

The EXEMPLAR BREAKPOINT DISTANCE for non-trivial genomes cannot be approximated

Guillaume Blin¹, Guillaume Fertin², Florian Sikora¹, and Stéphane Vialette¹

¹ Université Paris-Est, IGM-LabInfo - UMR CNRS 8049, France
{gblin,fsikora,vialette}@univ-mlv.fr

² LINA - UMR CNRS 6241 - Université de Nantes - France
guillaume.fertin@univ-nantes.fr

Abstract. A promising and active field of comparative genomics consists in comparing two genomes by establishing a one-to-one correspondence (i.e., a matching) between their genes. This correspondence is usually chosen in order to optimize a predefined measure. One such problem is the EXEMPLAR BREAKPOINT DISTANCE problem (or EBD, for short), which asks, given two genomes modeled by signed sequences of characters, to keep and match exactly one occurrence of each character in the two genomes (a process called *exemplarization*), so as to minimize the number of breakpoints of the resulting genomes.

Bryant [6] showed that EBD is **NP**-complete. In this paper, we close the study of the approximation of EBD by showing that no approximation factor can be derived for EBD considering non-trivial genomes – i.e. genomes that contain duplicated genes.

1 Introduction

Comparative genomics is a recent and active field of bioinformatics. One of the problems arising in this domain consists in comparing two species, and more specifically to look for conserved sets of genes between their genomes: a set of genes that is conserved in the same order during the evolution suggests that it participates to the same biological process. Finding conserved sets of genes in genomes is usually done by optimizing a given (dis)similarity measure. Many such measures have already been studied in the recent past: number of breakpoints, of adjacencies, of conserved intervals, of common intervals, etc. In this paper, we focus on the *number of breakpoints* between genomes.

All these measures are well-defined when genomes do not contain duplicates, and can usually be computed in polynomial time in this case. However, this assumption does not hold biologically. Moreover, by definition, the above mentioned measures do not apply when genes appear several times in a genome. A way to overcome this difficulty is to start from two genomes and to obtain a one-to-one correspondence (i.e., a matching) between their genes, in order to end up with a permutation on which the measure can then be computed. Among all possible matchings, the choice goes to the one that optimizes the studied measure. There exists several ways to achieve the desired matching. In this paper,

we are interested in the so-called *exemplar* model [9], where, for any gene family, exactly one gene is kept (and thus matched) in the genome. The motivation for this model is that the gene which is kept is assumed to be the ancestral gene, from which the other copies have derived.

Following the notations from Blin et al. [5], given an alphabet Σ of elements called *gene families*, a genome G on Σ is a sequence of signed elements of Σ , where the sign represents the DNA strand on which the gene lies. Each occurrence of an element of Σ in G is called a *gene*. For any gene family $g \in \Sigma$, we denote by $occ(G, g)$ the number of genes ($+g$ and $-g$) that appear in G . Let also $occ(G) = \max\{occ(G, g) \mid g \in \Sigma\}$.

For any genome G , a gene family g is said to be *trivial* if $occ(G, g) = 1$. Otherwise, g is said to be *non-trivial*. A gene belonging to a trivial (resp. non-trivial) family is said to be trivial (resp. non-trivial). A genome is called *trivial* if it only contains trivial genes, i.e. if it is a signed permutation. For convenience, we will use characters to represent each gene. In the following, given a genome G over an alphabet Σ , let $\chi(i, g, G)$ denote the i^{th} occurrence of character $g \in \Sigma$ in G (not taking signs into account). When there is no ambiguity, we will simply use $\chi(i, g)$. Moreover, we will refer to the i^{th} character of G as $G[i]$. We will also note $G[i] < G[j]$ for any $i < j$, that is when $G[i]$ appears before $G[j]$. For any gene g in G , we denote by \bar{g} the gene with opposite sign. As introduced by Chen et al. [7], a genome G is called an *s-span* genome if all the genes from the same gene family g are within distance at most s in G . For example, let $G = +a -d +c -b -d -a +e +b -b$. We have $occ(G, a) = 2$, $occ(G) = 3$, $\chi(2, b) = +b$ and G is a 5-span genome.

Given a trivial genome G , we say that gene $g = G[i]$ immediately precedes $g' = G[j]$ iff $j = i+1$. Given two trivial genomes G_1 and G_2 , if gene g immediately precedes gene g' in G_1 while neither (i) g immediately precedes g' nor (ii) \bar{g}' immediately precedes \bar{g} in G_2 , then they constitute a *breakpoint* in G_2 . The breakpoint distance between two trivial genomes G_1 and G_2 is then defined as the number of breakpoints in G_2 (we note that this distance is symmetric).

As previously mentioned, the breakpoint distance is not well defined for non-trivial genomes. The idea is then to establish a matching between genes of G_1 and genes of G_2 , in order to get back to a signed permutation on which the breakpoint distance can be computed. The *exemplar* model, introduced by Sankoff [9], is one of several ways to construct the matching, which consists in keeping, for each gene family, only one occurrence of its genes in G_1 and in G_2 . This raises the following problem, named EXEMPLAR BREAKPOINT DISTANCE problem, or EBD for short.

Given two genomes G_1 and G_2 , built over the same alphabet Σ , and an integer k , the EXEMPLAR BREAKPOINT DISTANCE Problem asks whether it is possible to establish an exemplar matching of G_1 and G_2 , such that the breakpoint distance between the resulting genomes is at most k .

Bryant [6] showed that EBD is **NP**-complete, even when one of the genomes is trivial, and the other has genes that appear at most twice in each genome. Concerning (in)approximability results, Angibaud et al. [2] proved that EBD is

APX-hard under the same conditions. Chen et al. [7] also showed that there exists no approximation algorithm for EBD, even when both genomes have genes that appear at most three times. However, this result does not completely close the question of EBD potential approximation. Indeed, whether EBD can be approximated on genomes that contain at most 2 copies of each gene remains unknown, and was actually raised as an open question by Chen et al. [7] and Angibaud et al. [4]. In this paper, we will answer this open question by showing that no approximation factor can be derived for EBD, even when considering genomes in which each gene occurs at most twice.

Chen et al. [7] also provided a logarithmic approximation ratio for the particular case in which one of the genomes is an s -span genome, with $s = O(\log m)$, $m = |\Sigma|$ being the number of gene families in the input genomes. It should also be noted that Nguyen et al. [8] designed a divide-and-conquer heuristic method in order to compute the EXEMPLAR BREAKPOINT DISTANCE while Angibaud et al. [3] proposed an exact method based on transforming the problem into a 0-1 linear programming problem.

In order to prove that EBD is not approximable, we will prove that a particular subproblem of EBD – called the ZERO EXEMPLAR BREAKPOINT DISTANCE problem (ZEBD for short) – is **NP**-complete. This decision problem asks whether there exists an exemplar matching of two genomes, such that the breakpoint distance between the resulting genomes is equal to zero. For sake of readability, for any $p \geq q \geq 1$, we will denote $\text{ZEBD}(p, q)$ the ZEBD problem in which $\text{occ}(G_1) = p$ and $\text{occ}(G_2) = q$. It is easy to see that $\text{ZEBD}(1, q)$ can be solved in linear time, for any $q \geq 1$. Chen et al. [7] showed that $\text{ZEBD}(3, 3)$ is **NP**-complete. Angibaud et al. [4] also showed that $\text{ZEBD}(2, q)$ is **NP**-complete, but with a value of q unbounded due to their reduction. Hence, the remaining unknown cases concern the complexity of $\text{ZEBD}(2, q)$, with fixed q . We will answer this question, by proving that $\text{ZEBD}(2, 2)$ (and thus, $\text{ZEBD}(2, q)$ for any $q \geq 2$) is **NP**-complete.

In this paper, we thus focus on ZEBD. More precisely, we first complete and close the study of the complexity of ZEBD, by proving that ZEBD is **NP**-complete, even when both genomes contain at most two occurrences of each gene. This result thus provides a full characterization of the polynomial and **NP**-complete cases for ZEBD, and also answers an open question raised by Chen et al. [7] and Angibaud et al. [4]. It also proves that no approximation factor can be derived for EBD, even when considering genomes in which each gene occurs at most twice; that is the simplest case considering non-trivial genomes. We then propose to overcome this difficulty by studying the fixed-parameter tractability of ZEBD which was also leaved as an open question in [7].

2 Inapproximability of EXEMPLAR BREAKPOINT DISTANCE

In this section, we prove that ZEBD(2, 2) is **NP**-complete. This result implies that EBD does not admit any approximation unless $\mathbf{P} = \mathbf{NP}$, even when both genomes contain at most two occurrences of each gene.

Theorem 1. *ZEBD(2, 2) is NP-complete.*

It is easy to see that ZEBD is in **NP**. In order to prove its **NP**-hardness, we propose a reduction from 3-SAT : let $V_n = \{x_1, x_2, \dots, x_n\}$ be a set of n boolean variables, and $C_q = \{c_1, c_2, \dots, c_q\}$ be a collection of q clauses, where each clause is a disjunction of three literals taken from V_n . The 3-SAT problem asks whether there exists an assignment of each variable of V_n such that each clause is satisfied. Let $\mathcal{I} = (C_q, V_n)$ be an instance of 3-SAT. From \mathcal{I} , we will build an instance $\mathcal{I}' = (G_1, G_2)$ of ZEBD, such that $occ(G_1) = occ(G_2) = 2$. In our construction, all genes carry a positive sign, which is omitted for sake of clarity. Moreover, for convenience, for any $1 \leq i \leq q$ and $1 \leq j \leq 3$, we let L_i^j denote the j^{th} literal of clause c_i in C_q . Moreover, for any $1 \leq k \leq n$, let N_{x_k} (resp. $N_{\bar{x}_k}$) denote the number of occurrences of x_k (resp. \bar{x}_k) in C_q . For each clause $c_i \in C_q$, $1 \leq i \leq q$, we first build a pair of sequences (U_i, V_i) , with $U_i = \mathbf{U}_i^1 d_i \mathbf{U}_i^2 d_i \mathbf{U}_i^3 t_i$ and $V_i = \mathbf{V}_i^1 d_i \mathbf{V}_i^2 d_i \mathbf{V}_i^3 t_i$, such that

$$\begin{array}{l} U_i = \overbrace{m_i^1 \mathbf{T}_{L_i^1} p_i^1 \mathbf{F}_{L_i^1} a_i m_i^1 d_i}^{U_i^1} \overbrace{a_i m_i^2 \mathbf{T}_{L_i^2} p_i^2 \mathbf{F}_{L_i^2} b_i m_i^2 d_i}^{U_i^2} \overbrace{b_i m_i^3 \mathbf{T}_{L_i^3} p_i^3 \mathbf{F}_{L_i^3} m_i^3 t_i}^{U_i^3} \\ V_i = \overbrace{p_i^1 \mathbf{F}_{L_i^1} m_i^1 \mathbf{T}_{L_i^1} p_i^1 a_i d_i}^{V_i^1} \overbrace{p_i^2 a_i \mathbf{F}_{L_i^2} m_i^2 \mathbf{T}_{L_i^2} p_i^2 b_i d_i}^{V_i^2} \overbrace{p_i^3 b_i \mathbf{F}_{L_i^3} m_i^3 \mathbf{T}_{L_i^3} p_i^3 t_i}^{V_i^3} \end{array}$$

Let us now define formally $T_{L_i^j}$ and $F_{L_i^j}$ for all $1 \leq i \leq q$ and $1 \leq j \leq 3$. Let $T_{L_i^j} = T_{L_i^j}^1 T_{L_i^j}^2$ (resp. $F_{L_i^j} = F_{L_i^j}^1 F_{L_i^j}^2$), defined as follows:

- if (i) L_i^j is the first occurrence of x_k or \bar{x}_k in C_q , and (ii) N_{x_k} and $N_{\bar{x}_k}$ are both strictly positive, then $T_{L_i^j}^1 = y_k^1$ and $F_{L_i^j}^1 = y_k^2$; otherwise, $T_{L_i^j}^1$ and $F_{L_i^j}^1$ are empty.
- if L_i^j is the l^{th} occurrence of x_k (resp. \bar{x}_k), let $p = l + 1$ if $l < N_{x_k}$ (resp. $l < N_{\bar{x}_k}$), and $p = 1$ otherwise. If $N_{x_k} > 1$ (resp. $N_{\bar{x}_k} > 1$), then $T_{L_i^j}^2 = x_k^l$ (resp. \bar{x}_k^l) and $F_{L_i^j}^2 = x_k^p$ (resp. \bar{x}_k^p); otherwise, $T_{L_i^j}^2$ and $F_{L_i^j}^2$ are empty.

Genomes G_1 and G_2 are then defined as $G_1 = U_1 U_2 \dots U_q$ and $G_2 = V_1 V_2 \dots V_q$. Clearly, this construction can be carried out in polynomial time, and it can also be seen that $occ(G_1) = occ(G_2) = 2$.

We now give an intuitive description of the different elements of this construction: each clause $c_i \in C_q$, $1 \leq i \leq q$, is represented by a pair (U_i, V_i) of sequences. Those sequences are both composed of three subsequences representing the literals of c_i . More precisely, the pair (U_i^j, V_i^j) represents a selection mechanism of

the j^{th} literal of c_i . For each variable $x_k \in V_n$, the set $\{x_k^1, x_k^2, x_k^3, \dots, x_k^{N_{x_k}}\}$ is used to propagate the selection of the literal x_k all over C_q , in order to make sure that if a literal satisfies a clause, then it satisfies all clauses where it appears. Similarly, the set $\{\overline{x_k^1}, \overline{x_k^2}, \overline{x_k^3}, \dots, \overline{x_k^{N_{\overline{x_k}}}}\}$ is a propagation mechanism for $\overline{x_k}$. Finally, for each variable $x_k \in V_n$ such that N_{x_k} and $N_{\overline{x_k}}$ are both strictly positive, the pair (y_k^1, y_k^2) in both G_1 and G_2 is a control mechanism that guarantees that a variable x_k cannot be TRUE and FALSE simultaneously.

Figure 1 illustrates an example of an instance (G_1, G_2) of ZEBD(2, 2) obtained from our construction, starting from $C_q = \{(x_1 \vee x_2 \vee x_3), (\overline{x_1} \vee x_2 \vee \overline{x_3}), (\overline{x_1} \vee \overline{x_2} \vee x_3), (x_1 \vee x_2 \vee x_4)\}$.

$$\begin{array}{c}
\begin{array}{ccc}
\overbrace{m_1^1 y_1^1 x_1^1 p_1^1 y_1^2 x_1^2 a_1 m_1^1}^{U_1^1} & \overbrace{d_1 a_1 m_1^2 y_2^1 x_2^1 p_1^2 y_2^2 x_2^2 b_1 m_1^2}^{U_1^2} & \overbrace{d_1 b_1 m_1^3 y_3^1 x_3^1 p_1^3 y_3^2 x_3^2 m_1^3}^{U_1^3} t_1 \\
\overbrace{p_1^1 y_1^2 x_1^2 m_1^1 y_1^1 x_1^1 p_1^1 a_1}^{V_1^1} & \overbrace{d_1 p_1^2 a_1 y_2^2 x_2^2 m_1^2 y_2^1 x_2^1 p_1^2 b_1}^{V_1^2} & \overbrace{d_1 p_1^3 b_1 y_3^2 x_3^2 m_1^3 y_3^1 x_3^1 p_1^3}^{V_1^3} t_1
\end{array} \\
\\
\begin{array}{ccc}
\overbrace{m_2^1 y_1^1 \overline{x_1^1} p_2^1 y_1^2 \overline{x_1^2} a_2 m_2^1}^{U_2^1} & \overbrace{d_2 a_2 m_2^2 x_2^2 p_2^2 x_2^3 b_2 m_2^2}^{U_2^2} & \overbrace{d_2 b_2 m_2^3 y_3^1 p_2^3 y_3^2 m_2^3}^{U_2^3} t_2 \\
\overbrace{p_2^1 y_1^2 \overline{x_1^2} m_2^1 y_1^1 \overline{x_1^1} p_2^1 a_2}^{V_2^1} & \overbrace{d_2 p_2^2 a_2 x_2^3 m_2^2 x_2^2 p_2^2 b_2}^{V_2^2} & \overbrace{d_2 p_2^3 b_2 y_3^2 m_2^3 y_3^1 p_2^3}^{V_2^3} t_2
\end{array} \\
\\
\begin{array}{ccc}
\overbrace{m_3^1 \overline{x_1^2} p_3^1 \overline{x_1^1} a_3 m_3^1}^{U_3^1} & \overbrace{d_3 a_3 m_3^2 y_2^1 p_3^2 y_2^2 b_3 m_3^2}^{U_3^2} & \overbrace{d_3 b_3 m_3^3 x_3^2 p_3^3 x_3^1 m_3^3}^{U_3^3} t_3 \\
\overbrace{p_3^1 \overline{x_1^1} m_3^1 \overline{x_1^2} p_3^1 a_3}^{V_3^1} & \overbrace{d_3 p_3^2 a_3 y_2^2 m_3^2 y_2^1 p_3^2 b_3}^{V_3^2} & \overbrace{d_3 p_3^3 b_3 x_3^1 m_3^3 x_3^2 p_3^3}^{V_3^3} t_3
\end{array} \\
\\
\begin{array}{ccc}
\overbrace{m_4^1 x_1^2 p_4^1 x_1^1 a_4 m_4^1}^{U_4^1} & \overbrace{d_4 a_4 m_4^2 x_2^3 p_4^2 x_2^1 b_4 m_4^2}^{U_4^2} & \overbrace{d_4 b_4 m_4^3 p_4^3 m_4^3}^{U_4^3} t_4 \\
\overbrace{p_4^1 x_1^1 m_4^1 x_1^2 p_4^1 a_4}^{V_4^1} & \overbrace{d_4 p_4^2 a_4 x_2^1 m_4^2 x_2^3 p_4^2 b_4}^{V_4^2} & \overbrace{d_4 p_4^3 b_4 m_4^3 p_4^3}^{V_4^3} t_4
\end{array}
\end{array}$$

Fig. 1. The instance (G_1, G_2) of ZEBD(2, 2) obtained starting from $C_q = \{(x_1 \vee x_2 \vee x_3), (\overline{x_1} \vee x_2 \vee \overline{x_3}), (\overline{x_1} \vee \overline{x_2} \vee x_3), (x_1 \vee x_2 \vee x_4)\}$.

In the following, let us denote by S_k (resp. S_d) the set of characters that are kept (resp. deleted) in an exemplarization of G_1 and G_2 , in a given solution for ZEBD. By definition, exactly one occurrence of each gene family must be kept. We also note that by construction, since, for $1 \leq i \leq q$, there is only one

occurrence of t_i in G_1 and in G_2 , characters of U_i may only be matched with characters of V_i . Moreover, for a given $1 \leq i \leq q$ and a given $1 \leq j \leq 3$, in U_i^j , $\chi(1, m_i^j) < p_i^j < \chi(2, m_i^j)$, whereas, in V_i^j , $\chi(1, p_i^j) < m_i^j < \chi(2, p_i^j)$. Those properties induce the following lemma.

Lemma 1. *In any solution for ZEBD on (G_1, G_2) , for any $1 \leq i \leq q$ and $1 \leq j \leq 3$, either (a) $\{\chi(1, m_i^j, U_i^j), \chi(2, p_i^j, V_i^j)\} \subseteq S_k$ or (b) $\{\chi(2, m_i^j, U_i^j), \chi(1, p_i^j, V_i^j)\} \subseteq S_k$.*

Lemma 2. *In any solution for ZEBD on (G_1, G_2) , for any $1 \leq i \leq q$ and $1 \leq j \leq 3$, at least one of $\chi(1, m_i^1, U_i^1)$, $\chi(1, m_i^2, U_i^2)$, $\chi(1, m_i^3, U_i^3)$ belongs to S_k .*

Proof. By contradiction, let us suppose that none of $\chi(1, m_i^1, U_i^1)$, $\chi(1, m_i^2, U_i^2)$, $\chi(1, m_i^3, U_i^3)$ belongs to S_k . Then, by Lemma 1, $\{\chi(2, m_i^j, U_i^j), \chi(1, p_i^j, V_i^j)\} \subseteq S_k$ for all $1 \leq j \leq 3$. Since in U_i , $\chi(1, a_i) < \chi(2, m_i^1, U_i^1) < \chi(1, d_i) < \chi(2, a_i) < p_i^2$, whereas in V_i , $m_i^1 < \chi(1, a_i) < \chi(1, d_i) < \chi(1, p_i^2, V_i^2) < \chi(2, a_i) < m_i^2$, we conclude that $\{\chi(1, a_i, U_i), \chi(2, a_i, V_i)\} \subseteq S_d$. Therefore, $\{\chi(1, d_i, U_i), \chi(1, d_i, V_i)\} \subseteq S_d$, whereas $\{\chi(2, a_i, U_i), \chi(1, a_i, V_i)\} \subseteq S_k$.

Moreover, since in U_i , $\chi(1, b_i) < \chi(2, m_i^2, U_i^2) < \chi(2, d_i) < \chi(2, b_i) < p_i^3$, whereas in V_i , $m_i^2 < \chi(1, b_i) < \chi(2, d_i) < \chi(1, p_i^3, U_i^3) < \chi(2, b_i) < m_i^3$, we conclude that $\{\chi(1, b_i, U_i), \chi(2, b_i, V_i)\} \subseteq S_d$. Thus, $\{\chi(2, d_i, U_i), \chi(2, d_i, V_i)\} \subseteq S_d$, whereas $\{\chi(2, b_i, U_i), \chi(1, b_i, V_i)\} \subseteq S_k$. Consequently, none of the occurrences of d_i can be kept in the exemplarization, a contradiction. \square

Lemma 3. *Let $I = \{(i_1, j_1), (i_2, j_2), \dots, (i_p, j_p)\}$ such that $\forall 1 \leq m \neq n \leq p$, $L_{i_m}^{j_m} = L_{i_n}^{j_n}$. Then either (a) $\{\chi(1, m_{i_m}^{j_m}, U_{i_m}^{j_m}), \chi(1, m_{i_n}^{j_n}, U_{i_n}^{j_n})\} \subseteq S_k$, or (b) $\{\chi(2, m_{i_m}^{j_m}, U_{i_m}^{j_m}), \chi(2, m_{i_n}^{j_n}, U_{i_n}^{j_n})\} \subseteq S_k$.*

Proof. Let us first suppose that $L_{i_1}^{j_1} = x_k$. Then, by construction, $x_k^r < p_{i_r}^{j_r} < x_k^{r+1}$ in $U_{i_r}^{j_r}$ and $x_k^{r+1} < m_{i_r}^{j_r} < x_k^r$ in $V_{i_r}^{j_r}$, for any $1 \leq r < p$ and $x_k^p < p_{i_p}^{j_p} < x_k^1$ in $U_{i_p}^{j_p}$ and $x_k^1 < m_{i_p}^{j_p} < x_k^p$ in $V_{i_p}^{j_p}$. If $\chi(1, m_{i_r}^{j_r}, U_{i_r}^{j_r}) \in S_k$, for a given $(i_r, j_r) \in I$ such that $r < p$, x_k^{r+1} in $U_{i_r}^{j_r}$ must be deleted. Therefore, the two occurrences of x_k^{r+1} in $U_{i_{r+1}}^{j_{r+1}}$ and $V_{i_{r+1}}^{j_{r+1}}$ must be kept. Consequently, $\chi(1, m_{i_{r+1}}^{j_{r+1}}, U_{i_{r+1}}^{j_{r+1}}) \in S_k$. By induction on r , we have $\chi(1, m_{i_p}^{j_p}, U_{i_p}^{j_p}) \in S_k$. This implies that x_k^1 in $U_{i_p}^{j_p}$ must be deleted, which in turn means that x_k^1 in $U_{i_1}^{j_1}$ must be kept, and thus $\chi(1, m_{i_1}^{j_1}, U_{i_1}^{j_1}) \in S_k$. The induction can then be continued from $U_{i_1}^{j_1}$, and we conclude that $\chi(1, m_{i_n}^{j_n}, U_{i_n}^{j_n}) \in S_k$ for any $1 \leq i \leq p$. This proves case (a) of the above lemma, when $L_{i_1}^{j_1} = x_k$.

Moreover, if $\chi(1, m_{i_r}^{j_r}, U_{i_r}^{j_r}) \in S_d$, for a given $(i_r, j_r) \in I$ such that $r > 1$, then $\chi(2, m_{i_r}^{j_r}, U_{i_r}^{j_r}) \in S_k$ and x_k^r in $U_{i_r}^{j_r}$ must be deleted. Therefore, the two occurrences of x_k^r in $U_{i_{r-1}}^{j_{r-1}}$ and $V_{i_{r-1}}^{j_{r-1}}$ must be kept. Consequently, $\chi(2, m_{i_{r-1}}^{j_{r-1}}, U_{i_{r-1}}^{j_{r-1}}) \in S_k$. By induction on r , $\chi(2, m_{i_1}^{j_1}, U_{i_1}^{j_1}) \in S_k$ and, thus, this implies that x_k^1 in $U_{i_1}^{j_1}$ must be deleted, which in turn means that x_k^1 in $U_{i_p}^{j_p}$ must be kept, and thus

$\chi(2, m_{i_p}^{j_p}, U_{i_p}^{j_p}) \in S_k$. The induction can then be continued from $U_{i_p}^{j_p}$, and we conclude that $\chi(2, m_{i_n}^{j_n}, U_{i_n}^{j_n}) \in S_k$ for any $1 \leq i \leq p$. This proves case (b) of the above lemma, when $L_{i_1}^{j_1} = x_k$.

By a similar reasoning, it is possible to prove that the same holds when $L_i^j = \overline{x_p}$. \square

Lemma 4. $\forall (i, j), (i', j')$ such that (1) $L_i^j = \overline{L_{i'}^{j'}}$ and (2) L_i^j and $\overline{L_{i'}^{j'}}$ are the first occurrences of the corresponding variable, only one of $\chi(1, m_i^j, U_i^j)$ and $\chi(1, m_{i'}^{j'}, U_{i'}^{j'})$ may be kept.

Proof. Let $L_i^j = \overline{L_{i'}^{j'}} = x_k$ and suppose L_i^j (resp. $L_{i'}^{j'}$) is the first occurrence of x_k (resp. $\overline{x_k}$). If $\chi(1, m_i^j, U_i^j)$ is kept, then y_k^2 in U_i^j (resp. $V_{i'}^{j'}$) must be deleted, and therefore of y_k^2 in $U_{i'}^{j'}$ (resp. $V_{i'}^{j'}$) must be kept. In that case, it can be seen that $\chi(1, m_{i'}^{j'}, U_{i'}^{j'})$ must be deleted. The proof is similar if we choose to keep $\chi(1, m_{i'}^{j'}, U_{i'}^{j'})$. \square

Thanks to the four above lemmas, we can prove the following theorem (the proof is omitted due to space constraints, and is available in Appendix).

Theorem 2. *Let \mathcal{I} be an instance of 3-SAT, and let $\mathcal{I}' = (G_1, G_2)$ be the instance of ZEBD(2,2) constructed from \mathcal{I} . There exists a truth assignment that satisfies each clause in \mathcal{I} iff there exists a zero breakpoint distance exemplarization in \mathcal{I}' .*

Altogether, this proves that ZEBD(2,2) is NP-complete.

3 Exponential-time algorithms for ZEBD

It can be easily seen that, for two genomes G_1 and $G_2 = g_1 g_2 \dots g_n$, if ZEBD is answered positively, the induced exemplarization is either (1) a common subsequence of G_1 and G_2 or (2) a common subsequence between G_1 and $\overleftarrow{G_2} = \overleftarrow{g_n} \dots \overleftarrow{g_2} \overleftarrow{g_1}$. Therefore, any algorithm that answers ZEBD should check both cases. For simplicity, we will only discuss case (1) in this section. Checking case (2) just requires to run the same algorithm on $(G_1, \overleftarrow{G_2})$, instead of (G_1, G_2) , which does not change the complexity.

Let G_1 and G_2 be two genomes defined over a set of m gene families, and such that $occ(G_1) = k_1$ and $occ(G_2) = k_2$. A brute-force algorithm for answering to ZEBD of G_1 and G_2 consists in computing all the possible exemplarizations of G_1 and G_2 , and then determining whether one of them leads to a zero breakpoint distance. Since for any gene family there are at most k_i occurrences in G_i , $1 \leq i \leq 2$, there are at most $(k_1)^m$ (resp. $(k_2)^m$) exemplarizations of G_1 (resp. G_2). Moreover, given two exemplar genomes (each of length m), one can check in $O(m)$ whether their breakpoint distance is equal to zero. Therefore, on the whole the time complexity of the brute force algorithm is $O(m \cdot (k_1 \cdot k_2)^m)$ time. In the following, we present two fixed parameter algorithms, which give more feasible solutions.

Theorem 3. *There exists an $O(m2^m)$ algorithm for solving ZEBD, where m is the number of gene families of the input genomes.*

Proof. The key idea is to decrease the time complexity of the brute-force algorithm by using a color-coding like method (see [1]). Color-coding is a technique to design fixed-parameter algorithms for several **NP**-complete subgraph isomorphism problems. This technique is based on a random coloring of each vertex of the input graph, using a small set of colors, and looking for a *colorful* path in the graph, that is a path whose vertices carry different colors and all colors are used.

Given two genomes G_1 and G_2 over a set Σ of m gene families, we build a directed graph (V, A) defined as follows. For each pair $(G_1[i], G_2[j])$ of genes of the same gene family that carry the same sign, add a vertex (i, j) to V . For all $\{(i, j), (p, q)\} \in V^2$ such that $i < p$ and $j < q$, add an edge from (i, j) to (p, q) in A . Finally, let $C : \Sigma \rightarrow \{c_1, c_2, \dots, c_m\}$ be a function that assigns a unique color to each gene family. For each vertex $(i, j) \in V$, we assign to (i, j) the color $C(f(G_1[i]))$, where $f(G[i])$ denotes the gene family corresponding to $G[i]$. An illustration is given in Figure 2.

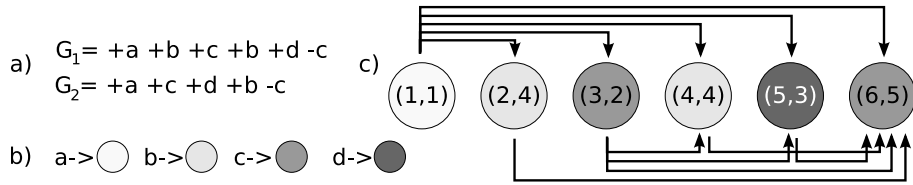


Fig. 2. a) Two genomes G_1 and G_2 ; b) A coloring function C ; c) The corresponding graph (V, A)

It can be seen that looking for a colorful path of length m in the graph (V, A) is equivalent to finding an zero breakpoint distance exemplarization of G_1 and G_2 , since (i) genes belonging to the same family carry the same color, and (ii) the order is preserved by construction of A . It can be easily seen that there exists a dynamic programming algorithm in $O(m2^m)$ to answer that question. Indeed, we only need to maintain the set of colors already selected in the current path, instead of the current selected vertices of the graph. Therefore, ZEBD is fixed parameter tractable with respect to the number m of gene families. \square

We now propose a second fixed-parameter algorithm for solving ZEBD, defined as follows. As previously, we transform the ZEBD problem into the problem of finding a path in a directed acyclic graph (or DAG, for short). But instead of having a vertex for each possible match between two genes of G_1 and G_2 , in this DAG, a vertex will represent a common subsequence between the two genomes (i.e. a sequence formed from the original sequence by deleting some of the elements without changing the relative order and signs of the remaining elements).

The complexity of this algorithm will stem from the construction of this graph, which is, as we will see, exponential on the span s of the input genomes, rather than on m .

Suppose, wlog, $|G_1| \leq |G_2|$. The algorithm is based on two functions : *CommonSubsequenceSet* (cf. Figure 3) and *BuildGraph* (cf. Figure 4). The function *CommonSubsequenceSet* consists in (1) computing – for all the common subsequences between non overlapping segments of G_1 of size s (i.e. $G_1[i..i + s - 1]$ for $i = \{1, s + 1, 2s + 1, \dots\}$) and G_2 – the starting and ending positions of the common subsequences in G_2 ; and (2) add a vertex represented as a triplet $(\alpha_j^i, s_j^i, e_j^i)$ (formally defined in Figure 3) for each of those common subsequences to a set V . This set will be returned by the function.

```

1 Function CommonSubsequenceSet ( $G_1, G_2$ ) {
2    $V = \emptyset; j = 1;$ 
3   for ( $i = 1; i \leq |G_1|; i = i + s$ ) do
4     foreach common subsequence  $\alpha_j^i$  between  $G_1[i..i + s - 1]$  and  $G_2$  do
5        $V = V \cup \{v_j^i\}$  such that  $v_j^i = (\alpha_j^i, s_j^i, e_j^i)$  where  $s_j^i$  (resp.  $e_j^i$ ) is the
        starting (resp. ending) position of  $\alpha_j^i$  in  $G_2$ ;
6     end
7      $j++;$ 
8 end
9 return  $V$  }

```

Fig. 3. Function CommonSubsequenceSet over two s -span genomes G_1 and G_2 .

Function *BuildGraph* consists in building a DAG $G = (V, A)$ from two s -span genomes G_1 and G_2 . The set of vertices V is obtained by calling the CommonSubsequenceSet function on G_1 and G_2 . *BuildGraph* then considers each pair of vertices $(v_j^i, v_{j+1}^{i'})$ and $(v_j^i, v_{j+2}^{i'})$, and adds an edge to A iff the corresponding common subsequences are compatible, i.e. if they have no gene of the same family in common (exemplar solution) and do not overlap. In the resulting graph (as illustrated in Figure 5), a path is an exemplarization of both G_1 and G_2 . In such a graph, if one decomposes each vertex v_j^i into a path of length $|\alpha_j^i|$ then, as we will show, the existence of a path of length m induces that there exists a zero breakpoint distance exemplarization of G_1 and G_2 .

Before proving the correctness and complexity of the above algorithm, let us prove an interesting property on the set V returned by *CommonSubsequenceSet*. In the following, δ_j will refer to the set $\{v_j^i | v_j^i \in V\}$.

Lemma 5. *Given two s -span genomes G_1 and G_2 , the function CommonSubsequenceSet, run on (G_1, G_2) , returns a set V of size less than or equal to $n2^s s$, where $n = |G_1|$.*

Proof. Given any segment of size s of G_1 , there are at most 2^s different subsequences. Moreover, for each such subsequence α_j^i , there are at most s positions in

```

1 Function BuildGraph ( $G_1, G_2$ ) {
2  $V = \text{CommonSubsequenceSet}(G_1, G_2)$ ;
3 foreach  $(v_j^i, v_{j+1}^{i'}) \in V^2$  do
4   if  $e_j^i < s_{j+1}^{i'}$  then
5     //the order is respected;
6     if  $\alpha_j^i$  and  $\alpha_{j+1}^{i'}$  have no gene of the same family in common then
7       //the concatenation is exemplar;
8        $A = A \cup \{(v_j^i, v_{j+1}^{i'})\}$ ;
9     end
10  end
11 end
12 foreach  $(v_j^i, v_{j+2}^{i'}) \in V^2$  do
13   if  $e_j^i < s_{j+2}^{i'}$  then
14     //by construction, the concatenation is exemplar;
15      $A = A \cup \{(v_j^i, v_{j+2}^{i'})\}$ ;
16   end
17 end
18 return  $G = (V, A)$  }

```

Fig. 4. Function BuildGraph over two s -span genomes G_1 and G_2 .

G_2 for the first (resp. the last) gene of α_j^i ; namely $\alpha_j^i[1]$ (resp. $\alpha_j^i[|\alpha_j^i|]$). Since in the function *CommonSubsequenceSet*, a vertex is added to V for each $(\alpha_j^i, s_j^i, e_j^i)$, V is of cardinality at most $\frac{|G_1|}{s} 2^s s^2$. \square

By construction, given any sequence of vertices along a path in the DAG obtained by *BuildGraph*(G_1, G_2), the order of the vertices in the path is similar to the one of the corresponding genes in both G_1 and G_2 . Moreover, by definition, for any j the set of genes involved in any common subsequence of δ_j is disjoint from the one for $\delta_{j'}$ with $j' \geq j + 2$. Therefore, a path in the DAG corresponds to an exemplarization of both G_1 and G_2 . Hence, for any s -span genome G_1 and G_2 , ZEBD is positively answered iff there exists a path of length m in the DAG obtained by *BuildGraph*(G_1, G_2).

Let us now analyze the time complexity of our algorithm. The function *CommonSubsequenceSet*(G_1, G_2) has a worst time complexity in $O(n2^s s)$, where n is the length of G_1 . Indeed, for each value of i (which is bounded by $\frac{n}{s}$), according to Lemma 5 there are at most 2^s different subsequences in G_1 and s^2 pairs (s_j^i, e_j^i) for any such given subsequence in G_2 . Moreover, any set δ_j , $1 \leq j \leq \frac{n}{s}$, is of size at most $2^s s^2$.

Function *BuildGraph*(G_1, G_2) has a worst time complexity in $O(n2^{2s} s^3)$. Indeed, there are less than $(2^s s^2)^2$ pairs $(v_j^i, v_{j+1}^{i'})$ in V , each of size at most $\frac{n}{s}$.

Finally, finding a longest path in the resulting DAG from *BuildGraph*(G_1, G_2) may be done polynomially in the size of the graph. On the whole, the algorithm has a time complexity in $O(n2^{2s} s^3)$.

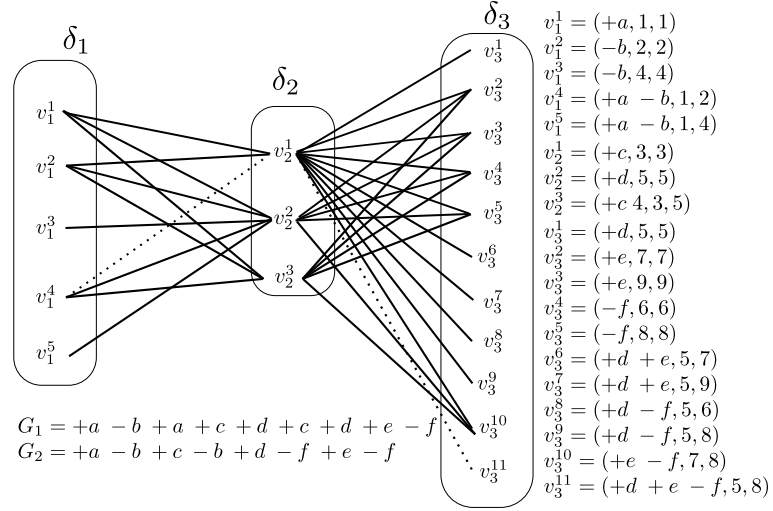


Fig. 5. Resulting graph from $BuildGraph(G_1, G_2)$ with $G_1 = +a - b + a + c + d + c + d + e - f$ and $G_2 = +a - b + c - b + d - f + e - f$. The dotted path corresponds to a positive solution for ZEBD. Links between δ_1 and δ_3 have not been drawn. All arcs are drawn without arrow, but should be understood from left to right.

Altogether, the above results lead to the following theorem.

Theorem 4. *There exists an $O(n2^{2s}s^3)$ algorithm for solving ZEBD, where n is the length of the shortest input genome, and s is the span of the input genomes.*

4 Conclusion

In this paper, we have given several results concerning the EXEMPLAR BREAKPOINT DISTANCE and ZERO EXEMPLAR BREAKPOINT DISTANCE problems. We first proved that that EBD cannot be approximated *at all* as soon as both genomes contain duplicates; this was done by showing that ZEBD(2,2) is **NP**-complete. This last result fills the remaining gaps concerning the knowledge of the complexity landscape of ZEBD. In particular, this answers an open question from Chen et al. [7] and Angibaud et al. [4]. We have also provided two fixed-parameter algorithms for ZEBD: one parameterized by the number of gene families, and the other by their span. Two questions remain open:

(1) since ZEBD(1, q), $q \geq 1$, is polynomial, there may exist approximation algorithms for EBD(1, q) (we recall that EBD(1, q) is **APX**-hard [2]). For instance, is it possible to approximate EBD(1, q) within a *constant* ratio ?

(2) as mentioned by Chen et al. [7], for problems whose optimal value could be equal to zero, it is sometimes better to look for so-called *weak approximations*. The natural question is thus the following: what can be said concerning weak

approximations for $EBD(p, q)$? Chen et al. [7] gave a negative result in the general case but restricting our study to particular values of p and q could lead to positive results.

References

1. N. Alon, R. Yuster, and U. Zwick. Color coding. *Journal of the ACM*, 42(4):844–856, 1995.
2. S. Angibaud, G. Fertin, and I. Rusu. On the approximability of comparing genomes with duplicates. In *Proc. 2nd Workshop on Algorithms and Computation (WALCOM 2008)*, volume 4921 of *Lecture Notes in Computer Science*, pages 34–45. Springer, 2008.
3. S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In *Proc. 5th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG)*, volume 4751 of *Lecture Notes in Bioinformatics*, pages 16–29. Springer, 2007.
4. S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. On the approximability of comparing genomes with duplicates. *Journal of Graph Algorithms and Applications*, 2008. Extended version of [2]. Submitted. Available at <http://www.arxiv.org/abs/0806.1103>.
5. G. Blin, C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Comparing genomes with duplications: a computational complexity point of view. *ACM/IEEE Trans. Computational Biology and Bioinformatics*, 14(4):523–534, 2007.
6. D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 207–212. Kluwer Academic Publisher, 2000.
7. Z. Chen, B. Fu, and B. Zhu. The approximability of the exemplar breakpoint distance problem. In *2nd International Conference on Algorithmic Aspects in Information and Management (AAIM)*, volume 4041 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2006.
8. C. Thach Nguyen, Y. C. Tay, and Louxin Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, 21:2171–2176, 2005.
9. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.

5 Appendix

Theorem 5. *Let \mathcal{I} be an instance of 3-SAT, and let $\mathcal{I}' = (G_1, G_2)$ be the instance of ZEBD(2,2) constructed from \mathcal{I} . There exists a truth assignment that satisfies each clause in \mathcal{I} iff there exists a zero breakpoint distance exemplarization in \mathcal{I}' .*

Proof. (\Leftarrow) Suppose we have a solution (S_k, S_d) for ZEBD(G_1, G_2). By Lemma 2, for any $1 \leq i \leq q$ and $1 \leq j \leq 3$, at least one of $\chi(1, m_i^1, U_i^1), \chi(1, m_i^2, U_i^2), \chi(1, m_i^3, U_i^3)$ belongs to S_k . Let us define the following truth assignment for V_n : $\forall 1 \leq i \leq q$, if $\chi(1, m_i^j, U_i^j) \in S_k$ then L_i^j is set to TRUE. Otherwise, L_i^j is set to FALSE. Let us prove that this assignment is a solution of 3-SAT. First, by Lemma 4, if $L_i^j = \overline{L_{i'}^j}$, then L_i^j and $L_{i'}^j$ cannot both be set to the same value (TRUE or FALSE): in other words, each literal is set a unique value TRUE or FALSE. Moreover, by Lemma 2, for any $1 \leq i \leq q$, at least one of the literals of clause c_i satisfies it. Altogether, this assignment is a positive solution for 3-SAT.

(\Rightarrow) Given a solution of 3-SAT, $\forall c_i \in C_q$, if L_i^j is TRUE (that is, L_i^j is one of the literals satisfying c_i), we let S_k contain $\{\chi(1, m_i^j), T_{L_i^j}, p_i^j\}$ of U_i^j and $\{m_i^j, T_{L_i^j}, \chi(2, p_i^j)\}$ of V_i^j ; otherwise, if L_i^j is FALSE, we let S_k contain $\{p_i^j, F_{L_i^j}, \chi(2, m_i^j)\}$ of U_i^j and $\{\chi(1, p_i^j), F_{L_i^j}, m_i^j\}$ of V_i^j . Moreover, $\forall c_i \in C_q$: (1) if L_i^1 is TRUE and L_i^2 is FALSE, then $\{\chi(1, a_i, U_i), \chi(1, a_i, V_i), \chi(1, d_i, U_i), \chi(1, d_i, V_i), \chi(2, b_i, U_i), \chi(1, b_i, V_i)\} \subseteq S_k$; (2) if L_i^2 is TRUE, then $\{\chi(2, a_i, U_i), \chi(2, a_i, V_i), \chi(1, d_i, U_i), \chi(1, d_i, V_i), \chi(1, b_i, U_i), \chi(1, b_i, V_i)\} \subseteq S_k$; (3) if L_i^3 is TRUE and L_i^1 and L_i^2 are FALSE, then $\{\chi(2, a_i, U_i), \chi(1, a_i, V_i), \chi(2, d_i, U_i), \chi(2, d_i, V_i), \chi(2, b_i, U_i), \chi(2, b_i, V_i)\} \subseteq S_k$; (4) $\{\chi(1, t_i, U_i), \chi(1, t_i, V_i)\} \subseteq S_k$.

Let us now prove that the corresponding set S_k is a solution for ZEBD. According to proof of Lemma 3, $\forall x_k \in V_n$ such that x_k (resp. $\overline{x_k}$) is TRUE, all $T_{L_i^j}$ such that $L_i^j = x_k$ (resp. $L_i^j = \overline{x_k}$) are kept. Therefore, for any $1 \leq l \leq N_{x_k}$ (resp. $1 \leq l \leq N_{\overline{x_k}}$), exactly one occurrence of x_k^l (resp. $\overline{x_k}^l$) is kept. Moreover, by construction, $T_{L_i^j}^1$ and $F_{L_i^j}^1$ are both empty if $N_{L_i^j} = 0$ or $N_{\overline{L_i^j}} = 0$; otherwise, $\forall x_k \in V_n$ such that $N_{x_k} > 0$ and $N_{\overline{x_k}} > 0$, there exist exactly two occurrences of y_k^1 (say, in $T_{L_i^j}^1$ and $T_{L_{i'}^j}^1$) and y_k^2 (in $F_{L_i^j}^1$ and $F_{L_{i'}^j}^1$) in both G_1 and G_2 . Since any variable of V_n is either TRUE or FALSE, exactly one of $(T_{L_i^j}^1, T_{L_{i'}^j}^1)$ and one of $(F_{L_i^j}^1, F_{L_{i'}^j}^1)$ has been kept. Therefore, exactly one occurrence of each y_k^1 and y_k^2 has been kept. Finally, it is easy to check that one of each remaining characters is kept, which shows that there exists a zero breakpoint distance exemplarization of G_1 and G_2 . \square