Heurísticas para Problemas de Rearranjo de Genomas com Genes Multiplicados

Gabriel Siqueira[1], **Andre Rodrigues Oliveira**[1] **e Zanoni Dias**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (IC/UNICAMP)
Av. Albert Einstein, 1251 – Cidade Universitária – Campinas – SP – Brasil

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

Abstract. *Genome rearrangement problems seek to estimate the evolutionary distance between genomes of different species using rearrangement events, which are mutations capable of altering the genetic sequence of genomes. Different rearrangement problems may be defined by the genome rearrangement event set and the characteristics of the considered genomes. Our goal was to develop heuristics for solving genome rearrangement problems considering replicated genes and the reversal and transposition events. We developed heuristics based on techniques used to solve combinatorial optimization problems, such as Local Search, Genetic Algorithms, and GRASP. The experimental tests showed that the distances obtained with the proposed heuristics are lower than the distances obtained with algorithms from the literature.*

Resumo. *Um dos objetivos dos problemas de rearranjo de genomas consiste em estimar a distância evolutiva entre genomas de diferentes espécies utilizando eventos de rearranjo, mutações capazes de alterar a sequência genética dos genomas. Diferentes problemas de rearranjo podem ser definidos de acordo com os eventos utilizados e as características dos genomas considerados. O objetivo do trabalho foi o desenvolvimento de heurísticas para resolver problemas de rearranjo de genomas considerando genes multiplicados e os eventos de rearranjo de reversão e transposição. Desenvolvemos heurísticas baseadas em técnicas amplamente adotadas para problemas de otimização combinatória, como Busca Local, Algoritmos Genéticos e GRASP. Nos testes realizados, verificamos que as distâncias obtidas pelas heurísticas propostas são menores que as distâncias obtidas por algoritmos da literatura.*

1. Introdução

A comparação genômica é uma área de estudo no campo da Biologia Computacional, em que o objetivo é a busca por similaridades e parentescos entre espécies utilizando informações genômicas. Parte dessa busca é realizada ao estimar a distância evolutiva entre genomas. Essa estimativa pode ser feita por meio da avaliação do número de mutações que afetaram o genoma de uma espécie ao longo da evolução, o que gera novas espécies com genomas distintos.

Para calcular essa estimativa, consideramos que o genoma consiste em uma sequência de genes, sendo que esses genes podem apresentar repetições, e suas

orientações podem ou não ser conhecidas. Chamamos de *eventos de rearranjo conservativos* as mutações que podem afetar apenas a ordem e orientação dos genes. A *distância de rearranjo* entre dois genomas é definida pelo tamanho da menor sequência de eventos de rearranjo capaz de transformar um genoma em outro.

Existem diferentes formas para representar os genomas, dependendo das características do problema de rearranjo abordado [Fertin et al. 2009]. Quando consideramos que os genomas apresentam repetições de genes, representamos os genomas por strings, onde cada caractere corresponde a um gene ou a um bloco conservado de genes durante o processo evolutivo.

Um *modelo de rearranjo* $\mathcal{M}$ representa o conjunto dos eventos de rearranjo permitidos para transformar um genoma em outro. Alguns modelos frequentemente utilizados em trabalhos de rearranjo de genomas são compostos pelos eventos de reversão [Hannenhalli and Pevzner 1999, Berman et al. 2002] e transposição [Elias and Hartman 2006]. Existem trabalhos considerando um modelo composto por ambos os eventos [Walter et al. 1998, Rahman et al. 2008].

Os trabalhos iniciais sobre rearranjos de genomas envolviam genomas sem repetições de genes. Mesmo nesse contexto restrito, poucos problemas têm algoritmos exatos polinomiais conhecidos [Hannenhalli and Pevzner 1999, Yancopoulos et al. 2005]. De fato, foi provado que alguns desses problemas pertencem à classe de problemas NP-Difícil [Caprara 1999, Bulteau et al. 2012, Oliveira et al. 2019], o que nos fornece um indício de que pode não ser possível encontrar algoritmos eficientes para tais problemas. Apesar dessa dificuldade em encontrar algoritmos exatos e eficientes, vários algoritmos com fatores de aproximação constantes foram propostos [Berman et al. 2002, Elias and Hartman 2006, Walter et al. 1998, Rahman et al. 2008].

Os trabalhos mais recentes passaram a considerar que os genes podem apresentar múltiplas cópias, ou seja, passaram a representar os genomas como strings. Nesse contexto, foi provado que alguns problemas que tinham algoritmos exatos polinomiais conhecidos para o caso sem repetições pertencem à classe de problemas NP-Difícil [Radcliffe et al. 2005, Shao et al. 2015]. A principal abordagem utilizada por algoritmos de aproximação envolve a relação entre os problemas de rearranjo e as variações do problema de Partição de Strings Comuns Miníma [Chen et al. 2005, Kolman and Waleń 2007, Shapira and Storer 2007].

Existem duas variações dos problemas de rearranjo dependendo do conhecimento da orientação dos genes. Quando a orientação dos genes é conhecida, temos um sinal associado a cada caractere das strings, portanto os problemas lidam com *strings com sinais*. Caso contrário, nenhum sinal é utilizado e os problemas lidam com *strings sem sinais*.

A dissertação teve como objetivo desenvolver e avaliar heurísticas para os problemas de rearranjo envolvendo os eventos conservativos de reversão e transposição. Adaptamos metaheurísticas clássicas da literatura com o intuito de obter bons resultados práticos. Consideramos os seguintes problemas de rearranjo envolvendo genomas com genes multiplicados: Distância de Reversão em Strings com Sinais ($D\bar{R}$); Distância de Reversão em Strings sem Sinais (DR); Distância de Transposição em Strings sem Sinais (DT);

Distância de Reversão e Transposição em Strings com Sinais ($D\bar{R}T$); Distância de Reversão e Transposição em Strings sem Sinais (DRT).

As implementações das heurísticas utilizadas e de outros algoritmos tratados na dissertação estão disponíveis em um repositório público[1].

O restante deste documento está estruturado da seguinte forma. Na Seção 2, definimos formalmente os problemas tratados. Na Seção 3, descrevemos brevemente as soluções propostas e experimentos realizados. Por fim, encerramos com algumas considerações finais na Seção 4.

2. Fundamentação Teórica

Dada uma string S com ou sem sinais representando um genoma, denotamos por $|S|$ o número de caracteres de S e por S_i o i-ésimo caractere de S. Além disso, o conjunto de caracteres distintos de S é chamado de *alfabeto* de S, denotado por Σ_S. Para evitar ambiguidade, utilizamos o termo *rótulo* quando falamos dos elementos de Σ_S e o termo caractere quando falamos dos elementos de S. Em strings com sinais, os caracteres incluem os sinais e os rótulos não, ou seja, os caracteres $-\alpha$ e $+\alpha$ são ambos correspondentes ao rótulo α.

Exemplo 1. Algumas das definições anteriores aplicadas em uma string sem sinais S e em uma string com sinais P.

$$
\begin{aligned}
&S = (\ \ E \quad B \quad A \quad C \quad D \quad E \quad E \quad D\ ),\ S_1 = E \text{ e } S_5 = D\\
&P = (+E\ -B\ -A\ +C\ +D\ +E\ +E\ -D\ ),\ P_4 = +C \text{ e } P_8 = -D\\
&\Sigma_S = \Sigma_P = \{A, B, C, D, E\},\ |S| = |P| = 8
\end{aligned}
$$

A *ocorrência* de um rótulo $\alpha \in \Sigma_S$, denotada por $occ(\alpha, S)$, representa o número de cópias do rótulo α na string S. A maior ocorrência de um rótulo em uma string S é denotada por $occ(S) = \max_{\alpha \in \Sigma_S}(occ(\alpha, S))$. O conjunto dos rótulos *multiplicados* de S, denotado por $mult(S)$, é composto pelos rótulos que aparecem mais de uma vez em S, ou seja, $mult(S) = \{\alpha : occ(\alpha, S) > 1, \forall\ \alpha \in \Sigma_S\}$. O conjunto dos rótulos *duplicados* de S, denotado por $dup(S)$, é composto pelos rótulos que aparecem exatamente duas vezes em S, ou seja, $dup(S) = \{\alpha : occ(\alpha, S) = 2, \forall\ \alpha \in \Sigma_S\}$. Usamos $|mult(S)|$ e $|dup(S)|$ para denotar o tamanho desses conjuntos. Quando $|mult(S)| = 0$, dizemos que a string S é uma *permutação*. Nas strings do Exemplo 1, temos $mult(S) = mult(P) = \{D, E\}$ e $dup(S) = dup(P) = \{D\}$.

Duas strings S e P são ditas *balanceadas* se possuem o mesmo alfabeto Σ ($\Sigma = \Sigma_S = \Sigma_P$) e a ocorrência dos caracteres é a mesma em ambas as strings, ou seja, $occ(\alpha, S) = occ(\alpha, P)$, $\forall\ \alpha \in \Sigma$. Quando um modelo $\mathcal{M}$ possui apenas eventos de rearranjo conservativos, precisamos que as strings consideradas sejam balanceadas.

Representamos um evento de rearranjo em um genoma $\mathcal{G}$ por uma operação aplicada na string S que o representa. A aplicação de um evento δ em uma string S é denotada por $S \circ \delta$.

[1] `https://github.com/compbiogroup/Heuristics-for-Genome-Rearrangement-with-Replicated-Genes`

Dada uma string sem sinais S, uma *reversão* $\rho(i,j)$ aplicada em S, onde $1 \leq i < j \leq |S|$, é um evento que inverte a ordem dos elementos do segmento de S entre as posições i e j. No caso de uma string com sinais S, uma *reversão* $\rho(i,j)$ aplicada em S, onde $1 \leq i \leq j \leq |S|$, é um evento que inverte a ordem e orientação dos elementos do segmento de S entre as posições i e j. Uma reversão $\rho(i,j)$ aplicada em uma string $S = (S_1 \dots\ S_{i-1}\underline{S_i \dots\ S_j}S_{j+1} \dots\ S_{|S|})$ resulta na string $S \circ \rho(i,j) = (S_1 \dots\ S_{i-1}\underline{S_j \dots\ S_i}\, S_{j+1} \dots\ S_{|S|})$, se S é uma string sem sinais e na string $S \circ \rho(i,j) = (S_1 \dots\ S_{i-1}\underline{-S_j \dots\ -S_i}\, S_{j+1} \dots\ S_{|S|})$, se S é uma string com sinais.

Dada uma string com ou sem sinais S, uma *transposição* $\tau(i,j,k)$ aplicada em S, onde $1 \leq i < j < k \leq |S|+1$, é um evento que troca o segmento de S entre as posições i e $j-1$ com o segmento de S entre as posições j e $k-1$. Esse evento não altera a ordem e a orientação dos elementos em cada segmento. Uma transposição $\tau(i,j,k)$ aplicado em uma string $S = (S_1 \dots\ S_{i-1}\underline{S_i \dots\ S_{j-1}}\ \underline{S_j \dots\ S_{k-1}}S_k \dots\ S_{|S|})$ com ou sem sinais resulta na string $S \circ \tau(i,j,k) = (S_1 \dots\ S_{i-1}\underline{S_j \dots\ S_{k-1}}\ \underline{S_i \dots\ S_{j-1}}S_k \dots\ S_{|S|})$.

Dado um modelo de rearranjo $\mathcal{M}$, a *distância de rearranjo* entre dois genomas representados pelas strings S e P, denotada por $d_{\mathcal{M}}(S,P)$, é o menor número de operações correspondentes a rearranjos de $\mathcal{M}$ necessárias para transformar S em P. Quando o modelo de rearranjo é composto apenas por reversões ($\mathcal{R}$) ou transposições ($\mathcal{T}$), usamos os termos distância de reversão ($d_{\mathcal{R}}(S,P)$) ou distância de transposição ($d_{\mathcal{T}}(S,P)$), respectivamente. Quando temos ambos os eventos de reversão e de transposição ($\mathcal{RT}$), usamos o termo distância de reversão e transposição ($d_{\mathcal{RT}}(S,P)$).

Existem algoritmos na literatura para estimar distâncias entre permutações com base em modelos compostos pelos eventos de reversão e transposição. Para aproveitarmos esses resultados em strings utilizamos o conceito de mapeamento de uma string em uma permutação.

Dada uma string S, para construir um mapeamento $\mathbf{x}$ de S em uma permutação $S^{\mathbf{x}}$, atribuímos um caractere distinto para cada ocorrência dos rótulos multiplicados de S. Cada um dos novos caracteres é denotado por S_i^p, onde S_i é o caractere original de S e $p \in \{1, \dots, occ(|S_i|, S)\}$ é um índice usado para diferenciar os caracteres de mesmo rótulo. Perceba que, no caso de strings com sinais, o mapeamento de caracteres não afeta os sinais que cada caractere já possui.

Para garantir que $S^{\mathbf{x}}$ seja uma permutação, dois caracteres de mesmo rótulo devem obrigatoriamente receber valores distintos de p (índices sobrescritos). Sendo assim, a lista com os valores de p para os caracteres de um rótulo α pode ser representada por uma permutação dos números de 1 a $occ(\alpha, S)$.

Combinando todas as permutações de valores de p para cada rótulo multiplicado α, podemos representar um mapeamento $\mathbf{x}$ por um vetor de permutações indexado pelos rótulos. O elemento $\mathbf{x}[\alpha]$ desse vetor contém a permutação correspondente à atribuição dos valores de p para os caracteres do rótulo α.

Para representar todas as atribuições possíveis de valores de p para os caracteres de um rótulo α, utilizamos o conjunto $Sym(\alpha, S)$, composto por todas as permutações dos números de 1 a $occ(\alpha, S)$. Com essa representação dos mapeamentos, é possível ver que existem $\Pi_{\alpha \in \Sigma_S} occ(\alpha, S)!$ mapeamentos possíveis da string S em permutações.

Exemplo 2. Aplicação de um mapeamento $\mathbf{x}$ em uma string com sinais S e em uma string sem sinais P.

$$
\begin{aligned}
S &= (+B\ +C\ -A\ -C\ -B\ +C\ -D\ +D\ +E\ +D\ ), \quad mult(S) = \{B, C, D\} \\
S^{\mathbf{x}} &= (+B^1 +C^2 -A\ -C^1 -B^2 +C^3 -D^2 +D^3 +E\ +D^1) \\
P &= (\ B\ \ C\ \ A\ \ C\ \ B\ \ C\ \ D\ \ D\ \ E\ \ D\ ), \quad mult(P) = \{B, C, D\} \\
P^{\mathbf{x}} &= (\ B^1\ C^2\ A\ \ C^1\ B^2\ C^3\ D^2\ D^3\ E\ \ D^1)
\end{aligned}
$$

$\mathbf{x} =$

B	C	D
1 2	2 1 3	2 3 1

Existe um conjunto de problemas relacionados aos problemas de rearranjo que são os problemas de partição de strings.

Definição 1. Dadas duas strings balanceadas sem sinais S e P, uma *partição direta* $(\mathbb{S}, \mathbb{P}, \phi)$ de S e P é composta por duas sequências $\mathbb{S}$ e $\mathbb{P}$ de strings e uma permutação ϕ, tais que:

- $\mathbb{S}$ e $\mathbb{P}$ têm o mesmo número de elementos ($|\mathbb{S}| = |\mathbb{P}|$);
- É possível obter S concatenando as strings de $\mathbb{S}$;
- É possível obter P concatenando as strings de $\mathbb{P}$;
- $\mathbb{P}_i = \mathbb{S}_{\phi_i}, \forall\ 1 \leq i \leq |\mathbb{S}|$.

Dada uma string S, dizemos que uma string $Q = inv(S)$ é a *inversa* de uma string S se ela consiste dos caracteres de S na ordem inversa, ou seja $Q_i = S_{|S|-i+1}, \forall\ 1 \leq i \leq |S|$. No caso de strings com sinais, o sinal de cada caractere é invertido, ou seja $Q_i = -S_{|S|-i+1},\ \forall\ 1 \leq i \leq |S|$. Se trocarmos a condição $\mathbb{P}_i = \mathbb{S}_{\phi_i}$ por $\mathbb{P}_i = \mathbb{S}_{\phi_i}$ ou $\mathbb{P}_i = inv(\mathbb{S}_{\phi_i})$ na definição de partição direta temos uma *partição reversa*, se as strings são sem sinais, ou uma *partição com sinais*, se as strings são com sinais. Dependendo da partição considerada temos 3 problemas que buscam minimizar o tamanho da partição:

- Partição de Strings Comuns Mínima ("Minimum Common String Partition" – MCSP): considerando partições diretas;
- Partição Reversa de Strings Comuns Mínima ("Reverse Minimum Common String Partition" – RMCSP): considerando partições reversas;
- Partição com Sinais de Strings Comuns Mínima ("Signed Minimum Common String Partition" – SMCSP): considerando partições com sinais.

3. Contribuições

Na dissertação, apresentamos uma nova abordagem baseada em metaheurísticas para os problemas de distância de rearranjo entre genomas (representados por strings) envolvendo os eventos de reversão e transposição. As heurísticas utilizadas são baseadas em mapeamentos e respeitam os seguintes passos:

1. Mapeamos a string de destino P em uma permutação $P^{\mathbf{y}}$ com um mapeamento $\mathbf{y}$ qualquer.
2. Geramos um conjunto $\mathbf{M}$ de mapeamentos da string de origem S em permutações, a partir de estratégias específicas de cada metaheurística.

3. Para cada mapeamento $\mathbf{x} \in \mathbf{M}$, calculamos uma estimativa para o valor $d(S^{\mathbf{x}}, P^{\mathbf{y}})$, utilizando algoritmos existentes para permutações.
4. O resultado da heurística é o valor da menor estimativa encontrada no passo anterior.

As heurísticas se distinguem pela forma como o conjunto $\mathbf{M}$ é gerado. As heurísticas Mapeamentos Aleatórias (MA) e Separação (Sep) geram os mapeamentos de forma aleatória, sendo que a MA utiliza uma distribuição uniforme e a Sep usa os mapeamentos já encontrados para adaptar o processo de geração. As demais heurísticas utilizam estratégias já estabelecidas na literatura: Busca Local (BL), *Greedy Randomized Adaptive Search Procedure* (GRASP), Algoritmos Genéticos (AG), Busca Tabu (BT), Busca Cuco (BC) e *Simulated Annealing* (SA).

Inicialmente, descrevemos e testamos nossas heurísticas considerando um cenário restrito onde cada gene dos genomas pode ter no máximo duas cópias. Nesse caso, a representação dos mapeamentos é simplificada, pois cada rótulo pode ter apenas dois valores associados a ele. Posteriormente, estendemos algumas heurísticas e adaptamos outras ideias da literatura para tratar do caso geral. O algoritmo desenvolvido para o caso geral pode ser decomposto em quatro etapas, representadas na Figura 1.

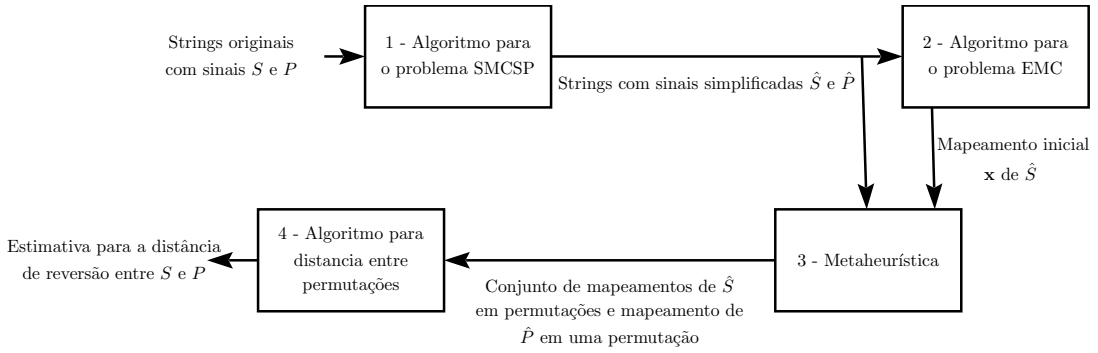

Figura 1. Etapas do algoritmo para estimar a distância de reversão em strings com sinais. As etapas são similares para os demais modelos, apenas os problemas de partição (SMCSP) e de distância entre permutações devem ser substituídos pelas variações correspondentes.

As heurísticas utilizadas compõe a etapa 3. Para as demais etapas, utilizamos adaptações de algoritmos existentes na literatura, realizando testes experimentais para determinar o melhor algoritmo para cada problema.

Na etapa 1, realizamos uma simplificação das strings a partir de algoritmos para problemas de partição de strings. Segundo trabalhos anteriores [Chen et al. 2005, Shapira and Storer 2007], ao simplificarmos as strings utilizando algoritmos de aproximação para os problemas SMCSP e MCSP, podemos garantir aproximações para os problemas $D\bar{R}$ e DT, respectivamente. Na dissertação adaptamos essas provas para estabelecer relações similares entre o problema RMCSP e os problemas DR e DRT, e entre o problema SMCSP e o problema $D\bar{R}T$. Também adaptamos alguns algoritmos da literatura para utilizá-los em todas as variações dos problemas de partição, e conseguimos melhorar os resultados de alguns algoritmos com uma nova heurística baseada em combinação de blocos.

Na etapa 2, utilizamos um algoritmo para um problema chamado Empacotamento Máximo de Ciclos (EMC), que nos permite encontrar um mapeamento inicial das strings em permutações.

Para nossos experimentos práticos, comparamos nossos algoritmos com o algoritmo SOAR da literatura [Chen et al. 2005]. Esse algoritmo foi desenvolvido para o problema $D\bar{R}$, mas realizamos adaptações para aplicá-lo nos demais problemas. Para esses experimentos, criamos os seguintes conjuntos de bases de dados: bases com caracteres duplicados para cada problema (REV, SREV, TRANS, REVTRANS e SREVTRANS); bases difíceis com caracteres duplicados (HARD e SHARD, para os problemas envolvendo strings sem e com sinais, respectivamente); bases aleatórias (RAND e SRAND, para os problemas envolvendo strings sem e com sinais, respectivamente); bases para cada problema, variando a ocorrência máxima (R-O, SR-O, T-O, RT-O e SRT-O); bases para cada problema, variando o tamanho do alfabeto (R-L, SR-L, T-L, RT-L e SRT-L).

A Tabela 1 apresenta um sumário das distâncias obtidas pelas heurísticas considerando as bases compostas por strings que contém apenas caracteres duplicados. Nesse caso, observamos que a melhor heurísticas na maioria dos casos foi a GRASP. Além disso, em geral as heurísticas mais sofisticadas (BL, GRASP, AG, BT, BC, SA e Sep) foram melhores que a heurística MA e que o algoritmo SOAR.

A Tabela 2 apresenta um sumário das distâncias obtidas pelas heurísticas considerando as bases com strings gerais. Nesses testes, as strings já foram simplificadas por algoritmos de partição. As heurísticas marcadas com asterisco utilizam o mapeamento gerados pelo problema EMC. Podemos ver que a heurística BL obteve os melhores resultados e que, quando estamos lidando apenas com o evento de reversão, as heurísticas GRASP e AG também se mostraram boas alternativas. Novamente as heurísticas mais sofisticadas foram melhores que a heurística MA e que o algoritmo SOAR. Além disso, utilizar o mapeamento do problema EMC tende a melhorar os resultados.

Tabela 1. Sumário dos resultados considerando genes duplicados. Os valores correspondem à média das razões entre as distâncias e o tamanho das strings em cada base de dados.

Problema	Base	MA	BL	GRASP	AG	BT	BC	SA	Sep	SOAR
DR	REV	0.42	0.26	**0.25**	**0.25**	0.26	0.28	**0.25**	0.26	0.32
$D\bar{R}$	SREV	0.41	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	0.26	**0.25**
DT	TRANS	0.35	0.27	**0.25**	0.27	0.30	0.32	0.31	0.27	0.27
DRT	REVTRANS	0.31	0.23	**0.22**	0.23	0.25	0.27	0.27	0.23	0.25
$D\bar{R}T$	SREVTRANS	0.31	0.23	**0.22**	0.23	0.26	0.27	0.37	0.23	0.24
DR	HARD	0.73	**0.65**	**0.65**	0.66	0.66	0.66	**0.65**	0.69	**0.65**
$D\bar{R}$	SHARD	0.94	0.92	0.92	0.92	0.92	0.93	**0.91**	0.92	0.94
DT	HARD	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	0.53
DRT	HARD	0.50	**0.49**	**0.49**	**0.49**	**0.49**	**0.49**	**0.49**	**0.49**	0.50
$D\bar{R}T$	SHARD	0.52	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	**0.51**	0.54

Tabela 2. Sumário dos resultados considerando genes multiplicados. Os valores correspondem à média das razões entre as distâncias e o tamanho das strings em cada base de dados.

Problema	Base	MA	BL	GRASP	AG	AG*	BL*	GRASP*	SOAR
DR	RAND	0.95	0.92	0.93	0.93	0.92	**0.91**	0.92	0.92
$D\bar{R}$	SRAND	0.97	0.95	0.95	0.95	**0.93**	**0.93**	0.94	0.94
DT	RAND	0.54	0.53	0.53	0.53	0.53	**0.52**	0.53	0.55
DRT	RAND	0.49	**0.48**	0.49	**0.48**	**0.48**	**0.48**	**0.48**	0.49
$D\bar{R}T$	SRAND	0.51	0.51	0.51	0.51	0.51	**0.50**	0.51	0.52
DR	R-O	0.59	**0.57**	**0.57**	**0.57**	**0.57**	**0.57**	**0.57**	0.58
$D\bar{R}$	SR-O	0.54	**0.52**	**0.52**	0.53	**0.52**	**0.52**	**0.52**	0.53
DT	T-O	0.40	0.40	0.40	0.40	0.40	**0.39**	0.40	0.42
DRT	RT-O	**0.35**	**0.35**	**0.35**	**0.35**	**0.35**	**0.35**	**0.35**	0.36
$D\bar{R}T$	SRT-O	0.36	**0.35**	0.36	0.36	0.36	**0.35**	0.36	0.37
DR	R-L	0.58	**0.57**	**0.57**	**0.57**	**0.57**	**0.57**	**0.57**	0.58
$D\bar{R}$	SR-L	0.55	0.52	0.52	0.52	0.52	0.52	**0.51**	0.54
DT	T-L	0.40	**0.39**	0.40	0.40	0.40	**0.39**	0.40	0.42
DRT	RT-L	**0.35**	**0.35**	**0.35**	**0.35**	**0.35**	**0.35**	**0.35**	0.36
$D\bar{R}T$	SRT-L	0.36	**0.35**	0.36	0.36	**0.35**	**0.35**	0.36	0.37

4. Conclusão

Durante o mestrado desenvolvemos e testamos heurísticas para os problemas de rearranjo de genomas com genes multiplicados considerando os eventos de reversão e transposição. Também adaptamos algoritmos da literatura para melhorar as soluções encontradas.

Dentre as metaheurísticas desenvolvidas, verificamos que nos experimentos práticos a heurística Busca Local obteve bons resultados no caso geral. Quando estamos lidando apenas com o evento de reversão ou com um limite de duas cópias para cada gene, as heurísticas GRASP e Algoritmo Genético também se mostraram boas alternativas.

Como trabalhos futuros, seria interessante explorar outras soluções para o problema EMC, para melhorar a qualidade do mapeamento inicial. Outro caminho para trabalhos futuros é a busca por um melhor fator de aproximação para os problemas de partição, o que garantiria um melhor fator de aproximação para os problemas de distância de rearranjo.

Referências

Berman, P., Hannenhalli, S., and Karpinski, M. (2002). 1.375-Approximation Algorithm for Sorting by Reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002)*, pages 200–210, London, UK. Springer-Verlag.

Bulteau, L., Fertin, G., and Rusu, I. (2012). Sorting by transpositions is difficult. *SIAM Journal on Discrete Mathematics*, 26(3):1148–1180.

Caprara, A. (1999). Sorting permutations by reversals and eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110.

Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., and Jiang, T. (2005). Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315.

Elias, I. and Hartman, T. (2006). A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):369–379.

Fertin, G., Labarre, A., Rusu, I., Tannier, E., and Vialette, S. (2009). *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. The MIT Press, London, England, 1 edition.

Hannenhalli, S. and Pevzner, P. A. (1999). Transforming cabbage into turnip. *Journal of ACM*, 46(1):1–27.

Kolman, P. and Waleń, T. (2007). Approximating reversal distance for strings with bounded number of duplicates. *Discrete Applied Mathematics*, 155(3):327–336.

Oliveira, A. R., Brito, K. L., Dias, U., and Dias, Z. (2019). On the complexity of sorting by reversals and transpositions problems. *Journal of Computational Biology*, 26(11):1223–1229.

Radcliffe, A. J., Scott, A. D., and Wilmer, E. (2005). Reversals and transpositions over finite alphabets. *SIAM Journal on Discrete Mathematics*, 19(1):224–244.

Rahman, A., Shatabda, S., and Hasan, M. (2008). An approximation algorithm for sorting by reversals and transpositions. *Journal of Discrete Algorithms*, 6(3):449–457.

Shao, M., Lin, Y., and Moret, B. M. (2015). An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. *Journal of Computational Biology*, 22(5):425–435.

Shapira, D. and Storer, J. A. (2007). Edit distance with move operations. *Journal of Discrete Algorithms*, 5(2):380–392.

Walter, M. E. M., Dias, Z., and Meidanis, J. (1998). Reversal and transposition distance of linear chromosomes. In *Proceedings of the String Processing and Information Retrieval: A South American Symposium (SPIRE'1998)*, pages 96–102, Los Alamitos, CA, USA.

Yancopoulos, S., Attie, O., and Friedberg, R. (2005). Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346.