Approximations for the Weighted Reversal, Transposition, and Indel Distance Problem with Intergenic Region Information

Gabriel Siqueira[1], **Alexsandro Oliveira Alexandrino**[1], **Zanoni Dias**[1]

[1]Instituto de Computação, Universidade Estadual de Campinas (Unicamp), Campinas, Brazil

{gabriel.siqueira,alexsandro,zanoni}@ic.unicamp.br

Abstract. *Genome rearrangement distances are an established method in genome comparison. Works in this area may include various rearrangement operations representing large-scale mutations, gene orientation information, the number of nucleotides in intergenic regions, and weights reflecting the expected frequency of each operation. In this article, we model genomes containing at most one copy of each gene by considering gene sequences, with orientations, and representing intergenic regions according to their nucleotide lengths. We looked at a problem called Weighted Reversal, Transposition, and Indel Distance, which seeks the minimal cost sequence composed by the rearrangement operations of reversals, transposition, and indels, capable of transforming one genome into another. We leverage a structure called Labeled Intergenic Breakpoint Graph to show an algorithm for that problem with guaranteed approximations considering some sets of weights for the operations.*

1. Introduction

In many scenarios of Computational Biology it is important to have some measure for the difference between two genomes considering some biological basis. If we are using genetic mutations, one such measure is the Genome Rearrangement distance, which seeks the shortest sequence of genome rearrangements (mutations that affect a large sequence of genetic material) capable of transforming one genome into the other.

Two common rearrangements that are normally adopted in rearrangement problems are reversals, which invert a sequence of genetic material, and transpositions, which exchange two consecutive sequences of genetic material. Those two operations are called conservative, because they only change the order and orientation of sequences of the genetic material. There are also non-conservative operations, such as the indel, which inserts or removes a sequence of genetic material.

In this work, we assume that each gene can have at most one copy. With these restrictions and considering only the sequence of genes, each with an associated orientation, we have the following known results:

- If we are only considering reversals, finding the rearrangement distance can be done in polynomial time [Hannenhalli and Pevzner 1999].
- If we are considering both operations of reversals and transpositions, finding the rearrangement distance is NP-hard [Oliveira et al. 2019a] and the best known approximation is 2 [Walter et al. 1998].

- If we are considering both operations of reversals and indels, finding the rearrangement distance can be done in polynomial time [Willing et al. 2021].
- If we are considering the operations of reversals, transpositions, and indels, finding the rearrangement distance is NP-hard and the best known approximation is 2 [Alexandrino et al. 2022, Alexandrino et al. 2023c].

Beyond using the sequence of genes, some works in genome rearrangements also consider information presented in intergenic regions (normally represented by the number of nucleotides contained therein). The following results consider this additional information:

- If we are only considering reversals, finding the rearrangement distance is NP-hard and the best known approximation is 2 [Oliveira et al. 2021a].
- If we are considering both operations of reversals and transpositions, finding the rearrangement distance is NP-hard and the best known approximation is 3 [Oliveira et al. 2021b].
- If we are considering both operations of reversals and indels, finding the rearrangement distance has unknown complexity and the best known approximation is 2.5 [Alexandrino et al. 2023a].
- If we are considering the operations of reversals, transpositions, and indels, finding the rearrangement distance is NP-hard and the best known approximation is 4 [Alexandrino et al. 2023b].

Another important variant of the genome rearrangement problems considers distinct weights for each operation. This is important to overcome a tendency the algorithms have to prefer transpositions, which may be a problem in populations where another rearrangement is more likely to occur [Blanchette et al. 1996]. For instance, if reversals have a weight of 2 and transpositions have a weight of 3 there is an 1.5-approximation algorithm for the rearrangement problem that does not consider intergenic regions [Oliveira et al. 2019b] (if the ratio between the cost of reversals and transpositions is between 1 and 3 the approximation is lower or equal to 2 in this algorithm). In 2020, there was a study of weighted operations considering intergenic regions [Brito et al. 2020] with the description of approximations for different weight functions considering reversal, transpositions and indels, but the indels were only allowed to affect intergenic regions.

In this work, we take another step in the study of genome rearrangement problems with weighted operations by considering weights for the operations of reversal, transposition, and indel, such that the indels may also affect genes. For that goal, we apply some known results from the literature and prove the necessary facts to develop an algorithm and ensure its approximation factor.

In Section 2, we provide some necessary definitions and a formal description of the problem. Afterward, in Section 3, we present the Labeled Intergenic Breakpoint Graph. Next, in Section 4 we describe an approximation algorithm for the rearrangement problem considering intergenic regions and different weights for the operations of reversal, transposition, and indel. Finally, Section 5 concludes the paper.

2. Definitions

A genome $\mathcal{G} = (S, \breve{S})$ is represented by a string S, with size n, and a list of integers $\breve{S}$, with size $n + 1$. For $1 \leq i \leq n$, the element S_i, in the i-th position of S, represents the

i-th gene of $\mathcal{S}$ and the i-th integer $\breve{S}_i$ of $\breve{S}$ represents the number of nucleotides (size of the intergenic region) between S_{i-1} and S_i, or before S_1 if $i = 0$. There is also an integer $\breve{S}_{n+1}$ representing the size of the intergenic region at the end of the genome. The alphabet Σ_S denotes the set of characters from S. We also encode gene orientation by assigning a $+$ or $-$ sign to each element of S.

Given a genome $\mathcal{G} = (S, \breve{S})$, the *reversal* $\rho^{(i,j)}_{(x,y)}$, with $1 \leq i \leq j \leq n$, $0 \leq x \leq \breve{S}_i$, and $0 \leq y \leq \breve{S}_{j+1}$ is an operation that inverts a segment of $\mathcal{G}$. This segment starts in the x-th nucleotide of the intergenic region $\breve{S}_i$ and ends in the y-th nucleotide of the intergenic region $\breve{S}_{j+1}$. The order and orientation of the genes and the order of the intergenic regions on this segment are inverted, resulting in a new genome $\mathcal{G}_1 \cdot \rho^{(i,j)}_{(x,y)} = (S', \breve{S}')$. Next, we show the application of this operation:

$$
\begin{aligned}
S &= (S_1\ S_2\ \ldots\ \underline{S_i\ S_{i+1}\ \ldots\ S_j}\ \ldots S_{n-1}\ S_n) \\
\breve{S} &= (\breve{S}_1\ \breve{S}_2\ \ldots\ \underline{(x{+}x')\ \breve{S}_{i+1}\ \breve{S}_{i+2}\ \ldots\ \breve{S}_j\ (y{+}y')}\ \ldots \breve{S}_n\ \breve{S}_{n+1}) \\
S' &= (S_1\ S_2\ \ldots\ \underline{-S_j\ \ldots\ -S_{i+1}\ -S_i}\ \ldots S_{n-1}\ S_n) \\
\breve{S} &= (\breve{S}_1\ \breve{S}_2\ \ldots\ \underline{(x{+}y)\ \breve{S}_j\ \ldots\ \breve{S}_{i+2}\ \breve{S}_{i+1}\ (x'{+}y')}\ \ldots \breve{S}_n\ \breve{S}_{n+1}) \\
x' &= \breve{S}_i - x,\ \ y' = \breve{S}_{j+1} - y
\end{aligned}
$$

Given a genome $\mathcal{G} = (S, \breve{S})$, the *transposition* $\tau^{(i,j,k)}_{(x,y,z)}$, with $1 \leq i < j < k \leq n{+}1$, $0 \leq x \leq \breve{S}_i$, $0 \leq y \leq \breve{S}_j$, and $0 \leq z \leq \breve{S}_k$, is an operation that exchanges two segments of $\mathcal{G}$. The first segment starts in the x-th nucleotide of the intergenic region $\breve{S}_i$ and ends in the y-th nucleotide of the intergenic region $\breve{S}_j$. The second segment starts in the y-th nucleotide of the intergenic region $\breve{S}_j$ and ends in the z-th nucleotide of the intergenic region $\breve{S}_k$. The genes and intergenic regions of these segments are swapped, resulting in a new genome $\mathcal{G} \cdot \tau^{(i,j,k)}_{(x,y,z)} = (S', \breve{S}')$. Next, we show the application of this operation:

$$
\begin{aligned}
S &= (S_1\ \ldots\ \underline{S_i\ \ldots\ S_{j-1}}\ \underline{S_j\ \ldots\ S_{k-1}}\ \ldots\ S_n) \\
\breve{S} &= (\breve{S}_1\ \ldots\ \underline{(x{+}x')\ \breve{S}_{i+1}\ \ldots\ \breve{S}_{j-1}\ (y}\ \underline{{+}y')\breve{S}_{j+1}\ \ldots\ \breve{S}_{k-1}\ (z}\ {+}\ z')\ \ldots\ \breve{S}_{n+1}) \\
S' &= (S_1\ \ldots\ \underline{S_j\ \ldots\ S_{k-1}}\ \underline{S_i\ \ldots\ S_{j-1}}\ \ldots\ S_n) \\
\breve{S}' &= (\breve{S}_1\ \ldots\ \underline{(x{+}y')\breve{S}_{j+1}\ \ldots\ \breve{S}_{k-1}\ (z}\ \underline{{+}x')\ \breve{S}_{i+1}\ \ldots\ \breve{S}_{j-1}\ (y}\ +\ z')\ \ldots\ \breve{S}_{n+1}) \\
x' &= \breve{S}_i - x,\ \ y' = \breve{S}_j - y,\ \ z' = \breve{S}_k - z
\end{aligned}
$$

Given a genome $\mathcal{G} = (S, \breve{S})$, a sequence of genes (characters) A, with size $|A|$, and a sequence of intergenic regions (integers) $\breve{A}$, with size $|A| + 1$, the *insertion* $\phi^{(i,A,\breve{A})}_{(x)}$, with $1 \leq i \leq n$ and $0 \leq x \leq \breve{S}_i$, is an operation that inserts the sequences A and $\breve{A}$ in $\mathcal{G}$. The insertion occurs after the x-th nucleotide of the intergenic region $\breve{S}_i$. After the insertion, we have a new genome $\mathcal{G} \cdot \phi^{(i,A,\breve{A})}_{(x)}$. Note that, if A is empty, the insertion only adds nucleotides in the intergenic region $\breve{S}_i$. Next, we show the application of this

operation:

$$
\begin{aligned}
S &= (S_1\ S_2\ \ldots\ S_i\ S_{i+1}\ \ldots S_{n-1}\ S_n)\\
\breve{S} &= (\breve{S}_1\ \breve{S}_2\ \ldots\ \breve{S}_{i-1}\ (x+x')\ \breve{S}_{i+1}\ \ldots \breve{S}_n\ \breve{S}_{n+1})\\
S' &= (S_1\ S_2\ \ldots\ S_i\ \underline{A_1\ A_2\ \ldots\ A_{|A|}}\ S_{i+1}\ \ldots S_{n-1}\ S_n)\\
\breve{S}' &= (\breve{S}_1\ \breve{S}_2\ \ldots\ \breve{S}_{i-1}\ (x+\underline{\breve{A}_1)\ \breve{A}_2\ \ldots\ (\breve{A}_{|A|+1}+x')}\ \breve{S}_{i+1}\ \ldots \breve{S}_n\ \breve{S}_{n+1})\\
x' &= \breve{S}_i - x
\end{aligned}
$$

Given a genome $\mathcal{G} = (S, \breve{S})$, the *deletion* $\psi_{(x,y)}^{(i,j)}$, with $1 \leq i \leq j \leq n+1$, $0 \leq x \leq \breve{S}_i$, and $0 \leq y \leq \breve{S}_j$, is an operation that removes a segment of $\mathcal{G}$. This segment starts in the x-th nucleotide of the intergenic region $\breve{S}_i$ and ends in the y-th nucleotide of the intergenic region $\breve{S}_j$. After the deletion, we have a new genome $\mathcal{G} \cdot \psi_{(x,y)}^{(i,j)}$. Note that, if $i = j$, the deletion only removes nucleotides in the intergenic region $\breve{S}_i$. In that case, we require that $0 \leq x \leq y \leq \breve{S}_j$. Next, we show the application of this operation:

$$
\begin{aligned}
S &= (S_1\ S_2\ \ldots\ S_{i-1}\ \underline{S_i\ S_{i+1}\ \ldots\ S_{j-1}}\ S_j\ \ldots S_{n-1}\ S_n)\\
\breve{S} &= (\breve{S}_1\ \breve{S}_2\ \ldots\ \breve{S}_{i-1}(x+\underline{x')\ \breve{S}_{i+1}\ \breve{S}_{i+2}\ \ldots\ \breve{S}_{j-1}\ (y+}y')\ \breve{S}_{j+1}\ \ldots \breve{S}_n\ \breve{S}_{n+1})\\
S &= (S_1\ S_2\ \ldots\ S_{i-1}\ S_j\ \ldots S_{n-1}\ S_n)\\
\breve{S} &= (\breve{S}_1\ \breve{S}_2\ \ldots\ \breve{S}_{i-1}(x+y')\ \breve{S}_{j+1}\ \ldots \breve{S}_n\ \breve{S}_{n+1})\\
x' &= \breve{S}_i - x,\ \ y' = \breve{S}_j - y
\end{aligned}
$$

When we want to discuss an operation that can be either an insertion or a deletion, we use the term indel. Given a genome $\mathcal{G}$ and a sequence of operations $R = (\beta_1, \beta_2, \ldots, \beta_{|R|})$, where each β_i can be a reversal, a transposition, or an indel, we denote by $\mathcal{G} \cdot R$ the genome $\mathcal{G} \cdot \beta_1 \cdot \beta_2 \cdot \ldots \cdot \beta_{|R|}$, which is the result of applying in order each operation of R to $\mathcal{G}$.

We are interested in comparing two genomes $\mathcal{G}_1$ and $\mathcal{G}_2$ that may have a distinct set of genes. So we need to insert or delete characters from the genomes. If we are allowed to insert or remove any segment of the genome then the problem becomes trivial, just remove the whole genome $\mathcal{G}_1$ and insert the genome $\mathcal{G}_2$. Consequently, we assume that only genes that appear exclusively in $\mathcal{G}_1$ can be deleted and only genes that appear exclusively in $\mathcal{G}_2$ can be inserted. However, we do not restrict which intergenic regions can be inserted or deleted.

Considering that each gene can have at most one copy, we use the following representation for a pair of genomes $\mathcal{G}_1$ and $\mathcal{G}_2$. The genome $\mathcal{G}_2$, with m genes, is represented by the identity string $\iota = (+1\ +2\ \ldots\ +m)$, with all elements positive, and by the integer list $\breve{\iota}$. The genome $\mathcal{G}_1$, with n genes, is represented by the string S and integer list $\breve{S}$, such that each character of S that corresponds to a gene present in $\mathcal{G}_2$ receives the same number used in ι, and the other characters are represented by the symbol α, because they need to be deleted and therefore do not need to be distinguished from one another. As we can delete or insert multiple consecutive characters and intergenic regions, we can combine any consecutive sequence of characters that appear only in ι or only in S (the intergenic

regions between those characters are removed from $\breve{\iota}$ and $\breve{S}$). Consequently, we assume that there are no two consecutive characters in ι that appear only in ι, and there are no two consecutive characters α in S.

We want to attribute a weight to each operation (reversal, transposition, or indel). We denote the weight of a rearrangement β by $W(\beta)$, and we denote the sum of the weights of all rearrangements of a sequence R by $W(R)$. In the Intergenic Weighted Reversal, Transposition, and Indel Distance (IWRTID) problem, we are interested in a sequence of operations R that turns $\mathcal{G}_1$ into $\mathcal{G}_2$ such that the value of $W(R)$ is minimized. This problem is NP-hard even if all operations have the same weight [Alexandrino et al. 2023b].

3. Labeled Intergenic Breakpoint Graph

To develop the new algorithms, we will use the Labeled Intergenic Breakpoint Graph presented in a previous study [Alexandrino et al. 2021]. To define this graph, we include two characters $S_0 = \iota_0 = 0$ and $S_{n+1} = \iota_{m+1} = m + 1$ at the beginning and the end of the genomes $\mathcal{G}_1$ and $\mathcal{G}_2$. Let π^S be the string S without the characters α. We denote by Σ_{π^S} the set of characters in π^S and by Σ_ι the set of characters in ι. We also defined $next(x, \Sigma_{\pi^S} \cap \Sigma_\iota) = min(y \in \Sigma_{\pi^S} \cap \Sigma_\iota | y > x)$, for $x \in \Sigma_{\pi^S} \setminus \{m + 1\}$.

Given two genomes $\mathcal{G}_1 = (S, \breve{S})$ and $\mathcal{G}_2 = (\iota, \breve{\iota})$, the *Labeled Intergenic Breakpoint Graph* $G(\mathcal{G}_1, \mathcal{G}_2) = (V, E, w, \ell)$ is a graph with the vertex set V, the edge set E, the weight function $w : E \to \mathbb{N}$, and the label function $\ell : E \to (\Sigma_\iota \setminus \Sigma_{\pi^S}) \cup \{\alpha\}$. The set V has a vertex $+\pi^S_0$, a vertex $-\pi^S_{n+1}$, and, for each character $\pi^S_i, 1 \leq i \leq n$, two vertices $-\pi^S_i$ and $+\pi^S_i$. The edge set E is separated into the set E^o of origin edges and the set E^t of target edges. For each $1 \leq i \leq n + 1$, there is an origin edge $o_i = (+\pi^S_{i-1}, -\pi^S_i)$ with weight $w(o_i)$ equal to the sum of intergenic regions in $\breve{S}$ between the elements π^S_{i-1} and π^S_i. The label $\ell(o_i)$ is equal to α, if there is one α between π^S_{i-1} and π^S_i, or is empty otherwise. For every $x \in \Sigma_{\pi^S} \setminus \{m + 1\}$, there is a target edge $t_x = (+x, -next(x, \Sigma_{\pi^S} \cap \Sigma_\iota))$ with weight $w(t_x)$ equal to the sum of the intergenic regions in $\breve{\iota}$ between x and $next(x, \Sigma_{\pi^S} \cap \Sigma_\iota)$. The label $\ell(t_x)$ is equal to $x + 1$, if $x + 1 \neq next(x, \Sigma_{\pi^S} \cap \Sigma_\iota)$, or is empty otherwise.

Each vertex from V has only one incident target edge and one incident origin edge. Consequently, there is a unique decomposition of $G(\mathcal{G}_1, \mathcal{G}_2)$ into alternating cycles (cycles formed by interleave target and origin edges). One such cycle is called *trivial* if it has only one target and one origin edge, and it is called *non-trivial* otherwise. If some edge of a cycle has a label the cycle is called *labeled*, and it is called *clean* otherwise. A cycle C is *balanced* if the sum of the weights of the origin edges is equal to the sum of the weights of the target edges, and it is *unbalanced* otherwise. An unbalanced cycle is *positive* if the sum of the weights of the origin edges is lower than the sum of the weights of the target edges, and it is *negative* otherwise. If a cycle is balanced and clean it is called *good*, and it is called *bad* otherwise (if it is unbalanced or labeled).

We use the following standard drawing of the graph $G(\mathcal{G}_1, \mathcal{G}_2)$, which will help in the following definitions. We place all vertices aligned in the order $+\pi^S_0$, $-\pi^S_1$, $+\pi^S_1$, ..., $-\pi^S_n$, $+\pi^S_n$, $-\pi^S_{n+1}$. The origin edges are shown as horizontal lines. Note that these edges always connect two neighbor vertices, considering the order in which they are drawn. The target edges are shown as arcs above the vertices. Figure 1 shows an example of

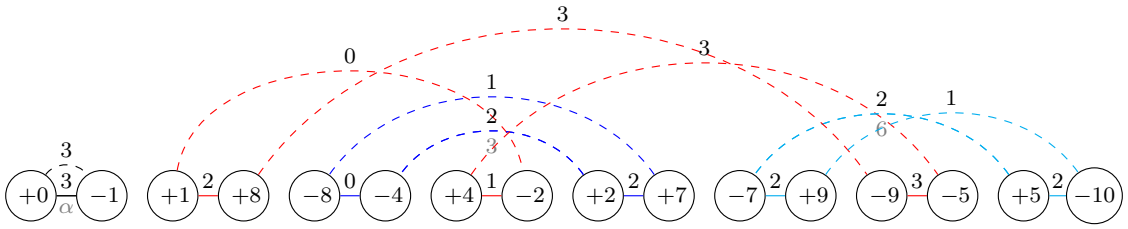

Figure 1. Standard drawing of the graph $G(\mathcal{G}_1, \mathcal{G}_2)$ created from two genomes $\mathcal{G}_1 = ((+0\ +\alpha\ +1\ -8\ +4\ +2\ -7\ -9\ +5\ +10), (1\ 2\ 2\ 0\ 1\ 2\ 2\ 3\ 2))$ and $\mathcal{G}_2 = ((+0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9\ +10), (3\ 0\ 1\ 1\ 3\ 0\ 2\ 1\ 3\ 1))$. Each color represents a different cycle of the cycle decomposition of $G(\mathcal{G}_1, \mathcal{G}_2)$. The weights (representing the intergenic regions) are indicated above each edge, and the labels (encoding the genes exclusive to one of the genomes) are indicated below each edge.

the standard drawing of a breakpoint graph. In this graph, the black cycle is trivial, bad, labeled, and balanced, the red cycle is good, the blue cycle is bad, labeled, and positive, and the cyan cycle is bad, labeled, and negative.

Considering the standard drawing, when traversing a cycle, starting from the rightmost vertices being traversed from right to left starting always using an origin edge, we may traverse an origin edge from right to left or from left to right. If all origin edges of cycles are traversed from right to left, we call this cycle *convergent*, and we call it *divergent* otherwise. Given a non-trivial convergent cycle C. If the edges are traversed in order from right to left, then the cycles C is called *non-oriented*, and it is called *oriented* otherwise. In Figure 1, the cyan cycle is divergent, the red cycle is convergent and oriented, and the blue cycle is convergent and non-oriented.

We will use $c(\mathcal{G}_1, \mathcal{G}_2)$ and $c_g(\mathcal{G}_1, \mathcal{G}_2)$ to denote the number of cycles and the number of good cycles in $G(\mathcal{G}_1, \mathcal{G}_2)$, respectively. For a rearrangement sequence R, let $\pi^S \cdot \beta$ be the resulting π^S after the application of the sequence R in $\mathcal{G}_1 = (S, \breve{S})$. We use $\Delta c(\mathcal{G}_1, \mathcal{G}_2, \beta) = (|\pi^S| + 1 - c(\mathcal{G}_1, \mathcal{G}_2)) - (|\pi^S \cdot R| + 1 - c(\mathcal{G}_1 \cdot R, \mathcal{G}_2))$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, \beta) = (|\pi^S| + 1 - c_g(\mathcal{G}_1, \mathcal{G}_2)) - (|\pi^S \cdot R| + 1 - c_g(\mathcal{G}_1 \cdot R, \mathcal{G}_2))$ to describe the effect of this rearrangement on the number of cycles and vertices in the graph.

Note that any distinction in the order of the genes of $\mathcal{G}_1$ and $\mathcal{G}_2$ will produce non-trivial cycles in $G(\mathcal{G}_1, \mathcal{G}_2)$, any difference in the size of the intergenic regions in $\mathcal{G}_1$ and $\mathcal{G}_2$ will produce unbalanced cycles in $G(\mathcal{G}_1, \mathcal{G}_2)$, and any difference in the gene content of $\mathcal{G}_1$ and $\mathcal{G}_2$ will produce labeled cycles in $G(\mathcal{G}_1, \mathcal{G}_2)$. This leads us to the following remark.

Remark 1. *$G(\mathcal{G}_1, \mathcal{G}_2)$ has only trivial good cycles if and only if $\mathcal{G}_1 = \mathcal{G}_2$.*

Note that any operation of reversal and transposition applied to $\mathcal{G}_1$ can be seen as an operation that affects the graph $G(\mathcal{G}_1, \mathcal{G}_2)$, changing the order of the vertices and replacing some origin edges according to the new order. The weights and labels of the new edges will be attributed according to the parameters of the operation and weights and labels of the edges that have been removed. Similarly, a deletion or insertion can be seen as changes to the weights and labels of the origin edges, with the possible inclusion of some vertices in the case of insertion. Note that an insertion that only affects intergenic regions corresponds to adding weight to one of the origin edges. Considering the convenience of this type of operation, we also define the operation of adding weight to

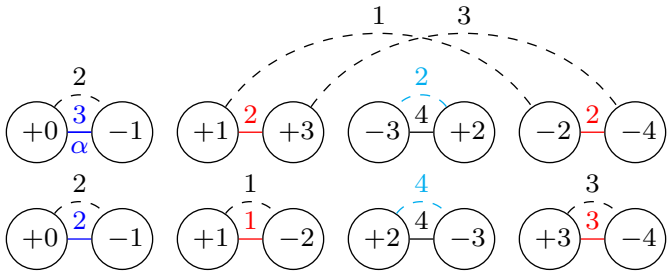

Figure 2. Given $\mathcal{G}_1 = ((+0\ +\alpha\ +1\ -3\ -2\ +4), (1\ 2\ 2\ 4\ 2))$ and $\mathcal{G}_2 = ((+0\ +1\ +2\ +3\ +4), (2\ 1\ 2\ 3))$. A sequence of rearrangement operations being applied to the graph $G(\mathcal{G}_1, \mathcal{G}_2)$, and the correspondent sequence applied to the genome $\mathcal{G}_1$ producing the genome $\mathcal{G}_2$. The sequence is composed of: a virtual insertion in the cyan edge correspondent to the deletion $\psi^{(4,4)}_{(0,2)}$; a reversal affecting the red edges correspondent to the reversal $\rho^{(4,5)}_{(1,0)}$; a deletion in the blue edge correspondent to the deletion $\psi^{(1,2)}_{(0,0)}$.

one of the target edges with a clean label. This is a virtual insertion because there is no rearrangement operation directly correspondent to it, but the following lemma allows us to use it. Figure 2 shows an example of the use of this lemma.

Lemma 1. *If a sequence R of rearrangement operations (including virtual insertions in target edges with clean label) can transform $G(\mathcal{G}_1, \mathcal{G}_2)$ into a graph with only good cycles, then there is a sequence R' with at most as many rearrangements as R that transforms $\mathcal{G}_1$ into $\mathcal{G}_2$.*

Proof. After the application of the operations from the sequence R in $G(\mathcal{G}_1, \mathcal{G}_2)$ we have a new graph $G(\mathcal{G}'_1, \mathcal{G}'_2)$, such that $\mathcal{G}'_1 = \mathcal{G}'_2$ (Remark 1). Note that $\mathcal{G}'_1$ is the result of applying all non-virtual operations in $\mathcal{G}_1$, because the virtual insertion only affects target edges. Besides, the virtual insertion is the only operation that affects target edges with clean labels. Consequently, the only distinction between $\mathcal{G}'_1$ and $\mathcal{G}_2$ is the size of some intergenic regions. We just have to apply deletions in each of these intergenic regions to turn $\mathcal{G}'_1$ into $\mathcal{G}_2$. We can take R' to be the sequence of non-virtual operations of R plus these deletions added to the end. Note that R' has at most as many operations as R, because the number of distinct intergenic regions between $\mathcal{G}'_1$ and $\mathcal{G}_2$ is at most the number of virtual insertions in R. □

4. Approximation Algorithms

In this section, we present the approximation algorithm for the IWRTID problem. First, we establish how the operations can affect the values of Δc and Δc_g.

Lemma 2. *[Bafna and Pevzner 1996, Theorem 1] For any reversal ρ, we have $\Delta c(\mathcal{G}_1, \mathcal{G}_2, \rho) \leq 1$.*

Lemma 3. *[Alexandrino et al. 2021, Lemma 2] For any reversal ρ, we have $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, \rho) \leq 1$.*

Lemma 4. *For any indel δ, we have $\Delta c(\mathcal{G}_1, \mathcal{G}_2, \delta) \leq 0$.*

Proof. We can only delete characters α, so deletions cannot have $\Delta c(\mathcal{G}_1, \mathcal{G}_2, \delta) \neq 0$. Besides, for each cycle inserted in $G(\mathcal{G}_1, \mathcal{G}_2)$ we must also insert at least one gene. Consequently, any change in $c(\mathcal{G}_1, \mathcal{G}_2))$ will be offset by a change in $|\pi^S|$, so insertions cannot have $\Delta c(\mathcal{G}_1, \mathcal{G}_2, \delta) > 0$. □

Lemma 5. *[Alexandrino et al. 2021, Lemma 3] For any indel δ, we have $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, \delta) \leq 1$.*

Lemma 6. *[Bafna and Pevzner 1998, Lemma 2.1] For any transposition τ, we have $\Delta c(\mathcal{G}_1, \mathcal{G}_2, \tau) \leq 2$.*

Lemma 7. *[Alexandrino et al. 2023b, Lemma 2] For any transposition τ, we have $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, \tau) \leq 2$.*

In the next lemmas, we describe operations that can produce a positive value for Δc or Δc_g depending on the characteristics of a given cycle C of the graph $G(\mathcal{G}_1, \mathcal{G}_2)$.

Lemma 8. *[Alexandrino et al. 2021, Lemma 5] If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ a trivial cycle C with an empty origin edge, such that C is unbalanced and the desire edge is empty or C is non-negative and the target edge has a non-empty label, then there exists an indel δ, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, (\delta)) = 0$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, (\delta)) = 1$.*

Lemma 9. *[Alexandrino et al. 2021, Lemma 6] If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ a trivial bad cycle C, then there exists a sequence R with up to 2 indels, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 0$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 1$.*

Lemma 10. *If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ an unbalanced clean cycle C, then there exists an indel δ, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, (\delta)) = 0$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, (\delta)) = 1$.*

Proof. If C is positive, we apply an insertion that increases the weight of an origin edge of C to make C balanced. If C is negative, we apply a virtual insertion (that will be turned into a deletion in the end) that increases the weight of a target edge of C to turn C balanced. □

Lemma 11. *[Oliveira et al. 2021b, Lemma 4.6] If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ an good oriented cycle C, then there exists a sequence R with 3 transpositions, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 2$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 2$.*

Lemma 12. *If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ an good oriented cycle C, then there exists a sequence R with 1 transposition and up to 2 indels, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 2$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 2$.*

Proof. By Lemma 4.3 from a previous work [Oliveira et al. 2021b] we can apply one transposition that splits C into three cycles such that one of them is balanced, and therefore good, because C is clean. We can balance the other two cycles with one indel each, as stated in Lemma 10. □

Lemma 13. *If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ a bad oriented cycle C, then there exists a sequence R with uma transposition and up to um indel, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 2$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 1$.*

Proof. By Lemma 12 from a previous work [Alexandrino et al. 2023b], there is a transposition that turns C into three cycles such that one of them is a trivial non-negative cycle with a clean source edge. If necessary, we can apply Lemma 8 to turn this cycle good with one indel. □

Lemma 14. *[Alexandrino et al. 2021, Lemma 7] If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ a labeled divergent cycle C, then there exists a sequence R with 1 reversal and up to 1 indel, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 1$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 1$.*

Lemma 15. *[Alexandrino et al. 2021, Lemma 8] If there is in $G(\mathcal{G}_1, \mathcal{G}_2)$ a good divergent cycle C, then there exists one reversal, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 1$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 1$.*

Lemma 16. *If there is no divergent cycles in $G(\mathcal{G}_1, \mathcal{G}_2)$ and there are still nontrivial or bad cycles, then there exists a sequence R with 3 reversals and up to 2 indels, such that $\Delta c(\mathcal{G}_1, \mathcal{G}_2, R) = 1$ and $\Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) = 2$.*

Proof. If C is oriented, the result is direct from Lemma 9 from a previous work [Alexandrino et al. 2023a]. If C is non-oriented, the result is direct from Lemma 10 from the same work [Alexandrino et al. 2023a]. □

Algorithm 1 The input is a pair of genomes $\mathcal{G}_1$ and $\mathcal{G}_2$. The output is a sequence S'' of reversal, transposition and indel operations such that $\mathcal{G}_1 \cdot S'' = \mathcal{G}_2$.

$G \leftarrow G(\mathcal{G}_1, \mathcal{G}_2)$
$S \leftarrow [\,]$
while G has a cycle that is not good and trivial. **do**
  if there is a trivial bad cycle in G. **then**
    $S' \leftarrow [\delta]$ or $[\delta_1, \delta_2]$ ▷ Step I: lemmas 8 or 9.
  else if there is an unbalanced clean cycle in G. **then**
    $S' \leftarrow [\delta]$ ▷ Step II: Lemma 10
  else if there is a good oriented cycle in G. **then**
    $S' \leftarrow [\tau_1, \tau_2, \tau_3]$ or $[\tau, \delta_1, \delta_2]$ ▷ Step III: lemmas 11 and 12
  else if there is a bad oriented cycle in G. **then**
    $S' \leftarrow [\tau, \delta]$ ▷ Step IV: Lemma 13.
  else if there is a labeled divergent cycle in G **then**
    $S' \leftarrow [\rho, \delta]$ ▷ Step V: Lemma 14.
  else if there is a good divergent cycle in G **then**
    $S' \leftarrow [\rho]$ ▷ Step VI: Lemma 15.
  else
    $S' \leftarrow [\rho_1, \rho_2, \rho_3, \delta_1, \delta_2]$ ▷ Step VII: Lemma 16.
  end if
  $G \leftarrow G \cdot S'$
  $S \leftarrow S + S'$
end while
$S'' \leftarrow S$ with the virtual insertions replaced by deletions at the end of the sequence.
return S''

Let us consider the weight of an operation as a cost to apply it. To prove the approximation of Algorithm 1, we use the following value to describe the variation in Δc_g and Δc per weight of the operations applied.

$$\Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{p_1 \Delta c_g(\mathcal{G}_1, \mathcal{G}_2, R) + p_2 \Delta c(\mathcal{G}_1, \mathcal{G}_2, R)}{W(R)}$$

In this equation for Δcc_g, the parameters p_1 and p_2 control the contribution of Δc_g and Δc and are useful to adjust this equation depending on the weights we are considering for the operations.

In the following theorem, we show approximations for some choices of weights. The weights are motivated by the fact that transpositions cut the genome in more points, which makes them preferred by most algorithms. To investigate how weight variations affect approximation performance, we also tested scenarios with equal weights for reversals and indels, as well as cases where reversals have higher weights compared to indels.

Theorem 17. *Algorithm 1 ensures the following approximations for the IWRTID problem:*

- 3.33, *if* $W(\rho) = 2$, $W(\tau) = 3$, $W(\delta) = 2$, $p_1 = 4$, *and* $p_2 = 1$.
- 2.67, *if* $W(\rho) = 2$, $W(\tau) = 3$, $W(\delta) = 1$, $p_1 = 1$, *and* $p_2 = 1$.
- 2.5, *if* $W(\rho) = 1$, $W(\tau) = 2$, $W(\delta) = 1$, $p_1 = 4$, *and* $p_2 = 1$.
- 2, *if* $W(\rho) = 2$, $W(\tau) = 4$, $W(\delta) = 1$, $p_1 = 1$, *and* $p_2 = 1$.

Proof. Considering Remark 1 and the fact that at each iteration the algorithm finds a rearrangement sequence with positive value for Δc or Δc_g, eventually, the algorithm stops producing the desired sequence S'', capable of transforming $\mathcal{G}_1$ into $\mathcal{G}_2$. We must prove that the weight of this sequence is at most the desire approximation factor times the optimal weight. For that goal, it is enough to prove that this fact is valid for each step.

For a reversal ρ, we know that $\Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, (\rho)) \leq \frac{p1+p2}{W(\rho)}$ (lemmas 2 and 3). Similarly, for a transposition τ, $\Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, (\tau)) \leq \frac{2(p1+p2)}{W(\tau)}$ (lemmas 6 and 7), and for an indel δ, $\Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, (\delta)) \leq \frac{p1}{W(\delta)}$ (lemmas 4 and 5). Consider $\Delta_{max} = max\left(\frac{p1+p2}{W(\rho)}, \frac{2(p1+p2)}{W(\tau)}, \frac{p1}{W(\delta)}\right)$, which is the maximum possible value of Δcc_g produce by a single operation. Note that if a sequence of operations reaches this value, then the sequence is optimal.

Let us look at the value of $\Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R)$ for each step of Algorithm 1:

- Step I: $\Delta_I = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{p_1}{2W(\delta)}$
- Step I: $\Delta_I = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{p_1}{W(\delta)}$
- Step III: $\Delta_{II} = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{2p_1+2p_2}{min(3W(\tau), W(\tau)+2W(\delta))}$
- Step IV: $\Delta_{III} = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{p_1+2p_2}{W(\tau)+W(\delta)}$
- Step V: $\Delta_{IV} = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{p_1+p_2}{W(\rho)+W(\delta)}$
- Step VI: $\Delta_{IV} = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{p_1+p_2}{W(\rho)}$
- Step VII: $\Delta_V = \Delta cc_g(p_1, p_2, \mathcal{G}_1, \mathcal{G}_2, R) = \frac{2p_1+2p_2}{3W(\rho)+2W(\delta)}$

For each step K, the applied sequence is within a factor of $\frac{\Delta_{max}}{\Delta_k}$ from the optimal. Consequently, the approximation factor of algorithm 1 is $max(\frac{\Delta_{max}}{\Delta_I}, \frac{\Delta_{max}}{\Delta_{II}}, \frac{\Delta_{max}}{\Delta_{III}}, \frac{\Delta_{max}}{\Delta_{IV}}, \frac{\Delta_{max}}{\Delta_V}, \frac{\Delta_{max}}{\Delta_{VI}}, \frac{\Delta_{max}}{\Delta_{VII}})$. By replacing the values for $W(\rho)$, $W(\tau)$, $W(\delta)$, p_1, and p_2, we reach the informed approximations. □

5. Conclusion

In this work, we investigated the genome rearrangement problem considering intergenic regions represented by their lengths and allowing reversal, transposition, and indel operations. We developed approximation algorithms with factors between 2 and 3.33 for distinct weights attributed to each operation.

Our contribution represents a meaningful step toward developing algorithms for rearrangement distances that consider operation weights. The main novelty of our approach is the integration of intergenic regions and indels in such problems. Nevertheless, as an initial effort, our study remains restricted to single-copy genes and a simplified representation of intergenic regions.

In future works, this problem can be explored with some alternative representation of intergenic regions or without the restriction of a single copy of each gene. It is also relevant to perform tests with real genomes to observe if the weights are helpful in the estimation of the evolutionary distance between them.

Acknowledgment

This work was supported by the São Paulo Research Foundation, FAPESP (grant 2021/13824-8).

References

Alexandrino, A. O., Brito, K. L., Oliveira, A. R., Dias, U., and Dias, Z. (2021). 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.

Alexandrino, A. O., Brito, K. L., Oliveira, A. R., Dias, U., and Dias, Z. (2023a). Reversal and indel distance with intergenic region information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 20(3):1628–1640.

Alexandrino, A. O., Oliveira, A. R., Dias, U., and Dias, Z. (2022). Labeled Cycle Graph for Transposition and Indel Distance. *Journal of Computational Biology*, 29(03):243–256.

Alexandrino, A. O., Oliveira, A. R., Jean, G., Fertin, G., Dias, U., and Dias, Z. (2023b). Reversal and transposition distance on unbalanced genomes using intergenic information. *Journal of Computational Biology*, 30(8):861–876.

Alexandrino, A. O., Siqueira, G., Brito, K. L., Oliveira, A. R., Dias, U., and Dias, Z. (2023c). Block interchange and reversal distance on unbalanced genomes. In *Advances in Bioinformatics and Computational Biology*, pages 1–13, Cham. Springer Nature Switzerland.

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

Bafna, V. and Pevzner, P. A. (1998). Sorting by Transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240.

Blanchette, M., Kunisawa, T., and Sankoff, D. (1996). Parametric Genome Rearrangement. *Gene*, 172(1):GC11–GC17.

Brito, K. L., Jean, G., Fertin, G., Oliveira, A. R., Dias, U., and Dias, Z. (2020). Sorting by Genome Rearrangements on both Gene Order and Intergenic Sizes. *Journal of Computational Biology*, 27(2):156–174.

Hannenhalli, S. and Pevzner, P. A. (1999). Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals. *Journal of the ACM*, 46(1):1–27.

Oliveira, A. R., Brito, K. L., Dias, U., and Dias, Z. (2019a). On the Complexity of Sorting by Reversals and Transpositions Problems. *Journal of Computational Biology*, 26:1223–1229.

Oliveira, A. R., Brito, K. L., Dias, Z., and Dias, U. (2019b). Sorting by Weighted Reversals and Transpositions. *Journal of Computational Biology*, 26:420–431.

Oliveira, A. R., Jean, G., Fertin, G., Brito, K. L., Bulteau, L., Dias, U., and Dias, Z. (2021a). Sorting Signed Permutations by Intergenic Reversals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2870–2876.

Oliveira, A. R., Jean, G., Fertin, G., Brito, K. L., Dias, U., and Dias, Z. (2021b). Sorting Permutations by Intergenic Operations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2080–2093.

Walter, M. E. M. T., Dias, Z., and Meidanis, J. (1998). 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. IEEE Computer Society.

Willing, E., Stoye, J., and Braga, M. (2021). Computing the Inversion-Indel Distance. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(6):2314–2326.