Pseudo-Boolean Approaches to Comparative Genomics (Extended Abstract)

João Benigno Delgado

July 1, 2009

Abstract

Comparative genomics is the study of structural and functional relationships between genomes. Primarily, the comparison of genomes in terms of genetic evolution was based on gene comparison. However, the increasing number of fully sequenced genomes has led to the study of genome rearrangements, i.e., to the comparison of two genomes in terms of gene order evolution.

Several approaches have been made to this topic, all of them being either too complex to be solved efficiently or too simple to be applied to the genomes of complex organisms. The latest challenge has been to overcome the problem of having genomes with duplicate genes. This led to the definition of two matching models: exemplar [14] and maximum [6]; and of several similarity measures, such as: the number of breakpoints [12], the number of common intervals [18] and the Summed Adjacency Disruption number (SAD) [16]. The idea is to find a matching, i.e., a correspondence between homologous genes in two genomes, in order to disambiguate the data of duplicate genes and calculate a similarity measure. The problem becomes that of finding a matching that best preserves the order of genes in two genomes, where gene order is evaluated by a chosen similarity measure.

This dissertation presents clearer formalisms of the approaches made on genome rearrangements and extends previous work using Pseudo-Boolean Optimization (PBO) to encode the problems of finding the number of common intervals [2] and the number of breakpoints [1]. Moreover, a new PBO encoding of the SAD problem is presented and evaluated.

1 Introduction

Genomes are known to evolve in two ways: through *point mutations* in DNA, that are responsible for the evolution of individual genes, and through wider operations, known as *rearrangement events*, that may affect the order by which genes occur in genomes.

Traditional methods for comparing genomes in terms of genetic evolution are based in the comparison of homologous versions of genes in two genomes [5, 3, 17]. These methods capture *point mutations* in the sequence of genes, but are limited in disregarding difference in the order by which genes occur in the considered genomes [17].

In *genome rearrangements* the idea is to compare two genomes in terms of gene order evolution. Several approaches have been made to this problem, each modeling genomes in a different way and considering different types of *rearrangement events* [10, 4, 9]. Their aim is to find a minimum number of the considered *rearrangement events* necessary to transform the order of genes in one genome into the order of genes in the other genome. Assuming that evolution proceeded only through the considered type of *rearrangement events*, finding such a scenario of evolution between two genomes, provides a lower bound on the actual number of *rearrangement events* that must have occurred since the genomes began to diverge [7].

The approaches made to *genome rearrangements* have been used to estimate how long ago the two genomes began to diverge from each other or, when comparing several genomes, to compute phylogenetic trees [8, 13, 17]. However, they all share the main bottleneck of not allowing the presence of duplicate genes in genomes (i.e., the presence of several homologous versions of genes in genomes), which has been found to be common [15, 11].

One of the proposed approaches to overcome this limitation has been the use of *matching models* and *similarity measures*, an idea first proposed by David Sankoff [14]. A *matching* is a one-to-one correspondence between homologous genes in two genomes. Such a correspondence allows to see the

order of genes in one genome as a permutation of the order of genes in the other genome. The idea is to find a *matching* for which this induced order is best preserved, where order preservation is evaluated by a chosen *similarity measure*.

In this dissertation, we provide a clear formalism for the use of *matching models* and *similarity measures* and propose a novel generic pseudo-Boolean optimization encoding for the problem of finding a *matching* between two genomes, of the exemplar [14] or the maximum [6] model, that minimizes the *summed adjacency disruption number* (SAD) [16]. Additionally, we test this encoding on the same data set of γ-Protobacteria used in [2, 1].

We begin by presenting our formalism in section 2, followed by the definition of SAD as a similarity measure and some of its properties in section 3. Our pseudo-Boolean optimization encoding is presented in section 4 (preprocessing and variable reduction rules are omitted due to space constraints) and results are presented in section 5. Section 6 presents our conclusions.

2 Optimal matchings for similarity measures

We now present the formalism for the use of *matching models* and *similarity measures*. We begin with the definition of genomes (definition 2.1) and of a matching between two genomes (definition 2.2). Definition 2.4 explains how two signed permutations (definition 2.3) are derived from two genomes and a matching between them. Following, similarity measures are introduced and the problem of finding a matching between two genomes that optimizes a given similarity measure is defined (definition 2.7).

Definition 2.1 (genome). *Consider a finite alphabet $\mathcal{A}$ of gene families. A gene is an element of $\mathcal{A}$ having a $+$ or $-$ sign prefixed to it, standing for transcriptional reading direction. If g is a gene then:*

- *$-g$ denotes the gene obtained from g by switching its sign.*
- *$s(g)$ denotes the sign of g.*
- *$|g|$ denotes the gene family of g.*

A genome G over $\mathcal{A}$ is a sequence $G[1]\cdots G[n_G]$ of genes, where n_G denotes the length of G. For all $a \in \mathcal{A}$ and for all $i, j \in \mathbb{N}$ such that $1 \leq i \leq j \leq n_G$, $occ_G(a, i, j)$ denotes the number of occurrences of genes g in a genome G, between positions i and j (inclusively), such that $|g| = a$. $occ_G(a, 1, n_G)$ is abbreviated to $occ_G(a)$. A genome G over $\mathcal{A}$ is said to be duplicate-free *if for all $a \in \mathcal{A}$, $occ_G(a) \leq 1$. G is said to have* duplicate genes *otherwise.*

Definition 2.2 (matching). *A matching $\mathcal{M}$ between two genomes G and H over the same finite alphabet of gene families $\mathcal{A}$, is a set $\mathcal{M} \subseteq \{1, \ldots, n_G\} \times \{1, \ldots, n_H\}$ of pairs that are pairwise disjoint and establish a correspondence between genes of the same gene family in both genomes, i.e:*

- *Every two distinct elements $(i, j), (i', j') \in \mathcal{M}$ verify that $i \neq i'$ and $j \neq j'$.*
- *Every element $(i, j) \in \mathcal{M}$ verifies that $|G[i]| = |H[j]|$.*

The number of elements in a matching $\mathcal{M}$ is denoted as $|\mathcal{M}|$. Genes $G[i]$ and $H[j]$ that are matched by a pair (i, j) in a matching $\mathcal{M}$ are said to be $\mathcal{M}$-saturated. A matching is said to be:

- *An exemplar matching: if, for each gene family $a \in \mathcal{A}$, such that $occ_G(a) \geq 1$ and $occ_H(a) \geq 1$, it saturates exactly one gene, in each genome, that belongs to that family.*
- *A maximum matching: if it saturates as many genes of each gene family as possible, i.e., if it is a matching of maximum cardinality.*

Prior to defining how the optimality of a matching $\mathcal{M}$ is evaluated, we define how a matching $\mathcal{M}$, between two genomes G and H, induces two signed permutations $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$ of size $|\mathcal{M}|$ (definition 2.4). The idea is to delete unmatched genes in G and H, assign a number from 1 to $|\mathcal{M}|$ to each pair in $\mathcal{M}$ and rename matched genes with the number given to the pair in $\mathcal{M}$ that matches them (the signs of genes are kept unchanged).

Definition 2.3 (signed permutation). *A signed permutation $\overrightarrow{\pi}$ of the elements of the set $\{1,\ldots,n\}$ is a tuple $\langle \pi, s\rangle$, where π is a permutation in S_n and s is a function $s : \{1,\ldots,n\} \longrightarrow \{+,-\}$. $\overrightarrow{S_n}$ denotes the set of all signed permutations of the set $\{1,\ldots,n\}$. A signed permutation $\overrightarrow{\pi} = \langle \pi, s\rangle \in \overrightarrow{S_n}$ can be generally represented by the sequence $\overrightarrow{\pi} = s(1)\pi(1)\cdots s(n)\pi(n)$.*

Definition 2.4 ($\mathcal{M}$-reduced genomes). *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$, of lengths n_G and n_H. Consider a matching $\mathcal{M}$ between G and H and a bijective function $f : \mathcal{M} \longrightarrow \{1,\ldots,|\mathcal{M}|\}$. Define $G_0 = G$, $G_1 = H$, $n_{G_0} = n_G$ and $n_{G_1} = n_H$. Define $(a,b)^{-0} \stackrel{def}{=} (a,b)$ and $(a,b)^{-1} \stackrel{def}{=} (b,a)$, for all $a,b \in \mathbb{N}$ and $\overline{x} \stackrel{def}{=} 1-x$, for all $x \in \{0,1\}$. Let:*

- *$P^x_{\mathcal{M}}(i) = 1$ if $G_x[i]$ is $\mathcal{M}$-saturated, i.e., if there exists $k \in \mathbb{N}$ such that $1 \leq k \leq n_{\overline{x}}$ and $(i,k)^{-x} \in \mathcal{M}$, for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$; $P^x_{\mathcal{M}}(i) = 0$ otherwise.*
- *$k^x_{\mathcal{M}}(i)$ denote the position of the ith $\mathcal{M}$-saturated gene in G_x, i.e., the minimum number t for which $\sum_{j=1}^{t} P^x_{\mathcal{M}}(j) = i$, for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$.*
- *$m^x_{\mathcal{M}}(i)$ denote the pair in $\mathcal{M}$ that matches the gene $G_x[i]$, i.e., the pair $(i,k)^{-x} \in \mathcal{M}$ (where $1 \leq k \leq n_{\overline{x}}$), for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$ and $P^x_{\mathcal{M}}(i) = 1$.*

The $\mathcal{M}$-reduced genomes $\mathcal{M}_f(G), \mathcal{M}_f(H) \in \overrightarrow{S_{|\mathcal{M}|}}$ are defined as follows:

$$\mathcal{M}_f(G) \stackrel{def}{=} \langle \pi_0, s_0\rangle \quad \wedge \quad \mathcal{M}_f(H) \stackrel{def}{=} \langle \pi_1, s_1\rangle \tag{1}$$

Where for all $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$ [1]:

$$\pi_x(i) \stackrel{def}{=} f(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) \quad \wedge \quad s_x(i) \stackrel{def}{=} s(G_x[k^x_{\mathcal{M}}(i)]) \tag{2}$$

Similarity measures are families $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$. Since $\mathcal{M}$-reduced genomes are signed permutations in $\overrightarrow{S_{|\mathcal{M}|}}$, the idea is to use $d_{|\mathcal{M}|}(\mathcal{M}_f(G), \mathcal{M}_f(H))$ to evaluate the optimality of $\mathcal{M}$. This value, however, should not depend on the choice of the bijective function $f : \mathcal{M} \longrightarrow \{1,\ldots|\mathcal{M}|\}$. Proposition 2.1 establishes a sufficient and necessary condition of $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ for this to be true.

Definition 2.5 (invariant on the left for permutations). *We define the composition of two signed permutations as the binary operation $\odot : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \overrightarrow{S_n}$ such that $\langle \pi_1, s_1\rangle \odot \langle \pi_2, s_2\rangle = \langle \pi, s\rangle$, where $\pi \stackrel{def}{=} \pi_1 \circ \pi_2$ and:*

$$s(i) \stackrel{def}{=} \begin{cases} + & , \text{ if } s_2(i) = s_1(\pi_2(i)) \\ - & , \text{ if } s_2(i) \neq s_1(\pi_2(i)) \end{cases} \tag{3}$$

A function $d : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ is said to be invariant on the left for permutations if:

$$\forall_{\overrightarrow{\pi_1},\overrightarrow{\pi_2}\in\overrightarrow{S_n}} \forall_{\pi\in S_n} \quad d(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d(\langle \pi, +\rangle \odot \overrightarrow{\pi_1}, \langle \pi, +\rangle \odot \overrightarrow{\pi_2}) \tag{4}$$

where $+ : \{1,\ldots,n\} \longrightarrow \{+,-\}$ is defined as $+(i) = +$.

Proposition 2.1. *Consider a family $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$. The two following conditions are equivalent:*

1. *for all $n \in \mathbb{N}$, d_n is invariant on the left for permutations.*
2. *for all finite alphabets of gene families $\mathcal{A}$, all genomes G and H over $\mathcal{A}$, all matchings $\mathcal{M}$ between G and H and all bijective functions $f_1, f_2 : \mathcal{M} \longrightarrow \{1,\ldots,|\mathcal{M}|\}$, it holds that:*

$$d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) \tag{5}$$

[1] $s(G_x[k^x_{\mathcal{M}}(i)])$ is the sign of the gene in position $k^x_{\mathcal{M}}(i)$ of G_x (see definition 2.1).

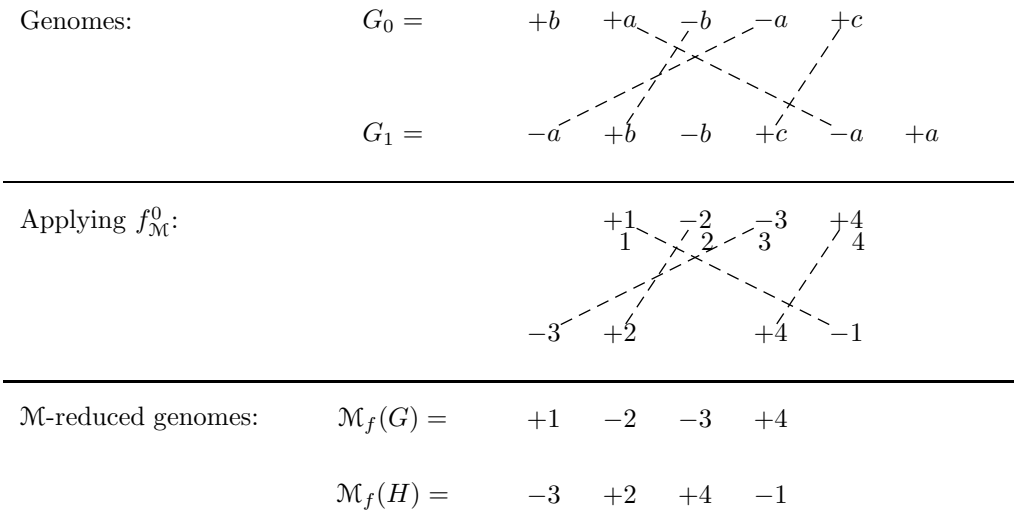

Figure 1: Building of the $\mathcal{M}$-reduced genomes $\mathcal{M}_{f^0_{\mathcal{M}}}(G)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(H)$ for the genomes $G = +b + a - b - a + c$ and $H = -a + b - b + c - a + a$ and the matching $\mathcal{M} = \{(2,5),(3,2),(4,1),(5,4)\}$ between G and H.

It is now important to define two particular functions: $f^0_{\mathcal{M}} : \mathcal{M} \longrightarrow \{1,\dots|\mathcal{M}|\}$, for which $\mathcal{M}_{f^0_{\mathcal{M}}}(G) = \pm 1 \cdots \pm |\mathcal{M}|$ and $f^1_{\mathcal{M}} : \mathcal{M} \longrightarrow \{1,\dots|\mathcal{M}|\}$, for which $\mathcal{M}_{f^1_{\mathcal{M}}}(H) = \pm 1 \cdots \pm |\mathcal{M}|$. $f^0_{\mathcal{M}}$ is chosen for building $\mathcal{M}$-reduced genomes (definition 2.7). See figure 1 for an illustration on the construction of $\mathcal{M}_{f^0_{\mathcal{M}}}(G)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(H)$. $f^1_{\mathcal{M}}$ will be called for in section 3.

Definition 2.6 ($f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$). *Let G and H be two genomes over a finite alphabet of gene families and $\mathcal{M}$ be a matching between G and H. Define $G_0 = G$ and $G_1 = H$. The bijective functions $f^0_{\mathcal{M}}, f^1_{\mathcal{M}} : \mathcal{M} \longrightarrow \{1,\dots,|\mathcal{M}|\}$ are defined as follows:*

$$f^0_{\mathcal{M}}((i,k)) \stackrel{def}{=} \sum_{j=1}^{i} P^0_{\mathcal{M}}(j) \qquad f^1_{\mathcal{M}}((i,k)) \stackrel{def}{=} \sum_{j=1}^{k} P^1_{\mathcal{M}}(j) \tag{6}$$

for all $(i,k) \in \mathcal{M}$. $P^x_{\mathcal{M}}(j)$ are defined as in definition 2.4.

We now define the problem of finding a matching $\mathcal{M}$, of a chosen model, that optimizes a chosen similarity measure (definition 2.7). Proposition 2.2 establishes a symmetry result on this problem.

Definition 2.7 ($OPT_e(G,H,d)$ and $OPT_m(G,H,d)$). *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations. Then:*

- *$OPT_e(G,H,d)$ is the problem of finding an exemplar matching $\mathcal{M}$, between G and H, such that $d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G), \mathcal{M}_{f^0_{\mathcal{M}}}(H))$ is optimized.*
- *$OPT_m(G,H,d)$ is the problem of finding maximum matching $\mathcal{M}$, between G and H, such that $d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G), \mathcal{M}_{f^0_{\mathcal{M}}}(H))$ is optimized.*

Proposition 2.2. *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations and symmetric. Then:*

1. *If $\mathcal{M}$ solves $OPT_e(G,H,d)$, then $T(\mathcal{M})$ solves $OPT_e(H,G,d)$.*
2. *If $\mathcal{M}$ solves $OPT_m(G,H,d)$, then $T(\mathcal{M})$ solves $OPT_m(H,G,d)$.*

where $T(\mathcal{M}) \stackrel{def}{=} \{(k,i) : (i,k) \in \mathcal{M}\}$.

3 The summed adjacency disruption number

Before presenting our pseudo-Boolean optimization encoding, we provide the definition of the summed adjacency disruption number (SAD) as well as some important results on this similarity measure, namely, the required properties for the problems $OPT_e(G, H, SAD)$ and $OPT_m(G, H, SAD)$ to verify symmetry presented in proposition 2.2 (proposition 3.1).

Definition 3.1 (summed adjacency disruption number). *Let $\overrightarrow{\pi_0} = \langle \pi_0, s_0 \rangle, \overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle \in \overrightarrow{S_n}$. $SAD_n(\overrightarrow{\pi_0}, \overrightarrow{\pi_1})$ is defined as the sum of two values $SAD_n^{0,1}(\overrightarrow{\pi_0}, \overrightarrow{\pi_1})$ and $SAD_n^{1,0}(\overrightarrow{\pi_0}, \overrightarrow{\pi_1})$, computed as follows:*

$$SAD_n^{0,1}(\overrightarrow{\pi_0}, \overrightarrow{\pi_1}) = \sum_{i=1}^{n-1} \left| (\pi_0^{-1} \circ \pi_1)(i) - (\pi_0^{-1} \circ \pi_1)(i+1) \right| \tag{7}$$

$$SAD_n^{1,0}(\overrightarrow{\pi_0}, \overrightarrow{\pi_1}) = \sum_{i=1}^{n-1} \left| (\pi_1^{-1} \circ \pi_0)(i) - (\pi_1^{-1} \circ \pi_0)(i+1) \right| \tag{8}$$

Proposition 3.1. *For all $n \in \mathbb{N}$, $SAD_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ is invariant on the left for permutations and symmetric.*

Proposition 3.2 states that $SAD_{|\mathcal{M}|}^{0,1}(\mathcal{M}_{f_{\mathcal{M}}^0}(G), \mathcal{M}_{f_{\mathcal{M}}^0}(H))$ equals the sum of the absolute difference of consecutive numbers in $\mathcal{M}_{f_{\mathcal{M}}^0}(H)$ and $SAD_{|\mathcal{M}|}^{1,0}(\mathcal{M}_{f_{\mathcal{M}}^0}(G), \mathcal{M}_{f_{\mathcal{M}}^0}(H))$ equals the sum of the absolute difference of consecutive numbers in $\mathcal{M}_{f_{\mathcal{M}}^1}(G)$.

Proposition 3.2. *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $\mathcal{M}$ be a matching between G and H. Let $\mathcal{M}_{f_{\mathcal{M}}^0}(G) = \langle \pi_0, s_0 \rangle$, $\mathcal{M}_{f_{\mathcal{M}}^0}(H) = \langle \pi_1, s_1 \rangle$ and $\mathcal{M}_{f_{\mathcal{M}}^1}(G) = \langle \pi_2, s_2 \rangle$. Then:*

$$\pi_0^{-1} \circ \pi_1 = \pi_1 \quad \wedge \quad \pi_1^{-1} \circ \pi_0 = \pi_2 \tag{9}$$

Therefore:

$$SAD_{|\mathcal{M}|}^{0,1}(\mathcal{M}_{f_{\mathcal{M}}^0}(G), \mathcal{M}_{f_{\mathcal{M}}^0}(H)) = \sum_{i=1}^{|\mathcal{M}|-1} |\pi_1(i) - \pi_1(i+1)| \tag{10}$$

$$SAD_{|\mathcal{M}|}^{1,0}(\mathcal{M}_{f_{\mathcal{M}}^0}(G), \mathcal{M}_{f_{\mathcal{M}}^0}(H)) = \sum_{i=1}^{|\mathcal{M}|-1} |\pi_2(i) - \pi_2(i+1)| \tag{11}$$

4 Pseudo-Boolean Optimization encoding for SAD

We now present our pseudo-Boolean optimization encoding for the problems $OPT_e(G_0, G_1, SAD)$ and $OPT_m(G_0, G_1, SAD)$. The encoding is resumed in **table 1**, were all constraints and the cost function are presented. Here we introduce the considered variables, explain the constraints (presented in table 1) and justify how a solution of this encoding solves the problems $OPT_e(G_0, G_1, SAD)$ and $OPT_m(G_0, G_1, SAD)$.

This encoding takes for input two genomes G_0 and G_1 over an alphabet of gene families $\mathcal{A}$. It is assumed that genes, that occur in only one of the two genomes, have been deleted from G_0 and G_1[2]. Therefore we have that for all $a \in \mathcal{A}$, $occ_{G_0}(a) \geq 1$ and $occ_{G_1}(a) \geq 1$. The lengths of G_0 and G_1 are abbreviated to n_0 and n_1 and $\overline{x}$ denotes $|x - 1|$, for $x \in \{0, 1\}$.

Variables and constraints

The considered Boolean variables are the following:

$$A = \{a(i,k) : i,k \in \mathbb{N} \wedge 1 \leq i \leq n_0 \wedge 1 \leq k \leq n_1 \wedge G_0[i] = G_1[k]\}$$
$$B = \{b_x(i) : x \in \{0,1\} \wedge i \in \mathbb{N} \wedge 1 \leq i \leq n_x\}$$
$$C = \{c_x(i,j) : x \in \{0,1\} \wedge i,j \in \mathbb{N} \wedge 1 \leq i < j \leq n_x\}$$
$$D = \{d_x(i,j,k,l) : x \in \{0,1\} \wedge i,j,k,l \in \mathbb{N} \wedge 1 \leq i < j \leq n_x \wedge 1 \leq k < l \leq n_{\overline{x}}\}$$
$$E = \{e_x(i,j,p) : x \in \{0,1\} \wedge i,j,p \in \mathbb{N} \wedge 1 \leq i < j \leq n_x \wedge i < p < j\}$$

[2]No such gene can matched and therefore, the problem remains equivalent after this preprocessing step.

Variables $a(i,k)$ represent pairs of a matching $\mathcal{M}$ and variables $b_x(i)$ represent that the gene $G_x[i]$ is matched by $\mathcal{M}$. If the *exemplar matching* is assumed, then constraints CE are added to force that, for each gene family $a \in \mathcal{A}$, exactly one gene, of the gene family a, is matched in G_0 and G_1. If a *maximum matching* is desired, constraints CM are added to force that, for each gene family $a \in \mathcal{A}$, exactly $min\{occ_{G_0}(a), occ_{G_1}(a)\}$ genes, of the gene family a, are matched in G_0 and G_1. Constraints $C1$ and $C2$ force variables $a(i,k)$ to define a valid matching between G_0 and G_1. Constraints $C1$ guarantee that every gene $G_0[i]$ is matched, at most, to one gene $G_1[k]$ and that this is only possible if $b_0(i)$ takes value 1. Constraints $C2$ guarantee that every gene $G_1[k]$ is matched, at most, to one gene $G_0[i]$ and that this is only possible if $b_1(k)$ takes value 1.

Variables $c_x(i,j)$ represent that no gene in G_x, strictly between positions i and j, is matched by $\mathcal{M}$. Constraints $C3$ force a variable $c_x(i,j)$ to take value 1 if no variable $b_x(p)$, where $i < p < j$, takes value 1. Constraints $C4$ force a variable $c_x(i,j)$ to take value 0 if some variable $b_x(p)$, where $i < p < j$, takes value 1.

Variables $d_x(i,j,k,l)$ represent that $G_{\overline{x}}[k]$ and $G_{\overline{x}}[l]$ correspond to consecutive numbers in $\mathcal{M}_{f^x_{\mathcal{M}}}(G_{\overline{x}})$ that are matched to $G_x[i]$ and $G_x[j]$ (or to $G_x[j]$ and $G_x[i]$), respectively. Variables $d_0(i,j,k,l)$ will be used to identify consecutive numbers in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ to compute the value of $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$. Variables $d_1(i,j,k,l)$ will be used to identify consecutive numbers in $\mathcal{M}_{f^1_{\mathcal{M}}}(G_0)$ to compute the value of $SAD^{1,0}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$.

Let $a(i,k)^{-0} \stackrel{def}{=} a(i,k)$ and $a(i,k)^{-1} \stackrel{def}{=} a(k,i)$ for all variables $a(i,k) \in A$. A variable $d_x(i,j,k,l)$, such that $G_x[i] = G_{\overline{x}}[k]$ and $G_x[j] = G_{\overline{x}}[l]$, will take value 1 if and only if variables $c_{\overline{x}}(k,l)$, $a(i,k)^{-x}$ and $a(j,l)^{-x}$ take value 1 (constraints $C5$). A variable $d_x(i,j,k,l)$, such that $G_x[i] = G_{\overline{x}}[l]$ and $G_x[j] = G_{\overline{x}}[k]$, will take value 1 if and only if variables $c_{\overline{x}}(k,l)$, $a(i,l)^{-x}$ and $a(j,k)^{-x}$ take value 1 (constraints $C6$). A variable $d_x(i,j,k,l)$, for which neither of the previous conditions is verified, is forced to take value 0 by constraints $C7$.

Notice that, due to constraints $C5$, $C6$ and $C7$ (and also due to constraints CE (or CM), $C1$ and $C2$ that force variables $a(i,k)$ to define a valid matching between G_0 and G_1), for any valoration $\mathcal{V} : A \cup B \cup C \cup D \longrightarrow \{0,1\}$ that satisfies all the constraints mentioned so far, and for all $d_x(i,j,k,l) \in D$, if $\mathcal{V}(d_x(i,j,k,l)) = 1$, then for every other variables $d_x(i,j,k',l'), d_x(i',j',k,l) \in D$ it holds that $\mathcal{V}(d_x(i,j,k',l')) = \mathcal{V}(d_x(i',j',k,l)) = 0$. Nevertheless, we impose this fact with constraints $C8$ and $C9$ that will, hopefully, improve the solving time of this pseudo-Boolean optimization encoding.

Variables $e_x(i,j,p)$ are introduced to count the number of non-$\mathcal{M}$-saturated genes in G_x, strictly between positions i and j, when it is the case that some variable $d_x(i,j,k,l)$ is assigned to 1. Constraints $C10$ force variables $e_x(i,j,p)$ to be assigned to 1 if and only if the following holds: some variable $d_x(i,j,k,l)$ is assigned to 1, $b_x(p)$ is assigned to 0 and both $b_x(i)$ and $b_x(j)$ are assigned to 1 [3].

Cost function

It is clear that for any solution, i.e., any valoration $\mathcal{V} : A \cup B \cup C \cup D \cup E \longrightarrow \{0,1\}$ that satisfies all the constraints, $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ is a valid matching (exemplar or maximum) between G_0 and G_1 and there is a one to one correspondence between variables $d_0(i,j,k,l) \in D$ such that $\mathcal{V}(d_0(i,j,k,l)) = 1$, and consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$. Moreover, if $d_0(i,j,k,l)$ is assigned to 1, then either $G_0[i]$ is matched to $G_1[k]$ and $G_0[j]$ is matched to $G_1[l]$, or $G_0[i]$ is matched to $G_1[l]$ and $G_0[j]$ is matched to $G_1[k]$. In either case, if a variable $d_0(i,j,k,l)$ is assigned to 1 then, the consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ to which $d_0(i,j,k,l)$ corresponds, will contribute to $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ with an amount equal to the number of $\mathcal{M}$-saturated genes in G_0, strictly between positions i and j, plus one (see figure 2).

Suppose that a variable $d_0(i,j,k,l)$ is assigned to 1 because $c_1(k,l)$, $a(i,k)$ and $a(j,l)$ are assigned to 1. Let a be the number with which $G_0[i]$ and $G_1[k]$ will be renamed in the construction of $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$. It is clear to see that a corresponds to the number of genes in G_0, in a position p such that $1 \leq p \leq i$, that are matched by $\mathcal{M}$ (see definitions 2.4, 2.6 and figure 1). From this, it is clear that $a + b + 1$ is the number with which $G_0[j]$ and $G_1[l]$ will be renamed in the construction of $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, where b is the number of $\mathcal{M}$-saturated genes in G_0, strictly between positions i and j. Therefore, since $G_1[k]$ and $G_1[l]$ correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, variable $d_0(i,j,k,l)$ will

[3] Including variables $b_x(i)$ and $b_x(j)$ in constraints $C10$ is unnecessary but it will hopefully improve the solving time.

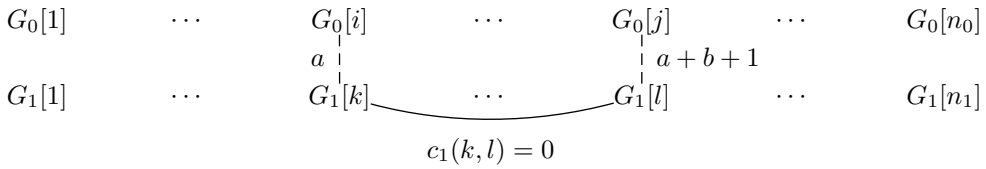

Figure 2: Variable $d_0(i,j,k,l)$ is assigned to 1 because $c_1(k,l)$, $a(i,k)$ and $a(j,l)$ are assigned to 1. Variable $d_0(i,j,k,l)$ will contribute do $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ with the value $|(a+b+1)-a| = b+1$.

contribute do $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ with the value $|(a+b+1)-a| = b+1$, i.e., the number of $\mathcal{M}$-saturated genes in G_0, strictly between positions i and j, plus one (where $\mathcal{M}$ is the matching defined by variables $a(i,k)$). If it was the opposite case of having $a(i,l)$ and $a(j,k)$ assigned to 1, the variable $d_0(i,j,k,l)$ would contribute to $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ with the same amount, because $|a-(a+b+1)| = |-b-1| = b+1$. Clearly, $b+1$ equals $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_0, strictly between positions i and j.

Following a similar reasoning over variables $d_1(i,j,k,l)$, we have that:

- $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ equals the sum, over all variables $d_0(i,j,k,l)$ assigned to 1, of $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_0, strictly between positions i and j.
- $SAD^{1,0}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ equals the sum, over all variables $d_1(i,j,k,l)$ assigned to 1, of $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_1, strictly between positions i and j.

We now explain how the value of the cost function (see table 1), that is to be minimized, represents the value of $SAD_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$. For every pair of positions $i<j$ in G_0, the following is added to the cost function:

$$\left(\sum_{k=1}^{n_1-1}\sum_{l=k+1}^{n_1}(j-i)\cdot d_0(i,j,k,l)\right) - \sum_{p=i+1}^{j-1} e_0(i,j,p) \tag{12}$$

The idea is that if, for a given pair of positions $i<j$ in G_0, there exists a variable $d_0(i,j,k,l)$ assigned to 1, then all other variables $d_0(i,j,k',l')$ are assigned to 0 and $\sum_{k=1}^{n_1-1}\sum_{l=k+1}^{n_1}(j-i)\cdot d_0(i,j,k,l) = (j-i)$. Also, in this case, variables $e_0(i,j,p)$ take the value 1 if and only if $G_0[p]$ is unmatched by $\mathcal{M}$. Therefore, if there exists a variable $d_0(i,j,k,l)$ assigned to 1, then we are adding $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_1, strictly between positions i and j, to the total. If no variable $d_0(i,j,k,l)$ is assigned to 1, then $\sum_{k=1}^{n_1-1}\sum_{l=k+1}^{n_1}(j-i)\cdot d_0(i,j,k,l) = 0$, variables $e_0(i,j,p)$ are forced to 0 and we are adding 0 to the total. Therefore, by adding, for every pair of positions $i<j$ in G_0, the term in equation 12 to the cost function, we have that $SAD^{0,1}_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ equals:

$$\sum_{i=1}^{n_0-1}\sum_{j=i+1}^{n_0}\left(\left(\sum_{k=1}^{n_1-1}\sum_{l=k+1}^{n_1}(j-i)\cdot \mathcal{V}(d_0(i,j,k,l))\right) - \sum_{p=i+1}^{j-1}\mathcal{V}(e_0(i,j,p))\right) \tag{13}$$

Following a similar reasoning over every pair of positions $i<j$ in G_1 and over variables $d_1(i,j,k,l)$, we have that an optimal solution $\mathcal{V}$ (i.e., a solution for which the cost function is minimized) defines a valid matching $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ (exemplar or maximum) between G_0 and G_1 for which the value of $SAD_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ is minimized. In other words $\mathcal{M}$ is a solution of $OPT_e(G_0, G_1, SAD)$ (or $OPT_m(G_0, G_1, SAD)$).

5 Experimental results

We tested this encoding on the same data set of γ-Protobacteria used in [2, 1]. Experiments were run using *cplex*[4], which is an Integer Linear Programming (ILP) solver, on an Intel Xeon 5160 server (3.0GHZ, 1333Mhz, 4GB) running Red Hat Enterprise Linux WS 4. We discarded instances which

[4]Version 11.2. Available at `http://www.ilog.com/products/cplex`

resulted in files with size greater than 1 Gigabyte and used 10 hours as the cut off value for solving the remaining instances.

The pseudo-Boolean optimization instances for SAD have proved to be harder to solve than the PBO instances for common intervals [2] or breakpoints [1]. This comes as no surprise because the variable reduction rules are much less strict than the ones for breakpoints [1] and therefore, the encoding of the problems $OPT_e(G, H, SAD)$ and $OPT_m(G, H, SAD)$ results in much larger instances due to a larger number of variables involved. Nonetheless, we have obtained the first results on these problems, namely when either G or H is the genome Baphi or Wglos. Solutions and running times are presented in figures 3 and 4.

Solution	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	32058	66510	60896‡	64894	31982	37082	55614	53018	56036	42556‡	43718‡
Wglos	63496‡	72908	75512‡	74676	63126‡		70272	68628	68304	62398‡	59420‡
Time											
Baphi	13312.76	778.94	§	823.08	23425.57	0.16	2339.82	4743.49	18.52	§	§
Wglos	§	2190.78	§	782.39	§		1177.17	1466.11	96.05	§	§

Figure 3: Results for the summed adjacency disruption number using the exemplar model.
(‡) sub-optimal solution (the greatest gap in sub-optimal solutions was of 6.55%).
(§) *Cplex* reached the time limit of ten hours.

Solution	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	34126‡	69462	64925‡	68336	34054‡	37906	58348	55546	59172	45011‡	45783‡
Wglos	67793‡	76669	80666‡	78286	66839‡		74012‡	72045	72140	66620‡	63927‡
Time											
Baphi	§	622.25	§	863.30	§	0.43	8887.94	3467.77	9.96	§	§
Wglos	§	1416.38	§	10795.27	§		§	9266.44	1257.11	§	§

Figure 4: Results for the summed adjacency disruption number using the maximum model.
(‡) sub-optimal solution (the greatest gap in sub-optimal solutions was of 3.10%).
(§) *Cplex* reached the time limit of ten hours.

6 Conclusion

While researching on the pseudo-Boolean optimization encodings presented in [2, 1], we provided some variable reduction rules and constraint value setting rules (in both exemplar and maximum matchings) for the number of breakpoints and common intervals. Experimental results show that *cplex* preforms extremely well on our instances. Moreover, we completed the work started in the number of breakpoints [1] and were able to obtain nearly all results for the number of common intervals [2] (we missed 6 out of 132). As for the new generic pseudo-Boolean optimization encoding of SAD proposed in this dissertation, we found the instances produced to be much harder than those of common intervals or breakpoints, mostly due to the greater amount of variables that are not eliminated by the variable cutting rules. Nevertheless, optimal solutions were obtained for some instances and these can be used to validate the accuracy of approximation algorithms. Moreover, the sub-optimal solutions found using *cplex* are valid, and provide upper bounds (and also lower bounds) on the optimal solutions.

References

[1] Sébastien Angibaud, Guillaume Fertin, Irena Rusu, Annelyse Thévenin, and Stéphane Vialette. A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In *5th RECOMB Comparative Genomics Statellite Workshop*, volume 4751 of *Lecture Notes in Computer Science*, pages 16–29, 2007.

[2] Sébastien Angibaud, Guillaume Fertin, Irena Rusu, and Stéphane Vialette. How pseudo-boolean programming can help genome rearrangement distance computation. In *4th RECOMB Comparative Genomics Statellite Workshop*, volume 4205 of *Lecture Notes in Computer Science*, pages 75–86, 2006.

[3] V. Bafna and P. A. Pevzner. Sorting by reversals: Genome rearrangements in plant organneles and evolutionary history of x chromosome. *Molecular Biology and Evolution*, 12(2):239–246, 1995.

[4] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM journal on computing*, 25(2):272–289, 1996.

[5] Alberto Caprara and See kiong Ng. A column-generation based branch-and-bound algorithm for sorting by reversals. In *Mathematical Support for Molecular Biology*, volume 47 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 213–226. AMS Press, 1999.

[6] Cedric Chauve, Guillaume Fertin, Romeo Rizzi, and Stéphane Vialette. Genomes containing duplicates are hard to compare. In *2nd International Workshop on Bioinformatics Research and Applications*, volume 3992 of *Lecture Notes in Computer Science*, pages 783–790, 2006.

[7] S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *36th Annual IEEE Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[8] Sridhar Hannenhalli, Colombe Chappey, Eugene V. Koonin, and Pavel A. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics*, 30:299–311, 1995.

[9] J. Kececioglu and R. Ravi. Of mice and men: Evolutionary distances between genomes under translocations. In *6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613, 1995.

[10] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangements. *Algorithmica*, 13:180–210, 1995.

[11] Wen-Hsiung Li, Zhenglong Gu, Haidong Wang, and Anton Nekrutenko. Evolutionary analyses of the human genome. *Nature*, 409:847–849, 2001.

[12] Joseph H. Nadeau and Benjamin A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences*, 81(3):814–818, 1984.

[13] Jeffrey D. Palmer and Laura A. Herbon. Plant mitochondrial dna evolved rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 28:87–97, 1988.

[14] David Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.

[15] David Sankoff. Gene and genome duplication. *Current Opinion in Genetics & Development*, 11(6):681–684, 2001.

[16] David Sankoff and Lani Haque. Power boost for cluster tests. In *3rd RECOMB Comparative Genomics Statellite Workshop*, volume 3678 of *Lecture Notes in Computer Science*, pages 121–130, 2005.

[17] David Sankoff, Guillaume Leduc, Natalie Antoine, Bruno Paquin, B. Franz Lang, and Robert Cedergren. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *Proceedings of the National Academy of Sciences of the United States of America*, 89(14):6575–6579, 1992.

[18] Takeaki Uno and Mutsunori Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.

minimize:	$\sum_{x=0}^{1}\sum_{i=1}^{n_x-1}\sum_{j=i+1}^{n_x}\left(\sum_{k=1}^{n_{\overline{x}}-1}\sum_{l=k+1}^{n_{\overline{x}}}(j-i)\cdot d_x(i,j,k,l)-\sum_{p=i+1}^{j-1}e_x(i,j,p)\right)$		
subject to:			
Condition	**Label**	**Constraints**	**Indexes**
Exemplar model	CE	$\sum_{\substack{1\le i\le n_x \\ m_x(i,a)}} b_x(i)=1$	$a\in\mathcal{A}$ $0\le x\le 1$ $1\le i\le n_x$
Maximum model	CM	$\sum_{\substack{1\le i\le n_x \\ m_x(i,a)}} b_x(i)=\min\{occ_{G_0}(a),occ_{G_1}(a)\}$	$a\in\mathcal{A}$ $0\le x\le 1$ $1\le i\le n_x$
	$C1$	$\sum_{\substack{1\le k\le n_1 \\ m(i,k)}} a(i,k)=b_0(i)$	$1\le i\le n_0$
	$C2$	$\sum_{\substack{1\le i\le n_0 \\ m(i,k)}} a(i,k)=b_1(k)$	$1\le k\le n_1$
	$C3$	$c_x(i,j)+\sum_{i<p<j} b_x(p)\ge 1$	$0\le x\le 1$ $1\le i<j\le n_x$
	$C4$	$c_x(i,j)+b_x(p)\le 1$	$0\le x\le 1$ $1\le i<j\le n_x$ $i<p<j$
$m(i,k,x)$ $\wedge$ $m(j,l,x)$	$C5$	$a(i,k)^{-x}+a(j,l)^{-x}+c_{\overline{x}}(k,l)-d_x(i,j,k,l)\le 2$ $a(i,k)^{-x}-d_x(i,j,k,l)\ge 0$ $a(j,l)^{-x}-d_x(i,j,k,l)\ge 0$ $c_{\overline{x}}(k,l)-d_x(i,j,k,l)\ge 0$	$0\le x\le 1$ $1\le i<j\le n_x$ $1\le k<l\le n_{\overline{x}}$
$m(i,l,x)$ $\wedge$ $m(j,k,x)$	$C6$	$a(i,l)^{-x}+a(j,k)^{-x}+c_{\overline{x}}(k,l)-d_x(i,j,k,l)\le 2$ $a(i,l)^{-x}-d_x(i,j,k,l)\ge 0$ $a(j,k)^{-x}-d_x(i,j,k,l)\ge 0$ $c_{\overline{x}}(k,l)-d_x(i,j,k,l)\ge 0$	$0\le x\le 1$ $1\le i<j\le n_x$ $1\le k<l\le n_{\overline{x}}$
$\neg((m(i,k,x)$ $\wedge m(j,l,x))$ $\wedge$ $\neg(m(i,l,x)$ $\wedge m(j,k,x))$	$C7$	$d_x(i,j,k,l)=0$	$0\le x\le 1$ $1\le i<j\le n_x$ $1\le k<l\le n_{\overline{x}}$
	$C8$	$\sum_{1\le k<n_{\overline{x}}}\sum_{k<l\le n_{\overline{x}}} d_x(i,j,k,l)\le 1$	$0\le x\le 1$ $1\le i<j\le n_x$
	$C9$	$\sum_{1\le i<n_x}\sum_{i<j\le n_x} d_x(i,j,k,l)\le 1$	$0\le x\le 1$ $1\le k<l\le n_{\overline{x}}$
	$C10$	$\sum_{1\le k<n_{\overline{x}}}\sum_{k<l\le n_{\overline{x}}} d_x(i,j,k,l)+b_x(i)+b_x(j)+$ $+(1-b_x(p))-e_x(i,j,p)\le 3$ $e_x(i,j,p)-\sum_{1\le k<n_{\overline{x}}}\sum_{k<l\le n_{\overline{x}}} d_x(i,j,k,l)\le 0$ $e_x(i,j,p)-b_x(i)\le 0$ $e_x(i,j,p)-b_x(j)\le 0$ $e_x(i,j,p)-(1-b_x(p))\le 0$	$0\le x\le 1$ $1\le i<j\le n_x$ $i<p<j$

Table 1: PBO encoding of $OPT_e(G_0,G_1,SAD)$ and $OPT_m(G_0,G_1,SAD)$. Define $a(i,k)^{-0}=a(i,k)$ and $a(i,k)^{-1}=a(k,i)$ for all variables $a(i,k)$. It is assumed that genes g for which $occ_{G_0}(g)=0\vee occ_{G_1}(g)=0$ have been deleted from the input genomes G_0 and G_1. The predicate $m(i,k,x)$ is defined true if $G_x[i]=G_{\overline{x}}[k]$. The predicate $m_x(i,a)$ is defined true if $G_x[i]=a$.