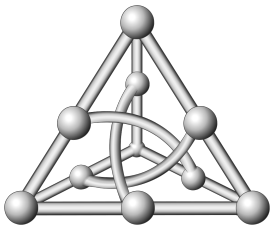
Avanços no cálculo da distância Double Cut and Join com regiões intergênicas para genomas reais

Victor Hugo Bagordakis da Rocha

Dissertação de Mestrado

Área: Biologia Computacional

Orientador: Prof. Fábio Henrique Viduani Martinez, Dr.
Co-orientador: Prof. Diego Padilha Rubert, Dr.



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Agosto, 2025

Avanços no cálculo da distância Double Cut and Join com regiões intergênicas para genomas reais

Victor Hugo Bagordakis da Rocha

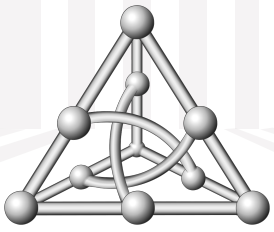
Dissertação de Mestrado

Área: Biologia Computacional

Orientador: Prof. Fábio Henrique Viduani Martinez, Dr.

Co-orientador: Prof. Diego Padilha Rubert, Dr.

Dissertação apresentada como parte dos requisitos necessários para a obtenção do título de Mestre em Ciência da Computação



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Agosto, 2025

Resumo

O problema da distância double cut and join (DCJ) tem sido amplamente estudado nos últimos anos, resultando no desenvolvimento de diversos algoritmos para suas diferentes variações. Apesar disso, ainda existem poucos estudos sobre o problema da distância DCJ para genomas com genes duplicados e regiões intergênicas. Os tamanhos dos genomas e as desigualdades nos comprimentos das regiões intergênicas desempenham um papel importante no tempo de execução dos algoritmos, porém, a influência desses aspectos costuma receber pouca atenção em estudos anteriores. Neste trabalho, estabelecemos uma definição formal para o problema da distância DCJ considerando regiões intergênicas e genes duplicados, e propomos diferentes estratégias para encontrar uma solução heurística para esse problema, através do uso de algoritmos exatos e de aproximação. Os métodos propostos oferecem ganhos em qualidade e eficiência em comparação com os já existentes, superando limitações de métodos atuais relacionadas ao número de genes e às desigualdades nas regiões intergênicas.

Palavras-chave: *Double cut and join* (DCJ), wDCJ, Regiões intergênicas, Duplicações, Programação Linear Inteira (PLI), Heurística, Rearranjo de genomas, Genômica comparativa

Abstract

The double cut and join (DCJ) distance problem has been extensively studied in recent years, leading to the development of multiple algorithms for its different variations. However, limited attention has been given to the DCJ distance problem in the context of genomes containing duplicated genes and intergenic regions. The genome sizes and the inequalities of the intergenic sizes play an important role in the running times of algorithms; yet these factors are often overlooked in existing studies. In this work, we establish a formal definition for the DCJ distance problem considering intergenic regions and duplicated genes, and propose different strategies to find a heuristic solution for it by using existing exact and approximation algorithms. The methods we propose improve upon existing approaches in both quality and efficiency, particularly overcoming severe limitations of the current methods regarding the number of genes and intergenic size inequalities.

Keywords: Double cut and join (DCJ), wDCJ, Intergenic regions, Duplications, Integer Linear Programming (ILP), Heuristic, Genome rearrangement, Comparative genomics

Agradecimentos

Agradeço aos meus pais, Amauri e Mariselma. O apoio deles foi fundamental para a conclusão desta etapa.

Agradeço aos meus queridos irmãos, Carlos e Isabela.

Agradeço ao meu orientador, Fábio Henrique Viduani Martinez, e ao meu co-orientador Diego Padilha Rubert, pela orientação e pelo conhecimento compartilhado.

Este trabalho foi realizado com o auxílio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) — Código de Financiamento 001 — Proc. No. 88887.836539/2023-00.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O *genoma* é o conjunto completo do material genético presente em um organismo. Esse material genético é codificado no DNA (ácido desoxirribonucleico) das células. O genoma contém as instruções hereditárias que, em interação com fatores epigenéticos e ambientais, orientam o crescimento, o desenvolvimento, o funcionamento e a reprodução do organismo. Em termos mais simples, pode-se pensar nele como o manual de instruções ou projeto para um organismo, ditando suas características e traços.

Os genomas variam em tamanho e complexidade entre diferentes espécies, indo de alguns milhares de pares de bases em alguns vírus até bilhões de pares de bases em organismos mais complexos como a *Tmesipteris oblanceolata*. Compreender os genomas é essencial para diversos campos da biologia, incluindo genética, biologia evolutiva e biotecnologia.

Durante o processo evolutivo, os genomas estão sujeitos a mutações e rearranjos. Rearranjos de larga escala alteram o número de genes, de cromossomos e podem mudar a posição e orientação de fragmentos de DNA. Várias métricas foram criadas com o intuito de comparar dois genomas. Elas podem ser utilizadas para estimar a distância evolutiva entre duas espécies, construir árvores filogenéticas, identificar genes ortólogos, entre outras aplicações.

Uma dessas métricas é chamada de *distância de rearranjo*, um problema clássico da genômica comparativa. Esse problema consiste em determinar o menor número de operações de rearranjo para transformar um genoma em outro. Exemplos de eventos de rearranjo incluem reversões (ou inversões), transposições, inserções, translocações, entre outros.

Para determinar a distância de rearranjo é necessário representar o genoma de alguma maneira. O nível de detalhe usado em cada representação varia conforme o problema a ser estudado. Para algumas aplicações, como no alinhamento de sequências, é necessário que cada nucleotídeo do genoma seja representado. Por outro lado, em problemas de distância de rearranjo, cada genoma é representado com um nível de abstração mais alto, sendo visto como um conjunto de cromossomos, cada um representado como uma sequência de genes.

Nos últimos anos, diversos tipos de distância de rearranjo foram estudados. Muitos dos

trabalhos iniciais foram feitos considerando genomas sem duplicações e operações como transposições, reversões e translocações [1, 2, 3, 4, 5, 6]. Em genomas sem duplicações, cada gene aparece uma única vez em cada genoma. Mesmo nessa situação, vários desses problemas são NP-difíceis [7, 8, 9, 10]. Por conta disso, vários algoritmos de aproximação foram propostos para esses problemas ao longo dos anos [11, 12, 13, 14].

O foco deste trabalho é uma operação de rearranjo conhecida como *Double Cut and Join* (DCJ), introduzida em [15]. Essa operação consiste em cortar um genoma em dois pontos distintos e recombinar as extremidades geradas após o corte de uma maneira diferente. Essa operação pode ser usada para representar outros eventos de rearranjo, como transposições e reversões.

O problema da *distância DCJ* consiste em determinar o menor número de operações DCJ para transformar um determinado genoma em outro. Quando consideramos genomas sem duplicações, com o mesmo conjunto de genes, e somente a posição e a orientação dos genes, a distância DCJ exata pode ser encontrada em tempo linear [16].

Quando duplicações são consideradas e os genomas possuem o mesmo conjunto de genes, o problema da distância DCJ se torna NP-difícil [17]. Nesse caso, para resolver o problema de forma exata, foram propostas uma formulação de programação linear inteira (PLI) e uma formulação de satisfatibilidade booliana (SAT) [18]. Um algoritmo de aproximação de tempo linear também foi proposto [19].

Todos os trabalhos citados anteriormente consideram somente os genes e suas orientações no cálculo da distância de rearranjo. Alguns estudos argumentaram que essa abordagem pode resultar em estimativas que são consideravelmente menores do que o número real de eventos de rearranjo, e essa estimativa pode ser melhorada considerando a distribuição dos comprimentos das regiões intergênicas [20, 21].

Trabalhos mais recentes começaram a considerar a influência dos comprimentos das regiões intergênicas em diferentes distâncias de rearranjo [22, 23, 24, 25, 26]. A distância DCJ considerando os comprimentos das regiões intergênicas também é chamada de distância wDCJ, e foi definida inicialmente para genomas sem genes duplicados. Esse problema é NP-difícil, e, para ele, foram propostos uma formulação de PLI e um algoritmo de 4/3-aproximação [22].

Uma variação da distância DCJ que ainda foi pouco estudada é o problema da distância DCJ considerando genes duplicados e os comprimentos das regiões intergênicas, que chamamos de problema da distância wDCJ com genes duplicados. Até onde sabemos, um único trabalho abordou esse problema [27]. Nele, foram propostas três heurísticas baseadas em *adjacency graph packing*, que utilizam uma estrutura conhecida como grafo de adjacências.

O objetivo deste trabalho é propor uma heurística eficiente para o problema da distância wDCJ com genes duplicados capaz de lidar com genomas com um grande número de genes e grande número de desigualdades nas regiões intergênicas — características que os aproximam de genomas reais. Assumimos que os genomas são unicromossomais, balanceados, circulares e possuem o mesmo conjunto de genes. Portanto, os experimentos foram realizados em *genomas simulados*, isto é, genomas gerados artificialmente.

O nosso método é dividido em dois passos. Primeiro, estabelecemos uma associação

entre os genes homólogos dos genomas. Após isso, os genomas são tratados como se não houvesse duplicações, e calculamos a distância wDCJ para a associação de genes definida no passo anterior.

A principal diferença em relação ao método proposto em [27] está na maneira em que os genes homólogos são associados. Para encontrar uma associação de genes entre os pares de genomas (primeiro passo), utilizamos a formulação de PLI para o problema da distância DCJ-indel para genomas naturais (veja a Seção 2.3) [28].

Após obter uma associação de genes, calculamos a distância wDCJ com genes duplicados (segundo passo) de duas maneiras distintas. A primeira abordagem utiliza o PLI para o problema da distância DCJ considerando regiões intergênicas, sem genes duplicados [22]. A segunda abordagem usa o algoritmo de aproximação para a distância wDCJ [22].

1.1 Organização do texto

Este trabalho está organizado da seguinte maneira. No Capítulo 2 serão apresentados os conceitos iniciais necessários para definir o problema abordado neste trabalho. O Capítulo 3 introduz a definição do problema da distância DCJ e algumas de suas diferentes variações. No Capítulo 4 descreveremos a nossa abordagem para obter uma heurística para o problema da distância wDCJ com genes duplicados, e também apresentaremos o critério utilizado para avaliar o desempenho dos algoritmos. As ferramentas utilizadas para criar os datasets usados nos experimentos serão detalhadas no Capítulo 5. Os experimentos realizados e os resultados obtidos serão descritos e analisados no Capítulo 6. Por fim, no Capítulo 7, serão abordadas as considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Conceitos iniciais

Neste capítulo, serão apresentadas as definições e conceitos iniciais necessários para compreender o problema abordado neste trabalho. Assumimos que o leitor está familiarizado com o conceito de grafos e com as notações relacionadas a esse conceito.

2.1 Definições genômicas

Um *gene* é um segmento específico de DNA que codifica informações para características hereditárias. Uma *família* de genes é um grupo de genes que compartilham uma origem em comum e que possuem funções bioquímicas similares. Genes que pertencem à mesma família são chamados de *homólogos*.

Um gene é representado por um símbolo g, que também representa sua família. Devido à estrutura de dupla hélice do DNA, cada gene pode estar localizado em qualquer uma das duas fitas complementares, e, por essa razão, pode ser lido em uma das duas direções possíveis. Isso dá origem ao conceito de orientação de um gene. Um gene com orientação direta é representado simplesmente como g, enquanto o seu complemento reverso, correspondente à mesma sequência de genes lida na direção oposta, é denotado como $-g$. Em um genoma com *genes duplicados*, um símbolo g pode aparecer mais de uma vez no mesmo genoma.

Um gene g possui duas extremidades chamadas de *cabeça* e *cauda*, denotadas como g^h e g^t, respectivamente. Duas extremidades de genes distintos que aparecem consecutivamente em um cromossomo formam uma *adjacência*. Uma adjacência formada por dois genes g e h é representada como gh.

Um genoma é representado como um conjunto de cromossomos, e cada cromossomo pode conter muitos genes. Um cromossomo circular com n genes, de um genoma sem regiões intergênicas é representado por uma string delimitada por parênteses $\pi = (g_1\ g_2\ \ldots\ g_n)$. Portanto, um genoma A com m cromossomos, é representado como $A = \{\pi_1, \pi_2, \ldots, \pi_m\}$. O conjunto de genes de A é denotado como $\mathcal{G}(A)$. Dois genomas A e B são *balanceados* se $\mathcal{G}(A) = \mathcal{G}(B)$.

Regiões intergênicas são sequências de nucleotídeos entre dois genes consecutivos. O número de nucleotídeos entre os genes é chamado de *comprimento da região intergênica*.

Um cromossomo circular com n genes possui n regiões intergênicas. Os comprimentos das regiões intergênicas de um cromossomo são representados por uma sequência de números inteiros não negativos. Para um cromossomo circular $\pi = (g_1\ g_2\ \ldots\ g_n)$, os comprimentos das regiões intergênicas são representados pela string $\breve{\pi} = (\breve{g}_1\ \breve{g}_2\ \ldots\ \breve{g}_n)$, onde cada $\breve{g}_i \in \mathbb{N}$ denota a região intergênica entre g_i e g_{i-1}, para $1 < i \leq n$, e $\breve{g}_1$ representa o comprimento da região intergênica entre g_1 e g_n.

Os cromossomos de um genoma $\mathcal{A}$ com regiões intergênicas são compostos por duas componentes. O conjunto de todas as strings com os genes de cada cromossomo de $\mathcal{A}$ é denotado como $A = \{\pi_1, \pi_2, \ldots, \pi_m\}$, onde m é o número de cromossomos de $\mathcal{A}$. O conjunto de todas as sequências com os comprimentos das regiões intergênicas de cada cromossomo de $\mathcal{A}$ é denotado como $\breve{A} = \{\breve{\pi}_1, \breve{\pi}_1, \ldots, \breve{\pi}_m\}$. Portanto, denotamos o genoma como $\mathcal{A} = (A, \breve{A})$.

O conjunto de genes de um genoma $\mathcal{A} = (A, \breve{A})$ é representado como $\mathcal{G}(\mathcal{A})$, e o conjunto com todas as extremidades de genes é denotado como $ext(\mathcal{A})$. O número de genes de $\mathcal{A}$ é denotado como $|A|$, e o número de regiões intergênicas é denotado como $|\breve{A}|$. Neste trabalho, consideramos somente cromossomos circulares. Nesse caso, temos $|A| = |\breve{A}|$.

O conjunto de famílias de um genoma $\mathcal{A}$ é $\mathcal{F}(\mathcal{A})$. O número de ocorrências de um gene g em um genoma $\mathcal{A}$ é denotado por $o_{\mathcal{A}}(g)$. Para que cada ocorrência de g seja representada de forma única, numeramos as ocorrências de g em $\mathcal{A}$ de 1 a $o_{\mathcal{A}}(g)$.

Sejam $\mathcal{A}$ e $\mathcal{B}$ dois genomas com genes duplicados e regiões intergênicas. Dois genomas $\mathcal{A}$ e $\mathcal{B}$ são *balanceados* se a soma dos comprimentos das regiões intergênicas for igual em ambos os genomas, e se $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{B})$. Uma *região conservada* de $\mathcal{A}$ em $\mathcal{B}$ é um segmento do genoma $\mathcal{A}$ que aparece em $\mathcal{B}$.

Por exemplo, seja $\mathcal{A} = (A, \breve{A})$ um genoma com $A = \{(a_1\ -b_1\ c_1\ b_2)\}$ e $\breve{A} = \{(2\ 10\ 6\ 4)\}$. Esse genoma possui $|A| = 4$ e $|\breve{A}| = 4$. O conjunto de famílias de $\mathcal{A}$ é $\mathcal{F}(\mathcal{A}) = \{a, b, c\}$ e o conjunto de genes é $\mathcal{G}(\mathcal{A}) = \{a_1, b_1, b_2, c_1\}$. O conjunto de extremidades de $\mathcal{A}$ é $ext(\mathcal{A}) = \{{a_1}^t, {a_1}^h, {b_1}^t, {b_1}^h, {b_2}^t, {b_2}^h, {c_1}^t, {c_1}^h\}$.

2.2 Grafo de adjacências e decomposições consistentes

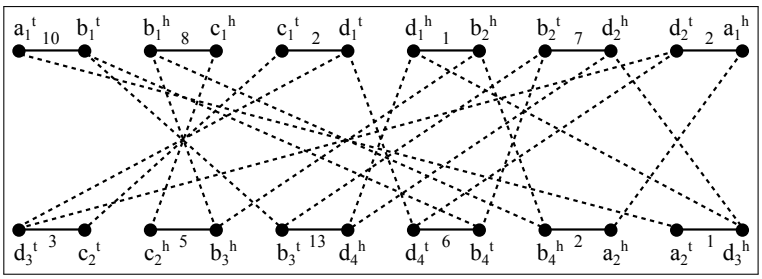
Sejam $\mathcal{A} = (A, \breve{A})$ e $\mathcal{B} = (B, \breve{B})$ dois genomas balanceados com conjuntos de extremidades $ext(\mathcal{A})$ e $ext(\mathcal{B})$. Para nos referirmos a cada ocorrência de um gene g de $\mathcal{B}$ de forma única, numeramos as ocorrências de g em $\mathcal{B}$ de $o_{\mathcal{A}}(g) + 1$ a $o_{\mathcal{A}}(g) + o_{\mathcal{B}}(g)$.

O *grafo de adjacências* [16] $AG(\mathcal{A}, \mathcal{B})$ para os genomas $\mathcal{A}$ e $\mathcal{B}$ é definido da seguinte maneira. O conjunto de vértices é $V = ext(\mathcal{A}) \cup ext(\mathcal{B})$ e o conjunto de arestas E é composto por dois tipos de arestas, chamadas de arestas de adjacências e arestas de extremidades. Uma *aresta de adjacência* é uma aresta com peso que conecta duas extremidades de genes se elas são adjacentes em um genoma, e o seu peso é igual ao comprimento da região intergênica entre essas duas extremidades. Uma *aresta de extremidade* conecta dois genes homólogos que pertencem a conjuntos de extremidades distintos. O conjunto de arestas de adjacências é representado como E_{adj}, enquanto o conjunto de arestas de extremidade é representado como E_{ext}. Um exemplo de grafo de adjacências pode ser visto na Figura 2.1a.

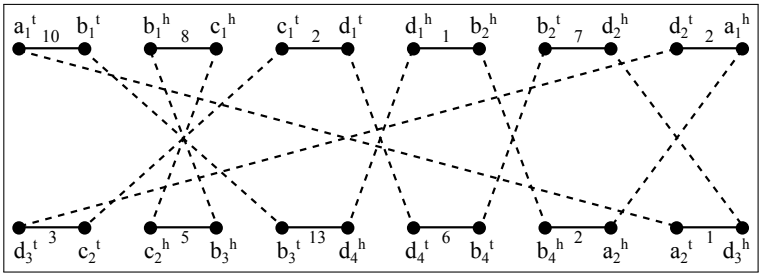
Também podemos representar dois genomas A e B sem regiões intergênicas em um

grafo de adjacências $AG(A, B)$. Nesse caso, as arestas de adjacência não possuem pesos.

Se fizermos uma associação de um para um entre os genes de $\mathcal{A}$ e os genes de $\mathcal{B}$, podemos tratar os genomas como se não houvesse duplicações. Essa associação pode ser feita estabelecendo uma bijeção entre os genes de $\mathcal{A}$ e os genes de $\mathcal{B}$.



(a)



(b)

Figura 2.1: Grafo de adjacências e uma decomposição consistente para um par de genomas balanceado. (a) Grafo de adjacências para dois genomas balanceados $\mathcal{A} = (A, \breve{A})$ e $\mathcal{B} = (B, \breve{B})$, com $A = \{(-a_1\ b_1\ -c_1\ d_1\ -b_2\ -d_2)\}$, $\breve{A} = \{(2\ 10\ 8\ 2\ 1\ 7)\}$, $B = \{(-d_3\ c_2\ -b_3\ -d_4\ b_4\ -a_2)\}$ e $\breve{B} = \{(1\ 3\ 5\ 13\ 6\ 2)\}$. Arestas de adjacências são representadas por linhas contínuas e arestas de extremidades são representadas por linhas tracejadas. (b) Uma decomposição consistente D de $AG(\mathcal{A}, \mathcal{B})$, com a associação de genes $b_1 \rightarrow b_3$, $b_2 \rightarrow b_4$, $d_1 \rightarrow d_4$ e $d_2 \rightarrow d_3$.

Uma *bijeção válida* de $\mathcal{A} = (A, \breve{A})$ para $\mathcal{B} = (B, \breve{B})$ associa um gene g_i de $\mathcal{A}$ à exatamente um de seus homólogos em $\mathcal{B}$, formando n pares de genes homólogos, onde $n = |A| = |B|$. Se o genoma não possui genes duplicados, existe somente uma maneira de fazer essa associação. Em genomas com genes duplicados, existem múltiplas bijeções válidas. Uma bijeção válida define uma associação de um para um entre os genes homólogos de $\mathcal{A}$ e $\mathcal{B}$. Uma associação de um gene g_i de $\mathcal{A}$ com um de seus homólogos g_j de $\mathcal{B}$ é denotada como $g_i \rightarrow g_j$.

Um *ciclo alternante* em $AG(\mathcal{A}, \mathcal{B})$ é um ciclo no qual duas arestas são adjacentes se uma delas é uma aresta de adjacência e a outra é uma aresta de extremidade. Uma decomposição D do grafo de adjacências é um conjunto de ciclos alternantes com vértices disjuntos que cobre todos os vértices e todas as arestas de adjacência de $AG(\mathcal{A}, \mathcal{B})$. O conjunto de todas as arestas de uma decomposição D é representado como $E(D)$. Uma decomposição é consistente se, para quaisquer genes homólogos g_i e g_j, tais que $g_i \in \mathcal{G}(\mathcal{A})$ e $g_j \in \mathcal{G}(\mathcal{B})$, uma das seguintes condições é satisfeita:

1. $({g_i}^t, {g_j}^t) \in E(D)$ e $({g_i}^h, {g_j}^h) \in E(D)$;
2. $({g_i}^t, {g_j}^t) \notin E(D)$ e $({g_i}^h, {g_j}^h) \notin E(D)$.

Uma decomposição consistente D de $AG(\mathcal{A}, \mathcal{B})$ fornece uma bijeção válida entre os genes de $\mathcal{A}$ e os genes de $\mathcal{B}$. A Figura 2.1b mostra um exemplo de uma decomposição consistente.

2.3 Tipos de genomas

Ao longo deste trabalho, iremos utilizar diferentes nomenclaturas para diferentes tipos de genomas. A definição de cada uma dessas nomenclaturas pode ser vista abaixo.

- *Genomas simulados*: genomas gerados artificialmente por meio de um algoritmo;
- *Genomas reais*: genomas encontrados na natureza;
- *Genomas naturais*: conceito utilizado em [28], onde não é considerada a existência de regiões intergênicas. Em um par de genomas naturais, qualquer gene pode ocorrer múltiplas vezes em qualquer um dos dois genomas, e o número de ocorrências de uma determinada família não precisa ser igual em ambos os genomas. Em outras palavras, dois genomas naturais podem possuir tamanhos e conjuntos de genes distintos. Genomas balanceados com genes duplicados são um caso particular de genomas naturais.

2.4 Reversões e transposições

Seja $(\pi, \breve{\pi})$ um cromossomo circular de um genoma com regiões intergênicas, com $\pi = (g_1 \ \dots \ g_{i-1} \ g_i \ \dots \ g_j \ g_{j+1} \ \dots \ g_n)$ e $\breve{\pi} = (\breve{g}_1 \ \dots \ \breve{g}_{i-1} \ \breve{g}_i \dots \ \breve{g}_j \ \breve{g}_{j+1} \ \dots \ \breve{g}_n)$.

Uma **reversão** $\rho^{(i,j)}$ no cromossomo $(\pi, \breve{\pi})$, com $1 \leq i \leq j \leq n$, e $\{i, j\} \in \mathbb{Z}$, é uma operação de rearranjo que gera o cromossomo $(\pi', \breve{\pi}')$, tal que:

1. $\pi' = (g_1 \ \dots \ g_{i-1} \underline{-g_j \ -g_{j-1} \ \dots \ -g_{i+1} \ -g_i} \ g_{j+1} \ \dots \ g_n)$, e
2. $\breve{\pi}' = (\breve{g}_1 \ \dots \ \breve{g}'_i \ \underline{\breve{g}_j \ \dots \ \breve{g}_{i+1}} \ \breve{g}'_{j+1} \ \dots \ \breve{g}_n)$, com $\breve{g}'_i + \breve{g}'_{j+1} = \breve{g}_i + \breve{g}_{j+1}$ e $\{\breve{g}'_i, \breve{g}'_{j+1}\} \in \mathbb{N}$.

Uma **transposição** $\tau^{(i,j,k)}$ no cromossomo $(\pi, \breve{\pi})$, com $1 \leq i < j < k \leq n$, e $\{i, j, k\} \in \mathbb{Z}$, é uma operação de rearranjo que gera o cromossomo $(\pi', \breve{\pi}')$, tal que:

1. $\pi' = (g_1 \ \dots \ g_{i-1} \ \underline{g_j \ \dots \ g_{k-1}} \ \underline{g_i \ \dots \ g_{j-1}} \ g_k \ \dots g_n)$, e
2. $\breve{\pi}' = (\breve{g}_1 \ \dots \ \breve{g}'_i \ \underline{\breve{g}_{j+1} \ \dots \ \breve{g}_{k-1}} \ \breve{g}'_{i+k-j} \ \underline{\breve{g}_{i+1} \ \dots \ \breve{g}_{j-1}} \ \breve{g}'_k \ \dots \ \breve{g}_n)$, com $\breve{g}'_i + \breve{g}'_{i+k-j} + \breve{g}'_k = \breve{g}_i + \breve{g}_{i+k-j} + \breve{g}_k$ e $\{\breve{g}'_i, \breve{g}'_{i+k-j}, \breve{g}'_k\} \in \mathbb{N}$.

2.5 Complexidade computacional

Esta seção foi baseada em [29, 30, 31], e apresenta resumidamente alguns conceitos importantes da complexidade computacional que são necessários para uma melhor compreensão do problema apresentado neste trabalho.

Um problema pode ser classificado como um problema de otimização ou de decisão. Em um *problema de otimização* estamos interessados em encontrar a solução ótima dentre todas as soluções viáveis para o problema. Por exemplo, o problema de determinar o caminho mínimo entre dois vértices de um grafo G é um problema de otimização. Um *problema de decisão* simplesmente responde uma pergunta de "sim" ou "não", como, por exemplo, determinar se existe um ciclo de tamanho k em um grafo G.

A teoria da NP-completude foi definida apenas para problemas de decisão, contudo, muitos problemas de otimização possuem um problema de decisão relacionado, de modo que a solução do problema de decisão pode ser utilizada para resolver o problema de otimização relacionado a ele. Por exemplo, o problema de otimização que consiste em encontrar o caminho mínimo entre dois vértices u e v de um grafo G está relacionado com o seguinte problema de decisão: "Dado um grafo G, dois vértices u, v e um valor K, existe um caminho de u a v que possua no máximo K arestas?". Resolvendo o problema de decisão para diferentes valores de K, podemos encontrar o valor mínimo de K para o qual a resposta do problema de decisão é "sim", encontrando assim uma solução para o problema do caminho mínimo. Portanto, apesar de ser definida somente para problemas de decisão, a teoria da NP-completude possui implicações para problemas de otimização [31].

A classe de problemas P contém todos os problemas de decisão que podem ser resolvidos em tempo polinomial por um algoritmo determinístico (i.e., usando uma máquina de Turing determinística). A classe NP é composta por todos os problemas de decisão que podem ser resolvidos em tempo polinomial por uma máquina de Turing não-determinística. Equivalentemente, podemos dizer que a classe NP é composta por todos os problemas de decisão que são "verificáveis" em tempo polinomial por uma máquina de Turing determinística. Um determinado problema é *verificável* em tempo polinomial se, dado um "certificado" da solução desse problema, é possível verificar em tempo polinomial se esse certificado é correto. Claramente, todo problema que pode ser resolvido em tempo polinomial por um algoritmo determinístico também é verificável em tempo polinomial por um algoritmo determinístico, e, portanto, $\mathrm{P} \subseteq \mathrm{NP}$.

Uma *redução em tempo polinomial* de um problema $\mathcal{P}$ para um problema $\mathcal{P}'$ é denotada como $\mathcal{P} \leq_{\mathrm{P}} \mathcal{P}'$ e é definida por um par de funções (f, g). A função f transforma cada entrada x de $\mathcal{P}$ em uma entrada $f(x) = x'$ de $\mathcal{P}'$ em tempo polinomial. Seja y' a solução do problema $\mathcal{P}'$ para a entrada x'. A função g recebe como entrada x e y' e gera uma saída $g(x, y') = y$ para o problema $\mathcal{P}$ em tempo polinomial. Portanto, se $\mathcal{P} \leq_{\mathrm{P}} \mathcal{P}'$ e $\mathcal{P}'$ pode ser resolvido em tempo polinomial, então $\mathcal{P}$ também pode ser resolvido em tempo polinomial.

A redução em tempo polinomial $\mathcal{P} \leq_{\mathrm{P}} \mathcal{P}'$ é uma forma de mostrar que $\mathcal{P}'$ é pelo menos tão difícil quanto $\mathcal{P}$, e, dessa forma, se $\mathcal{P}$ não pode ser resolvido em tempo polinomial, então $\mathcal{P}'$ também não pode.

Um problema $\mathcal{P}$ é *NP-completo* se:

1. $\mathcal{P} \in \mathrm{NP}$, e
2. $\mathcal{P}' \leq_{\mathrm{P}} \mathcal{P},\ \forall\ \mathcal{P}' \in \mathrm{NP}$.

Se um problema $\mathcal{P}$ satisfaz a condição 2, mas não necessariamente a condição 1, então

o problema é *NP-difícil*. Alguns problemas de otimização podem satisfazer a condição 2 (i.e., podem ser NP-difíceis), como por exemplo, o problema de determinar o caminho mais longo entre dois vértices u e v de um grafo G. No entanto, nenhum problema de otimização satisfaz a condição 1, porque a classe NP é definida somente para problemas de decisão.

Para mostrar que um problema $\mathcal{P}$ é NP-difícil, basta mostrar que $\mathcal{P}' \leq_{\mathrm{P}} \mathcal{P}$ para algum problema $\mathcal{P}' \in$ NP-completo ou $\mathcal{P}' \in$ NP-difícil. Para mostrar que um problema de decisão $\mathcal{P}$ é NP-completo, é necessário mostrar que $\mathcal{P}$ é verificável em tempo polinomial por um algoritmo determinístico e que $\mathcal{P}' \leq_{\mathrm{P}} \mathcal{P}$ para algum problema $\mathcal{P}' \in$ NP-completo.

Se existir um algoritmo determinístico que encontra uma solução para um problema NP-completo ou NP-difícil em tempo polinomial, então todo problema em NP poderia ser resolvido em tempo polinomial, o que implicaria que P = NP. Contudo, até hoje não foi descoberto um algoritmo de tempo polinomial que resolva um problema NP-completo ou NP-difícil, e, portanto, mostrar que um problema é NP-completo ou NP-difícil é uma forte evidência de que esse problema, provavelmente, não pode ser resolvido em tempo polinomial. O problema de determinar se P = NP ou se P $\neq$ NP é um problema em aberto da Ciência da Computação.

2.6 Heurísticas e algoritmos de aproximação

Esta seção foi baseada em [30, 31, 32, 33], e apresenta resumidamente os conceitos de heurísticas e algoritmos de aproximação.

Conforme foi visto na seção anterior, não se conhecem algoritmos que resolvam problemas NP-completos ou NP-difíceis em tempo polinomial para todas as entradas, o que significa, em termos práticos, que resolver esses problemas de forma exata requer algoritmos de tempo exponencial. Esse tempo de execução pode ser satisfatório para entradas pequenas, contudo, com o aumento do tamanho da entrada, o tempo de execução cresce exponencialmente, o que torna o uso desses algoritmos inviável para entradas suficientemente grandes.

Existem estratégias que podem ser utilizadas para "contornar" o tempo de execução exponencial desses problemas. Uma delas consiste em determinar um conjunto especial de entradas com propriedades específicas, para as quais é possível encontrar uma solução em tempo polinomial.

Para problemas de otimização NP-difíceis, é possível utilizar algoritmos de aproximação ou heurísticas, que não encontram a solução exata do problema, mas encontram uma solução viável que pode ser "suficientemente boa" dependendo da aplicação.

Seja x uma entrada de um problema de otimização $\mathcal{P}$, seja $OPT(x)$ o valor da solução ótima para x e seja A um algoritmo de aproximação para $\mathcal{P}$ que produz uma solução $A(x)$ para a entrada x. Um *algoritmo de aproximação* A para um problema $\mathcal{P}$ possui uma razão de aproximação $R_A(x)$, tal que, para qualquer entrada x:

$$R_A(x) = \begin{cases} \frac{A(x)}{OPT(x)}, & \text{se } \mathcal{P} \text{ é um problema de minimização,} \\ \\ \frac{OPT(x)}{A(x)}, & \text{se } \mathcal{P} \text{ é um problema de maximização.} \end{cases} \tag{2.1}$$

Em ambos os casos da Equação (2.1), temos que $R_A(x) \geq 1$. Seja n o tamanho de uma entrada x. O algoritmo de aproximação A é um algoritmo de $r(n)$-aproximação se, para toda entrada x de tamanho n, temos:

$$R_A(x) \leq r(n) \ . \tag{2.2}$$

Um *algoritmo heurístico* constrói uma solução viável para um problema sem garantir um limite formal de qualidade. A construção de uma solução heurística é feita utilizando estratégias baseadas na intuição, experiência ou regras empíricas. Considere, por exemplo, o problema NP-difícil da cobertura mínima de vértices, que consiste em encontrar um subconjunto de vértices de tamanho mínimo de um grafo G, tal que cada aresta de G incide em pelo menos um vértice do subconjunto. Uma possível heurística para esse problema consiste em selecionar repetidamente o vértice de maior grau, e em seguida, remover todas as arestas incidentes nesse vértice. O processo é repetido até que todas as arestas de G sejam removidas.

Capítulo 3

Distância DCJ

A operação conhecida como *double cut and join* (DCJ) consiste em cortar um genoma em duas adjacências distintas g_ig_j, g_kg_l e criar duas novas adjacências g_ig_l, g_jg_k, ou as adjacências g_ig_k, g_jg_l.

Quando regiões intergênicas são consideradas, essa operação é chamada de *wDCJ* e também altera o comprimento da região intergênica entre g_i e g_j, representado por $\breve{g}_j$, e o comprimento da região intergênica entre g_k e g_l, representado como $\breve{g}_l$, criando as novas regiões intergênicas $\breve{g}_j'$ e $\breve{g}_l'$. Para que a soma das regiões intergênicas seja a mesma antes e depois da operação wDCJ, os novos comprimentos das regiões intergênicas devem ser atribuídos de forma que $\breve{g}_j' + \breve{g}_l' = \breve{g}_j + \breve{g}_l$.

O problema da distância DCJ consiste em determinar o número mínimo de operações DCJ para transformar um genoma em outro. A versão mais simples desse problema considera dois genomas balanceados A e B sem duplicações e sem regiões intergênicas. Nesse caso, considerando somente cromossomos circulares, a distância DCJ pode ser obtida em tempo linear através da Equação (3.1) [16].

$$\mathrm{DCJ}(A, B) = n - c \ , \tag{3.1}$$

onde $n = |A| = |B|$, e c é o número de ciclos em $AG(A, B)$.

Existem várias variações da distância DCJ que consideram diferentes aspectos dos genomas. Na Seção 3.1, será apresentado o problema da distância DCJ com genes duplicados sem considerar regiões intergênicas. Na Seção 3.2, abordaremos o problema da distância DCJ com regiões intergênicas sem duplicações. Na Seção 3.3 uniremos ambos os conceitos para formular o problema da distância DCJ considerando duplicações e regiões intergênicas.

3.1 Distância DCJ com duplicações

Sejam A e B dois genomas com duplicações e sem regiões intergênicas, com $|A| = |B| = n$. Seja D uma decomposição consistente do grafo de adjacências $AG(A, B)$ e c_D o número de ciclos na decomposição D. Para uma decomposição D, a distância DCJ é calculada como em [17]:

$$\mathrm{DCJ}(D) = n - c_D \ . \tag{3.2}$$

O problema da distância DCJ para dois genomas A e B com genes duplicados é NP-difícil e consiste em encontrar a menor decomposição D que minimiza a Equação (3.2). Seja $\mathcal{D}$ o conjunto de todas as decomposições consistentes de $AG(A, B)$. A distância DCJ exata pode ser calculada utilizando a Equação (3.3) proposta em [17].

$$\mathrm{DCJ}_{\mathrm{dup}}(A, B) = \min_{D \in \mathcal{D}}(\mathrm{DCJ}(D)) \ . \tag{3.3}$$

A distância DCJ para genomas com genes duplicados pode ser obtida de forma exata com a formulação de PLI apresentada em [17], que pode ser vista na Figura 3.1. Essa formulação de PLI possui $O(|E|)$ variáveis e $O(|E|)$ restrições, onde $|E|$ é o número de arestas de $AG(A, B)$. Cada restrição do PLI será explicada a seguir.

$$\begin{aligned}
\max \quad & \sum_{1 \le i \le |V|} z_i && \text{(3.4a)}\\
\text{sujeito a} \quad & \sum_{(u,v) \in E,\ v \in ext(\mathcal{B})} x_{(u,v)} = 1, \quad \forall u \in ext(\mathcal{A}) && \text{(3.4b)}\\
& \sum_{(u,v) \in E,\ u \in ext(\mathcal{A})} x_{(u,v)} = 1, \quad \forall v \in ext(\mathcal{B}) && \text{(3.4c)}\\
& x_e = 1, \quad \forall e \in E_{adj} && \text{(3.4d)}\\
& x_{(f^t, g^t)} = x_{(f^h, g^h)}, \quad \forall f \in \mathcal{A} \text{ e } \forall g \in \mathcal{B} \text{ que são homólogos} && \text{(3.4e)}\\
& 0 \le y_i \le i, \quad \forall 1 \le i \le |V| && \text{(3.4f)}\\
& y_i \le y_j + i \cdot (1 - x_e), \quad \forall e = (v_i, v_j) \in E && \text{(3.4g)}\\
& y_j \le y_i + j \cdot (1 - x_e), \quad \forall e = (v_i, v_j) \in E && \text{(3.4h)}\\
& i \cdot z_i \le y_i, \quad \forall 1 \le i \le |V| && \text{(3.4i)}\\
& x_e \in \{0, 1\}, \quad \forall e \in E && \text{(3.4j)}\\
& z_i \in \{0, 1\}, \quad \forall 1 \le i \le |V| && \text{(3.4k)}
\end{aligned}$$

Figura 3.1: Formulação de PLI para o problema da distância wDCJ.

Para cada aresta $e \in E$, é criada uma variável binária x_e que indica se a aresta e fará parte da decomposição final. A restrição (3.4d) garante que todas as arestas de adjacência façam parte da decomposição final.

As restrições (3.4b) e (3.4c) garantem que, para cada vértice, somente uma aresta de extremidade que incide nesse vértice seja escolhida. A restrição (3.4e) garante que a decomposição seja consistente.

Para identificar cada vértice $v_i \in V$, para $1 \le i \le |V|$, é criada uma variável y_i que indica o rótulo de v_i. As restrições (3.4g) e (3.4h) garantem que todos os vértices que pertencem ao mesmo ciclo na decomposição final, possuam o mesmo rótulo.

A restrição (3.4i) utiliza a variável binária z_i para indicar quando y_i é igual ao seu limitante superior i. Como todos os vértices de um mesmo ciclo possuem o mesmo rótulo,

e cada y_i possui um limitante superior distinto, somente um vértice do ciclo terá $y_i = i$. Portanto, cada variável $z_i = 1$ identifica um ciclo da decomposição final.

Por fim, visando minimizar a Equação (3.2), o PLI encontra a decomposição que maximiza o número de ciclos através da função objetivo (3.4a).

3.2 Grafo de breakpoint e distância DCJ com regiões intergênicas

Sejam $\mathcal{A}$ e $\mathcal{B}$ dois genomas balanceados com regiões intergênicas e sem genes duplicados. Como os genomas são balanceados, temos que $\mathcal{G}(\mathcal{A}) = \mathcal{G}(\mathcal{B})$ e $ext(\mathcal{A}) = ext(\mathcal{B})$. Esses genomas podem ser representados em uma estrutura conhecida como grafo de breakpoint [1], denotada por $BG(\mathcal{A}, \mathcal{B})$.

Um *grafo de breakpoint* $BG(\mathcal{A}, \mathcal{B})$ possui como conjunto de vértices o conjunto de extremidades de um dos dois genomas. Uma aresta conecta duas extremidades se elas formam uma adjacência em $\mathcal{A}$ ou se formam uma adjacência em $\mathcal{B}$. O peso de uma aresta é igual ao comprimento da região intergênica entre os genes adjacentes. A Figura 3.2 ilustra um exemplo de grafo de breakpoint.

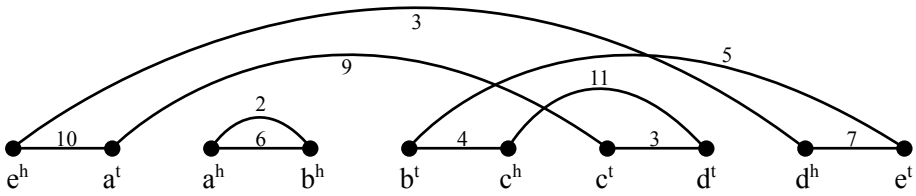


Figura 3.2: Grafo de breakpoint $BG(\mathcal{A}, \mathcal{B})$ para dois genomas balanceados com regiões intergênicas e sem duplicações, $\mathcal{A} = (A, \breve{A})$ e $\mathcal{B} = (B, \breve{B})$, com $A = \{(a\ -b\ -c\ d\ e)\}$, $\breve{A} = \{(10\ 6\ 4\ 3\ 7)\}$, $B = \{(b\ -a\ c\ d\ -e)\}$ e $\breve{B} = \{(5\ 2\ 9\ 11\ 3)\}$. Arestas retas representam as adjacências do genoma $\mathcal{A}$. Arestas curvas representam as adjacências do genoma $\mathcal{B}$.

Seja C um ciclo de $BG(\mathcal{A}, \mathcal{B})$. Denotamos como $w_{\mathcal{A}}(C)$ a soma do peso das arestas de C que pertencem ao genoma $\mathcal{A}$, e denotamos como $w_{\mathcal{B}}(C)$ a soma dos pesos das arestas de C que pertencem à $\mathcal{B}$. O *desequilíbrio* de um ciclo C é definido como $I(C) = w_{\mathcal{A}}(C) - w_{\mathcal{B}}(C)$. Um ciclo C é balanceado se $I(C) = 0$, e é desbalanceado caso contrário. O número de ciclos desbalanceados em $BG(\mathcal{A}, \mathcal{B})$ é denotado como n_u.

Por exemplo, considere o ciclo $C = e^h\, a^t\, c^t\, d^t\, c^h\, b^t\, e^t\, d^h\, e^h$ do grafo da Figura 3.2. Para esse ciclo, temos $w_{\mathcal{A}}(C) = 10 + 4 + 3 + 7 = 24$ e $w_{\mathcal{B}}(C) = 9 + 11 + 5 + 3 = 28$. Portanto, temos que $I(C) = 24 - 28 = -4$, o que significa que esse ciclo é desbalanceado.

Uma operação DCJ com regiões intergênicas (wDCJ) pode influenciar no número de ciclos do grafo de breakpoint de três maneiras diferentes [22]:

1. o número de ciclos permanece o mesmo;
2. o número de ciclos aumenta em um (*divisão de ciclos*);
3. o número de ciclos diminui em um (*fusão de ciclos*).

O problema da distância wDCJ para dois genomas $\mathcal{A}$ e $\mathcal{B}$ com regiões intergênicas e sem duplicações é NP-difícil [22] e consiste em determinar o número mínimo de operações wDCJ para transformar $\mathcal{A}$ em $\mathcal{B}$.

Seja n o número de genes em $\mathcal{A}$ e em $\mathcal{B}$, c o número de ciclos em $BG(\mathcal{A}, \mathcal{B})$, e m o número mínimo de fusões de ciclos necessárias para obter um conjunto de ciclos balanceados a partir dos ciclos desbalanceados de $BG(\mathcal{A}, \mathcal{B})$. A distância wDCJ entre $\mathcal{A}$ e $\mathcal{B}$ é calculada da seguinte maneira [22]:

$$\text{wDCJ}(\mathcal{A}, \mathcal{B}) = n - c + 2m \ . \tag{3.5}$$

Uma aproximação para a distância wDCJ pode ser calculada usando o algoritmo de 4/3-aproximação proposto em [22] (veja a Seção 3.2.1). A distância wDCJ exata pode ser obtida com a formulação de PLI proposta em [22], que pode ser vista na Figura 3.3. Essa formulação de PLI, na verdade, resolve um problema conhecido como *Max-Zero-Sum-Partition* (MZSP).

O problema MZSP recebe como entrada um multiconjunto $\mathcal{S} = \{s_1, s_2, \ldots, s_k\}$ de k números inteiros não-negativos e gera como saída uma partição de cardinalidade máxima $\{S_1, S_2, \ldots, S_p\}$ de $\mathcal{S}$, tal que $\sum_{s_i \in S_j} s_i = 0$, $\forall 1 \leq j \leq p$. Seja $\mathcal{C}_d = \{C_1, C_2, \ldots, C_{n_u}\}$ o conjunto de ciclos desbalanceados de $BG(\mathcal{A}, \mathcal{B})$. Se o multiconjunto $\mathcal{S}$ for composto pelos desequilíbrios dos ciclos de $\mathcal{C}_d$, a cardinalidade máxima p pode ser utilizada para calcular o número de merges $m = n_u - p$. O tamanho do PLI depende de n_u e ele possui $\Theta({n_u}^2)$ variáveis e $\Theta(n_u)$ restrições. Cada uma das restrições do PLI será explicada a seguir.

$$\begin{aligned}
\max \quad & \sum_{1 \leq i \leq n_u/2} p_i && \text{(3.6a)}\\
\text{sujeito a} \quad & \sum_{1 \leq j \leq n_u/2} x_{i,j} = 1, \quad \forall 1 \leq i \leq n_u && \text{(3.6b)}\\
& \sum_{1 \leq i \leq n_u} I(C_i) \cdot x_{i,j} = 0, \quad \forall 1 \leq j \leq n_u/2 && \text{(3.6c)}\\
& \sum_{1 \leq i \leq n_u} x_{i,j} \geq p_j, \quad \forall 1 \leq j \leq n_u/2 && \text{(3.6d)}\\
& x_{i,j} \in \{0,1\}, \quad \forall 1 \leq i \leq n_u \text{ e } \forall 1 \leq j \leq n_u/2 && \text{(3.6e)}\\
& p_i \in \{0,1\}, \quad \forall 1 \leq i \leq n_u/2 && \text{(3.6f)}
\end{aligned}$$

Figura 3.3: Formulação de PLI para o problema da distância DCJ com regiões intergênicas.

Cada variável $x_{i,j}$ do PLI da Figura 3.3 indica se o ciclo $C_i \in \mathcal{C}_d$ foi escolhido para fazer parte do subconjunto S_j na solução final. A restrição (3.6b) garante que cada ciclo C_i seja atribuído a um único subconjunto S_j. O objetivo da restrição (3.6c) é garantir que a soma dos desequilíbrios dos ciclos de um subconjunto S_j seja igual a zero.

Cada variável p_i indica se o subconjunto S_i foi usado na solução, isto é, $p_i = 1$ se $S_i \neq \emptyset$, e $p_i = 0$ caso contrário. A restrição (3.6d) garante que $p_i = 0$ se o subconjunto S_i não for usado na solução.

Por fim, a função objetivo (3.6a) garante que a partição de $\mathcal{C}_d$ encontrada possua cardinalidade máxima.

3.2.1 Algoritmo de aproximação

Sejam $\mathcal{A}$ e $\mathcal{B}$ dois genomas com regiões intergênicas e sem duplicações. Seja $BG(\mathcal{A}, \mathcal{B})$ o grafo de breakpoint para os genomas $\mathcal{A}$ e $\mathcal{B}$, e seja $\mathcal{C}_d = \{C_1, C_2, \ldots, C_{n_u}\}$ o conjunto de ciclos desbalanceados de $BG(\mathcal{A}, \mathcal{B})$. O objetivo do algoritmo de aproximação é encontrar uma aproximação m' para o número mínimo de fusões de ciclos m necessário para obter um conjunto de ciclos desbalanceados a partir dos ciclos desbalanceados de $BG(\mathcal{A}, \mathcal{B})$. O algoritmo proposto em [22] é dividido em quatro etapas, que serão descritas a seguir.

1. Crie uma cópia do conjunto de ciclos desbalanceados $\mathcal{C}_d$. Chamamos essa cópia de $\mathcal{C}_d{}'$;
2. Encontre o número máximo de pares de ciclos $\{C_i, C_j\}$ de $\mathcal{C}_d{}'$, tal que $I(C_i)+I(C_j) = 0$. Isso pode ser feito por meio de um algoritmo guloso que procura iterativamente por um número e seu oposto entre os desequilíbrios dos ciclos de $\mathcal{C}_d{}'$. Para cada par de ciclos C_i, C_j encontrados, tal que $I(C_i) = -I(C_j)$, removemos C_i e C_j de $\mathcal{C}_d{}'$ e incrementamos o valor de m' em 1;
3. Encontre entre os ciclos restantes de $\mathcal{C}_d{}'$, uma aproximação para o número máximo de trios de ciclos que podem ser unidos para formar um ciclo balanceado. Encontrar uma solução exata para esse problema é equivalente a um problema conhecido como *Max-Zero-Sum-Triplets* (MZS3), que é fortemente NP-completo [22]. Portanto, nesse passo é utilizado um algoritmo de aproximação. A aproximação pode ser encontrada por meio de um algoritmo guloso que, para cada ciclo C_i, procura dois outros ciclos C_j e C_k, tal que $I(C_i) = -(I(C_j) + I(C_k))$. Para cada trio de ciclos encontrado, removemos C_i, C_j e C_k de $\mathcal{C}_d{}'$, e incrementamos o valor de m' em 2;
4. Una os ciclos que restaram em $\mathcal{C}_d{}'$ para formar um único ciclo balanceado. O valor de m' é incrementado em $|\mathcal{C}_d{}'| - 1$.

Após essas etapas, obteremos o valor m', que pode ser utilizado para obter uma aproximação para a distância wDCJ utilizando a Equação (3.5).

3.3 Distância DCJ com duplicações e regiões intergênicas

Unindo os problemas apresentados na Seção 3.1 e na Seção 3.2, formularemos a definição do problema da distância DCJ para genomas com regiões intergênicas e duplicações. Chamamos esse problema de distância wDCJ com duplicações.

Sejam $\mathcal{A} = (A, \check{A})$ e $\mathcal{B} = (B, \check{B})$ dois genomas com regiões intergênicas e genes duplicados. Dada uma decomposição consistente D do grafo de adjacências $AG(\mathcal{A}, \mathcal{B})$, renomeamos os genes de $\mathcal{B}$ e denotamos o genoma com os genes renomeados como $\mathcal{B}^D$. Os rótulos dos genes do genoma $\mathcal{B}^D$ são construídos da seguinte maneira. Para cada associação de genes $g_i \to g_j$ definida por D, onde $g_i \in \mathcal{G}(\mathcal{A})$ e $g_j \in \mathcal{G}(\mathcal{B})$, renomeamos g_j como g_i, de forma que $\mathcal{G}(\mathcal{B}^D) = \mathcal{G}(\mathcal{A})$. Como $\mathcal{A}$ e $\mathcal{B}^D$ possuem o mesmo conjunto de genes, podemos representar $\mathcal{A}$ e $\mathcal{B}^D$ no grafo de breakpoint $BG(\mathcal{A}, \mathcal{B}^D)$. A Figura 3.4 mostra um exemplo de grafo de breakpoint para a decomposição D da Figura 2.1b.

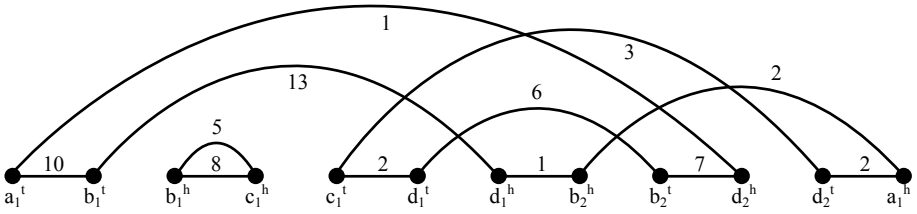


Figura 3.4: Grafo de breakpoint $BG(\mathcal{A}, \mathcal{B}^D)$ para a decomposição D da Figura 2.1b, com $A = \{(-a_1\ b_1\ -c_1\ d_1\ -b_2\ -d_2)\}$, $\breve{A} = \{(2\ 10\ 8\ 2\ 1\ 7)\}$, $B^D = \{(-d_2\ c_1\ -b_1\ -d_1\ b_2\ -a_1)\}$ e $\breve{B} = \{(1\ 3\ 5\ 13\ 6\ 2)\}$.

Para dois genomas $\mathcal{A}$ e $\mathcal{B}$ com n genes, a distância wDCJ para uma decomposição consistente D é dada por:

$$\text{wDCJ}(\mathcal{A}, \mathcal{B}^D) = n - c_D + 2m_D\ , \tag{3.7}$$

onde c_D é o número de ciclos na decomposição D e m_D é o número mínimo de fusões de ciclos necessárias para obter um conjunto de ciclos balanceados a partir dos ciclos desbalanceados de $BG(\mathcal{A}, \mathcal{B}^D)$.

O problema da distância wDCJ com duplicações para dois genomas $\mathcal{A}$ e $\mathcal{B}$ consiste em determinar a decomposição que minimiza a Equação (3.7). Seja $\mathcal{D}$ o conjunto de todas as decomposições consistentes de $AG(\mathcal{A}, \mathcal{B})$. A distância wDCJ considerando genes duplicados é dada por:

$$\text{wDCJ}_{\text{dup}}(\mathcal{A}, \mathcal{B}) = \min_{D \in \mathcal{D}}(\text{wDCJ}(\mathcal{A}, \mathcal{B}^D))\ . \tag{3.8}$$

Como calcular a distância DCJ com regiões intergênicas para uma decomposição D já é um problema NP-difícil, encontrar uma decomposição consistente de $AG(\mathcal{A}, \mathcal{B})$ que minimiza a Equação (3.7) também é NP-difícil. Até o momento, não existe uma formulação de PLI que compute a distância wDCJ exata para genomas com genes duplicados.

Capítulo 4

Métodos

Para obter uma solução heurística para o problema da distância wDCJ com genes duplicados para dois genomas $\mathcal{A}$ e $\mathcal{B}$, primeiro, precisamos obter uma decomposição consistente D do grafo de adjacências $AG(\mathcal{A}, \mathcal{B})$. Após encontrar essa decomposição, podemos tratar os genomas $\mathcal{A}$ e $\mathcal{B}^D$ como se não houvesse duplicações, e calcular a distância wDCJ para a decomposição D utilizando o PLI da Figura 3.3, proposto em [22], ou o algoritmo de 4/3-aproximação descrito na Seção 3.2.1, proposto em [22]. O algoritmo de aproximação foi implementado em Python.

Utilizamos dois métodos diferentes para encontrar uma decomposição D do grafo de adjacências. O primeiro deles consiste em encontrar uma decomposição consistente de forma aleatória, estabelecendo uma associação aleatória entre os genes homólogos dos dois genomas. O segundo visa encontrar uma decomposição consistente que fornece a solução ótima para o problema da distância DCJ para genomas com genes duplicados sem regiões intergênicas. Chamamos essa decomposição de *decomposição ótima*. Para encontrar tal decomposição, utilizamos um projeto chamado DING [1] [28], que implementa o PLI para o problema da distância DCJ-indel em genomas naturais.

O programa principal do DING, chamado "dingII.py", recebe como entrada um genoma sem regiões intergênicas e gera uma formulação de PLI para o problema da distância DCJ-indel em genomas naturais, que é armazenada em um arquivo no formato LP. Esse arquivo é usado como entrada para o *solver* Gurobi Optimizer, que encontra uma solução para o problema. Após isso, outro programa do DING, chamado "dingII_parsesol.py", recebe a solução encontrada pelo Gurobi e retorna a decomposição consistente que forneceu essa solução. A Figura 4.1 ilustra o método heurístico.

Todos os experimentos foram realizados em genomas simulados gerados com as ferramentas descritas no Capítulo 5. Para avaliar a qualidade dos resultados obtidos com a nossa abordagem, nós comparamos os nossos resultados com os obtidos pelas heurísticas baseadas em *adjacency graph packing* [27].

As heurísticas propostas em [27] se baseiam na ideia de produzir k decomposições consistentes distintas e selecionar a melhor utilizando como critério o número de ciclos balanceados na decomposição — em [27], uma decomposição consistente é chamada de

[1] https://gitlab.ub.uni-bielefeld.de/gi/dingiiofficial

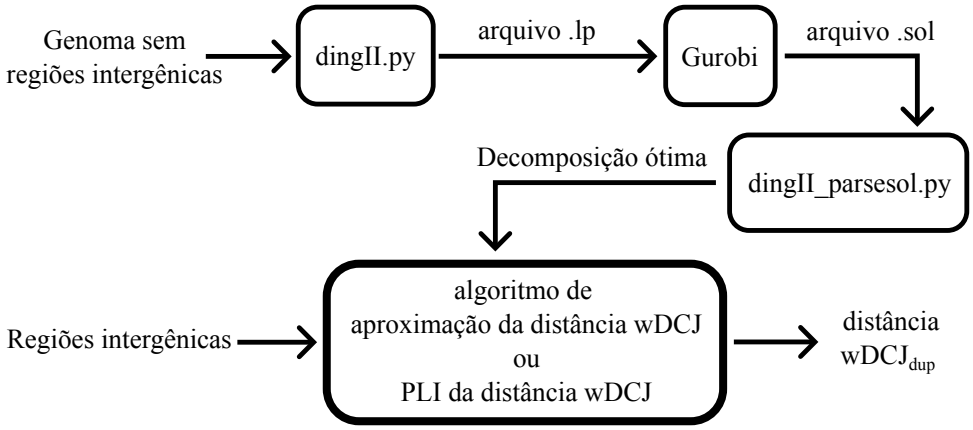


Figura 4.1: Visão geral da heurística proposta.

alternating cycle packing. As heurísticas propostas incluem um algoritmo aleatório, um algoritmo genético (GA) e um algoritmo de busca em largura que utiliza uma abordagem gulosa, chamado *Greedy Breadth-First-Search* (GBFS). Após selecionar a melhor decomposição, os genomas são tratados como não tivessem duplicações, e então, é utilizado o algoritmo de aproximação descrito na Seção 3.2.1 para obter uma solução heurística para a distância wDCJ com genes duplicados. As três heurísticas estão disponíveis no repositório do GitHub [2] fornecido pelos autores.

Os seguintes parâmetros das heurísticas podem ser modificados: o número de iterações k, a taxa de mutação do algoritmo genético mr, a taxa de crossover cr, e o número de elementos em cada torneio do algoritmo genético t. Em todos os experimentos, definimos $k = 1000$, $t = 2$, $cr = 50$ e $mr = 20$, da mesma maneira em que foi feito em [27].

[2] https://github.com/compbiogroup/Heuristics-based-on-Adjacency-Graph-Packing-for-DCJ-Distance-Considering-Intergenic-Regions

Capítulo 5

Datasets

Este capítulo apresenta as ferramentas utilizadas para construir os nossos datasets. Cada genoma é representado por dois arquivos. Um dos arquivos contém somente os genes dos cromossomos, e o outro contém os comprimentos das regiões intergênicas dos cromossomos.

Como nenhuma ferramenta existente oferece a flexibilidade e os recursos necessários, — especialmente no que diz respeito ao controle das regiões intergênicas e ao número de ciclos desbalanceados — utilizamos uma combinação de ferramentas, juntamente com etapas de pré e pós-processamento, dependendo do dataset.

5.1 Zombi

O Zombi [1] [34] é um programa que simula a evolução de espécies. Inicialmente, ele simula a evolução de uma árvore de espécies a partir de um genoma ancestral, continuando até que um limite de tempo seja alcançado, ou que um número predefinido de linhagens seja obtido. Com base nessa árvore, ele então simula a evolução do genoma considerando o tamanho do genoma ancestral, os comprimentos das regiões intergênicas e uma variedade de eventos evolutivos com suas respectivas taxas.

Exigimos, para todas as espécies existentes, (i) genomas balanceados e (ii) soma dos comprimentos das regiões intergênicas idênticos. Portanto, desabilitamos todos os eventos que modificam o conteúdo do genoma, como deleções e duplicações. Note que, sem duplicações, os genomas existentes conterão somente uma cópia de cada gene, pois o genoma ancestral contém somente genes únicos. Consequentemente, ao usar o Zombi, a definição de famílias é feita manualmente. Após a simulação, pós-processamos a saída para atribuir o mesmo rótulo de família aos genes que consideramos pertencer à mesma família.

Sequências de genes podem ser geradas em uma etapa final, entretanto, elas não são necessárias para os nossos experimentos.

[1] https://github.com/AADavin/Zombi

5.2 Gerador de genomas com regiões conservadas

Desenvolvemos um gerador de genomas que cria dois genomas descendentes $\mathcal{A}$ e $\mathcal{B}$ a partir de um ancestral em comum $\mathcal{G}_0$.

Para gerar $\mathcal{G}_0$, primeiro criamos um genoma com n genes distintos. Posteriormente, são realizadas s duplicações de segmentos, cada uma duplicando um segmento aleatório que possui um tamanho entre 1 e m, resultando em um genoma com n' genes. A criação de $\mathcal{G}_0$ é finalizada atribuindo um comprimento aleatório entre 1 e i a cada região intergênica.

Após criar o genoma ancestral, r eventos de rearranjo são realizados para gerar o genoma $\mathcal{A}$, e outros r eventos de rearranjo independentes são realizados para produzir o genoma $\mathcal{B}$. Os eventos de rearranjo consistem em reversões e translocações, que "cortam" o genoma em posições aleatórias em regiões intergênicas, modificando possivelmente os comprimentos das regiões, mas preservando o comprimento total do genoma. Cada genoma descendente possui n' genes e n' regiões intergênicas.

Em seguida, modificamos somente o genoma $\mathcal{B}$, aplicando k perturbações às suas regiões intergênicas. Uma *perturbação* subtrai um valor inteiro arbitrário do comprimento de uma região intergênica selecionada aleatoriamente e o adiciona ao comprimento de outra região, alterando, portanto, dois comprimentos de regiões intergênicas, e preservando sua soma. As perturbações são adicionadas com dois propósitos: (i) aumentar o número de ciclos desbalanceados no grafo de adjacências, e (ii) aproximar mutações intergênicas reais de maneira simplificada enquanto preservamos o comprimento total das regiões intergênicas. O valor k não é um parâmetro de entrada do nosso gerador de genomas, e sim derivado de uma porcentagem $p \in (0, 100]$ definida pelo usuário, isto é, $k = \lfloor (p/100) \cdot n' \rfloor$.

O número de perturbações possui um impacto significativo no tempo de execução dos algoritmos, e, ao mesmo tempo, gerar perturbações é um passo importante para simular genomas que se aproximem mais de dados reais.

Durante a aplicação dos eventos de rearranjo para transformar o genoma ancestral em cada um dos dois genomas descendentes, é feita a contagem do número de operações wDCJs realizadas. Cada reversão pode ser realizada com uma única operação DCJ. Como, no nosso caso, as reversões podem alterar os comprimentos das regiões intergênicas, então, cada reversão corresponde a uma operação wDCJ. Cada operação de transposição pode ser representada por duas operações wDCJs, e, portanto, cada operação de transposição contribui com duas unidades para a contagem total de operações wDCJs. Cada perturbação corresponde a uma única operação wDCJ, na qual somente os comprimentos das regiões intergênicas são alterados, e as adjacências permanecem as mesmas.

Seja w_1 o número total de operações wDCJs para transformar $\mathcal{G}_0$ em $\mathcal{A}$, e seja w_2 o número total de operações wDCJs para transformar $\mathcal{G}_0$ em $\mathcal{B}$. Se cada operação realizada é única, o número de operações wDCJs necessárias para transformar $\mathcal{A}$ em $\mathcal{B}$ é dado por $w_1 + w_2$, pois são necessárias w_1 operações para transformar $\mathcal{A}$ em $\mathcal{G}_0$, e w_2 operações para transformar $\mathcal{G}_0$ em $\mathcal{B}$.

Cada um dos parâmetros do gerador de genomas será apresentado a seguir:

- n: tamanho inicial do genoma ancestral;
- m: comprimento máximo de um segmento que será duplicado ou rearranjado;

- s: número de duplicações de segmento aplicadas ao genoma ancestral;
- r: número de rearranjos aplicados a cada genoma descendente;
- i: tamanho máximo atribuído a uma região intergênica;
- p: porcentagem utilizada para calcular o número de perturbações k aplicado ao genoma $\mathcal{B}$.

Capítulo 6

Experimentos

Nesta seção, apresentaremos os experimentos realizados, que visam avaliar o tempo de execução e a qualidade da disstância wDCJ encontrada pelos algoritmos.

Na Seção 6.1 será feita uma comparação entre as distâncias wDCJs obtidas por meio de decomposições aleatórias e as distâncias wDCJs obtidas por meio de decomposições ótimas. Na Seção 6.2, avaliaremos a influência do número de perturbações nos tempos de execução do algoritmo de PLI para a distância DCJ com regiões intergênicas e do algoritmo de aproximação para a distância wDCJ. Na Seção 6.3 comparamos o tempo de execução do PLI da distância DCJ-indel com os das heurísticas propostas em [27]. Por fim, na Seção 6.4 será feita uma comparação entre as distâncias wDCJs obtidas através da decomposição ótima, e as distâncias wDCJs obtidas através das decomposições encontradas utilizando as heurísticas baseadas em *adjacency graph packing*.

Especificações do sistema

Todos os experimentos foram realizados em um computador com 32 GB de memória RAM e com um Intel Core i9-14900KF com 24 núcleos físicos e 32 threads.

6.1 Decomposições aleatórias

O objetivo do experimento desta seção é avaliar se a decomposição que fornece a melhor distância DCJ (sem regiões intergênicas) também fornece a melhor distância wDCJ (considerando regiões intergênicas).

Neste experimento, geramos múltiplas decomposições aleatórias do grafo de adjacências, e, para cada uma, computamos a distância wDCJ usando o algoritmo de PLI proposto em [22]. Conforme foi visto anteriormente, uma decomposição define uma associação de um para um entre os genes homólogos dos genomas (como em genomas sem duplicações), o que nos permite aplicar esse algoritmo. Também geramos uma decomposição ótima utilizando o DING, e, novamente, calculamos a distância wDCJ para essa decomposição com a mesma formulação de PLI.

Os genomas usados neste experimento foram gerados com o Zombi, e as famílias foram definidas seguindo a mesma distribuição de tamanhos de famílias encontrada na bactéria Escherichia coli. Geramos 10 genomas com 1000 genes e 1000 regiões intergênicas cada.

Formando todos os pares possíveis com os 10 genomas, foram obtidos 45 pares de genomas distintos. Para cada par de genes, criamos 6 pares adicionais de regiões intergênicas, através da inserção de perturbações com valores entre 1 e 6, totalizando $45 \times 7 = 315$ pares de genomas. Limitamos o número de perturbações a 6 devido ao número elevado de decomposições geradas para cada par de genomas — executamos o PLI da distância wDCJ para 1000 decomposições aleatórias de cada grafo de adjacências, e para mais uma fornecida pelo DING, totalizando $45 \times 7 \times 1001 = 315.315$ decomposições.

Cada PLI foi resolvido utilizando o software Gurobi Optimizer, com um tempo limite de 51 segundos. Esse tempo limite foi escolhido devido à elevada quantidade de PLIs a serem resolvidos, e baseado em um experimento prévio que mostrou que as soluções não melhoravam depois dos 50 segundos, mesmo estendendo o tempo de execução para 10 minutos. A Figura 6.1 mostra as distâncias wDCJs médias e os desvios padrão obtidos para as decomposições aleatórias e ótimas para cada número de perturbações. O tempo de execução total do experimento para os 45 pares de genomas para cada número de perturbações pode ser visto na Tabela 6.1.

Observamos que, para todos os pares de genomas e todos os números de perturbações, a melhor distância DCJ considerando regiões intergênicas foi obtida com a decomposição ótima. Comparando os resultados obtidos na Figura 6.1, podemos verificar que a média das distâncias wDCJ das decomposições ótimas são muito menores do que a média das distâncias wDCJ das decomposições aleatórias.

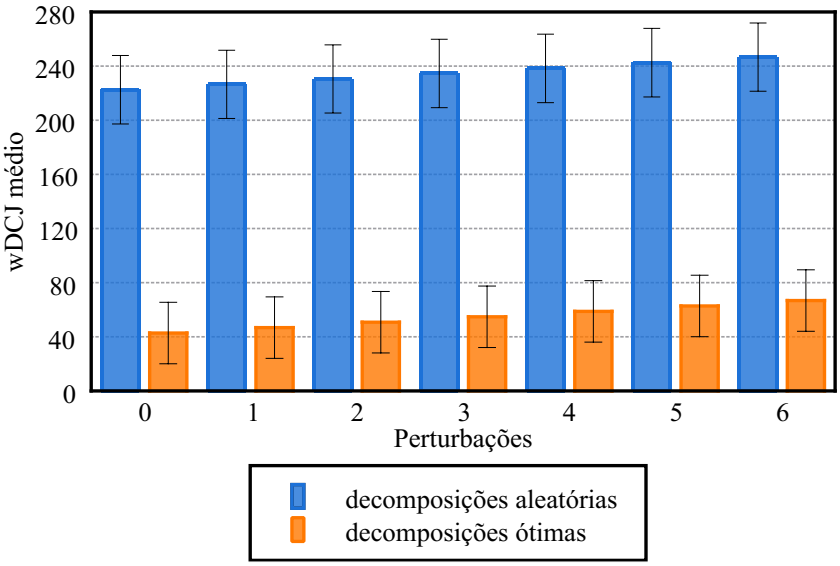


Figura 6.1: Distâncias wDCJs médias e desvios padrão para as decomposições aleatórias e ótimas de 45 pares de genomas. Cada barra azul mostra a média de 45.000 distâncias wDCJs obtidas com todas as decomposições aleatórias geradas para um determinado número de perturbações. Cada barra laranja mostra a média de 45 distâncias wDCJs obtidas com todas as decomposições ótimas obtidas para um número de perturbações. As barras pretas com linhas horizontais nas extremidades representam os desvios padrão.

Nota-se também que o tempo de execução do experimento se torna muito grande mesmo com a introdução de uma pequena quantidade de perturbações nas regiões in-

tergênicas. Para genomas sem perturbações, o experimento foi concluído em um pouco menos de 10 minutos. Com 1 perturbação, esse tempo triplicou, e com a inserção de 6 perturbações, foram necessários 3 dias para executar o experimento para todos os pares de genomas. Esse grande aumento no tempo de execução a cada incremento no número de perturbações tornou inviável executar o experimento para um número maior de perturbações.

Tabela 6.1: Tempo total do experimento das decomposições aleatórias por número de perturbações

Perturbações	**Tempo total (s)**
0	565,01
1	1.740,60
2	7.086,68
3	17.269,38
4	59.478,86
5	146.666,17
6	265.111,58

6.2 Tempo de execução por número de perturbações

Os dois experimentos que serão abordados nesta seção foram realizados para avaliar como o PLI e o algoritmo de aproximação para a distância wDCJ são afetados pelo aumento do número de perturbações. Nesses experimentos, usamos os 45 pares de genomas do experimento anterior, e, para cada um deles, encontramos uma decomposição ótima utilizando o DING. Para cada decomposição ótima, utilizamos regiões intergênicas com diferentes números de perturbações.

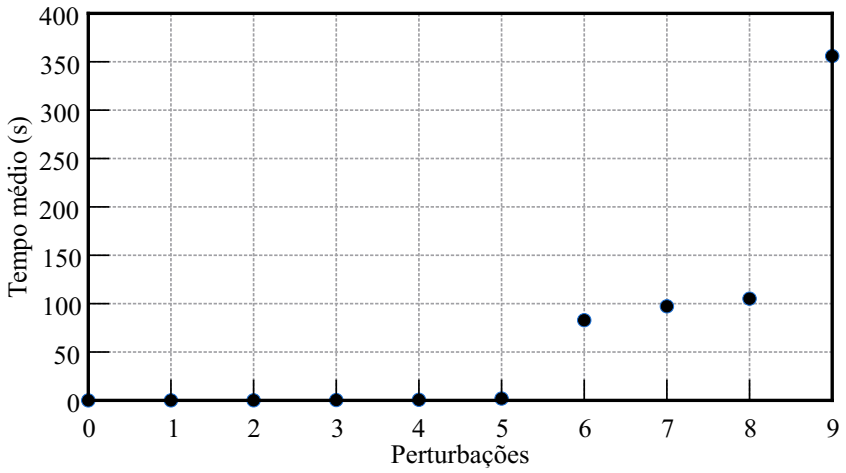


Figura 6.2: Tempo de execução médio do PLI para a distância DCJ com regiões intergênicas por número de perturbações. Cada ponto do gráfico indica a média do tempo de execução para as 45 decomposições ótimas.

Executamos o PLI da distância wDCJ para as 45 decomposições ótimas, com o número

de perturbações variando de 0 a 9. Para cada PLI, definimos um tempo limite de 1 hora. Tentativas de executar o PLI para perturbações maiores do que 9 resultaram em erros por falta de memória durante a otimização. A Figura 6.2 mostra o tempo de execução médio para cada número de perturbações, evidenciando que o tempo de execução cresce substancialmente conforme o número de perturbações aumenta.

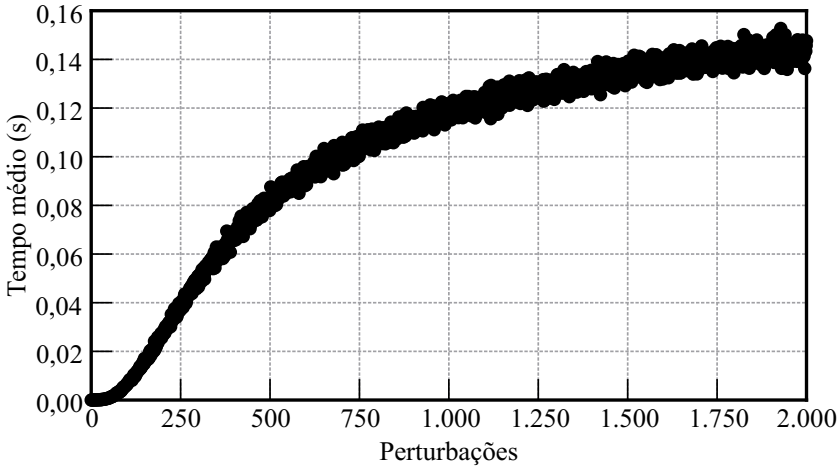


Figura 6.3: Tempo de execução médio por número de perturbações do algoritmo de aproximação para a distância wDCJ. Cada ponto do gráfico representa a média do tempo de execução para as 45 decomposições ótimas.

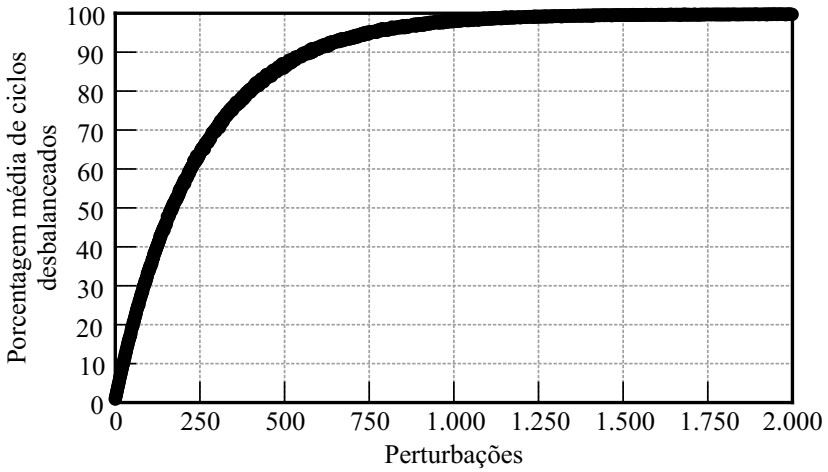


Figura 6.4: Porcentagem média de ciclos desbalanceados obtidos para as 45 decomposições ótimas para cada número de perturbações.

Como o algoritmo de aproximação é muito mais rápido do que o PLI, foi possível executá-lo para um número muito maior de perturbações. Executamos esse algoritmo para as 45 decomposições ótimas com perturbações variando de 0 a 2000. Os resultados podem ser vistos na Figura 6.3. Não estendemos a nossa análise para um número maior de perturbações porque o número de ciclos desbalanceados varia muito pouco após certo ponto. Como é possível observar na Figura 6.4, a partir de aproximadamente 1250 perturbações, quase todos os ciclos serão desbalanceados.

6.3 Tempo de execução por número de duplicações

Como foi dito no Capítulo 4, para encontrar uma solução heurística para a distância DCJ com genes duplicados e regiões intergênicas para dois genomas $\mathcal{A}$ e $\mathcal{B}$, primeiro encontramos uma decomposição consistente de $AG(\mathcal{A}, \mathcal{B})$ e em seguida calculamos a distância wDCJ para a decomposição encontrada. Em [27], essa decomposição é encontrada através das heurísticas baseadas em *adjacency graph packing*. Na nossa abordagem, a decomposição é encontrada através do PLI para a distância DCJ-indel em genomas naturais.

No experimento a seguir, comparamos o tempo de execução do PLI da distância DCJ-indel para genomas naturais [28] (DING) com os tempos de execução das heurísticas baseadas em *adjacency graph packing*. Utilizamos novamente os mesmos 10 genomas (45 pares) gerados pelo Zombi no experimento da seção anterior, porém as famílias foram definidas de uma maneira diferente.

Para analisar como o tempo de execução dos algoritmos se comporta conforme o número de duplicações aumenta, agrupamos os genes em famílias de forma aleatória até que um determinado número de duplicações fosse atingido. Para cada par de genomas, definimos as famílias de 6 maneiras distintas, correspondendo aos seguintes números de duplicações: 100, 200, 300, 400, 500 e 600. Em seguida, executamos o DING e as duas heurísticas para todos os pares de genomas criados, totalizando $45 \times 6 = 270$ execuções. Para cada PLI, foi definido um tempo limite de 1 hora, contudo, todos foram resolvidos muito antes do tempo limite ser alcançado. O tempo máximo de execução de um PLI foi de 0,31 segundos. Os resultados obtidos podem ser vistos na Figura 6.5.

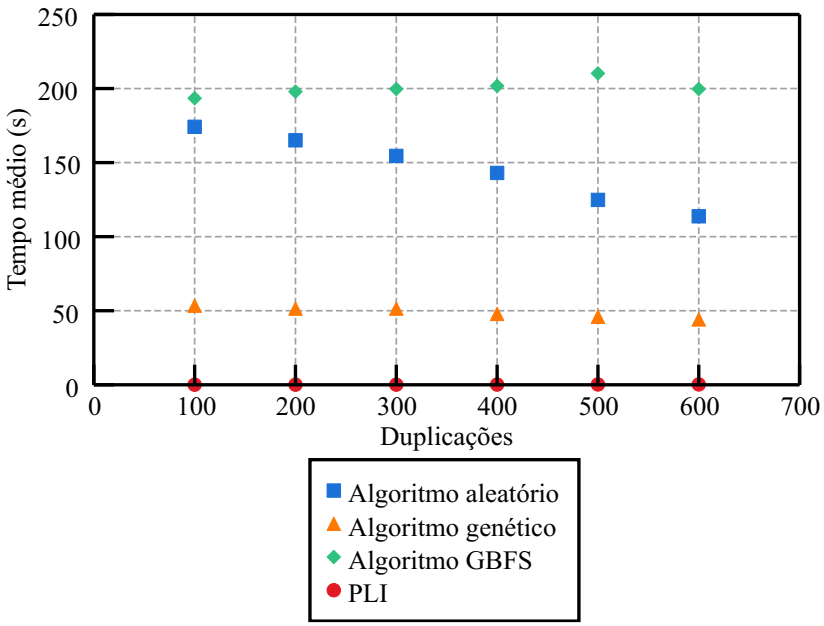


Figura 6.5: Tempo de execução por número de duplicações. Os círculos vermelhos representam o tempo de execução médio obtido pelo PLI da distância DCJ-indel para genomas naturais. Os quadrados azuis, triângulos laranjas e losangos verdes correspondem ao algoritmo aleatório, genético e GBFS, respectivamente.

O tempo médio de execução variou muito pouco para todos os algoritmos com o au-

mento do número de duplicações, e a abordagem utilizando o PLI apresentou o melhor tempo médio para todos os valores de duplicações.

6.4 Genomas com regiões conservadas

Os genomas utilizados neste experimento foram gerados com o nosso gerador de genomas (veja a Seção 5.2). Para cada valor de n de 1000 a 10000 (em intervalos de 1000), geramos 10 pares de genomas (note que o tamanho dos genomas pode aumentar devido às duplicações de segmentos), totalizando 100 pares. Para cada valor de n, aplicamos s duplicações de segmentos e r eventos de rearranjo, onde ambos s e r são 5% de n. Para o tamanho máximo de um segmento para duplicar ou rearranjar foi definido $m = 10$, para o tamanho máximo de cada região intergênica foi definido $i = 100$, e para a porcentagem de perturbações foi definido $p = 20$.

O intuito do experimento desta seção é comparar duas abordagens. Na primeira delas, calculamos como primeiro passo, para cada par de genomas, uma decomposição ótima usando a implementação de PLI para a distância DCJ-indel. Definimos um tempo limite de 1 hora para cada PLI, no entanto, todas as instâncias foram resolvidas antes do tempo limite ser atingido, demorando no máximo 501,22 segundos. Como segundo passo, executamos o algoritmo de 4/3-aproximação para a distância DCJ com regiões intergênicas, utilizando como entrada as decomposições ótimas encontradas no passo anterior. A Figura 4.1 ilustra esse processo.

Na segunda abordagem, executamos os algoritmos aleatório e genético para cada par de genomas (primeiro passo), e, após isso, executamos novamente o algoritmo de aproximação como um segundo passo.

A Figura 6.6 mostra os tempos de execução obtidos com ambas as abordagens. As distâncias wDCJs obtidas com cada abordagem, assim como o número real de operações wDCJs aplicadas para transformar um genoma em outro, podem ser vistas na Figura 6.7.

A abordagem que utiliza o PLI apresentou o melhor tempo de execução para todos os pares de genomas. O mesmo pode ser visto para a qualidade das distâncias obtidas. Para os genomas com menos de 3000 genes, as distâncias wDCJs obtidas através do PLI, do algoritmo genético e do algoritmo GBFS apresentaram valores similares, mas mesmo assim, as aproximações obtidas utilizando o PLI foram um pouco melhores. O algoritmo aleatório obteve as piores distâncias para todos os pares de genomas. Conforme o número de genes aumenta, as distâncias obtidas com o PLI se distanciam mais das distâncias encontradas pelas heurísticas.

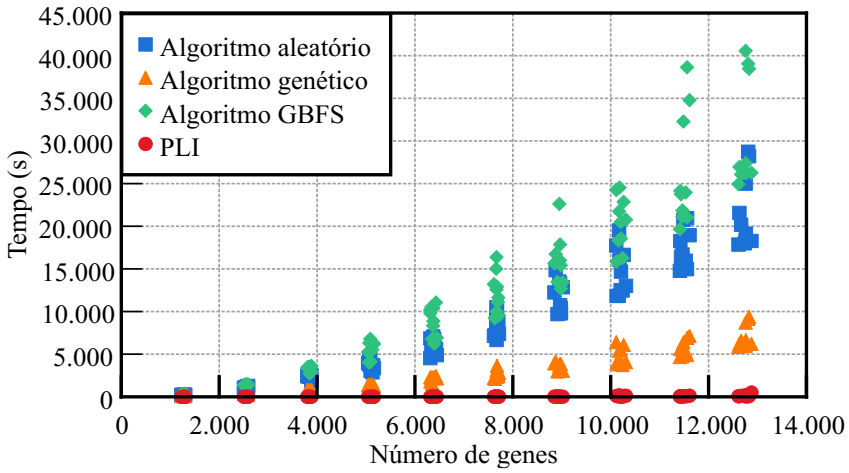


Figura 6.6: Tempo de execução de cada algoritmo como função do número de genes. Círculos vermelhos mostram o tempo de execução do PLI para a distância DCJ-indel (primeira abordagem). Quadrados azuis, triângulos laranjas e losangos verdes correspondem aos algoritmos aleatório, genético e GBFS (segunda abordagem), respectivamente.

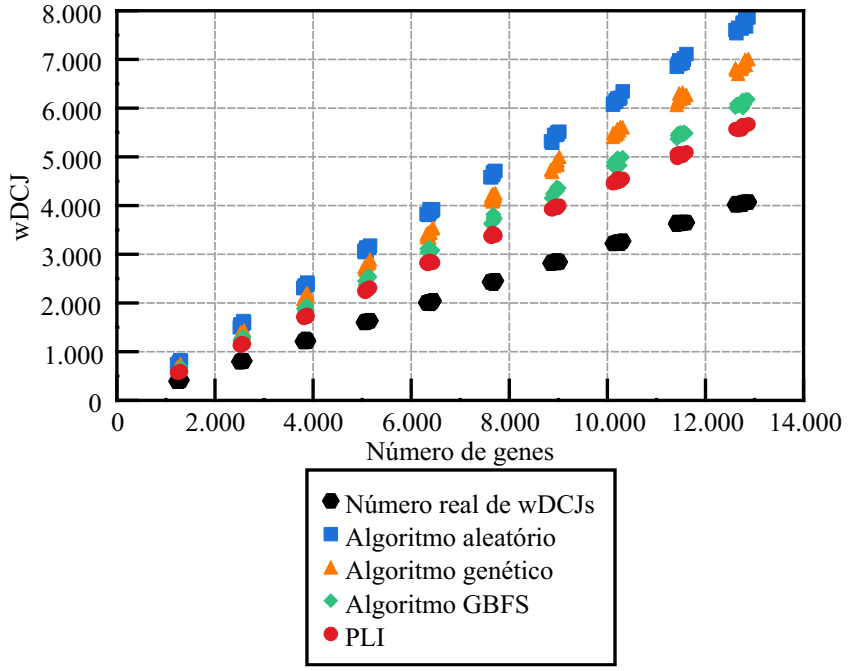


Figura 6.7: Tamanhos dos genomas e valores da distância wDCJ obtidas por diferentes algoritmos. Círculos vermelhos indicam os resultados obtidos com a primeira abordagem: obtendo uma decomposição ótima usando o PLI para a distância DCJ-indel, seguido pelo algoritmo de aproximação da distância wDCJ. Resultados obtidos com a segunda abordagem são representados por quadrados azuis, triângulos laranjas e losangos verdes, correspondendo ao algoritmo aleatório, algoritmo genético e algoritmo GBFS, respectivamente. O número real de operações wDCJ para transformar um genoma em outro está representado por hexágonos pretos.

Capítulo 7

Conclusão

Neste trabalho, foi apresentada uma nova heurística para o problema da distância wDCJ com genes duplicados. Nossos resultados demonstram que a combinação de métodos proposta permite o cálculo prático da distância wDCJ para genomas significativamente mais próximos dos reais, tanto em termos de tamanho quanto de complexidade estrutural.

Ao integrar decomposições ótimas baseadas em PLI com um algoritmo eficiente de aproximação, conseguimos lidar com genomas com milhares de genes e com uma quantidade substancial de desigualdades nos comprimentos das regiões intergênicas — cenários que anteriormente tornavam o cálculo da distância wDCJ inviável devido a restrições de memória ou tempos de execução excessivos. Nossos resultados também sugerem que a qualidade da solução wDCJ está fortemente relacionada com a decomposição que fornece a solução ótima para a distância DCJ sem regiões intergênicas.

Notavelmente, esse avanço foi alcançado sem comprometer a qualidade das soluções. Na verdade, nossa abordagem produziu soluções tão boas quanto ou melhores do que as obtidas por heurísticas anteriores, especialmente para genomas maiores e com mais desigualdades nos comprimentos de suas regiões intergênicas. Além disso, os tempos de execução foram significativamente reduzidos em comparação com métodos anteriores.

Esses resultados indicam um caminho promissor e prático para o cálculo da distância DCJ com regiões intergênicas em genomas com quantidades realistas de duplicações e de variações intergênicas. Trabalhos futuros podem se concentrar em integrar o processo atual de duas etapas em um único método unificado, e em calcular a distância wDCJ para genomas desbalanceados, reduzindo ainda mais a diferença entre dados simulados e dados biológicos reais.

Referências Bibliográficas

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

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

[3] John Kececioglu and David Sankoff. Exact and approximation algorithms for the inversion distance between two chromosomes. In *Combinatorial Pattern Matching*, pages 87–105, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[4] Sridhar Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71(1):137–151, 1996.

[5] Maria E.M.T. Walter, Zanoni Dias, and João Meidanis. Reversal and transposition distance of linear chromosomes. In *Proceedings. String Processing and Information Retrieval: A South American Symposium (Cat. No.98EX207)*, pages 96–102, 1998.

[6] Maria E.M.T. Walter, Zanoni Dias, and João Meidanis. A new approach for approximating the transposition distance. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 199–208, 2000.

[7] Alberto Caprara. Sorting by reversals is difficult. In *Proceedings of the First Annual International Conference on Computational Molecular Biology*, RECOMB '97, page 75–83, New York, NY, USA, 1997. Association for Computing Machinery.

[8] Andre Rodrigues Oliveira, Klairton Lima Brito, Ulisses Dias, and Zanoni Dias. On the complexity of sorting by reversals and transpositions problems. *Journal of Computational Biology*, 26(11):1223–1229, November 2019.

[9] Daming Zhu and Lusheng Wang. On the complexity of unsigned translocation distance. *Theoretical Computer Science*, 352(1):322–328, 2006.

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

[11] Luís Felipe I Cunha, Luis Antonio B Kowada, Rodrigo de A Hausen, and Celina M H de Figueiredo. A faster 1.375-approximation algorithm for sorting by transpositions. *J. Comput. Biol.*, 22(11):1044–1056, November 2015.

[12] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-approximation algorithm for sorting by reversals. *Electronic Colloquium Computational Complexity*, TR01, 2002.

[13] Yun Cui, Lusheng Wang, and Daming Zhu. A 1.75-approximation algorithm for unsigned translocation distance. In Xiaotie Deng and Ding-Zhu Du, editors, *Algorithms and Computation*, pages 392–401, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[14] 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.

[15] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 06 2005.

[16] Anne Bergeron, Julia Mixtacki, and Jens Stoye. A unifying view of genome rearrangements. In Philipp Bücher and Bernard M. E. Moret, editors, *Algorithms in Bioinformatics*, pages 163–173, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[17] Mingfu Shao, Yu Lin, and Bernard Moret. An exact algorithm to compute the dcj distance for genomes with duplicate genes. In Roded Sharan, editor, *Research in Computational Molecular Biology*, pages 280–292, Cham, 2014. Springer International Publishing.

[18] Aaryan Sarnaik, Ke Chen, Austin Diaz, and Mingfu Shao. An exact and fast sat formulation for the dcj distance. In *Research in Computational Molecular Biology*, Lecture Notes in Computer Science, pages 175–189. Springer Nature Switzerland, 04 2025.

[19] Diego P Rubert, Pedro Feijão, Marília Dias Vieira Braga, Jens Stoye, and Fábio Henrique Viduani Martinez. Approximating the DCJ distance of balanced genomes in linear time. *Algorithms for Molecular Biology*, 12(1):3, March 2017.

[20] Priscila Biller, Laurent Guéguen, Carole Knibbe, and Eric Tannier. Breaking good: Accounting for fragility of genomic regions in rearrangement distance estimation. *Genome Biology and Evolution*, 8(5):1427–1439, May 2016.

[21] Priscila Biller, Carole Knibbe, Guillaume Beslon, and Eric Tannier. Comparative genomics on artificial life. In Arnold Beckmann, Laurent Bienvenu, and Nataša Jonoska, editors, *Pursuit of the Universal*, pages 35–44, Cham, 2016. Springer International Publishing.

[22] G. Fertin, G. Jean, and E. Tannier. Algorithms for computing the double cut and join distance on both gene order and intergenic sizes. *Algorithms Mol Biol*, 12 16, Jun 5 2017.

[23] Andre Rodrigues Oliveira, Geraldine 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, November 2021.

[24] Gabriel Siqueira, Alexsandro Oliveira Alexandrino, and Zanoni Dias. Signed rearrangement distances considering repeated genes, intergenic regions, and indels. *Journal of Combinatorial Optimization*, 46(2):16, Sep 2023.

[25] Klairton Lima Brito, Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Ulisses Dias, and Zanoni Dias. Genome rearrangement distance with a flexible intergenic regions aspect. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(3):1641–1653, 2023.

[26] Alexsandro Oliveira Alexandrino, Andre Rodrigues Oliveira, Géraldine Jean, Guillaume Fertin, Ulisses Dias, and Zanoni Dias. Reversal and transposition distance on unbalanced genomes using intergenic information. *Journal of Computational Biology*, 30(8):861–876, August 2023.

[27] Gabriel Siqueira, Alexsandro Alexandrino, Andre Oliveira, and Zanoni Dias. Heuristics based on adjacency graph packing for DCJ distance considering intergenic regions. In *Anais do XVII Simpósio Brasileiro de Bioinformática*, pages 71–82, Porto Alegre, RS, Brasil, 2024. SBC.

[28] Leonard Bohnenkämper, Marília D. V. Braga, Daniel Doerr, and Jens Stoye. Computing the rearrangement distance of natural genomes. In Russell Schwartz, editor, *Research in Computational Molecular Biology*, pages 3–18, Cham, 2020. Springer International Publishing.

[29] Diego P. Rubert. *Distance and Similarity Measures in Comparative Genomics*. Ph.d. thesis, Universidade Federal de Mato Grosso do Sul, Campo Grande, Brasil, December 2019.

[30] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979.

[31] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Algoritmos - Teoria e Prática*. LTC, 4. edition, 2024.

[32] Sachin Desale, Akhtar Rasool, Sushil Andhale, and Priti Rane. Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey. *International Journal of Computer Engineering in Research Trends*, 2(5):296–304, May 2015.

[33] Nivio Ziviani. *Projeto de Algoritmos com Implementações em Pascal e C*. Cengage Learning, 3. edition, 2018.

[34] Adrián A Davín, Théo Tricou, Eric Tannier, Damien M de Vienne, and Gergely J Szöllősi. Zombi: a phylogenetic simulator of trees, genomes and sequences that accounts for dead linages. *Bioinformatics*, 36(4):1286–1288, 09 2019.