mathcal{R}$ e $\mathcal{B}$, e isso depende do caso em que a instância se encaixa.

Se for o caso `(i)`, então um conjunto $\mathcal{R}$ de tamanho mínimo pode ser composto do menor número de ciclos em que a seguinte restrição seja cumprida:

$$\sum_{C \in \mathcal{R}} gap_{\min}(C) + \sum_{C \in \mathcal{F}} gap_{\min}(C) \geq 0$$

Se for o caso (ii), então um conjunto $\mathcal{R}$ de tamanho mínimo pode ser composto do menor número de ciclos em que a seguinte restrição seja cumprida:

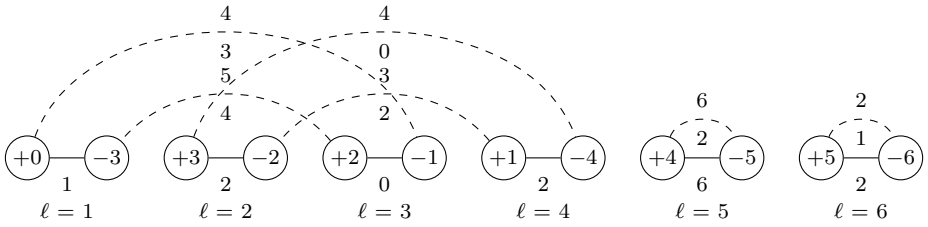
$$\sum_{C\in\mathcal{R}} gap_{\max}(C) + \sum_{C\in\mathcal{F}} gap_{\max}(C) \geq 0$$

Observe que em ambos os casos, o conjunto $\mathcal{R}$ pode ser facilmente obtido após a ordenação, de forma decrescente, dos ciclos verdadeiros pelos valores $gap_{\min}$ e $gap_{\max}$ considerando os casos (i) e (ii), respectivamente. Então, seguindo a ordem decrescente, os ciclos são rotulados como ruins até satisfazerem a restrição. O conjunto de ciclos bons $\mathcal{B}$ é obtido por $\mathcal{V} - R$.

Dada uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$, denotamos por $c_v(G(\mathcal{I}))$ e $c_b(G(\mathcal{I}))$ o número de ciclos verdadeiros e bons em $G(\mathcal{I})$, respectivamente. Dada uma sequência de eventos de rearranjo S, denotamos por $\Delta c_v(G(\mathcal{I}), S) = c_v(G(\mathcal{I}'))$ $- c_v(G(\mathcal{I}))$ 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 verdadeiros e bons, respectivamente, após aplicar a sequência S no genoma de origem $(\pi, \breve{\pi})$ de $\mathcal{I}$.

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

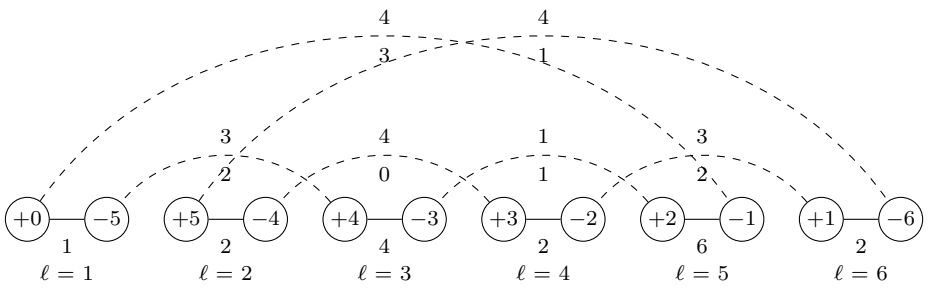
Exemplo 2.6.4.



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 $\mathcal{F} = \{C_1\}$ e $\mathcal{V} = \{C_2, C_3, C_4\}$. Observe que a instância intergênica flexível $\mathcal{I}$ do Exemplo 2.6.4 pertence ao caso (i): $1 = W_b(C_1) < W_g^{\min}(C_1) = 7$, onde apenas o ciclo falso C_1 precisa aumentar o seu peso total para ser transformado em um ciclo verdadeiro. Note que $gap_{\min}(C_2) = 2$, $gap_{\min}(C_3) = 4$ e $gap_{\min}(C_4) = 1$. Portanto, temos que $\mathcal{R} = \{C_2, C_3\}$ e $\mathcal{B} = \{C_4\}$.

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 $\mathcal{F} = \{C_1\}$ e $\mathcal{V} = \{C_2\}$. Observe que a instância intergênica flexível $\mathcal{I}$ do Exemplo 2.6.5 pertence ao caso `(ii)`: $11 = W_b(C_1) > W_g^{\max}(C_1) = 8$, onde apenas o ciclo falso C_1 precisa reduzir o seu peso total para ser transformado em um ciclo verdadeiro. Note que $gap_{\max}(C_2) = 5$. Portanto, temos que $\mathcal{R} = \{C_2\}$ e $\mathcal{B} = \varnothing$.

Observação 2.6.4. Uma instância intergênica flexível $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}^{\min}, \breve{\iota}^{\max}))$ tal que $c_v(G(\mathcal{I}) = c_b(G(\mathcal{I}) = n + 1$ implica que $\pi = \iota$ e $\breve{\iota}_i^{\min} \leq \breve{\pi}_i \leq \breve{\iota}_i^{\max}$ para todo $\breve{\pi}_i \in \breve{\pi}$.

Capítulo 3

Modelos com Porporção 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*. Nesse abordagem, cada tipo de evento de rearranjo possui um peso associado que é contabilizado na distância evolutiva entre os genomas. A abordagem ponderada geralmente é utilizada para mapear um cenário em que queremos que determinado eventos de rearrajo tenham uma possibilidade maior de ocorrer do que outros. Para isso, basta atribuir um peso menor nos eventos de rearranjo que esperados que ocorram mais. Esses pesos podem ser atribuídos com base em observações empíricas de determinados organismos ou através de análises realizadas especificamente para esse objetivo [5, 38].

Os eventos de rearranjo de reversão e transposição são dois dos eventos mais estudados na literatura [11, 37, 60]. Considerando uma representação clássica e uma abordagem não ponderada, temos o problema de Ordenção 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 [52], para a variação com sinais do problema existe um algoritmo de aproximação com fator 2 [64]. Para a variação sem sinais, existe um algoritmo de aproximação com fator $2k$ [59], onde k [27] é o fator de aproximação do algoritmo utilizado para a decomposição de ciclos do Grafo de Ciclos [26].

Considerando um abordagem poderada, temos o problema de Ordenção de Permutações por Reversões e Transposições Ponderadas ($\mathbf{Sb_WRT}$) na variação com e sem sinais. In 2002, Eriksen [39] apresentou um algoritmo com factor de aproximação 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.*[53] desenvolveram um algoritmo de aproximação com fator 1.5 para a variação com sinais do problema $\mathbf{Sb_WRT}$ utilizando 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 [6] apresentaram o problema de Ordenção 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 adjacentes 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, no intervalo [1..2], para os eventos de transposição e transposição inversa. Em 2020, Alexandrino *et al.*[2] 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 mais prioridade para determinados tipos de eventos de rearranjo. Entretanto, ela não garantem que os rearranjos de menor custo, que são supostamente os mais frequentes em um cenário evolutivo, serão os mais utilizados pelos algoritmos. Para contornar esse ponto, propomos e investigamos o problema de Ordenção de Permutações por Reversões e Transposições com Restrição de Proporção ($\mathbf{Sb_PRT}$) em instâncias clássicas com e sem sinais. Neste 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 as abordagens ponderada e proporcional tentam incorporar no modelo a frequência na qual 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 características comuns. Uma característica que difere da abordagem de proporção é que uma vez conhecida a frequência na qual 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 ótimo 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 custo maior para o evento de transposição.

Exemplo 3.0.1.

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

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 problem $\mathbf{Sb_PRT}$ 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.

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

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 Ordenção de Permutações por Reversões e Transposições com Restrição de Proporção.

Ordenção de Permutações por Reversões e Transposições com Restrição de Proporção ($\mathbf{Sb_PRT}$)
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 propoção* entre π e ι, denotada por $dp_k(\mathcal{I})$, é o tamanho da menor sequências 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, nesse capítulo iremos nos referir a um breakpoint clássico simplesmente como um breakpoint.

Nesse capítulo, provaremos que o problema $\mathbf{Sb_PRT}$ 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 práticos dos algoritmo propostos.

Estes resultados foram publicados em 2021 na revista *Journal of Bioinformatics and Computational Biology* [15].

3.1 Limitantes Inferiores

Nessa seção, apresentamos limitantes inferiores para as variações com e sem sinais do problema $\mathbf{Sb_PRT}$.

Lema 3.1.1 (Kececioglu e Sankoff [47])**.** *Dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$, para qualquer reversão ρ temos que $\Delta b_1(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 3.1.2 (Walter *et al.* [64])**.** *Dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$, para qualquer transposição τ temos que $\Delta b_1(\mathcal{I}, S = (\tau)) \geq -3$.*

Lema 3.1.3. *Dada uma instância clássica sem sinais $\mathcal{I} = (\pi, \iota)$ 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 [45])**.** *Dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$, para qualquer reversão ρ temos que $\Delta b_2(\mathcal{I}, S = (\rho)) \geq -2$.*

Lema 3.1.5. *Dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$, 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. *Dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ 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 dois removidos por cada evento de S, em média, é menor ou igual a $3 - k$.*

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

Lema 3.1.7 (Hannenhalli e Pevzner [45])**.** *Dada uma instância clássica com sinais* $\mathcal{I} = (\pi, \iota)$*, para qualquer reversão* ρ *temos que* $\Delta c(G(\mathcal{I}), S = (\rho)) \leq 1$*.*

Lema 3.1.8 (Bafna e Pevzner [7]; Walter *et al.* [64])**.** *Dada uma instância clássica com ou sem sinais* $\mathcal{I} = (\pi, \iota)$*, para qualquer transposição* τ *temos que* $\Delta c(G(\mathcal{I}), S = (\tau)) \leq 2$*.*

Lema 3.1.9. *Dada uma instância clássica com sinais* $\mathcal{I} = (\pi, \iota)$ *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. *Dada uma instância clássica sem sinais* $\mathcal{I} = (\pi, \iota)$ *para o problema* ***Sb$_P$RT*** *e uma proporção* $k \in [0..1]$*, temos que* $dp_k(\mathcal{I}) \geq \frac{b_1(\mathcal{I})}{3-k}$*.*

Demonstração. Como $b_1(\mathcal{I})$ 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. *Dada uma instância clássica com sinais* $\mathcal{I} = (\pi, \iota)$ *para o problema* ***Sb$_P$RT*** *e 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 a 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. *Dada uma instância clássica com sinais* $\mathcal{I} = (\pi, \iota)$ *para o problema* ***Sb$_P$RT*** *e 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 Obervação 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

Nessa seção, apresentamos uma análise de complexidade do problema **Sb$_P$RT** em instâncias clássicas com e sem sinais para todos os possível valores de k. A seguir descrevemos formalmente a versão de decisão do problema **Sb$_P$RT**.

Sb$_P$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$_P$RT** é possível fornecer como entrada diferentes valores para k. Entretanto, quando utilizamos o valor de $k = 0$ obtemos o problema **SbRT**, uma vez que estipulamos que em uma solução não é necessário obter uma porcentagem mínima de eventos de reversão em comparação ao tamanho da sequência de eventos de rearranjo. Por outro lado, quando adotamos o valor de $k = 1$, obtemos o problema de Ordenação de Permutações por Reversões (**SbR**). Note que, nesse caso, toda solução para o problema deve ser composta exclusivamente por eventos de reversão. Com base nessa característica do problema obtemos os seguintes lemmas.

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

Demonstração. Quando $k = 0$ o problema **Sb$_P$RT** em instâncias clássicas com sinais torna-se a variação com sinais do problema **SbRT**, que é NP-difícil [52]. Por outro lado, quando $k = 1$ o problema **Sb$_P$RT** em instâncias clássicas com sinais torna-se a variação com sinais do problema **SbR**, que possui um algoritmo exato polinomial [45]. □

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$_P$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 [52, 26]. □

A seguir investigamos a complexidade do problema **Sb$_P$RT** quando k pertence ao intervalo (0..1). Para isso, apresentamos definições que serão 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 o transformação de duplicação sendo aplicado na representação clássica sem sinais $\pi = (4\ 1\ 5\ 3\ 2)$.

Exemplo 3.2.1.
$\pi \quad = \quad (4\ 1\ 5\ 3\ 2)$
$\pi' \quad = \quad (7\ 8\ 1\ 2\ 9\ 10\ 5\ 6\ 3\ 4)$

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 sendo aplicado na representação clássica sem sinais $\pi = (4\ 1\ 5\ 3\ 2)$.

Exemplo 3.2.2.
$\pi \quad = \quad (4\ 1\ 5\ 3\ 2)$
$\pi' \quad = \quad (+4\ +1\ +5\ +3\ +2)$

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 sendo aplicada na representação clássica com sinais $\pi = (+4\ +1\ +5\ -3\ -2)$.

Exemplo 3.2.3.
$\pi \quad = \quad (+4\ +1\ +5\ -3\ -2)$
$\pi' \quad = \quad (+4\ +1\ +5\ -3\ -2\ +6\ +7\ +8)$

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

Exemplo 3.2.4.
$\pi \quad = \quad (4\ 1\ 5\ 3\ 2)$
$\pi' \quad = \quad (4\ 1\ 5\ 3\ 2\ 6\ 7\ 8)$

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) $\pi'_j = -(n+4-j)$; (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) $\pi'_j = n+4-j$; (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 sendo aplicada na representação clássica com sinais $\pi = (+4\ +1\ +5\ -3\ -2)$.

Exemplo 3.2.5.
$\pi \quad = \quad (+4\ +1\ +5\ -3\ -2)$
$\pi' \quad = \quad (+4\ +1\ +5\ -3\ -2\ -8\ -7\ -6\ +9\ +10\ +11)$

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

Exemplo 3.2.6.
π = (4 1 5 3 2)
π' = (4 1 5 3 2 8 7 6 9 10 11)

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 não nulo.
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 [24] provaram que o problem **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.* [52])**.** *Se uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ 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 e 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 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 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 existe 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 e 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. Contúdo, cada reversão $\rho \in S'$ remove dois breakpoints tipo dois (criados pela extensão gadget). Logo, $|S'| = \ell + p$, $\pi' \cdot S' = \iota'$ e $\frac{|S'_\rho|}{|S'|}$.

($\Leftarrow$) 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 $d_k(\mathcal{I}') \geq \ell+p$. Suponha que existe 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 exitir no máximo ℓ transpositions in S'. Caso contrário, a proporção $\frac{|S'_\rho|}{|S'|}$ não seria satisfeita. Dessa forma, temos que existe 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 Lemma 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 restantes eventos de rearranjo 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* $\boldsymbol{Sb_PRT}$ *temos no passo 1 a obteção 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.* [52])**.** *Se uma instância clássica sem sinais* $\mathcal{I} = (\pi, \iota)$ *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 e 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. Estas 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 existe 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 $\mathcal{I}'$. Contúdo, cada reversão $\rho \in S'$ remove dois breakpoints tipo dois (criados pela extensão gadget). Logo, $|S'| = \ell + p$, $\pi' \cdot S' = \iota'$ e $\frac{|S'_\rho|}{|S'|}$.

($\Leftarrow$) 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 $d_k(\mathcal{I}') \geq \ell+p$. Suponha que existe 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 exitir no máximo ℓ transpositions in S'. Caso contrário, a proporção $\frac{|S'_\rho|}{|S'|}$ não seria satisfeita. Dessa forma, temos que existe 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 Lemma 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 restantes eventos de rearranjo de S', ou seja, as p reversões, devem remover $2p$ bre-

akpoints 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: (i) 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 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* $\boldsymbol{Sb_PRT}$ *temos no passo 1 a obteção 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

Nessa 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 de $3-k$ para o problema $\mathbf{Sb_PRT}$ em instâncias clássicas sem sinais.

Lema 3.3.1 (Kececioglu e Sankoff [47])**.** *Dada uma instância clássica sem sinais* $\mathcal{I} = (\pi, \iota)$*, 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 e 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 algoritmo também garante um fator de aproximação de $3 - k$ para instâncias clássicas sem sinais do problem $\mathbf{Sb_PRT}$ e para qualquer valor de k. Além disso, a propoção entre a quantidade de reversões e o tamanho da sequência de eventos de rearranjo fornecida pelo o algoritmo tende a ser um valor próximo de k.

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

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 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 [47]
8          π ← π · S'
9          S ← S + S'
10
11 retorna 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 de $3-k$. 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 $|S| \leq b_1(\pi) \leq n + 1$.

3.3.2 Instâncias Clássicas com Sinais

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

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

Demonstração. Se $G(\mathcal{I})$ possuir um ciclo divergente C, então existe uma reversão que quando aplicada em C aumenta o número de ciclos em uma unidade (Teorema 5 de [64]). Caso contrário, todos os ciclos não triviais devem ser convergentes. Isso significa que um dos seguintes cenários deve ocorrer obrigatoriamente [53]:

- 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 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 que os open gates são fechado 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 possivel 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 em 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 curto, então é possivel 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. □

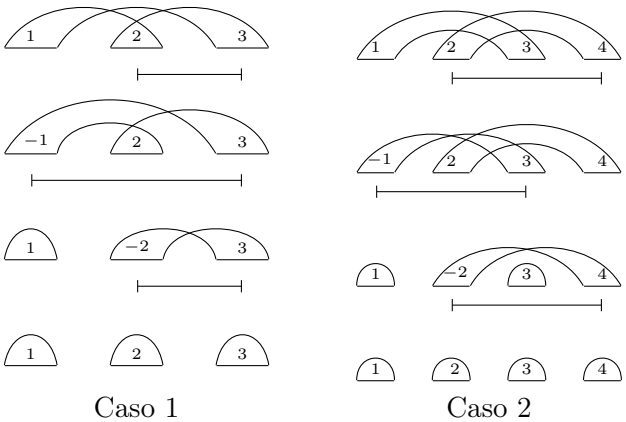
Lema 3.3.4. *Dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$, é possível transforma π em ι utilizando no máximo $\frac{3(n+1-c(G(\mathcal{I})))}{2}$ reversões.*

Demonstração. O Lema 3.3.3 resulta em uma sequência de reversões que sempre aumenta o número de ciclos. Logo, podemos aplicarmos 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, em média, é maior ou igual a 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. □

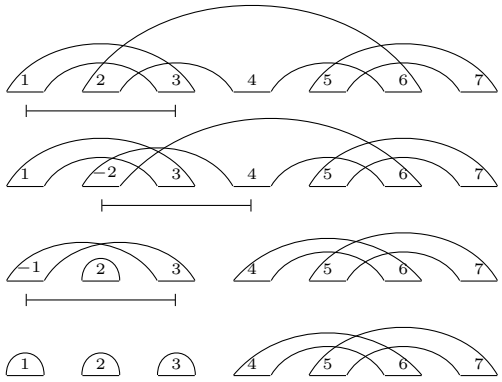
Teorema 3.3.5. *Existe um algoritmo de aproximação com fator $3 - \frac{3k}{2}$ para o problema $\boldsymbol{Sb_PRT}$ em instâncias clássicas com sinais e para uma proporção $k \in [0..1]$.*

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 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,

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



Caso 1 Caso 2



Caso 3

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 $C_1 = (3, 1)$ tem os open gates fechados pelo ciclo $C_2 = (4, 2)$. Por fim, no Caso 3, o ciclo longo não orientado $C_2 = (6, 4, 2)$ tem os open gates fechados pelos ciclos $C_1 = (3, 1)$ e $C_3 = (7, 5)$. Para os três casos, é mostrado uma sequência de três reversões que aumenta o número de ciclos em duas unidades.

□

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 algoritmo também garante um fator de aproximação de $3 - \frac{3k}{2}$ para instâncias clássicas com sinais do problem $\mathbf{Sb_PRT}$ 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 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 da sequência S aumenta, em média, o número de ciclos em pelo menos 2/3 unidades. 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 encontradas

Algoritmo 2: Um algoritmo de aproximação para 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 π ≠ ι faça
3      se |S_ρ|/(|S|+1) ≥ k e existe uma transposição τ que Δc(G(I),(τ)) ≥ 1 então
4          π ← π · τ
5          S ← S + (τ)
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
           unidades (Lema 3.3.3)
8          π ← π · S'
9          S ← S + S'
10
11 retorna S
```

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

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

Definição 3.3.1. Dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$, 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.

Observação 3.3.1 (Oliveira *et al.* [53]). 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 $\mathcal{O}(n^3)$), 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, em tempo 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 de Algoritmo 3 é $\mathcal{O}(n^4)$.

Lema 3.3.6. *Dada uma instância clássica com sinais $\mathcal{I} = (\pi, \iota)$ 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 $\frac{|S_\rho|}{|S|} \geq k$.*

Algoritmo 3: Um algoritmo de aproximação assintótica para 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** $|\mathcal{A}_\rho(\mathcal{I})| > k(|S| + |\mathcal{A}_\rho(\mathcal{I})|)$ **faça**
3 **se** *em* $G(\mathcal{I})$ *existe um ciclo divergente* **então**
4 Seja ρ uma reversão que aumenta o número de ciclos em uma unidade (Teorema 5 de [64])
5 $\mathcal{I} = (\pi \cdot \rho, \iota)$
6 $S \leftarrow S + (\rho)$
7 **senão**
8 Seja S' uma sequência de no máximo duas transposições que aumenta o número de ciclos em duas unidade (Teorema 3.4 de [9])
9 $\mathcal{I} = (\pi \cdot S', \iota)$
10 $S \leftarrow S + S'$
11
12 $S \leftarrow S + \mathcal{A}_\rho(\mathcal{I})$
13 **se** $|S_\rho| < k|S|$ **então**
14 Substitua até duas transposições de S por uma sequência equivalente de reversões (Observação 3.3.1)
15 **retorna** S

Demonstração. Note que a sequência forneceida pelo algoritmo $\mathcal{A}_\rho$ (linha 12) trasforma 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 unidades. 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 $\frac{|S_\rho|}{|S|} \geq k$ subtituindo 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* $\boldsymbol{Sb_PRT}$ *em instâncias clássicas com sinais e para uma proporção* $k \in [0..1]$.

Demonstração. Pelo Lema 3.3.6 e Teorema 3.1.12, 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 $|S| \leq$

$(n+1-c(G(\mathcal{I})))/(1-k/3)+4 \leq \frac{2-k}{1-k/3} dp_k(\mathcal{I})+4$. 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 $\mathbf{Sb_PRT}$, ou seja, para garantir que a restrição de propoção 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, usamos seis algoritmos da literatura como baselines para comparar com os resultados fornecidos por nossos algoritmos. 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 $\mathbf{Sb_PRT}$. 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 modificações realizado.

- Instâncias clássicas sem sinais:
  - UR: Algoritmo de aproximação com fator 2 para o problema de Ordenação de Permutações por Reversões [47].
  - UT: Algoritmo de aproximação com fator 1.5 para o problema de Ordenação de Permutações por Transposições [9].
  - URT: Algoritmo de aproximação com fator 2α para o problema de Ordenação de Permutações por Reversões e Transposições [59], onde α é o fator de aproximação do algoritmo utilizado para a decomposição de ciclos do Grafo de Ciclos (valor adotado $\alpha = 1.4193 + \epsilon$ [49]).
- Instâncias clássicas com sinais:
  - SR: Algoritmo exato e polinomial para o problema de Ordenação de Permutações por Reversões [45].
  - SRT: Algoritmo de aproximação com fator 2 para o problema de Ordenação de permutações por Reversões e Transposições [64].
  - SWRT: Algoritmo de aproximação com fator 1.5 para o problema de Ordenação de Permutações por Reversões e Transposições Ponderadas [53] (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 descito 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 algoritmo 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 descito 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 10.000 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 parâmetros 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. Este processo é repetido até que o grupo tenha um total de 10.000 instâncias. As proporções utilizadas variaram de 0 a 1, em intervalos de 0.1, totalizando 11 grupos. Esta 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 220.000 instâncias.

DB2 - Esta base de dados foi desenvolvido 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 10.000 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, adotamos: 10%, 20%, 30%, 40% e 50%. As etapas finais do processo são semelhantes ao que descrevemos anteriormente na base de dados DB1. Esta base de dados possui uma versão apenas para instâncias clássicas com sinais e um total de 250.000 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 mínimo, média e máximo, 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.

A Tabela 3.1 mostra os resultados dos algoritmos UR, UT, URT e UPRT aplicados em instâncias sem sinal 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 em valores menores 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 possível notar que a razão de aproximação média é muitas vezes melhor do que o fator de aproximação teórica $3 - k$ 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 pela 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 supeior a cinco vezes o número de eventos utilizados para criar as instâncias (40 operações). Isso indica que a técnica de modificação de solução aplicada ao algoritmo desenvolvido para o problema considerando apenas transposições não fornece bons resultados para valores maiores de k.

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 se encaixa perfeitamente no caso em que $k = 1,0$ e, quando $k = 0,0$, uma sequência de transposições não precisa passar o processo de modificação para respeitar a restrição de proporção. Nesse caso, modificar as soluções para se adequarem à proporção mínima

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

exigida pela instância resultou em bons resultados independente 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 independente 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 obter 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 valor 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 $k = 1$, temos o problema de Ordenando 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.

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

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.

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

Pela Tabela 3.3, é possível notar que a variação da razão de aproximação média é muito pequena independente 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 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 instâncias 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 aplicado em instâncias clássicas com sinais da base de dados DB2. A coluna OP mostra o número de operações para criar as instâncias. A razão de aproximação foi calculada adotando-se o limite inferior apresentado no Teorema 3.1.12.

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

A partir da Tabela 3.4, é possível notar que o algoritmo SPRT apresenta uma tendência de diminuir a razão de aproximação média linearmente à medida que o tamanho da permutação e o número de operações utilizadas para criar as instâncias (OP) crescem. Outro fato 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 equivalente ou melhores resultados 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 de modificação da solução proposto para viabilizar soluções obtidas atráves de algoritmo para outros problemas também apresentou bons resultados, principalmente quando aplicado ao algoritmo SWRT.

3.5 Conclusões

Neste capítulo, investigamos o problema de Ordenção 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 considerando 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 assintótico 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 algoritmo 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 presentes ou associadas diretamente aos genes são descartadas, o que implica em uma perda de informação. Em particular, informações referente às regiões intergênicas, que são regiões entre cada par consecutivo de genes e nas extremidades de um genoma linear, acabam não sendo consideradas pelos modelos que adotam uma representação clássica de um genoma. Estudos [12, 13] sugerem que incorporar tais estruturas aos modelos pode resultar em resultados mais realistas para a distância evolutiva entre os organismos. Cada região intergênica possui uma quantidade de nucleotídeos, essa quantidade de nucleotídeos é denominada de *tamanho*. Nesse 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:

- Ordenação de Permutações por Reversões Intergênicas ($\mathbf{Sb_IR}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Indel ($\mathbf{Sb_IRI}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Move ($\mathbf{Sb_IRM}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Move e Indel ($\mathbf{Sb_IRMI}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão e Transposição ($\mathbf{Sb_IRT}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Indel ($\mathbf{Sb_IRTI}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição e Move ($\mathbf{Sb_IRTM}$)
- Ordenação de Permutações por Operações Intergênicas de Reversão, Transposição, Move e Indel ($\mathbf{Sb_IRTMI}$)

Neste capítulo, iremos nos 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, iremos nos referir a um breakpoint intergênico simplesmente como um breakpoint.

Dada uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, a *distância* entre $(\pi, \breve{\pi})$ e $(\iota, \breve{\iota})$, denotada por $d_{\mathcal{M}}(\mathcal{I})$, é o tamanho da menor sequências 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 considerados neste capítulo são identificados por siglas apresentadas na Tabela 4.1.

Tabela 4.1: Siglas dos modelos de rearranjo considerados para instâncias intergênicas rígidas.

Sigla	**Conjunto de Eventos de Rearranjo**
$\mathbf{Sb_IR}$	$\{\rho\}$
$\mathbf{Sb_IRI}$	$\{\rho, \delta\}$
$\mathbf{Sb_IRM}$	$\{\rho, \mu\}$
$\mathbf{Sb_IRMI}$	$\{\rho, \mu, \delta\}$
$\mathbf{Sb_IRT}$	$\{\rho, \tau\}$
$\mathbf{Sb_IRTI}$	$\{\rho, \tau, \delta\}$
$\mathbf{Sb_IRTM}$	$\{\rho, \tau, \mu\}$
$\mathbf{Sb_IRTMI}$	$\{\rho, \tau, \mu, \delta\}$

Quando estivermos adotando um modelo de rearranjo composto exclusivamente por eventos de rarranjo conservativos assumimos que a instância intergênica rígida para o problema será sempre balanceada. Caso contrário, seria impossível transformar o genoma de origem no genoma alvo.

Parte dos resultados que serão apresentados neste capítulo foram publicados nas revistas *Journal of Computational Biology* [19] e *Algorithms for Molecular Biology* [20] em 2020 e 2021, respectivamente.

4.1 Limitantes Inferiores

Nesta seção, apresentaremos limitantes inferiores para as variações com e sem sinais dos problemas investigados neste capítulo.

Em instâncias intergênicas rígidas com e sem sinais utilizaremos o conceito de breakpoint tipo dois e um, respectivamente. Os eventos de rearranjo de reversão, transposição, move e indel afetam, respectivamente, a seguinte quantidade de regiões intergênicas: duas, três, duas e uma. No melhor cenário, cada uma das regiões intergênicas faz parte de um breakpoint que é removido após o evento de rearranjo ser aplicado. Com isso, obtemos os seguintes lemas.

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

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

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

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

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

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

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

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

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

$$
\begin{array}{lcl}
d_{\boldsymbol{Sb_IR}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{2}, \\
d_{\boldsymbol{Sb_IRI}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{2}, \\
d_{\boldsymbol{Sb_IRM}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{2}, \\
d_{\boldsymbol{Sb_IRMI}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{2}, \\
d_{\boldsymbol{Sb_IRT}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{3}, \\
d_{\boldsymbol{Sb_IRTI}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{3}, \\
d_{\boldsymbol{Sb_IRTM}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{3} \\
e\ d_{\boldsymbol{Sb_IRTMI}}(\mathcal{I}) & \geq & \frac{ib_1(\mathcal{I})}{3}.
\end{array}
$$

Demonstração. Pela Obervação 2.4.1, 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. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*. Se* $\mathcal{I}$ *for balanceada, então temos que* $d_{\boldsymbol{Sb_IRTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$*. Caso contrário, temos que* $d_{\boldsymbol{Sb_IRTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})+2}{3}$.

Demonstração. Pela Obervação 2.4.1, 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, no máximo três breakpoints tipo um 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. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, *temos que:*

$$
\begin{aligned}
d_{\boldsymbol{Sb_IR}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{2},\\
d_{\boldsymbol{Sb_IRI}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{2},\\
d_{\boldsymbol{Sb_IRM}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{2},\\
d_{\boldsymbol{Sb_IRMI}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{2},\\
d_{\boldsymbol{Sb_IRT}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{3},\\
d_{\boldsymbol{Sb_IRTI}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{3},\\
d_{\boldsymbol{Sb_IRTM}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{3}\\
e\ d_{\boldsymbol{Sb_IRTMI}}(\mathcal{I}) &\geq \tfrac{ib_2(\mathcal{I})}{3}.
\end{aligned}
$$

Demonstração. A prova é similar a 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. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$. *Se* $\mathcal{I}$ *for balanceada, então temos que* $d_{\boldsymbol{Sb_IRTI}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$. *Caso contrário, temos que* $d_{\boldsymbol{Sb_IRTI}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})+2}{3}$.

Demonstração. Pela Obervação 2.4.1, 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, no máximo três breakpoints tipo um 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 criado a partir de uma instância intergênica rígida com sinais, é possível notar que 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 move também afeta duas arestas pretas do grafo, mas pode aumentar somente o número de ciclos balanceados no grafo. Já o evento de indel

afeta apenas uma aresta preta do grafo e pode aumentar somente o número de ciclos balanceados no grafo. 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. De maneira similar, temos que $\Delta c(G(\mathcal{I}), S = (\mu)) = 0$ e $\Delta c_b(G(\mathcal{I}), S = (\mu)) \in \{2, 1, 0, -1, -2\}$ para qualquer move μ, 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. Com isso, obtemos os seguintes limitantes inferiores.

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

$$d_{\boldsymbol{Sb_IRM}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I})) + c_b(G(\mathcal{I}))}{2},$$
$$e\ d_{\boldsymbol{Sb_IRMI}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I})) + c_b(G(\mathcal{I}))}{2}.$$

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). Reversões, moves e indels podem aumentar $c(G(\mathcal{I})) + c_b(G(\mathcal{I}))$ em no máximo duas unidades, então o teorema segue. □

4.2 Análise de Complexidade

Nesta seção realizamos uma análise de complexidade considerando as variações dos problemas resultantes dos modelos de rearranjo investigados neste capítulo.

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 Ordenação de Permutações por Reversões e Transposições(**SbRT**)
- O problema 3-Partição.

Esses problemas que pertencem à classe NP-difícil [26, 52, 44].

Ordenação de Permutações por Reversões (**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 $\|S\| \leq d$?

Ordenação de Permutações por Reversões e Transposições (**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-Partição (**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.

$\mathbf{Sb_I R}$(Versão de Decisão)
Entrada: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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 $|S| \leq t$?

$\mathbf{Sb_I RI}$(Versão de Decisão)
Entrada: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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 $|S| \leq t$?

$\mathbf{Sb_I RM}$(Versão de Decisão)
Entrada: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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 $|S| \leq t$?

$\mathbf{Sb_I RMI}$(Versão de Decisão)
Entrada: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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 $|S| \leq t$?

$\mathbf{Sb_I RT}$(Versão de Decisão)
Entrada: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ 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 $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$, tal que $|S| \leq t$?

$\mathbf{Sb_IRTI}$(Versão de Decisão)
Entrada: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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: Uma instância intergênica rígida com ou sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e um número natural t.
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 pertencem à classe NP-difícil.

Teorema 4.2.1. *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 satisfeita 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 satisfeita.

($\Rightarrow$) Suponha que existe 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 existe uma sequência S' com t eventos de rearranjo, tal que $(\pi, \breve{\pi}) \cdot S' = (\iota, \breve{\iota})$. Primeiramente mostraremos que a sequência S' é composta exclusivamente por reversões intergênicas. Suponha por contradição que S' é uma sequência ótima 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'', 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,

$(\pi, \breve{\pi}) \cdot S'' = (\iota, \breve{\iota})$, o que contradiz a suposição de que S' é uma sequência ótima para a instância $(\mathcal{I}', t)$ do problema $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$ ou $\mathbf{Sb_IRMI}$. Sabendo que S' é 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 $\rho^{(i,j)}$. Note que $\pi \cdot S = \iota$ e $|S| = d = t$. Logo, o teorema segue. □

Teorema 4.2.2. *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 a descrita no Teorema 4.2.1, 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 $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ ou $\mathbf{Sb_IRTMI}$ é composta por reversões intergênicas e transposições intergênicas ao invés de reversões intergênicas exclusivamente. □

Os problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRT}$ e $\mathbf{Sb_IRTM}$ em instâncias intergênicas rígidas com sinais pertencem à classe NP-difícil [56, 54]. A seguir mostramos que a variação com sinais dos problemas $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRTI}$ e $\mathbf{Sb_IRTMI}$ também pertencem à classe NP-difícil. Para a variação com sinais dos problemas $\mathbf{Sb_IRM}$ e $\mathbf{Sb_IRMI}$ iremos realizar uma redução do problema **3-PART**, enquanto para a variação com sinais dos problemas $\mathbf{Sb_IRTI}$ e $\mathbf{Sb_IRTMI}$ utilizaremos um redução da variação com sinais do problema **SbRT**.

Teorema 4.2.3. *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 assinada $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ do problema $\mathbf{Sb_IRM}$ ou $\mathbf{Sb_IRMI}$ da seguinte forma:

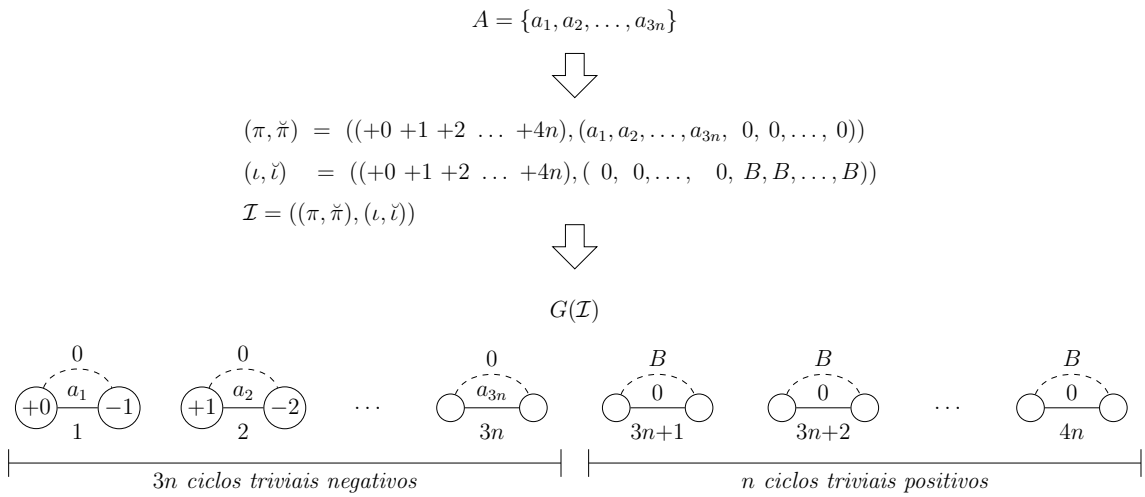
i) $\pi = \iota = (+0\ +1\ +2\ \ldots\ +4n)$.

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 $3n + 1 \leq j \leq 4n$.

O Exemplo 4.2.1 ilustra o processo de criação de uma instância $\mathcal{I}$ do problema $\mathbf{Sb_IRM}$ ou $\mathbf{Sb_IRMI}$ 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 4.2.1.



Agora, mostramos que a instância A da **3-PART** é satisfeita $\iff$ $d_{\mathbf{Sb_IRM}}(\mathcal{I}) \leq 3n$ e $d_{\mathbf{Sb_IRMI}}(\mathcal{I}) \leq 3n$.

($\Rightarrow$) 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 $a_i \in A_j$. Repeta este procedimento em todas as n triplas produzindo uma sequência de $3n$ operações de move que resulta em $c(G(\mathcal{I})) = 4n$ e $c_b(G(\mathcal{I})) = 4n$. Assim, $d_{\mathbf{Sb_IRM}}(\mathcal{I}) \leq 3n$ e $d_{\mathbf{Sb_IRMI}}(\mathcal{I}) \leq 3n$.

($\Leftarrow$) Agora suponha que $d_{\mathbf{Sb_IRM}}(\mathcal{I}) \leq 3n$ e $d_{\mathbf{Sb_IRMI}}(\mathcal{I}) \leq 3n$. Observe que, para atingir o genoma alvo desejado, os ciclos triviais com aresta 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 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 no máximo dois ciclos. Se o movimento 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 $\Delta c_\alpha(G(\mathcal{I}), S = (\mu)) = 0$. Observe que a operação de move não altera o tamanho dos ciclos.

- Se pelo menos um dos ciclos for trivial, então o movimento 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 $\Delta c_\alpha(G(\mathcal{I}), S = (\mu)) = 0$. Observe 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 $\rho_{(x,y)}^{(i,j)}$ pode criar no máximo 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 em que 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 acaba unindo 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) seguido por uma reversão ρ_2 que divide esse ciclo de tamanho dois que foi gerado em dois ciclos alpha novamente. Observe que neste caso a reversão ρ_2 desfaz o que ρ_1 criou, 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 (uma reversão para realizar esta tarefa deve primeiro dividir ou unir o ciclo). 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. Este 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 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 necessária para balancear a instância $\mathcal{I}$ novamente e, como esse indel não pode aumentar o número de ciclos alpha resulta em mais de $3n$ operações em uma sequência S que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

Neste 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. □

Teorema 4.2.4. *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 a descrita no Teorema 4.2.1, 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 reversões intergênicas exclusivamente. □

4.3 Instâncias Intergênicas Rígidas sem Sinais

Nesta seção apresentamos algoritmos para os problemas resultantes dos modelos de rearranjo investigados neste capítulo e considerando uma representação intergênica rígida sem sinais de um genoma. Inicialmente iremos apresentar alguns lemas que serão utilizados por múltiplos algoritmos.

Lema 4.3.1. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, tal que* $\sum_{i=1}^{n+1} \breve{\pi}_i \geq \sum_{i=1}^{n+1} \breve{\iota}_i$ *e* $ib_1(\mathcal{I}) > 1$*, então sempre é possível encontrar um par conectado de breakpoints.*

Demonstração. Como $ib_1(\mathcal{I}) > 1$, então deve existir pelo menos um par de breakpoints tipo um (π_i, π_{i+1}) e (π_j, π_{j+1}). Agora vamos mostrar que pelo menos um desses pares de breakpoints está conectado. Suponha por contradição que existe 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 não existe nenhum par conectado de breakpoints em $\mathcal{I}$. Com isso, temos que avaliar dois possíveis casos:

- Para todo par de breakpoints tipo um (π_i, π_{i+1}) e (π_j, π_{j+1}), os elementos (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}) não são consecutivos na permutação identidade ι. Entretanto, isso não pode acontecer uma vez que, por construção da instância, π e ι são permutações que compartilham o mesmo conjunto de valores.

- Para todo par de breakpoints tipo um (π_i, π_{i+1}) e (π_j, π_{j+1}), a quantidade de nucleotídeos nas regiões intergênicas $\breve{\pi}_{i+1}$ e $\breve{\pi}_{j+1}$ não é sufuciente para remover qualquer breakpoint, ou seja, $\breve{\pi}_{i+1} + \breve{\pi}_{j+1} < \breve{\iota}_k$ onde é tamanho da região intergênica entre o par de elementos consecutivos correspondentes na permutação identidade ι. Entretanto, se isso for verdade temos 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 $\sum_{i=1}^{n+1} \breve{\pi}_i \geq \sum_{i=1}^{n+1} \breve{\iota}_i$.

□

Lema 4.3.2. *Não existe uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, tal que* $ib_1(\mathcal{I}) = 1$.

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 existe 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. Caso contrário, teríamos 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. □

4.3.1 Reversão

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

Lema 4.3.3. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *e sejam* (π_i, π_{i+1}) *e* (π_j, π_{j+1}) *breakpoints conectados, 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, uma das seguintes possibilidades deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não formam uma adjacência intergênica, são 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 $\rho_{(x,y)}^{(i+1,j)}$ 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 do que a região intergênica entre os mesmo elementos no genoma alvo (Figura 4.1a)).

ii. O par de elementos (π_i, π_{j+1}) não formam uma adjacência intergênica, são 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 existe 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. Entretando, se ambas afirmações forem verdadeiras, isso implica que os valores dos elementos no segmento $(\pi_{i+1}, \ldots, \pi_j)$ não estão presentes em ι, isso contradiz a construção da instância $\mathcal{I}$, já que 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 4.1b)). 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 4.1c)).

iii. O par de elementos (π_{i+1}, π_j) não formam uma adjacência intergênica, são 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 existe 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 afirmações forem verdadeiras, isso implica que $|\pi_j - \pi_{i+1}| > 1$ e 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 4.1d)).

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não formam uma adjacência intergênica, são 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 $(i)(Figura\ 4.1e)$).

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

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

Lema 4.3.4. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 4 transforma* $(\pi, \breve{\pi})$ *em* $(\iota, \breve{\iota})$ *utilizando no máximo* $2ib_1(\mathcal{I})$ *reversões.*

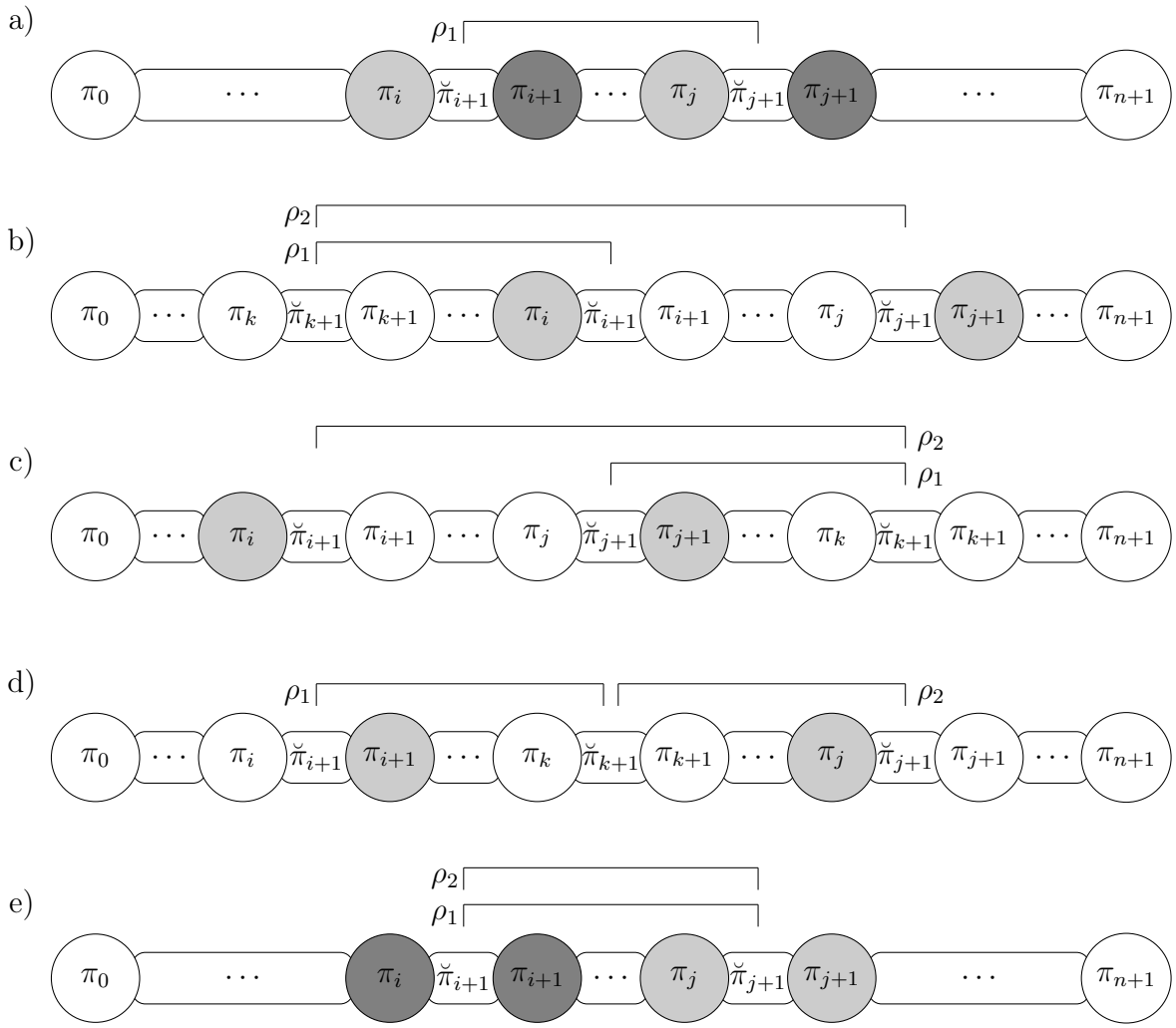


Figura 4.1: As possibilidades que podem surgir quando existe um par de breakpoints conectados e as reversões que podem ser aplicadas para remover pelo menos um breakpoint tipo um. O par de elementos que são consecutivos no genoma alvo é representado com uma cor em tons de cinza.

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 2.4.1 e Lema 4.3.2), o seguinte procedimento é aplicado: pelos lemas 4.3.1 e 4.3.3, sempre podemos encontrar um par conectado de breakpoints e remover pelo menos um breakpoint tipo um após aplicar no máximo duas reversões. 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 duas reversões. Logo, $2ib_1(\mathcal{I})$ reversões, no máximo, são utilizadas para transformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ e 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 4.3.3, 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*.

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
enquanto $ib(\mathcal{I}) > 1$ **faça**
  `// Lema 4.3.1`
  (π_i, π_{i+1}), $(\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  `// Lema 4.3.3`
  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**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso iii* **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso iv* **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorna S

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

Teorema 4.3.5. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 4 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IR}$*.*

Demonstração. Pelo Lema 4.3.4, 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 $d_{\mathbf{Sb_IR}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

4.3.2 Reversão e Indel

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

Lema 4.3.6. *Dada uma instância intergênica rígida desbalanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, tal que* $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$*, então sempre é possível aplicar um indel* δ *de forma que* $\Delta ib_1(\mathcal{I}, S = (\delta)) \leq 0$ *e* $\mathcal{I}$ *é tranformada em 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 4.3.7. *Dada uma instância intergênica rígida desbalanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, tal que* $ib_1(\mathcal{I}) = 1$*, então sempre é 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 obrigatoriamente ele deve ser um breakpoint forte. 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. Como o breakpoint (π_i, π_{i+1}) acaba sendo removido após a aplicação do evento de indel, o lema segue. □

A seguir apresentamos o Algoritmo 5 para a variação sem sinais do problema $\mathbf{Sb_I RI}$.

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

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
`// Lema 4.3.6`
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 $ib(\mathcal{I}) > 1$ **faça**
  `// Lema 4.3.1`
  $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  `// Lema 4.3.3`
  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**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso iii* **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se $(\pi_i, \pi_{i+1}), (\pi_j, \pi_{j+1})$ *pertence ao caso iv* **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
`// Lema 4.3.7`
se $ib(\mathcal{I}) = 1$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorna S

Lema 4.3.8. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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:

- $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$, neste cenários o Algoritmo 5 aplica um indel (linhas 2-5) que pode não remover nenhum breakpoint tipo um, mas torna $\mathcal{I}$ 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 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 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 4.3.2, 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 cenários 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.

- $\sum_{i=1}^{n+1} \breve{\pi}_i > \sum_{i=1}^{n+1} \breve{\iota}_i$, neste último cenário enquanto $ib(\mathcal{I})$ 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) transformando $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$. Note que no pior caso deste cenário cada breakpoint tipo um é removido após a aplicação de duas reversões.

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

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

Teorema 4.3.9. *Dada uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, o Algoritmo 5 é uma 4-aproximação para o problema $\boldsymbol{Sb_IRI}$.*

Demonstração. Pelo Lema 4.3.8, 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 $d_{\mathbf{Sb_IRI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

4.3.3 Reversão e Move

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

Lema 4.3.10. *Dada uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ e sejam (π_i, π_{i+1}) e (π_j, π_{j+1}) breakpoints conectados, então é possível remover pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando no máximo duas reversões ou um move.*

Demonstração. Note a aplicação do Lema 4.3.3 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, uma das seguintes possibilidades deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não formam uma adjacência intergênica, são 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 4.3.3.

ii. O par de elementos (π_i, π_{j+1}) não formam uma adjacência intergênica, são 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 4.3.3.

iii. O par de elementos (π_{i+1}, π_j) não formam uma adjacência intergênica, são 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 4.3.3.

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não formam uma adjacência intergênica, são 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, obrigatoriamente (π_i, π_{i+1}) ou (π_j, π_{j+1}) deve ser 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 $\mu^{(i+1,j+1)_{(x)}}$, tal que $x = \breve{\pi}_{i+1} - \breve{\iota}_{max(\pi_i,\pi_{i+1})}$. Caso contrário, aplicamos um move $\mu^{(j+1,i+1)_{(x)}}$, 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 $\mu^{(j+1,i+1)_{(x)}}$, tal que $x = \breve{\pi}_{j+1} - \breve{\iota}_{max(\pi_j,\pi_{j+1})}$. Caso contrário, aplicamos um move $\mu^{(i+1,j+1)_{(x)}}$, 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 um 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 garatem que nenhum novo breakpoint é gerado e 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. □

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

Lema 4.3.11. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 6 transforma* $(\pi, \breve{\pi})$ *em* $(\iota, \breve{\iota})$ *utilizando no máximo* $2ib_1(\mathcal{I})$ *eventos de reversão e move.*

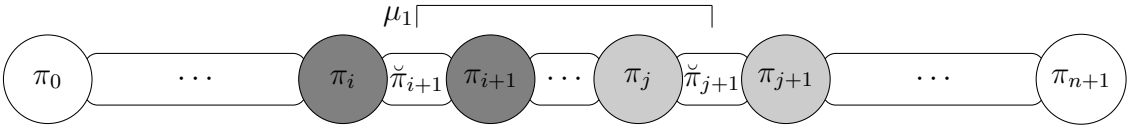


Figura 4.2: Exemplo de remoção de pelo menos um breakpoint tipo um após a aplicação de um move μ.

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto ib(I) > 1 faça
       // Lema 4.3.1
3      (π_i, π_{i+1}), (π_j, π_{j+1}) ← encontre um par de breakpoints conectados
       // Lema 4.3.10
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 iv então
7          S' ← (μ_1)
8      senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso ii então
9          S' ← (ρ_1, ρ_2)
10     senão se (π_i, π_{i+1}), (π_j, π_{j+1}) pertence ao caso iii então
11         S' ← (ρ_1, ρ_2)
12     S ← S + S'
13     I ← ((π, π̆) · S', (ι, ῐ))
14 retorna S
```

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 2.4.1 e Lema 4.3.2), o seguinte procedimento é aplicado: pelos lemas 4.3.1 e 4.3.10, 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 também pode ser realizado em tempo constante. Logo, o tempo de execução do Algoritmo 6 também é $\mathcal{O}(n^2)$.

Teorema 4.3.12. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 6 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRM}$*.*

Demonstração. Pelo Lema 4.3.11, 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 $d_{\mathbf{Sb_IRM}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

4.3.4 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**.

A seguir apresentamos o Algoritmo 7 para a variação sem sinais do problema **Sb$_\mathbf{I}$RMI**.

Algoritmo 7: Um algoritmo de aproximação para o problema **Sb$_\mathbf{I}$RMI**.

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 4.3.6`
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 $ib(\mathcal{I}) > 1$ **faça**
  `// Lema 4.3.1`
  (π_i, π_{i+1}), $(\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  `// Lema 4.3.10`
  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 iv* **então**
    $S' \leftarrow (\mu_1)$
  senão se (π_i, π_{i+1}), (π_j, π_{j+1}) *pertence ao caso ii* **então**
    $S' \leftarrow (\rho_1, \rho_2)$
  senão se (π_i, π_{i+1}), (π_j, π_{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 4.3.7`
se $ib(\mathcal{I}) = 1$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorna S

Lema 4.3.13. *Dada uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, 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 a descrita no Lema 4.3.8. □

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

Teorema 4.3.14. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 7 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRMI}$*.*

Demonstração. Pelo Lema 4.3.13, 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 $d_{\mathbf{Sb_IRMI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{2}$. Logo, o teorema segue. □

4.3.5 Reversão e Transposição

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

Lema 4.3.15. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *e sejam* (π_i, π_{i+1}) *e* (π_j, π_{j+1}) *breakpoints conectados, então é possível remover pelo menos um breakpoint tipo um de* $\mathcal{I}$ *utilizando no máximo duas reversões ou uma transposição.*

Demonstração. Note a aplicação do Lema 4.3.3 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, uma das seguintes possibilidades deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não formam uma adjacência intergênica, são 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 4.3.3.

ii. O par de elementos (π_i, π_{j+1}) não formam uma adjacência intergênica, são 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. Nesta caso sabemos que deve existir um breakpoint tipo um (π_k, π_{k+1}), tal que $k < i$ ou $k > j$ (caso *ii*, Lema 4.3.3). 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 elemento (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 formam uma adjacência intergênica, são 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. Nesta caso sabemos que deve existir um breakpoint tipo um (π_k, π_{k+1}), tal que $i < k < j$ (caso *iii*, Lema 4.3.3). 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 elemento (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 formam uma adjacência intergênica, são 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 4.3.3.

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

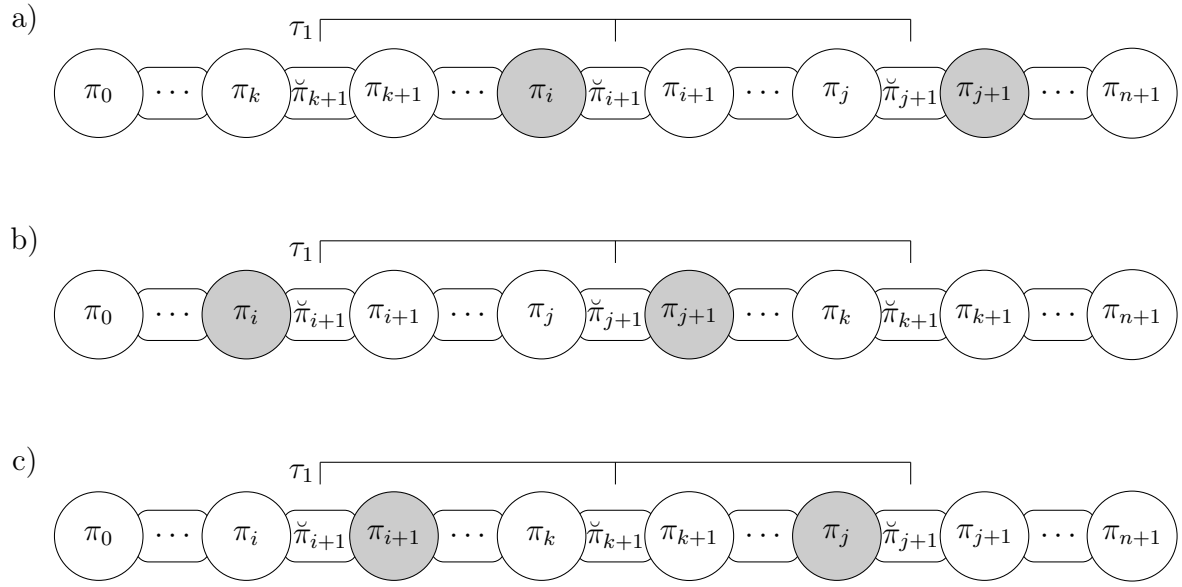


Figura 4.3: Exemplo de remoção de pelo menos um breakpoint tipo um após a aplicação de uma transposição τ.

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

Lema 4.3.16. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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 2.4.1 e Lema 4.3.2), o seguinte procedimento é aplicado: pelos lemas 4.3.1 e 4.3.15, 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. □

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

Seja $S \leftarrow ()$
enquanto $ib(\mathcal{I}) > 1$ **faça**
  `// Lema 4.3.1`
  (π_i, π_{i+1}), $(\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  `// Lema 4.3.15`
  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**
    $S' \leftarrow (\rho_1, \rho_2)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorna S

Note que o Algoritmo 8 difere do Algoritmo 4 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 4.3.17. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 8 é uma* 6*-aproximação para o problema* $\boldsymbol{Sb_IRT}$*.*

Demonstração. Pelo Lema 4.3.16, 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 $d_{\mathbf{Sb_IRT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. 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 4.3.18. *Dada uma representação intergênica rígida* $(\pi, \breve{\pi})$ *e duas transposições consecutivas no formato:*

$$(\pi, \breve{\pi}) \cdot \tau^{(i,j,k)}_{(\varphi_i, \varphi_j, \varphi_k)} \cdot \tau^{(i,i+k-j,k)}_{(\varphi'_i, \varphi'_{i+k-j}, \varphi'_k)},$$

então é possível realizar qualquer redistribuição de nucleotídeos nas regiões intergênicas $\breve{\pi}_i$*,* $\breve{\pi}_j$ *e* $\breve{\pi}_k$*.*

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 esta instância que satisfaça as restrições de redistribuição das regiões intergênicas $\breve{\pi}_i$, $\breve{\pi}_j$ e $\breve{\pi}_k$.

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 do Lemma 4.3.18). Note que cada região intergênica pode enviar nucleotídeos para duas regiões intergênicas distintas. Assim, podemos criar um grafo, onde cada vértice corresponde a uma região intergênica, e onde existe um arco (i, j) com capacidade ilimitada se a região intergênica i pode enviar nucleotídeos para a região intergênica j.

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. Figura 4.4 (direita) mostra a instância I_{MF} do problema de fluxo máximo obtido.

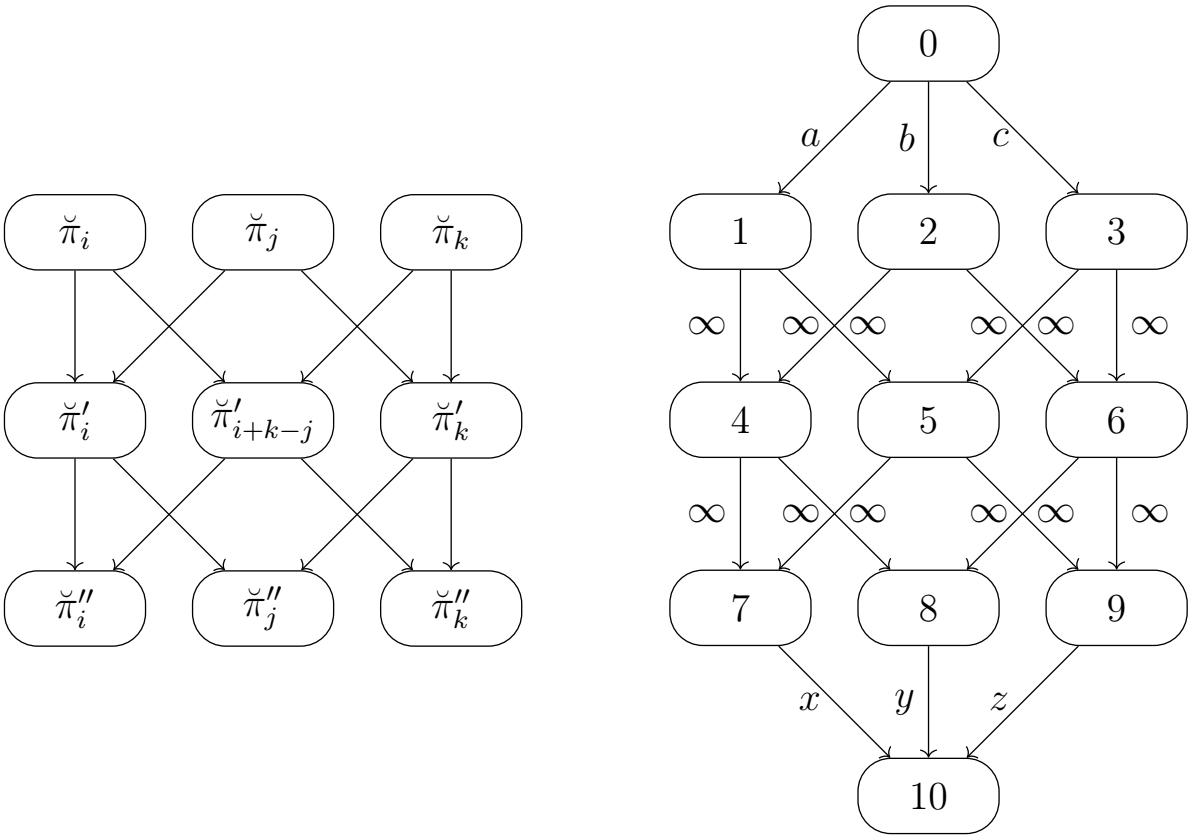


Figura 4.4: No lado esquerdo, uma representação do fluxo de nucleotídeos entre as regiões intergênicas. No lado direito, a instância I_{MF} do problema de fluxo máximo com os vértices de origem (0) e destino (10).

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 fracionária. Observe que o fluxo máximo para esta etapa é menor ou igual a x, pois $(a+b+c)\geq x$. Além disso, existe uma solução que atinge exatamente x (por exemplo, envie a pelo caminho $(1,4,7)$; envie b pelo caminho $(2,4,7)$; envie 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 este 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 esta etapa é menor ou igual a y, pois $a'+b'+c'\geq y$, e existe uma solução que atinge exatamente y (por exemplo, envie a' pelo caminho $(1,4,8)$; envie b' pelo caminho $(2,4,8)$; envie 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 esta etapa é igual a z, pois $a''+b''+c''=z$, e existe uma solução que atinja 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.

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

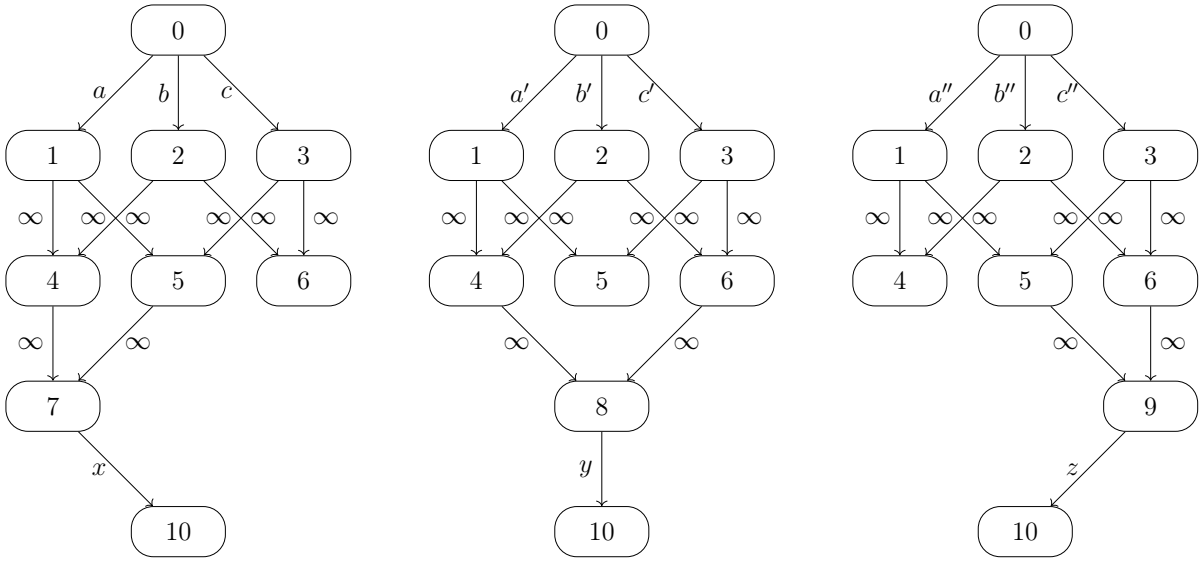


Figura 4.5: Divisão da instância I_{MF} do problema de fluxo máximo em três etapas.

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

Lema 4.3.19. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *com pelo menos dois breakpoints sobrecarregados, então existe uma sequência de duas transposições que remove pelo menos dois breakpoints tipo um de* $\mathcal{I}$.

Demonstração. Seja (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 4.3.18, 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. Neste caso, o excesso de nucleotídeos nos breakpoits 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 4.3.20. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *com apenas um breakpoint sobrecarregado* (π_i, π_{i+1}) *e pelo menos um breakpoint subcarregado* (π_j, π_{j+1})*, tal que* (π_i, π_{i+1}) *e* (π_j, π_{j+1}) *estão conectados, então 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 a 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 obrigatoriamente ser de um dos tipos da lista de prioridade. Em seguida, aplicamos o mesmo processo descrito no Lema 4.3.19. Caso contrário, 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 4.3.3, e o lema segue. □

Lema 4.3.21. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *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* $y = \max(\pi_j, \pi_{j+1})$*, então 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 4.3.3, 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 4.3.22. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *sem breakpoints sobrecarregados e com* $ib_1(\mathcal{I}) > 0$*, então deve existir em* $\mathcal{I}$ *pelo menos um par suavemente conectado de breakpoints.*

Demonstração. Suponha por contradição que $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ sem breakpoints sobrecarregados e $ib_1(\mathcal{I}) > 0$ é uma instância intergênica rígida balanceada sem sinais, sem breakpoints sobrecarregados, com $ib_1(\mathcal{I}) > 0$ e não existe em $\mathcal{I}$ 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 contradiz a suposição de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. 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 é suficiente 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 a suposição de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. □

Lema 4.3.23. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *e sejam* (π_i, π_{i+1}) *e* (π_j, π_{j+1}) *breakpoints suavemente conectados, então é possível remover pelo menos um breakpoint tipo um de* $\mathcal{I}$ *utilizando no máximo uma reversão ou uma transposição.*

Demonstração. O Lema 4.3.15 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. □

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

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
1  Seja S ← ()
2  enquanto ib(I) > 1 faça
3      se existir pelo menos dois breakpoints sobrecarregados em I então
           // Lema 4.3.19
4          S' ← (τ1, τ2)
5      senão se existir apenas um breakpoints sobrecarregado (πi, πi+1) em I então
6          se existir um breakpoints subcarregado (πj, πj+1) em I, tal que (πi, πi+1) e
           (πj, πj+1) estão conectados então
               // Lema 4.3.20
7              S' ← (τ1, τ2) or (ρ1, ρ2)
8          senão
               // Lema 4.3.21
9              S' ← (ρ1, ρ2)
10         
11     senão
           // Lemas 4.3.22 e 4.3.23
12         (πi, πi+1), (πj, πj+1) ← encontre um par suavemente conectado de
           breakpoints
13         se (πi, πi+1), (πj, πj+1) pertence ao caso i então
14             S' ← (ρ1)
15         senão se (πi, πi+1), (πj, πj+1) pertence ao caso ii então
16             S' ← (τ1)
17         senão se (πi, πi+1), (πj, πj+1) pertence ao caso iii então
18             S' ← (τ1)
19     
20     S ← S + S'
21     I ← ((π, π̆) · S', (ι, ῐ))
22 retorna S
```

Lema 4.3.24. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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, 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 algoritmo chegou até esse ponto 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.

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 reverão 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 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 4.3.25. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 9 é uma* 4.5*-aproximação para o problema* $\boldsymbol{Sb_IRT}$*.*

Demonstração. Pelo Lema 4.3.24, 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 $d_{\mathbf{Sb_IRT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. 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 4.3.26. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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 existe 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 a suposição de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. Neste 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 é suficiente 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 a suposição de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. □

Lema 4.3.27. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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* (π_i, π_{i+1}) *e* (π_j, π_{j+1}) *estão conectados.*

Demonstração. Se (π_i, π_{i+1}) e (π_j, π_{j+1}) estão conectados, 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 $y = \max(\pi_j, \pi_{j+1})$. 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 breakpoits 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 a suposição de que $\mathcal{I}$ é uma instância intergênica rígida balanceada. □

Lema 4.3.28. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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. Pelo Lema 4.3.27, sabemos que (π_i, π_{i+1}) e (π_j, π_{j+1}) estão conectados. Logo, podemos aplicar o Lema 4.3.20, e o lema segue. □

Note que a sequência de transposições aplicadas pelo Lema 4.3.28, gera no máximo um breakpoint sobrecarregado. Entretanto, se isso ocorrer implica que a instância $\mathcal{I}$ não possui breakpoints suaves. Este fato é decorrente da lista de prioridade para a seleção do terceito breakpoint no Lema 4.3.20. Uma vez que adicionar ou remover nucleotídeos em uma região intergênica de um breakpoint suave não transforma-o em um breakpoint forte.

Observação 4.3.1. Nenhum breakpoint super forte pode ser removido por uma operação de reversão ou transposição resultante do Lema 4.3.23.

Lema 4.3.29. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, se* $\mathcal{I}$ *não possui um par suavemente conectado de breakpoints, então é possível, após aplicar uma reversão ou um 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 $\mathcal{I}$, deve existir 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 [47]. 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, 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 [64]. 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) seja um breakpoint super forte subcarregado, e o lema segue. □

Lema 4.3.30. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *com apenas um breakpoint sobrecarregado, sem nenhum breakpoint subcarregado e não existe nenhum par suavemente conectado de breakpoints em* $\mathcal{I}$*, então 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 4.3.29, temos duas possibilidades: (i) um breakpoint subcarregado é criado mantendo $\mathcal{I}$ sem um par suavemente conectado de breakpoints, então podemos aplicar o Lema 4.3.28 (resultando em uma sequência com três operações que remove pelo menos dois breakpoints tipo um); (ii) um breakpoint super forte subcarregado é criado. Neste caso, se $\mathcal{I}$ continuar sem nenhum par suavemente conectado de breakpoints, então podemos aplicar o Lema 4.3.28 (resultando também em uma sequência com três operações que remove pelo menos dois breakpoints tipo um). Caso contrário, o Lema 4.3.23 pode ser aplicado. Pela Observação 4.3.1, nenhum breakpoint super forte pode ser removido por uma operação de reversão ou transposição resultante do Lema 4.3.23. Logo um dos seguintes casos pode ocorrer:

- Um novo breakpoint sobrecarregado é criado, e podemos aplicar o Lema 4.3.19 (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 4.3.23 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 4.3.28 é aplicado (resultando em uma sequência com quatro operações que remove pelo menos três breakpoints tipo um).

□

Observação 4.3.2. Note que se apenas dois breakpoints tipo um forem removidos pelo Lema 4.3.30, então isso implica que o genoma alvo ainda não foi alcançado, ou seja, $(\pi, \breve{\pi})$ é diferente de $(\iota, \breve{\iota})$.

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

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
Seja S ← ()
enquanto ib(I) > 1 faça
    se existir pelo menos dois breakpoints sobrecarregados em I então
        // Lema 4.3.19
        Seja S' uma sequência com duas transposições que remove pelo menos dois
        dois breakpoints tipo um de I
    senão se existir um par suavemente conectado de breakpoints em I então
        // Lema 4.3.23
        Seja S' uma sequência com uma reversão ou uma transposição que remove
        pelo menos um breakpoint tipo um de I
    senão
        // Existe exatamente um breakpoint sobrecarregado (Lema 4.3.26)

        se existir pelo menos um breakpoint subcarregado em I então
            // Lema 4.3.28
            Seja S' uma sequência com duas transposições ou duas reversões que
            remove pelo menos dois dois breakpoints tipo um de I
        senão
            // Lema 4.3.30
            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

    S ← S + S'
    I ← ((π, π̆) · S', (ι, ῐ))
retorna S
```

Lema 4.3.31. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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 sem nenhum um par suavemente conectado de breakpoints (linhas 8-9).
- $\mathcal{I}$ tem apenas um breakpoint sobrecarregado, nenhum breakpoint subcarregado e sem nenhum um par suavemente conectado de breakpoints (linhas 10-11).

Note que, a cada iteração do Algoritmo 9, 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 4.3.26) e sem nenhum um par suavemente conectado de breakpoints. Caso contrário, o caso 1 ou 2 seria aplicado.

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 4.3.2 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, e segue o lema. □

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 4.3.32. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 9 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRT}$*.*

Demonstração. Pelo Lema 4.3.31, 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 $d_{\mathbf{Sb_IRT}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

4.3.6 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 4.3.33. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *e seja* S *uma sequência de reversões, transposições e indels que transforma* $(\iota, \breve{\iota})$ *em* $(\pi, \breve{\pi})$*, então 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 um reversão $\rho^{(i,j)}_{(x,y)}$, então adicione em S' a reversão $\rho^{(i,j)}_{(x,x')}$.
- Se β for um 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 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 em que as operações são aplicadas também são invertidas. 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.

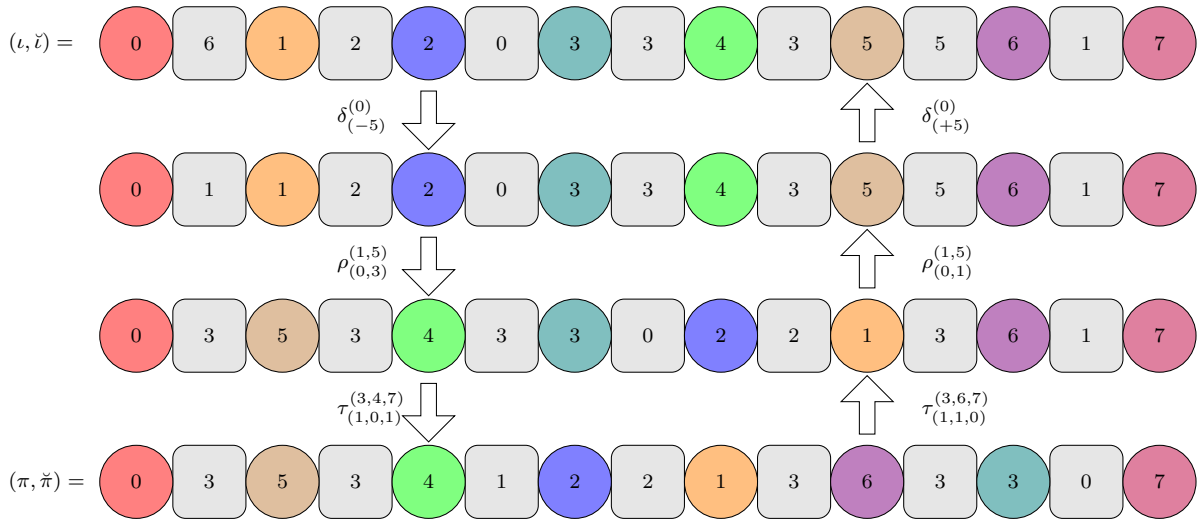


Figura 4.6: Exemplo de construção de uma sequência S' que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ a partir de uma sequência S que transforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$.

Observação 4.3.3. 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 $(\pi, \breve{\pi})$.

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 o genes do genoma de origem (elementos da permutação π) com os valores padrão da permutação identidade e também 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.

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

Algoritmo 11: Um algoritmo de aproximação para o problema $\mathbf{Sb_IRTI}$.

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 um sequência de operações fornecida pelo Algoritmo 10 para a instância $\mathcal{I}$
  retorna S
senão
  se $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$ **então**
    `// Lema 4.3.6`
    Seja δ_1 um indel que torna $\mathcal{I}$ em uma instância balanceada
    $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot \delta_1, (\iota, \breve{\iota}))$
    Seja S um sequência de operações fornecida pelo Algoritmo 10 para a instância $\mathcal{I}$
    retorna $(\delta_1) + S$
  senão
    $\mathcal{I}' \leftarrow ((\pi^{-1} \circ \iota, \breve{\iota}), (\pi^{-1} \circ \pi, \breve{\pi}))$
    `// Lema 4.3.6`
    Seja δ_1 um indel que torna $\mathcal{I}'$ em uma instância balanceada
    $\mathcal{I}' \leftarrow ((\pi^{-1} \circ \iota, \breve{\iota}) \cdot \delta_1, (\pi^{-1} \circ \pi, \breve{\pi}))$
    Seja S um sequência de operações fornecida pelo Algoritmo 10 para a instância $\mathcal{I}'$
    $S \leftarrow (\delta_1) + S$
    `// Lema 4.3.33`
    Seja S' uma sequência, criada a partir de S e $\mathcal{I}$, que transforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$
    retorna S'

Lema 4.3.34. *Dada uma instância intergênica rígida sem sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, 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 4.3.31, a sequência eventos fornecida pelo Algoritmo 10 transnforma $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$ utilizando no máximo $\frac{4b_1(\mathcal{I})}{3}$ eventos de reversão e transposição. Caso contrário, temos duas possibilidades:

- $\sum_{i=1}^{n+1} \breve{\pi}_i < \sum_{i=1}^{n+1} \breve{\iota}_i$: Neste caso, pelo Lema 4.3.6, existe um indel que torna $\mathcal{I}$ uma instância intergênica rígida balanceada e em seguida o Algoritmo 10 pode ser aplicado, resultado no máximo em $\frac{4ib_1(\mathcal{I})}{3} + 1$ eventos de reversão, transposição e indel para transnformar $(\pi, \breve{\pi})$ em $(\iota, \breve{\iota})$.

- $\sum_{i=1}^{n+1} \breve{\pi}_i > \sum_{i=1}^{n+1} \breve{\iota}_i$: Neste caso, pela Observação 4.3.3,uma instância auxiliar $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$ é criada, onde $\breve{\pi}' = \breve{\iota}$, $\breve{\iota}' = \breve{\pi}$ e $\sum_{i=1}^{n+1} \breve{\pi}'_i < \sum_{i=1}^{n+1} \breve{\iota}'_i$. Pelo Lema 4.3.6, existe um indel que torna $\mathcal{I}'$ uma instância intergênica rígida balanceada e em seguida o Algoritmo 10 pode ser aplicado, resultado em uma sequência S com no máximo $\frac{4ib_1(\mathcal{I})}{3}+1$ eventos de reversão, transposição e indel que transnforma $(\iota, \breve{\iota})$ em $(\pi, \breve{\pi})$. Pelo Lema 4.3.33, 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 Algoritmo 11 não possui laços de repetição e a subrotina com maior gasto computacional de tempo é decorrente do uso do Algoritmo 10 ($\mathcal{O}(n^2)$). Logo, o tempo de execução do algoritmo Algoritmo 11 também é $\mathcal{O}(n^2)$.

Teorema 4.3.35. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 11 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRTI}$*.*

Demonstração. Pelo Lema 4.3.34, 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. Caso $\mathcal{I}$ seja uma instância intergênica rígida balanceada, $d_{\mathbf{Sb_IRTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Caso contrário, $d_{\mathbf{Sb_IRTI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})+2}{3}$. 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. □

4.3.7 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 4.3.36. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *e sejam* (π_i, π_{i+1}) *e* (π_j, π_{j+1}) *breakpoints conectados, então é possível remover pelo menos um breakpoint tipo um de* $\mathcal{I}$ *utilizando no máximo 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, uma das seguintes possibilidades deve ocorrer:

i. O par de elementos (π_i, π_j) ou (π_{i+1}, π_{j+1}) não formam uma adjacência intergênica, são 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 4.3.3.

ii. O par de elementos (π_i, π_{j+1}) não formam uma adjacência intergênica, são 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 4.3.15.

iii. O par de elementos (π_{i+1}, π_j) não formam uma adjacência intergênica, são 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 4.3.15.

iv. O par de elementos (π_i, π_{i+1}) ou (π_j, π_{j+1}) não formam uma adjacência intergênica, são 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 4.3.10.

Note que os casos *i*, *ii*, *iii* e *iv* removem pelo menos um breakpoint tipo um de $\mathcal{I}$ utilizando no máximo um eventos de reversões, transposição ou move. Logo, o lema segue. □

A seguir apresentamos, para a variação sem sinais do problema **Sb$_\mathbf{I}$RTM**, o Algoritmo 12.

Algoritmo 12: Um algoritmo de aproximação para o problema **Sb$_\mathbf{I}$RTM**.

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 $ib(\mathcal{I}) > 1$ **faça**
  `// Lema 4.3.1`
  (π_i, π_{i+1}), $(\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  `// Lema 4.3.36`
  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**
    $S' \leftarrow (\mu_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorna S

Lema 4.3.37. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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 2.4.1 e Lema 4.3.2), o seguinte procedimento é aplicado: pelos lemas 4.3.1 e 4.3.36, sempre podemos encontrar um par conectado de breakpoints e remover pelo menos um breakpoint tipo um após aplicar no máximo 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 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. □

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 4.3.38. *Dada uma instância intergênica rígida balanceada sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 9 é uma* 3*-aproximação para o problema* $\boldsymbol{Sb_IRTM}$*.*

Demonstração. Pelo Lema 4.3.37, 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 $d_{\mathbf{Sb_IRTM}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

4.3.8 Reversão, Transposição, Move e Indel

Nesta seção apresentaremos um algoritmo de aproximação com fator 3 para a variação sem sinais do problema $\mathbf{Sb_IRTMI}$. A seguir apresentamos o Algoritmo 13.

Lema 4.3.39. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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 a descrita no Lema 4.3.8. □

Podemos notar que Algoritmo 13, em comparação com o Algoritmo 9, possui adicionalmente apenas subrotinas que podem ser feitas em tempo constante. Logo, o tempo de execução do Algoritmo 13 é $\mathcal{O}(n^2)$.

Teorema 4.3.40. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 13 é uma* 3*-aproximação para o problema* $\boldsymbol{Sb_IRTMI}$*.*

Demonstração. Pelo Lema 4.3.39, 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 $d_{\mathbf{Sb_IRTMI}}(\mathcal{I}) \geq \frac{ib_1(\mathcal{I})}{3}$. Logo, o teorema segue. □

4.3.9 Resultados Práticos

Nesta seção apresentamos os resultados práticos dos algoritmos apresentados para a varaiação sem sinas dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ e $\mathbf{Sb_IRTMI}$.

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 4.3.6
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 $ib(\mathcal{I}) > 1$ **faça**
  // Lema 4.3.1
  (π_i, π_{i+1}), $(\pi_j, \pi_{j+1}) \leftarrow$ encontre um par de breakpoints conectados
  // Lema 4.3.36
  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 4.3.7
se $ib(\mathcal{I}) = 1$ **então**
  $S' \leftarrow (\delta_1)$
  $S \leftarrow S + S'$
  $\mathcal{I} \leftarrow ((\pi, \breve{\pi}) \cdot S', (\iota, \breve{\iota}))$
retorna S

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. 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 genona 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 $\mathcal{S} = \mathcal{S} \cdot \sigma$. 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 $[-\breve{\pi}_i..2\breve{\pi}_i]$, também de maneira aleatória. Quando não houverem operações disponíveis para serem aplicadas, então temos a instância $\mathcal{I}$, que é 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.

A quantidade de operações disponíveis para gerar cada instância é baseada em uma porcentagem do identificador de cada grupo e difere entre as bases de dados. A Tabela 4.2 mostra, para cada base de dados, a porcentagem adotada por operação ao criar uma instância.

Tabela 4.2: 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%

Note que, para cada base de dados, a soma das porcentagens adotadas para cada operação é igual a 50%. Logo, toda instância de um grupo G foi gerada a partir de $\frac{G}{2}$ operações, sendo que o a quantidade de operações por tipo dependende 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 4.3 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 4.3: 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 4.3, 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, os fatores de aproximação observados pelo algoritmo na base de dados foram significativamente menores do que o limite teórico provado para o algoritmo.

A Tabela 4.4 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 4.4: 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 4.4, podemos perceber que o Algoritmo 5, em comparação com o Algoritmo 4, apresentou uma distância média menor em todos os grupos, mas apresentou uma aproximação média superior em todos os grupos. 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 4.5 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.

Tabela 4.5: 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

Podemos perceber pela Tabela 4.5 que o Algoritmo 6 apresenta um comportamento similar aos algoritmos 4 e 5, mas com uma variação na aproximação média levemente maior entre os grupos. A maior e menor aproximação observadas foram, respectivamente, 3.16 e 2.22, com ambos os registros ocorrendo no grupo 100.

A Tabela 4.6 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 4.6: 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 4.6 podemos observar que a aproximação média do Algoritmo 7 variou de 2.71 até 2.80. A menor e 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 a aproximação mínima e máxima com o valor de 1.04. O grupo que apresentou a menor variação entre a aproximação 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 4.7 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 4.7, podemos perceber que os três algoritmos apresentaram resultados bem similares considerando distância média 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 do que a distância média fornecida média fornecida pelo Algoritmo 9, em todos os grupos. Note que a diferença foi extremamente pequena, sendo que a diferência absoluta para cada grupo foi menor do que 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 desempenho prático dos três algoritmos apresentando fatores de aproximação significativamente menores do que os limites teóricos provados para cada um deles.

A Tabela 4.8 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.

Pela Tabela 4.8, 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 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

Tabela 4.7: 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

Tabela 4.8: 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

observada em todos os grupos foi menor que 3.00.

A Tabela 4.9 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.

Na Tabela 4.9, 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 maior aproximação foram 2.76 e 2.99, respectivamente. A menor aproximação foi registrada no grupo

Tabela 4.9: 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

100, enquanto a maior aproximação foi registrada no grupo 500.

A Tabela 4.10 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 4.10: 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 4.10 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 apresentam um bom desempenho na prática. Vale ressaltar o desempenho dos algoritmos 8 e 9 para a variação sem sinais do problema $\mathbf{Sb_IRT}$, que mesmo possuindo um limite teórico mais elevado para a aproximação, apresentaram resultados compatíveis com os resultados do melhor algoritmo conhecido para o problema até o momento (Algoritmo 10).

4.3.10 Heurística Gulosa

Visando obter resultados práticos melhores pelos algoritmos apresentados para a variação sem sinais dos problemas investigados neste capítulo, nos propomos heurísticas baseadas em uma estratégia gulosa. As heurísticas consistem em verificar se existe em 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 consdera 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, 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**
  $S \leftarrow (\rho_1)$
retorna 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**
  $S \leftarrow (\mu_1)$
retorna 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 **Sb$_\mathbf{I}$R**, **Sb$_\mathbf{I}$RI**, **Sb$_\mathbf{I}$RM**, **Sb$_\mathbf{I}$RMI**, **Sb$_\mathbf{I}$RT**, **Sb$_\mathbf{I}$RTI**, **Sb$_\mathbf{I}$RTM** e **Sb$_\mathbf{I}$RTMI**, mantendo-se 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 **Sb$_\mathbf{I}$RM**, **Sb$_\mathbf{I}$RMI**, **Sb$_\mathbf{I}$RTM** e **Sb$_\mathbf{I}$RTMI**, alterando-se para $\mathcal{O}(n^2 \log n)$.

4.4 Instâncias Intergênicas Rígidas com Sinais

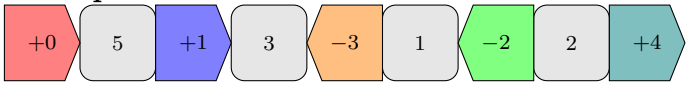
Nesta seção apresentamos algoritmos para os problemas resultantes dos modelos de rearranjo investigados neste capítulo e considerando uma representação intergênica rígida com sinais de um genoma. Inicialmente iremos apresentar algumas definições e lemas que serão utilizados por múltiplos algoritmos.

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.

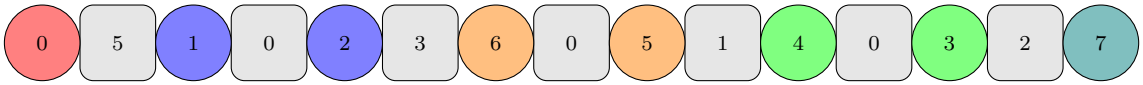
Definição 4.4.1. Dada uma representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi})$ com n genes, a *duplicação* cria uma representação intergênica rígida sem sinais $\mathcal{R}' = (\pi', \breve{\pi}')$ com $2n$ genes, de forma que cada elemento $\pi_i \in \pi$ é mapeado em dois 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, é atribuído para cada região intergênica $\breve{\pi}'_{2i-1}$ o tamanho $\breve{\pi}_i$, para $i \in [1..n+1]$, e para cada região intergênica $\breve{\pi}'_{2j}$ é atribuído o tamanho zero, para $j \in [1..n]$.

Dada uma representação intergênica rígida com sinais $\mathcal{R} = (\pi, \breve{\pi})$, denotamos por $\mathcal{D}(\mathcal{R})$ o reultado da transformação de duplicação aplicada em $\mathcal{R}$. O Exemplo 4.4.1 mostra a transformação de duplicação sendo aplicada em uma representação intergênica rígida com sinais $\mathcal{R} = ((+0\ -3\ +1\ +2\ +4), (5, 3, 1, 2))$.

Exemplo 4.4.1.



π = (+0 −3 +1 +2 +4)
$\breve{\pi}$ = (5, 3, 1, 2)



π' = (0 1 2 6 5 4 3 7)
$\breve{\pi}'$ = $(5, 0, 3, 0, 1, 0, 2)$

Lema 4.4.1. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$ *com* n *genes, então é possível criar 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. 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 sequinte forma: $(\pi', \breve{\pi}') = \mathcal{D}((\pi, \breve{\pi}))$ e $(\iota', \breve{\iota}') = \mathcal{D}((\iota, \breve{\iota}))$.

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 $|\pi'_{2i-1}$ e $\pi'_{2i}| = 1$ 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 4.4.2. *Dada uma instância intergênica rígida sem sinais* $\mathcal{I}' = ((\pi', \breve{\pi}'), (\iota', \breve{\iota}'))$, *tal que* $\mathcal{I}' = \mathcal{D}(\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota})))$. *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* $\mathcal{I}'$, *então é possível criar 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 um 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 um 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 um 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 causado 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.

4.4.1 Reversão

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema $\mathbf{Sb_IR}$. A seguir apresentamos o 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 4.4.1`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 4
`// Lema 4.4.2`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorna** S

Note que o tempo de execução do Algoritmo 16 será igual ao tempo de execução do Algoritmo 4, uma vez que para ler os dados de entrada é necessário um tempo linear e as funções $\mathcal{F}$ e $\mathcal{G}$ executam também em tempo linear.

Teorema 4.4.3. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 16 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IR}$*.*

Demonstração. Note que o Algoritmo 4 fornece um sequência de reversões que afetam apenas breakpoints tipo um. Logo, nunhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.3.4, Algoritmo 4 utiliza no máximo $2ib_1(\mathcal{I}')$ reversões para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$. Pelo Lema 4.4.2, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'| \leq 2ib_1(\mathcal{I}')$. Pelo Lema 4.4.1, 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 $d_{\mathbf{Sb_IR}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{2}$. Logo, o teorema segue. □

4.4.2 Reversão e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema $\mathbf{Sb_IRI}$. A seguir apresentamos o 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 4.4.1`
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 5
`// Lema 4.4.2`
3 $S \leftarrow \mathcal{G}(S')$
4 **retorna** S

Teorema 4.4.4. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 17 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRI}$*.*

Demonstração. A prova é similar a descrita no Teorema 4.4.3, 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 4.3.8). □

4.4.3 Reversão e Move

Nesta seção apresentaremos algoritmos de aproximação com fatores de aproximação 4 e 2 para a variação com sinais do problema $\mathbf{Sb_IRM}$. A seguir apresentamos o Algoritmo 18.

Algoritmo 18: 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 eventos de reversão e move S, tal que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

`// Lema 4.4.1`

1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$

2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 6

`// Lema 4.4.2`

3 $S \leftarrow \mathcal{G}(S')$

4 **retorna** S

Teorema 4.4.5. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 18 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRM}$*.*

Demonstração. A prova é similar a descrita no Teorema 4.4.3, 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 4.3.11). □

A seguir faremos uso da estrutura de grafo de ciclos ponderado rígido e apresentaremos lemas que serão utilizados para obtenção de uma algoritmo de aproximação com um fator 2.

Lema 4.4.6. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, tal que em* $G(\mathcal{I})$ *existe pelo menos um ciclo negativo* $C = (c^1)$*, então deve existir 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 pelo menos um ciclo negativo $C = (c^1)$ e não existe nenhum ciclo positivo. Como em $G(\mathcal{I})$ existe pelo menos um ciclos negativo em 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 4.4.7. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, tal que em* $G(\mathcal{I})$ *existe pelo menos um ciclo trivial negativo* $C = (c^1)$ *e pelo menos um ciclo* $D = (d^1, \ldots, d^k)$ *desbalanceado, então é possível aumentar o número de cilos 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 torna-se balanceado, e o lema segue. □

Lema 4.4.8. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, tal que em $G(\mathcal{I})$ existe pelo menos um ciclo divergente negativo ou balanceado C, então é possível aumentar o número de cilos de $G(\mathcal{I})$ em uma unidade e o número de cilos balanceados de $G(\mathcal{I})$ em uma unidade após aplicar uma operação de reversão.*

Demonstração. Direto do Lema 5 de Oliveira *et al.* [54]. □

Lema 4.4.9. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, tal que todos os ciclos não triviais negativos ou balanceados de $G(\mathcal{I})$ são convergentes, então é possível aumentar o número de cilos de $G(\mathcal{I})$ em uma unidade e o número de cilos balanceados de $G(\mathcal{I})$ em uma unidade após aplicar uma sequência de duas reversões.*

Demonstração. Direto do Lema 6 de Oliveira *et al.* [54]. □

Agora considere o Algoritmo 19 e observe que ele possui três passos: (i) operações de move aplicadas a ciclos negativos triviais (Lemas 4.4.6 e 4.4.7); (ii) reversões aplicadas em ciclos divergentes negativos ou balanceados (Lema 4.4.8); (iii) duas reversões consecutivas aplicadas em ciclos convergentes negativos ou balanceados (Lema 4.4.9).

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
Seja S ← ()
enquanto c(G(I)) < n + 1 e c_b(G(I)) < n + 1 faça
    se existir um ciclo trivial negativo em G(I) então
        // Lemas 4.4.6 e 4.4.7
        S' ← (μ_1)
    senão se existir um ciclo divergente negativo ou balanceado em G(I) então
        // Lema 4.4.8
        S' ← (ρ_1)
    senão
        // Lema 4.4.9
        S' ← (ρ_1, ρ_2)
    S ← S + S'
    I ← ((π, π̆) · S', (ι, ῐ))
retorna 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 mais ciclos balanceados criados por uma 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 4.4.10. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, 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 4.4.6 e 4.4.7. 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 4.4.8). 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 4.4.9 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 $2n$ vezes, então o tempo de execução do Algoritmo 19 é $\mathcal{O}(n^3)$.

Teorema 4.4.11. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 19 é uma* 2*-aproximação para o problema* $\boldsymbol{Sb_IRM}$*.*

Demonstração. Pelo Lema 4.4.10, 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 $d_{\mathbf{Sb_IRM}}(\mathcal{I}) \geq n+1-\frac{c(G(\mathcal{I}))+c_b(G(\mathcal{I}))}{2}$. Assim, o teorema segue. □

4.4.4 Reversão, Move e Indel

Nesta seção apresentaremos algoritmos de aproximação com fatores de aproximação 4 e 2 para a variação com sinais do problema $\mathbf{Sb_IRMI}$. A seguir apresentamos o Algoritmo 20.

Teorema 4.4.12. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 20 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRMI}$*.*

Demonstração. A prova é similar a descrita no Teorema 4.4.3, 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 4.3.13). □

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 4.4.1
```
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 7
```
// Lema 4.4.2
```
3 $S \leftarrow \mathcal{G}(S')$
4 **retorna** S

A seguir faremos uso da estrutura de grafo de ciclos ponderado rígido e apresentaremos lemas que serão utilizados para obtenção de uma algoritmo de aproximação com um fator 2.

Lema 4.4.13. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, tal que em $G(\mathcal{I})$ existe pelo menos um ciclo trivial negativo $C = (c^1)$, então é possível aumentar o número de cilos 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 areta preta e uma areta cinza. Neste caso, basta aplica um indel 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 4.4.14. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, tal que em $G(\mathcal{I})$ existe pelo menos um ciclo positivo C, então é possível aumentar o número de cilos balanceados de $G(\mathcal{I})$ em uma unidade após aplicar uma operação de indel.*

Demonstração. Neste caso, basta aplica 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 4.4.7 e 4.4.13); (ii) reversões aplicadas em ciclos divergentes negativos ou balanceados (Lema 4.4.8); (iii) duas reversões consecutivas aplicadas em ciclos convergentes negativos ou balanceados (Lema 4.4.9); (iv) indels aplicados em ciclos positivos (Lema 4.4.14).

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 4.4.15. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, 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

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 $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$

```
Seja S ← ()
enquanto c(G(I)) < n + 1 e c_b(G(I)) < n + 1 faça
    se existir um ciclo trivial negativo C em G(I) então
        se existir um ciclo desbalanceado D em G(I) então
            // Lema 4.4.7
            S' ← (μ_1)
        senão
            // Lema 4.4.13
            S' ← (δ_1)
    senão se existir um ciclo divergente negativo ou balanceado em G(I) então
        // Lema 4.4.8
        S' ← (ρ_1)
    senão se existir um ciclo convergente negativo ou balanceado em G(I) então
        // Lema 4.4.9
        S' ← (ρ_1, ρ_2)
    senão
        // Lema 4.4.14
        S' ← (δ_1)
    S ← S + S'
    I ← ((π, π̆) · S', (ι, ῐ))
retorna S
```

satisfeita o Algoritmo 21 garante que todo ciclo trivial negativo seja transformado em balanceado pelos lemas 4.4.7 e 4.4.13. 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 4.4.8). 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). Se existir pelo menos um ciclo convergente negativo ou balanceado, então o Lema 4.4.9 pode ser aplicado. Caso o passo (iii) não seja aplicado, então 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 balanceado em uma unidade (Lema 4.4.14). 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 difere do Algoritmo 19 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 4.4.16. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 21 é uma* 2*-aproximação para o problema* ***Sb$_I$RMI****.*

Demonstração. Pelo Lema 4.4.15, 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 $d_{\mathbf{Sb_I RMI}}(\mathcal{I}) \geq n + 1 - \frac{c(G(\mathcal{I}))+c_b(G(\mathcal{I}))}{2}$. Assim, o teorema segue. □

4.4.5 Reversão e Transposição

Nesta seção, com base na estrutura do Algoritmo 16, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema **Sb$_I$RT**. A seguir apresentamos o Algoritmo 22.

Algoritmo 22: Um algoritmo de aproximação para o problema **Sb$_I$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 4.4.1
```
$\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$

Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 10

```
// Lema 4.4.2
```
$S \leftarrow \mathcal{G}(S')$

retorna S

Teorema 4.4.17. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 22 é uma* 4*-aproximação para o problema* ***Sb$_I$RT****.*

Demonstração. Note que o Algoritmo 10 fornece um sequência de reversões e tranposições que afetam apenas breakpoints tipo um. Logo, nunhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.3.31, 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 4.4.2, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'| \leq \frac{4ib_1(\mathcal{I}')}{3}$. Pelo Lema 4.4.1, 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 $d_{\mathbf{Sb_I RT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$. Logo, o teorema segue. □

4.4.6 Reversão, Transposição e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 4 para a variação com sinais do problema **Sb$_I$RTI**. A seguir apresentamos o Algoritmo 23.

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 4.4.1
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 11
// Lema 4.4.2
3 $S \leftarrow \mathcal{G}(S')$
4 **retorna** S

Teorema 4.4.18. *Dada uma instância intergênica rígida com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 23 é uma* 4*-aproximação para o problema* $\boldsymbol{Sb_IRTI}$*.*

Demonstração. Note que o Algoritmo 11 fornece um sequência de reversões, tranposições e indels que afetam apenas breakpoints tipo um. Logo, nunhuma adjacência intergênica de $\mathcal{I}'$ é afetada. Além disso, pelo Lema 4.3.34, o Algoritmo 11 utiliza no máximo $\frac{4ib_1(\mathcal{I}')}{3}$ e $\frac{4ib_1(\mathcal{I}')}{3} + 1$ eventos de reversão, transposição e indel para transformar $(\pi', \breve{\pi}')$ em $(\iota', \breve{\iota}')$ se $\mathcal{I}'$ for balanceada e desbalanceada, respectivamente. Pelo Lema 4.4.2, temos que $(\pi, \breve{\pi}) \cdot S = (\iota, \breve{\iota})$ e $|S| = |S'|$. Pelo Lema 4.4.1, 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: Se $\mathcal{I}$ for balanceada, então $d_{\mathbf{Sb_IRT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})}{3}$. Caso contrário, $d_{\mathbf{Sb_IRT}}(\mathcal{I}) \geq \frac{ib_2(\mathcal{I})+2}{3}$.Se $\mathcal{I}$ for balanceada, temos que $\frac{\frac{4b_2(\mathcal{I})}{3}}{\frac{ib_2(\mathcal{I})}{3}} = 4$. 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 $\frac{4ib_2(\mathcal{I})+3}{ib_2(\mathcal{I})+2} < \frac{4(ib_2(\mathcal{I})+2)}{ib_2(\mathcal{I})+2} = 4$, e o teorema segue. □

4.4.7 Reversão, Transposição e Move

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação com sinais do problema $\mathbf{Sb_IRTM}$. A seguir apresentamos o Algoritmo 24.

Algoritmo 24: 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 4.4.1
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 12
// Lema 4.4.2
3 $S \leftarrow \mathcal{G}(S')$
4 **retorna** S

Teorema 4.4.19. *Dada uma instância intergênica rígida balanceada com sinais* $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$*, o Algoritmo 24 é uma* 3*-aproximação para o problema* $\boldsymbol{Sb_IRTM}$*.*

Demonstração. A prova é similar a descrita no Teorema 4.4.17, 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 4.3.37). □

4.4.8 Reversão, Transposição, Move e Indel

Nesta seção, apresentaremos um algoritmo de aproximação com fator 3 para a variação com sinais do problema **Sb$_\mathbf{I}$RTMI**. A seguir apresentamos o Algoritmo 25.

Algoritmo 25: Um algoritmo de aproximação para o problema **Sb$_\mathbf{I}$RTMI**.

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 4.4.1
```
1 $\mathcal{I}' \leftarrow \mathcal{F}(\mathcal{I})$
2 Seja S' uma sequência de eventos de reversão fornecida pelo Algoritmo 13
```
// Lema 4.4.2
```
3 $S \leftarrow \mathcal{G}(S')$
4 **retorna** S

Teorema 4.4.20. *Dada uma instância intergênica rígida com sinais $\mathcal{I} = ((\pi, \breve{\pi}), (\iota, \breve{\iota}))$, o Algoritmo 25 é uma 3-aproximação para o problema **Sb$_I$RTMI**.*

Demonstração. A prova é similar a descrita no Teorema 4.4.17, 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 4.3.39). □

4.4.9 Resultados Práticos

Nesta seção, apresentamos os resultados práticos dos algoritmos apresentados para a varaiação com sinas dos problemas **Sb$_\mathbf{I}$R**, **Sb$_\mathbf{I}$RI**, **Sb$_\mathbf{I}$RM**, **Sb$_\mathbf{I}$RMI**, **Sb$_\mathbf{I}$RT**, **Sb$_\mathbf{I}$RTI**, **Sb$_\mathbf{I}$RTM** e **Sb$_\mathbf{I}$RTMI**.

Nós também criamos 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$_\mathbf{I}$R**, **Sb$_\mathbf{I}$RI**, **Sb$_\mathbf{I}$RM**, **Sb$_\mathbf{I}$RMI**, **Sb$_\mathbf{I}$RT**, **Sb$_\mathbf{I}$RTI**, **Sb$_\mathbf{I}$RTM** e **Sb$_\mathbf{I}$RTMI**, respectivamente. As bases de dados foram criadas de forma similar ao processo descrito na Seção 4.3.9, diferindo apenas que cada instância foi criada a partir de um par de representação intergênica rígida com sinais. Logo, ao aplicar um evento de reversão os genes no segmento afetado também acabam tendo a orientação invertida.

Além das bases de dados criadas por nós, utilizamos também duas bases de dados apresentadas por Oliveira *et al.* [56]. 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 e cada valor de $\breve{\iota}_i$, com $1 \leq i \leq 101$, sendo aleatoriamente e de maneira uniforme escolhido 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:

- $\mathrm{DB}_{\mathbf{Sb_IRT}}$: d operações aleatórias de reversões e transposições (sendo 50% de cada) em cada genoma alvo $(\iota, \breve{\iota})$.

- $\mathrm{DB}_{\mathbf{Sb_IRTM}}$: d operações aleatórias de reversões e transposições genéricas (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 d variou de 10 até 100, em intervalos de 10. Para cada valor de d, foi gerado um grupo com 10.000 instâncias. As bases de dados $\mathrm{DB}_{\mathbf{Sb_IRT}}$ e $\mathrm{DB}_{\mathbf{Sb_IRTM}}$ têm um total de 100.000 instâncias cada.

A seguir apresentamos os resultados dos algoritmos propostos para a variação com sinais dos problemas $\mathbf{Sb_IR}$, $\mathbf{Sb_IRI}$, $\mathbf{Sb_IRM}$, $\mathbf{Sb_IRMI}$, $\mathbf{Sb_IRT}$, $\mathbf{Sb_IRTI}$, $\mathbf{Sb_IRTM}$ 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 4.11 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 4.11: 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 4.11, podemos perceber que, em média, a distância fornecida pelo Algoritmo 16 é um valor próximo de 90% do tamamho 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 o fator 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 [54] baseado na estrutura de grafo de ciclos. O Algoritmo 16 garante um fator de aproximação 4, mas utiliza o conceito de breakpoint.

A Tabela 4.12 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.

Pela Tabela 4.12, podemos notar que o Algoritmo 17 apresentou um desempenho similar ao Algoritmo 16. É importante mencionar que o fator de aproximação médio

Tabela 4.12: 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

foi levemente maior em todos os grupos. Entretanto, a distância média fornecida pelo Algoritmo 17, em todos os grupos, foi menor.

As Tabela 4.13 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 4.13: 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, pela Tabela 4.13, 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, 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 e inferior ao 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 pra o grupo 100 apresentou um

valor de 1.13 e para os demais grupo apresentou um valor 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 4.14 mostra os resultados dos algoritmo 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

Tabela 4.14: 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

Pela Tabela 4.14, podemos notar um comportamento similar ao observado nos algoritmos da Tabela 4.13, com o Algoritmo 21 (baseado na estrutura de grafo de ciclos ponderado rígido) tendendo a fornecer soluções melhores para a variação com sinais do problema $\mathbf{Sb_IRMI}$, em comparação com o Algoritmo 20 (baseado no conceito de breakpoints). É importante notar que em ambos os algoritmos e em todos os grupos houve um queda no valor da distância média em comparação com os resultados dos algoritmos da Tabela 4.13.

A Tabela 4.15 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.

Pela Tabela 4.15, podemos notar que o fator de aproximação máximo observado pelo Algoritmo 22 foi de 3.0 e ocorreu em todos os grupos. Além disso, o fator de aproximação médio tende a subir conforme o tamanho das instâncias cresce. 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 $\mathbf{Sb_IRT}$ um algoritmo com fator de aproximação 3 (baseado na estrutura de grafo de ciclos ponderado rígido), que iremos

Tabela 4.15: 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

chamar de 3$\mathbf{Sb_IRT}$. Como os fatores de aproximação dos algoritmos 22 e 3$\mathbf{Sb_IRT}$ estão próximos, nós realizamos um comparativo entre eles utilizando a base de dados $\mathrm{DB}_{\mathbf{Sb_IRT}}$. O Algoritmo 22 foi executado adotando a heurística I, Algoritmo 14. A Tabela 4.16 mostra os resultados dos algoritmos 22 e 3$\mathbf{Sb_IRT}$ utilizando a base de dados $\mathrm{DB}_{\mathbf{Sb_IRT}}$. As colunas M e ME indicam, para cada grupo, a porcentagem de soluções fornecidas pelo Algoritmo 22 com tamanho estritamente menor e com tamanho menor ou igual, respectivamente, 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 [56, Teorema 3.8].

Tabela 4.16: Comparação entre os algoritmos 22 e 3$\mathbf{Sb_IRT}$ utilizando a base de dados $\mathrm{DB}_{\mathbf{Sb_IRT}}$.

OP	3$\mathbf{Sb_IRT}$		Algoritmo 22		M	ME
	Distância Média	Aproximação Média	Distância Média	Aproximação Média		
10	12.60	1.68	12.19	1.63	51.37%	68.70%
20	26.37	1.77	25.60	1.71	54.72%	66.48%
30	40.23	1.81	36.91	1.66	78.60%	85.53%
40	53.29	1.85	46.10	1.60	95.84%	97.62%
50	63.78	1.87	53.21	1.56	99.46%	99.74%
60	71.88	1.89	58.97	1.55	99.81%	99.92%
70	77.83	1.90	63.29	1.55	99.97%	99.98%
80	82.45	1.91	66.73	1.55	99.95%	99.99%
90	85.98	1.92	69.48	1.55	99.97%	99.99%
100	88.78	1.93	71.64	1.56	99.99%	99.99%

A partir da Tabela 4.16, é possível observar que o Algoritmo 22, em todos os grupos, foi capaz de fornecer melhores resultados considerando as métricas de de aproximação média e distância média. Além disso, considerando os grupos com d maior que 20, o algoritmo forneceu melhores soluções em mais de 75% das instâncias (coluna M). Considerando os grupos com d maior que 30, o algoritmo Algoritmo 22 forneceu soluções de tamanho melhor ou equivalente (coluna ME) em mais de 97% das instâncias. É importante observar que, à medida que o valor de d aumenta, a diferença absoluta entre a distância média fornecida pelos algoritmos 22 e 3$\mathbf{Sb_IRT}$ também aumenta significativamente. Quando d é maior que 50, a diferença absoluta entre as distâncias médias é superior a 10, o que indica que o algoritmo Algoritmo 22 tende a oferecer melhores soluções em cenários onde

foi utilizado um número maior de operações.

A Tabela 4.17 mostra os resultados do Algoritmo 23 utilizando a base de dados $S_{\mathbf{Sb_IRTI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.12.

Tabela 4.17: Resultados do Algoritmo 23 utilizando a base de dados $S_{\mathbf{Sb_IRTI}}$.

-	-	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

Pela Tabela 4.17, podemos perceber que o Algoritmo 23 apresentou um comportamento similar ao Algoritmo 22 na Tabela 4.15. Um ponto de diferença foi que 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.

A Tabela 4.18 mostra os resultados do Algoritmo 24 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 4.18: Resultados do Algoritmo 24 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 4.18, é possível notar que as métricas de mínimo, média e máximo para a aproximação fornecida pelo Algoritmo 24 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 $\mathbf{Sb_IRTM}$ um algoritmo com fator de aproximação 2.5 (baseado na estrutura de grafo de ciclos ponderado rígido), que iremos chamar de 2.5$\mathbf{Sb_IRTM}$. Como os fatores de aproximação dos algoritmos 24 e 2.5$\mathbf{Sb_IRTM}$ estão próximos, nós realizamos um comparativo entre eles utilizando a base de dados $\mathrm{DB}_{\mathbf{Sb_IRTM}}$. O Algoritmo 24 foi executado adotando a heurística I, Algoritmo 14. A Tabela 4.19 mostra os resultados dos algoritmos 24 e 2.5$\mathbf{Sb_IRTM}$ utilizando a base de dados $\mathrm{DB}_{\mathbf{Sb_IRTM}}$. As colunas M e ME indicam, para cada grupo, a porcentagem de

soluções fornecidas pelo Algoritmo 24 com tamanho estritamente menor e com tamanho menor ou igual, respectivamente, 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 [56, Teorema 7.6].

Tabela 4.19: Comparação entre os algoritmos 24 e 2.5$\mathbf{Sb_IRT}$ utilizando a base de dados $DB_{\mathbf{Sb_IRTM}}$.

OP	2.5$\mathbf{Sb_IRTM}$		Algoritmo 24		M	ME
	Distância Média	Aproximação Média	Distância Média	Aproximação Média		
10	11.79	1.57	12.25	1.64	32.30%	53.18%
20	24.97	1.68	25.57	1.72	34.77%	49.58%
30	38.21	1.74	36.69	1.67	63.31%	73.75%
40	50.47	1.79	45.61	1.62	89.60%	93.69%
50	60.46	1.81	52.65	1.58	97.69%	98.75%
60	68.17	1.83	58.19	1.56	99.45%	99.78%
70	74.06	1.84	62.54	1.55	99.75%	99.90%
80	78.68	1.85	65.96	1.55	99.93%	99.96%
90	82.09	1.85	68.67	1.55	99.92%	99.97%
100	84.85	1.86	70.82	1.55	99.97%	100.00%

Pela Tabela 4.19, podemos notar que o algoritmo 2.5$\mathbf{Sb_IRTM}$, quando comparado ao Algoritmo 24 , apresentou um resultado um pouco melhor em relação à aproximação média e distância média nos grupos com $d = 10$ e $d = 20$. Considerando esses dois grupos ($d = 10$ e $d = 20$), a diferença absoluta entre a distância média fornecida pelos algoritmos foi menor que 0.61. Além disso, pela coluna M, podemos notar que nos grupos $d = 10$ e $d = 20$ o Algoritmo 24 forneceu melhores soluções em 32.30% e 34.77% das instâncias, respectivamente. Isso mostra que o Algoritmo 24 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 d é maior que 20, o algoritmo Algoritmo 24 forneceu melhores resultados considerando a aproximação média e distância média. Além disso, nos mesmos grupos, o Algoritmo 24 forneceu soluções de tamanho menor ou equivalente (coluna ME) em mais de 73% das instâncias.

A Tabela 4.20 mostra os resultados do Algoritmo 25 utilizando a base de dados $S_{\mathbf{Sb_IRTMI}}$. Os fatores de aproximação foram computados utilizando o limitante inferior apresentado no Teorema 4.1.11.

Pela Tabela 4.20, é possível notar que em pelo menos uma instância de cada os grupo foi observado um fator de aproximação (computado com base no limitante inferior) que atingiu o limite teórico, que é 3 (coluna aproximação máxima). Considerando todos os grupos, a aproximação média foi maior que 2.91 e menor que 2.97. Levando em conta a distância fornecida pelo Algoritmo 25 temos uma variação relativamente pequena, uma vez que a distância máxima menos a distância mínima observada em cada grupo foi menor

Tabela 4.20: Resultados do Algoritmo 25 utilizando a base de dados $S_{\mathbf{Sb_IRTMI}}$.

-	-	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

do que 45.

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 25. Vale ressaltar que isso não significa que o fator de aproximação teórico provado para o algoritmo é justo. 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 obter a informação referente ao fator de aproximação obtido em cada instância pelos algoritmos.

4.4.10 Resultados com Genomas Reais

Nesta seção, apresentamos 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 [43], que é 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 a quantidade total de genes, na média, foi menor que 5%. Realizamos uma etapa de pré-processamento para garantir que o dados se ajustam às restrições do modelo, que é dividido em duas etapas:

- 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 blocos conservados compartilhados por ambos os genomas fossem mantidos, enquanto o restante dos genes foram removidos através de um processo que simula uma sequência de deleções.

Para realizarmos um comparativo, utilizamos o Algoritmo 24 (com a heurística I, Algoritmo 14), que leva em consideração tanto a estrutura dos genes como o tamanho regiões intergênica, e o Algoritmo 2**SbRT** [64], que é uma 2-aproximação para a variação com sinais do problema de Ordenação de Permutações por Reversões e Transposições (**SbRT**).

Executamos os algoritmos para cada instância resultante do emparelhamento, mas note que o algoritmo **2SbRT** considera apenas a estrutura dos genes. Logo, o algoritmo **2SbRT** 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 mais o tamanho da sequência de reversões, transposições e moves fornecidas 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 [50]. Para analisar as características topológicas das árvores filogenéticas, realizamos uma comparação com a árvore filogenética apresentada por Laurence *et al.* [42] usando uma ferramenta [32] baseada nas subárvores de concordância máxima (MAST) para determinar a congruência topológica entre duas árvores filogenéticas. A Tabela 4.21 mostra os resultados obtidos.

Tabela 4.21: Análise das características topológicas das árvores filogenéticas, geradas pelos resultados dos algoritmos 24 e **2SbRT**, comparadas com a árvore filogenética apresentada por Laurence *et al.* [42].

	MAST	I_{cong}	P-value
2SbRT	46 folhas	3.17	6.76e-22
Algoritmo 24	51 folhas	3.52	2.77e-25

A Tabela 4.21 indica que ambas as árvores filogenéticas têm uma alta concordância com as árvores filogenéticas apresentadas por Laurence *et al.* [42], com a árvore filogenética obtida através do Algoritmo 24 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 uma árvore filogenética construída usando o método de Reconstrução de Ordem Circular [50] com a matriz de distâncias do Algoritmo 24.

Pela Figura 4.7 (criada usando o pacote `treeio` da linguagem R [65]), observamos que a abordagem separa os organismos considerando as espécies e realizou bons agrupamentos. Vale ressaltar que a árvore foi baseada exclusivamente em informações de eventos de rearranjo.

4.5 Conclusões

Neste capítulo, investigamos a variação com e sem sinais de oito problemas considerando instâncias intergênicas rígidas e os eventos de rearranjo de reversão, transposição, move

Figura 4.7: Árvore filogenética baseada em rearranjos de genomas usando o Algoritmo 24 com a estratégia gulosa e 97 genomas do sistema Cyanorak 2.1.

e indel. Para todas as variações dos problemas em que a complexidade ainda era desconhecida, nos 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 nos apresentamos pelo menos um algoritmo de aproximação com fator constante baseado no conceito de breakpoint intergênico. Além disso, apresentamos algoritmos com fatores de aproximação melhor baseados na estrutura de grafo de ciclos ponderado rígido. Por fim, 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 mesmo 97 genomas, e o resultado apontou que existe uma alta concordância entre elas.

Capítulo 5

Modelos Intergênicos Flexíveis

Capítulo 6

Conclusões

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 rearranjo em uma solução, até a inclusão de novas estruturas genéticas na representação computacional que é adota nos modelos. Além disso, foram investigados problemas em que é considerado um grau de flexibilidade nas caracteríticas de um genoma alvo que é desejado. Para a grande maioria dos problemas que investigamos nós apresentamos a prova de NP-dificuldade e desenvolvemos algoritmos de aproximação. Além disso, sempre que possível, criamos mecanismos que visam melhorar os resultados práticos dos algoritmos que foram propostos. Os resultados apresentados nesta 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), realizado em La Jolla, USA, no mês de Maio de 2022 [22].

- "*Genome Rearrangement Distance with a Flexible Intergenic Regions Aspect*", aceito para publicação na revista IEEE-ACM Transactions on Computational Biology and Bioinformatics, e em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [17]. Uma versão preliminar deste artigo foi apresentada na 8th International Conference on Algorithms for Computational Biology (AlCoB), realizado de forma virtual, no mês de Novembro de 2021 [16]. 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 [21].

- "*An Improved Approximation Algorithm for the Reversal and Transposition Distance Considering Gene order and Intergenic Sizes*", publicado na revista Algorithms for Molecular Biology, e em coautoria com Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias e Zanoni Dias [20].

- "*Reversals and Transpositions Distance with Proportion Restriction*", publicado na revista Journal of Bioinformatics and Computational Biology, e em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [15]. Uma versão preliminar deste artigo foi apresentada no 13th Brazilian Symposium on Bioinformatics (BSB), realizado de forma virtual, no mês de Novembro de 2020 [14].

- "*Sorting by Genome Rearrangements on Both Gene Order and Intergenic Sizes*", publicado na revista Journal of Computational Biology, e em coautoria com Géraldine Jean, Guillaume Fertin, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [19]. 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 [18].

Além dos artigos listados acima, que estão diretamente relacionados com esta tese, outras contribuições foram apresentadas nos seguintes artigos:

- "*Reversal Distance on Genomes with Different Gene Content and Intergenic Regions Information*", em coautoria com Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias, foi apresentado na 8th International Conference on Algorithms for Computational Biology (AlCoB), realizado de forma virtual, no mês de Novembro de 2021 [1]

- "*Sorting Permutations by Intergenic Operations*", publicado na revista IEEE-ACM Transactions on Computational Biology and Bioinformatics, e em coautoria com Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Ulisses Dias e Zanoni Dias [56]. Uma versão preliminar deste artigo foi apresentada na 7th International Conference on Algorithms for Computational Biology (AlCoB), realizado de forma virtual, no mês de Novembro de 2021 [55].

- "*Sorting Signed Permutations by Intergenic Reversals*", publicado na revista IEEE-ACM Transactions on Computational Biology and Bioinformatics, e em coautoria com Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Laurent Bulteau, Ulisses Dias e Zanoni Dias [54].

- "*Heuristics for Genome Rearrangement Distance with Replicated Genes*", publicado na revista IEEE-ACM Transactions on Computational Biology and Bioinformatics, e em coautoria com Gabriel Siqueira, Ulisses Dias e Zanoni Dias [62]. Uma versão preliminar deste artigo foi apresentada na 7th International Conference on Algorithms for Computational Biology (AlCoB), realizado de forma virtual, no mês de Novembro de 2021 [61].

- "*On the Complexity of Sorting by Reversals and Transpositions Problem*", publicado na revista Journal of Computational Biology, e em coautoria com Andre Rodrigues Oliveira, Ulisses Dias e Zanoni Dias [52].

- "*Block-Interchange Distance Considering Intergenic Regions*", em coautoria com Ulisses Dias, Andre Rodrigues Oliveira e Zanoni Dias, foi apresentado no 12th Brazilian Symposium on Bioinformatics (BSB), realizado em Fortaleza, Brasil, no mês de Outubro de 2019 [35].

A partir das contribuições que foram 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 multiplas cópias de um gene. iv) Por fim, uma possível linha de investigação seria o estudo de novos algoritmos visando obter resultados práticos ou teóricos melhores para os problemas que aqui foram investigados. Estas 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, Guilherme Henrique Santos Miranda, Carla Negri Lintzmayer, and Zanoni Dias. Length-weighted λ-rearrangement Distance. *Journal of Combinatorial Optimization*, pages 1–24, 2020.

[3] 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.

[4] Martin Bader. Sorting by Reversals, Block Interchanges, Tandem Duplications, and Deletions. *BMC bioinformatics*, 10(1):1–10, 2009.

[5] 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.

[6] Martin Bader and Enno Ohlebusch. Sorting by Weighted Reversals, Transpositions, and Inverted Transpositions. *Journal of Computational Biology*, 14(5):615–636, 2007.

[7] 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.

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

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

[10] Anne Bergeron. A Very Elementary Presentation of the Hannenhalli-Pevzner Theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.

[11] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In R. Möhring and R. Raman, editors, *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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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 1–1, 2022.

[18] 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.

[19] 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.

[20] Klairton Lima Brito, Andre Rodrigues Oliveira, Alexsandro Oliveira Alexandrino, Ulisses Dias, and Zanoni Dias. An Improved Approximation Algorithm for the Re-

versal and Transposition Distance Considering Gene Order and Intergenic Sizes. *Algorithms for Molecular Biology*, 16(1):1–21, 2021.

[21] 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.

[22] 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.

[23] 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.

[24] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by Transpositions is Difficult. *SIAM Journal on Discrete Mathematics*, 26(3):1148–1180, 2012.

[25] Laurent Bulteau, Guillaume Fertin, and Eric Tannier. Genome Rearrangements with Indels in Intergenes Restrict the Scenario Space. *BMC Bioinformatics*, 17(14):426, 2016.

[26] Alberto Caprara. Sorting Permutations by Reversals and Eulerian Cycle Decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110, 1999.

[27] Xin Chen. On Sorting Unsigned Permutations by Double-Cut-and-Joins. *Journal of Combinatorial Optimization*, 25(3):339–351, 2013.

[28] 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.

[29] David A. Christie. Sorting Permutations by Block-Interchanges. *Information Processing Letters*, 60(4):165–169, 1996.

[30] David A. Christie. A 3/2-Approximation Algorithm for Sorting by Reversals. In H. Karloff, editor, *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.

[31] David A. Christie and Robert W. Irving. Sorting Strings by Reversals and by Transpositions. *SIAM Journal on Discrete Mathematics*, 14(2):193–206, 2001.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] Nadia El-Mabrouk and David Sankoff. Analysis of Gene Order Evolution Beyond Single-Copy Genes. *Evolutionary Genomics*, pages 397–429, 2012.

[37] 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.

[38] Niklas Eriksen. *Combinatorics of Genome Rearrangements and Phylogeny*. Teknologie licentiat thesis, Kungliga Tekniska Högskolan, Stockholm, 2001.

[39] Niklas Eriksen. $(1+\epsilon)$-Approximation of Sorting by Reversals and Transpositions. *Theoretical Computer Science*, 289(1):517–529, 2002.

[40] 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.

[41] Guillaume Fertin, Anthony Labarre, Irena Rusu, Éric Tannier, and Stéphane Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. The MIT Press, London, England, 2009.

[42] Laurence Garczarek, Ulysse Guyet, Hugo Doré, Gregory K Farrant, Mark Hoebeke, Loraine Brillet-Guéguen, Antoine Bisch, Mathilde Ferrieux, Jukka Siltanen, Erwan Corre, et al. Cyanorak v2. 1: a scalable information system dedicated to the visualization and expert curation of marine and brackish picocyanobacteria genomes. *Nucleic acids research*, page 1, 2020.

[43] 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.

[44] 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.

[45] 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.

[46] Crystal L. Kahn and Benjamin J. Raphael. Analysis of Segmental Duplications via Duplication Distance. *Bioinformatics*, 24(16):i133–i138, 2008.

[47] John D. Kececioglu and David Sankoff. Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement. *Algorithmica*, 13:180–210, 1995.

[48] Petr Kolman and Tomasz Waleń. Reversal Distance for Strings with Duplicates: Linear Time Approximation Using Hitting Set. In *International Workshop on Approximation and Online Algorithms*, pages 279–289, 2006.

[49] Guohui Lin and Tao Jiang. A Further Improved Approximation Algorithm for Breakpoint Graph Decomposition. *Journal of Combinatorial Optimization*, 8(2):183–194, 2004.

[50] V Makarenkov and B Leclerc. Tree metrics and their circular orders: Some uses for the reconstruction and fitting of phylogenetic trees. *Mathematical Hierarchies and Biology, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 37:183–208, 1997.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] Eric Tannier, Anne Bergeron, and Marie-France Sagot. Advances on Sorting by Reversals. *Discrete Applied Mathematics*, 155(6-7):881–888, 2007.

[64] 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.

[65] Li-Gen Wang, Tommy Tsan-Yuk Lam, Shuangbin Xu, Zehan Dai, Lang Zhou, Tingze Feng, Pingfan Guo, Casey W Dunn, Bradley R Jones, Tyler Bradley, et al. treeio: an r package for phylogenetic tree input and output with richly annotated and associated data. *Molecular Biology and Evolution*, 37(2):599–603, 2020.

[66] Eyla Willing, Simone Zaccaria, Marília DV Braga, and Jens Stoye. On the Inversion-Indel Distance. *BMC Bioinformatics*, 14:S3, 2013.

[67] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient Sorting of Genomic Permutations by Translocation, Inversion and Block Interchange. *Bioinformatics*, 21(16):3340–3346, 2005.