# Duplication-Loss Genome Alignment: Complexity and Algorithm

Billel Benzaid[2], Riccardo Dondi[1], and Nadia El-Mabrouk[2]

[1] Dipartimento di Scienze Umane e Sociali, Università degli Studi di Bergamo, Via Donizetti 3, 24129 Bergamo - Italy
[2] Départment d'Informatique et Recherche Opérationnelle, Université de Montréal, Pavillon André-Aisenstadt, CP 6128 succ Centre-Ville, Montréal, Québec, Canada
benzaidb@iro.umontreal.ca, riccardo.dondi@unibg.it, mabrouk@iro.umontreal.ca

**Abstract.** Recently, an Alignment approach for the comparison of two genomes, based on an evolutionary model restricted to Duplications and Losses, has been presented. An exact linear programming algorithm has been developed and successfully applied to the Transfer RNA (tRNA) repertoire in Bacteria, leading to interesting observation on tRNA shift of identity. Here, we explore a direct dynamic programming approach for the Duplication-Loss Alignment of two genomes, which proceeds in two steps: (1) (The Dynamic Programming step) Outputs a best candidate alignment between the two genomes and (2) (Minimum Label Alignment problem) Finds an evolutionary scenario of minimum duplication-loss cost that is in agreement with the alignment. We show that the Minimum Label Alignment is APX-hard, even if the number of occurrences of a gene inside a genome is bounded by 5. We then develop a heuristic which is a thousands of times faster than the linear programming algorithm and exhibits a high degree of accuracy on simulated datasets. The heuristic has been implemented in JAVA and is available on request.

**Keywords:** Comparative Genomics, Genome Alignment, Duplication, Computational Complexity, Algorithms, Dynamic Programming.

## 1 Introduction

The abundance of completely sequenced and annotated genomes present in public repositories has reinforced the role of genome comparison as the primary approach to gain insight in the evolution of genomes and gene families. When comparing complete genomes, the mutations of interest are macro-evolutionary events such as rearrangements (inversions, transpositions, translocations etc.) and content modifying operations (duplications, losses, horizontal gene transfer etc.) affecting the overall organization of genes, rather than micro-evolutionary events, such as single nucleotide substitutions, affecting single gene sequences. In other words, genomes are modeled as strings of characters over an alphabet $\Sigma$ of gene families. The case of strings being permutations (i.e. each gene family with a

single representative in each genome) has been largely considered by the genome rearrangement community for pairwise comparison (for example [3, 8, 10, 13]) or multiple comparison in a phylogenetic framework (for example [4, 12, 14, 15]). An extra degree of difficulty is introduced in the case of strings containing multiple gene copies. Most of the methods used for comparing two genomes with duplicates (reviewed in [6, 7, 9]) rely mainly on rearrangement events. Contrariwise, we considered in [11], an evolutionary model restricted to content-modifying operations, and more specifically to duplications and losses. We showed that this model is required to study the evolution of certain gene families, such as Transfer RNAs (tRNAs). From a combinatorial point of view, the main consequence of ignoring rearrangements is the fact that gene organization is preserved, which allows reformulating the comparison of two genomes as a Duplication-Loss Alignment problem: find an alignment minimizing the cost of duplications and losses. As in [11], we consider in this paper the cost of an alignment to be the number of underlying segmental duplications (duplication of a string of adjacent genes) and single losses (loss of a single gene). Although alignments are *a priori* simpler to handle than rearrangements, we showed in [11] that a direct approach based on dynamic programming leads, at best, to an efficient heuristic, and we rather developed an exact pseudo-boolean linear programming algorithm. This algorithm is however exponential in the worst case, preventing from being applicable to relatively large genomes (more than 200 genes). The problem has actually been recently shown to be NP-hard [5].

In this paper, we further explore the suggested direct dynamic programming approach, which is in two steps: (1) (The Dynamic Programming step) Output a best candidate alignment between the two genomes and (2) (Minimum Label Alignment problem) Find an evolutionary scenario of minimum duplication-loss cost that is in agreement with the alignment. The way to solve the Minimum Label Alignment problem, as well as its complexity, were left open in [11]. We show in Section 4 that it is APX-hard, even if the number of occurrences of a gene inside a genome is bounded by 5. We then, in Section 5, present a heuristic for the Duplication-Loss Alignment problem, which is a thousands of times faster than the linear programming algorithm and exhibits optimal or near-optimal results on simulated datasets obtained with an evolutionary model consistent with that observed for the tRNA repertoire in *Bacillus*.

We begin by introducing the notations and alignment problems in Section 2, and the dynamic programming approach in Section 3.

## 2 Preliminaries

*Strings:* We consider single chromosomal (circular or linear) genomes, represented as gene orders with duplicates. More precisely, given an alphabet $\Sigma$, each character representing a specific gene family, a *genome* or *string* is a sequence of characters from $\Sigma$, where each character may appear many times. As the content-modifying operations considered in this paper do not change gene orientation, we can assume w.l.o.g. that genes are unsigned. For example, $X$ in

Figure 1 is a genome on the alphabet $\Sigma = \{a, b, c, d, e, f\}$, with four gene copies from the gene family identified by $b$, and a single copy from family $f$.

Given a string $Z$, we denote by $|Z|$ its length, by $Z[i]$, $1 \leq i \leq |Z|$, the $i$-th character of $Z$, and by $Z[i, j]$, $1 \leq i \leq j \leq |Z|$, the substring of $Z$ that starts at position $i$ and ends at position $j$. Finally, two substrings $Z[i_1, i_2]$ and $Z[j_1, j_2]$, $1 \leq i_2 \leq j_2 \leq |Z|$, overlap if $j_1 \leq i_2$.

*The Duplication-Loss Model of Evolution:* We assume that present-day genomes have evolved from an ancestral string through duplications and losses, where: (i) A *Duplication* of size $k$ is an operation that copies a substring of size $k$ of a current genome $X$ somewhere else in the genome. Given two identical non overlapping substrings $X[i, i+k-1]$ and $X[j, j+k-1]$ of $X$, we denote by $D = (X[i, i+k-1], X[j, j+k-1])$ a *duplication* from $X[i, i+k-1]$ to $X[j, j+k-1]$; the string $X[i, i+k-1]$ is called the *source*, and the string $X[j, j+k-1]$ the *target* of the duplication $D$; (ii) A *loss* of size $k$ is an operation $L = (X[i, i+k-1])$ that removes a substring $X[i, i+k-1]$ of size $k$ from genome $X$.

Consider a duplication $D = (X[i_1, i_2], X[j_1, j_2])$ of $X$. Such a duplication is called *maximal* if it cannot be extended using positions adjacent to the source and target substrings.

Given an integer $k \geq 1$, the cost of a duplication of size $k$ is denoted by $c(D(k))$, and the cost of a loss of size $k$ is denoted by $c(L(k))$.

*The Duplication-Loss Alignment Problem:* We introduced in [11] the concept of "Feasible" Labeled Alignment of two genomes $X$ and $Y$, and showed the one-to-one correspondence between the set of such alignments and the set of all possible "visible" evolutionary histories from a common ancestor $A$ to $X$ and $Y$. Definitions on alignments are given below, and illustrated in Figure 1.

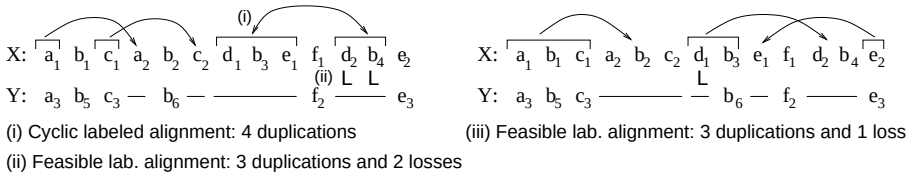


**Fig. 1.** Labeled alignments for strings $X =$ "abcabcdbefdbe" and $Y =$ "abcbfe". Costs are $c(D(k)) = 1$ and $c(L(k)) = k$ for any integer $k$. Losses are denoted by "L" and duplications by arrows from source (indicated by bracket) to target. Two different labeling are given for the left alignment: one (i) with "$d_2\ b_4$" being interpreted as the target of a duplication, and one (ii) with the same substring interpreted as two losses.

In the remaining of this paper, we consider two genomes $X$ and $Y$ on an alphabet $\Sigma$, with $|X| = n$ and $|Y| = m$. Let $\Sigma^- = \Sigma \cup \{-\}$ be the alphabet $\Sigma$ augmented with an additional character '-' called a gap.

**Definition 1.** *An* Alignment *of $X$ and $Y$ is a pair $(\mathcal{X}, \mathcal{Y})$ of strings on $\Sigma^- \times \Sigma^-$ obtained by filling $X$ and $Y$ respectively with gaps, such that the resulting* Aligned Genomes *$\mathcal{X}$ and $\mathcal{Y}$ are equal length. Moreover, each position $i$, with $1 \leq i \leq |\mathcal{X}|$, is such that either $\mathcal{X}[i] = \mathcal{Y}[i] \neq -$ (position $i$ is called a* Match*), or exactly one of $\mathcal{X}[i]$, $\mathcal{Y}[i]$ is equal to a gap (position $i$ is called a* Mismatch*).*

In order to uniquely match an alignment $(\mathcal{X}, \mathcal{Y})$ with a duplication-loss history leading to $X$ and $Y$ from a common ancestor, we need to label unmatched characters of the aligned genomes $\mathcal{X}$ and $\mathcal{Y}$ in terms of *duplications* and *losses*.

**Definition 2.** *A* Labeling *$\mathcal{L}(\mathcal{X})$ of an aligned genome $\mathcal{X}$ (or simply $\mathcal{L}$ if no ambiguity) is a set of losses and duplications, such that for each mismatched position $j$, $1 \leq j \leq |\mathcal{X}|$, $\mathcal{L}(\mathcal{X})$ contains either a loss $L = (\mathcal{X}[j_1, j_2])$ or exactly one duplication $D = (\mathcal{X}[i_1, i_2], \mathcal{X}[j_1, j_2])$, with $1 \leq j_1 \leq j \leq j_2 \leq |\mathcal{X}|$.*

*Now a* Labeling *of an alignment $(\mathcal{X}, \mathcal{Y})$ is a pair $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ where $\mathcal{L}(\mathcal{X})$ and $\mathcal{L}(\mathcal{Y})$ are labeling of $\mathcal{X}$ and $\mathcal{Y}$ respectively. The pair $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ is a* Labeled Alignment *of $X$ and $Y$. The cost of a labeling $\mathcal{L}(\mathcal{X})$ is the cost of the underlying operations (losses and duplications). The cost of a labeled alignment $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ is the sum of cost of the two labeling $\mathcal{L}(\mathcal{X})$ and $\mathcal{L}(\mathcal{Y})$.*

The above definition is not sufficient to ensure a correct interpretation of an alignment in term of duplication-loss history, as it does not prevent from a "cyclic" interpretation of an alignment. For example the labeled alignment (i) in Figure 1 is not feasible as it reflects a history with two circular duplications $D = (d_1 b_3 e_1, d_2 b_4 e_2)$ and $D' = (d_2 b_4, d_1 b_3)$. A "feasible labeling" is a non-cyclic labeling, where cycles are rigorously defined as follows.

**Definition 3.** *Consider a set of duplications $\mathcal{D}$. $\mathcal{D}$ induces a* Duplication Cycle *if there is a permutations $D_1 = (\mathcal{X}[i_1, r_1], \mathcal{X}[j_1, s_1])$, $D_2 = (\mathcal{X}[i_2, r_2], \mathcal{X}[j_2, s_2])$, $\ldots$, $D_h = (\mathcal{X}[i_h, r_h], \mathcal{X}[j_h, s_h])$ of the duplications in $\mathcal{D}$, such that the substrings $\mathcal{X}[j_p, s_p]$ and $\mathcal{X}[i_{p+1}, r_{p+1}]$ overlap, for each $1 \leq p \leq h-1$, and the substrings $\mathcal{X}[j_h, s_h]$ and $\mathcal{X}[i_1, r_1]$ overlap.*

Now, a labeling $\mathcal{L}(\mathcal{X})$ is Feasible if there is no subset of duplications in $\mathcal{L}(\mathcal{X})$ that induces a duplication cycle. Finally a Feasible Labeled Alignment $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ is a labeled alignment of $X$ and $Y$ where $\mathcal{L}(\mathcal{X})$ and $\mathcal{L}(\mathcal{Y})$ are feasible labeling. In Figure 1, (ii) and (iii) are two feasible labeled alignments of $X$ and $Y$, with (iii) being one of minimum cost.

We are now ready to give the main optimization problem allowing to infer a most parsimonious history of duplications and losses leading to present-day genomes from a common ancestor.

**Problem 1** ***Duplication-Loss Alignment****[DLA]*
***Input****: Two genomes $X$ and $Y$.*
***Output****: A Feasible Labeled Alignment $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ of minimum cost.*

This problem has been shown NP-hard in [5]. An exact pseudo-boolean linear programming algorithm has been developed in [11] for this problem. The next section presents an alternative approach based on dynamic programming.

## 3 A Dynamic Programming Approach

Let $|X| = n$ and $|Y| = m$. Let $C(i,j)$ ($C^f(i,j)$ respectively) be the minimum cost of a labeled (feasible labeled respectively) alignment of two prefixes $X[1,i]$ and $Y[1,j]$ of $X$ and $Y$. Then the problem is to compute $C^f(m,n)$. A natural approach sketched in [11] proceeds in two steps:

• Step 1. Unlabeled Alignment. Based on a dynamic programming approach, compute $C(i,j)$, for $1 \leq i \leq n$ and $1 \leq j \leq m$. Recurrences given in [11] allow to compute all values $M(i,j)$, $D_X(i,j)$, $D_Y(i,j)$, $L_X(i,j)$ and $L_Y(i,j)$ reflecting the minimum cost of an alignment $(\mathcal{X}_i, \mathcal{Y}_j)$ of $X[1,i]$ and $Y[1,j]$ satisfying respectively, the constraint that the last characters of $\mathcal{X}_i$ and $\mathcal{Y}_j$ represent a match, a duplication in $\mathcal{X}$ or in $\mathcal{Y}$, a loss in $\mathcal{X}$ or in $\mathcal{Y}$.

After computing all the values leading to $C(m,n)$, a bottom-up approach allows to output a labeled alignment $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ of minimum cost $C(m,n)$. Unfortunately, $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ is not necessarily a feasible alignment, as the recurrences for $D_X(i,j)$ may lead to invalid cyclic evolutionary scenarios. Notice that, as the DLA problem has been recently shown to be NP-complete [5], unless $P = NP$, no alternative recurrences would lead to a polynomial-time algorithm for computing $C^f(m,n)$.

• Step 2. Minimum Labeling Alignment. Consider an (unlabeled) alignment $(\mathcal{X}, \mathcal{Y})$ output by Step 1, and label it in an optimal way, e.g. find labeling $\mathcal{L}(\mathcal{X})$ and $\mathcal{L}(\mathcal{Y})$ for $\mathcal{X}$ and $\mathcal{Y}$ respectively, such that $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ is a feasible labeled alignment of minimum cost over all possible labeling of $(\mathcal{X}, \mathcal{Y})$. Notice that once the genomes are aligned, each labeling can be computed independently. Hence, the Minimum Labeling Alignment problem can be formulated as follows:

**Problem 2** ***Minimum Labeling Alignment****[MLA]*
***Input****: An aligned genome $\mathcal{X}$.*
***Output****: A Feasible Labeling $\mathcal{L}(\mathcal{X})$ of minimum cost.*

The complexity of the MLA problem, as well as an appropriate algorithm to solve it, were left open in [11]. These are precisely the goals of our paper. It has to be noted that this approach cannot lead to an exact algorithm, as an alignment of minimum cost $C(m,n)$ does not necessarily lead to a feasible alignment of minimum cost $C^f(m,n)$. For example in Figure 1, an optimal labeling for alignment (i) of minimum cost $C(m,n) = 4$ leads to the feasible alignment (ii) of cost 5, which is not optimal, as (iii) is a better feasible alignment of cost 4.

*Cost:* As in [11], we will consider $c(D(k)) = 1$ and $c(L(k)) = k$. This leads to a natural weight of an evolutionary history in term of number of segmental duplications (duplication of a string of adjacent genes) and single losses (loss of a single gene). Although segmental deletions are also likely to occur during evolution, accumulation of mutations transforming a single gene into a pseudogene is the most frequent cause of gene loss. From an optimization point of view, the DLA problem is trivial if we count segmental losses as single events in

the same way as duplications, that is $c(L(k)) = 1$. Indeed, in this case, a most parsimonious labeled alignment can always be obtained by ignoring duplications.

## 4 Hardness of Minimum Labeling Alignment

In this section, we prove that the MLA problem is APX-hard, even if each character (gene) has at most 5 occurrences in a genome $X$, by giving an $L$-reduction from the Minimum Vertex Cover problem on Cubic graphs (MVCC), known to be APX-hard [1], to MLA (for details on $L$-reduction see [2]). A graph is cubic iff each vertex of the graph has degree 3. Given a cubic graph $G = (V, E)$, with $V = \{v_1, \ldots, v_n\}$, MVCC asks for a minimum cardinality set $V' \subseteq V$, such that for each $\{v_i, v_j\} \in E$, at least one of $v_i$, $v_j$ belongs to $V'$.

Next, we present the L-reduction from MVCC to MLA. Let $G = (V, E)$ be a cubic graph. Define the following ordering on the edges in $E$: $\{v_i, v_j\} < \{v_x, v_y\}$ if and only if $i < x$, or (in case $i = x$) $j < y$. Based on this ordering, we denote the edges incident on $v_i$, as the first, the second and the third edges of $v_i$. In what follows, given $v_i \in V$, we denote with $\{v_i, v_j\}$, $\{v_i, v_h\}$, $\{v_i, v_k\}$ the first, the second and the third edges respectively of $G$ incident on $v_i$.

First, we define the aligned genome $\mathcal{X}$ corresponding to the cubic graph $G$. We present an overview of the construction of $\mathcal{X}$, then we give the details of the construction. The aligned genome $\mathcal{X}$ consists of two *parts* (see Fig. 2): the leftmost part is called the *Vertex-Edge-set Part* (VE-Part), the rightmost part is called the *Auxiliary Part* (A-Part). Each part is then divided into substrings, called *blocks*. Each position of $\mathcal{X}$ in the A-part is a match, while positions in the VE-part can be either matches or mismatches. Hence a labeling $\mathcal{L}$ of $\mathcal{X}$ is computed by labeling the mismatched positions in the VE-part of $\mathcal{X}$.

The VE-part of $\mathcal{X}$ consists of the concatenation of $|V| + |E|$ blocks (see Fig. 2). For each vertex $v_i \in V$ there is one block $B_{VE}(v_i)$ in the VE-part of $\mathcal{X}$; for each edge $\{v_i, v_j\} \in E$, there is one block $B_{VE}(e_{i,j})$ in the VE-part of $\mathcal{X}$.

The A-part of $\mathcal{X}$ consists of the concatenation of $2|V|$ blocks (see Fig. 2). For each $v_i \in V$, there exist two blocks $B_{A,1}(v_i)$, $B_{A,2}(v_i)$ in the A-part of $\mathcal{X}$. Now,

$$\mathcal{X} = \underbrace{B_{VE}(v_1) \ldots B_{VE}(v_n) B_{VE}(e_{1,a}) \ldots B_{VE}(e_{z,w})}_{\text{VE-part}} \cdot \underbrace{B_{A,1}(v_1) B_{A,2}(v_1) \ldots B_{A,1}(v_n) B_{A,2}(v_n)}_{\text{A-part}}$$

**Fig. 2.** The structure of the aligned genome $\mathcal{X}$

we define the specific values of the blocks of $\mathcal{X}$. Given an edge $\{v_i, v_j\} \in E$, where $i < j$, $\{v_i, v_j\}$ is the $p$-th edge of $v_i$, $1 \leq p \leq 3$, and the $q$-th edge of $v_j$, $1 \leq q \leq 3$, we define its associated block $B_{VE}(e_{i,j})$ as follows:

$$B_{VE}(e_{i,j}) = s_{e,i,j} x_{i,p} e_{i,j,1} e_{i,j,2} x_{j,q}$$

where the first position of $B_{VE}(e_{i,j})$, that is the position containing character $s_{e,i,j}$, is a match and each other position of $B_{VE}(e_{i,j})$ is a mismatch.

Now, we define the block $B_{VE}(v_i)$, with $v_i \in V$. First, define the $i$-encoding of $\{v_i, v_j\}$, denoted as $i\text{-}enc_{i,j}$, as the following string: $i\text{-}enc_{i,j} = x_{i,p}e_{i,j,1}e_{i,j,2}$. Moreover, let $i\text{-}enc^l_{i,j} = x_{i,p}$, and $i\text{-}enc^r_{i,j} = e_{i,j,1}e_{i,j,2}$. The $j$-encoding of $\{v_i, v_j\}$, denoted as $j\text{-}enc_{i,j}$, is defined as follows: $j\text{-}enc_{i,j} = e_{i,j,1}e_{i,j,2}x_{j,q}$, and $j\text{-}enc^l_{i,j} = e_{i,j,1}e_{i,j,2}$, $j\text{-}enc^r_{i,j} = x_{j,q}$.

The block $B_{VE}(v_i)$ is defined as follows:

$$B_{VE}(v_i) = s_i z_{i,1} z_{i,2}\ i\text{-}enc_{i,j}\ z_{i,3}z_{i,4}\ i\text{-}enc_{i,h}\ z_{i,5}z_{i,6}\ i\text{-}enc_{i,k}\ z_{i,7}z_{i,8}$$

$B_{VE}(v_i)$ contains one matched position, the first position containing character $s_i$, and 17 mismatched positions (from position 2 to position 18 of $B_{VE}(v_i)$).

Now, we define the A-part of $\mathcal{X}$. Recall that each position of the A-part of $\mathcal{X}$ is a match. The block $B_{A,1}(v_i)$ is defined as follows:

$$B_{A,1}(v_i) = w_{i,1}z_{i,1}z_{i,2}w_{i,2}z_{i,3}z_{i,4}w_{i,3}z_{i,5}z_{i,6}w_{i,4}z_{i,7}z_{i,8}$$

The block $B_{A,2}(v_i)$ is defined as follows:

$$B_{A,2}(v_i) = u_{i,1}z_{i,2}\ i\text{-}enc^l_{i,j}\ u_{i,2}\ i\text{-}enc^r_{i,j}\ z_{i,3}u_{i,3}z_{i,4}\ i\text{-}enc^l_{i,h}\ u_{i,4}\ i\text{-}enc^r_{i,h}\ z_{i,5}\cdot$$

$$\cdot u_{i,5}z_{i,6}\ i\text{-}enc^l_{i,k}\ u_{i,6}\ i\text{-}enc^r_{i,k}\ z_{i,7}$$

Before giving the details of the proof, we give a high-level description of the reduction. We will show that each block $B_{VE}(v_i)$ can be labeled essentially in two possible ways (see Remark 4):
1. with a *type a labeling*, defining seven maximal duplications from substrings of blocks $B_{VE}(e_{i,j})$, $B_{VE}(e_{i,h})$, $B_{VE}(e_{i,k})$, $B_{A,1}(v_i)$ to substrings of block $B_{VE}(v_i)$; a *type a labeling* is the optimal labeling of $B_{VE}(v_i)$ (see Lemma 6) and has a cost of 7;
2. with a *type b labeling*, defining six maximal duplications from substrings of block $B_{A,2}(v_i)$ to substrings of block $B_{VE}(v_i)$ and two losses; a *type b labeling* is a suboptimal labeling of $B_{VE}(v_i)$ (see Lemma 6) and has a cost of 8.

Thanks to the property of block $B_{VE}(e_{i,j})$ (see Remark 5 and Lemma 7), we can relate these two kinds of labeling with a cover of $G$ (see Lemma 8 and Lemma 9): a *type b labeling* of $B_{VE}(v_i)$ corresponds to a vertex $v_i$ in a vertex cover $V'$ of $G$, a *type a labeling* of $B_{VE}(v_i)$ corresponds to a vertex $v_i$ in $V \setminus V'$ of $G$.

Now, we give the details of the reduction. First, we introduce some preliminaries properties of $\mathcal{X}$.

*Remark 4.* Consider a cubic graph $G = (V, E)$, and the corresponding instance $\mathcal{X}$ of MLA. Let $v_i$ be a vertex of $V$, with $\{v_i, v_j\}$, $\{v_i, v_h\}$, $\{v_i, v_k\}$ the first, the second and the third edges of $v_i$ respectively. A *type a labeling* of $B_{VE}(v_i)$ consists of the following 7 duplications:

- four duplications, each one from the substring $z_{i,2p-1}, z_{i,2p}$, $1 \leq p \leq 4$, of block $B_{A,1}(v_i)$, to the substring $z_{i,2p-1}, z_{i,2p}$, of block $B_{VE}(v_i)$;

– a duplication from the substring $i\text{-}enc_{i,x}$, with $x \in \{j, h, k\}$, of block $B_{VE}(e_{ix})$ to the substring $i\text{-}enc_{i,x}$ of block $B_{VE}(v_i)$.

A *type b labeling* labeling of $B_{VE}(v_i)$) consists of the following 6 duplications and 2 losses (hence it has a cost of 8):

– six duplications from substrings of $B_{A,2}(v_i)$ to substrings of $B_{VE}(v_i)$ (specifically for the six substrings $z_{i,2}$ $i\text{-}enc^l_{i,j}$, $i\text{-}enc^r_{i,j}$ $z_{i,3}$, $z_{i,4}$ $i\text{-}enc^l_{i,h}$, $i\text{-}enc^r_{i,h}$ $z_{i,5}$, $z_{i,6}$ $i\text{-}enc^l_{i,k}$, $i\text{-}enc^r_{i,k}$ $z_{i,7}$);
– two losses for the leftmost position and the rightmost position of $B_{VE}(v_i)$.

Notice that in a *type b labeling* for $B_{VE}(v_i)$, there is no duplication of $B_{VE}(v_i)$ from substrings of $B_{VE}(e_{ij})$, $B_{VE}(e_{ih})$, $B_{VE}(e_{ik})$.

*Remark 5.* Let $G = (V, E)$ be a cubic graph, let $\{v_i, v_j\} \in E$, with $i < j$, be the $p$-th edge of $v_i$, $1 \leq p \leq 3$, and the $q$-th edge of $v_j$, $1 \leq q \leq 3$. Let $\mathcal{X}$ be the corresponding instance of MLA. The following two labeling of $B_{VE}(e_{i,j})$ (recall that the first position of $B_{VE}(e_{i,j})$ is a match) have cost 2:

– one duplication from the substring $x_{i,p}e_{i,j,1}e_{i,j,2}$ of $B_{VE}(v_i)$ to the substring $x_{i,p}e_{i,j,1}e_{i,j,2}$ of $B_{VE}(e_{i,j})$, one loss for the last position of $B_{VE}(e_{i,j})$
– one duplication from the substring $e_{i,j,1}e_{i,j,2}x_{j,q}$ of $B_{VE}(v_j)$ to the substring $e_{i,j,1}e_{i,j,2}x_{j,q}$ $B_{VE}(e_{i,j})$, one loss for the second position of $B_{VE}(e_{i,j})$

Now, we are ready to show that a *type a labeling* is the only optimal labeling for $B_{VE}(v_j)$.

**Lemma 6.** *Let $G = (V, E)$ be an instance of MVCC and let $\mathcal{X}$ be the corresponding instance of MLA. Then, given a block $B_{VE}(v_i)$, with $v_i \in V$: (1) any feasible labeling of $B_{VE}(v_i)$ has a cost of at least 7; (2) if a labeling has cost of 7, then such a labeling is a* type a labeling *of $B_{VE}(v_i)$.*

*Proof.* (*Sketch.*) (1) The proof follows from a simple counting argument. Block $B_{VE}(v_i)$ contains 17 unmatched positions. By construction the leftmost position and the rightmost position of $B_{VE}(v_i)$ are labeled by duplications of length at most 2. The remaining positions are at least 13, and since by construction are labeled by duplications of length at most 3, it follows that at least $2 + \lceil \frac{13}{3} \rceil = 7$ duplications are required for each feasible labeling of $B_{VE}(v_i)$.

(2) It is easy to see that if a feasible labeling of $B_{VE}(v_i)$ contains only duplications from substrings of $B_{VE}(e_{i,j})$, $B_{VE}(e_{i,h})$, $B_{VE}(e_{i,k})$, $B_{A,1}(v_i)$, then it has a cost of 7 iff is a *type a labeling*. Similarly if a feasible labeling of $B_{VE}(v_i)$ contains only duplications from substrings of $B_{A,2}(v_i)$, it has a cost of at least 8. Assume that a feasible labeling $\mathcal{L}$ of $B_{VE}(v_i)$ contains a duplication $D = (\mathcal{X}[i_1, i_2], \mathcal{X}[j_1, j_2])$, where $\mathcal{X}[i_1, i_2]$ is a substring of $B_{A,2}(v_i)$, and a duplication from a substring of one of $B_{VE}(e_{i,j})$, $B_{VE}(e_{i,h})$, $B_{VE}(e_{i,k})$, $B_{A,1}(v_i)$. It is easy to see that $D$ can (eventually) be extended so that it is a maximal duplication. Then by replacing each other duplication of $\mathcal{L}$ having as a target a substring of $B_{VE}(v_i)$ with a duplication from substrings of $B_{A,2}(v_i)$ (or a loss), we obtain a *type b labeling*. This implies that $\mathcal{L}$ has a cost of at least 8. □

Now, we prove a property on the labeling of a block $B_{VE}(e_{i,j})$.

**Lemma 7.** *Let $G = (V, E)$ be an instance of MVCC and let $\mathcal{X}$ be the corresponding instance of MLA. Then, each feasible labeling of $B_{VE}(e_{i,j})$, with $\{v_i, v_j\} \in E$, has a cost of at least 2, in which case $B_{VE}(e_{i,j})$ must be labeled with one duplication having target in $B_{VE}(v_i)$ or in $B_{VE}(v_j)$.*

*Proof.* By construction, since there is no other substring in $\mathcal{X}$ identical to $B_{VE}(e_{i,j})$, it follows that any labeling of $B_{VE}(e_{i,j})$ requires a cost of at least 2. Now, assume that $B_{VE}(e_{i,j})$ is not labeled by a duplication having a target in $B_{VE}(v_i)$ or in $B_{VE}(v_j)$. Then, by construction, either each position of $B_{VE}(e_{i,j})$ is labeled as a loss (the cost of such labeling is 4) or the position corresponding to the substring $e_{i,j,1}, e_{i,j,2}$ of $B_{VE}(e_{i,j})$ is labeled as a duplication from a substring of $B_{A,2}(v_i)$, implying a cost of 3 for the labeling. □

Now, we are ready to prove the two main properties of the reduction in Lemma 8 and in Lemma 9.

**Lemma 8.** *Let $G$ be an instance of MVCC and let $\mathcal{X}$ be the corresponding instance of MLA. Then, given a vertex cover $V' \subseteq V$ of $G$, we can compute in polynomial time a solution of MLA over instance $\mathcal{X}$ of cost $8|V'|+7|V\setminus V'|+2|E|$.*

*Proof.* (*Sketch*). Given a cover $V'$ of $G$, we define a solution of MLA over instance $\mathcal{X}$ having cost $8|V'| + 7|V \setminus V'| + 2|E|$ as follows: (1) for each $v_i \in V'$, define a *type b labeling* for the corresponding block $B_{VE}(v_i)$ (of cost of 8, see Remark 4); (2) for each $v_i \in V \setminus V'$, define a *type a labeling* for the corresponding block $B_{VE}(v_i)$ (of cost of 7, see Remark 4); (3) a duplication of cost 2 for each $B_{VE}(e_{i,j})$ associated with edge $\{v_i, v_j\} \in E$ (see Remark 5). Since $V'$ is a vertex cover of $G$, at least one of $v_i, v_j \in V'$, hence this labeling is feasible. □

**Lemma 9.** *Let $G$ be an instance of MVCC and let $\mathcal{X}$ be the corresponding instance of MLA. Then, given a feasible labeling of $\mathcal{X}$ of cost $8p+7(|V|-p)+2|E|$, we can compute in polynomial time a vertex cover of $G$ of size at most $p$.*

*Proof.* (*Sketch*). Let $\mathcal{L}$ be a feasible labeling of $\mathcal{X}$ of cost $8p + 7(|V| - p) + 2|E|$. First, by Lemma 6, we can assume that $B_{VE}(v_i)$ is associated in $\mathcal{L}$ either with a *type a labeling* or with a *type b labeling*.

Now, consider a block $B_{VE}(e_{i,j})$, with $\{v_i, v_j\} \in E$. We show that we can assume that at least one of $B_{VE}(v_i)$, $B_{VE}(v_j)$ has a *type b labeling* in $\mathcal{L}$. If this is not the case, $B_{VE}(e_{i,j})$ cannot be labeled with a duplication having source in $B_{VE}(v_i)$, $B_{VE}(v_j)$, hence by Lemma 7, the cost of the labeling of $B_{VE}(e_{i,j})$ is at least 3. We compute in polynomial time a feasible labeling $\mathcal{L}'$, such that $c(\mathcal{L}') \leq c(\mathcal{L})$, as follows: (1) define a *type b labeling* for one of $B_{VE}(v_i)$, $B_{VE}(v_j)$, w.l.o.g. $B_{VE}(v_i)$; (2) define a duplication $D$ from the substring *i-enc*$_{i,j}$ of $B_{VE}(v_i)$ to the substring *i-enc*$_{i,j}$ of $B_{VE}(e_{i,j})$, and a loss for the unmatched position of $B_{VE}(e_{i,j})$ not contained in the target of $D$. Hence we can assume that, for each block $B_{VE}(e_{i,j})$, at least one of $B_{VE}(v_i)$, $B_{VE}(v_j)$ has a *type b labeling* in $\mathcal{L}$. It follows that we can define a vertex cover $V'$ of $G$ as follows: $V' = \{v_i : B_{VE}(v_i)$ has a *type b labeling* in $\mathcal{L}\}$. Since the cost of $\mathcal{L}$ is at most $8p + 7(|V| - p) + 2|E|$, it follows that $|V'| \leq p$. □

The following result is a direct consequence of Lemmas 8 and 9.

**Theorem 1** *MLA is APX-hard.*

## 5 An efficient heuristic

We now present DLAlign , which is a heuristic based on the dynamic programming approach (Section 3) for the Duplication-Loss Alignment (DLA, Problem 1) of two genomes $X$ and $Y$. Recall that $|X| = n$ and $|Y| = m$.

• **Step 1.** Dynamic Programming:
− Compute all the values of $C(i, j)$, for $1 \leq i \leq n$ and $1 \leq j \leq m$;
− Output a labeled alignment $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$ of cost $C(m, n)$. To limit the possibility of creating cycles we do the following: (i) in the bottom-up approach used to output a labeled alignment after filling the dynamic programming table $C$, we choose a match operation whenever possible; (ii) for any duplication involving a given string $Z$, the rightmost position of $Z$ in the genome is always chosen to be the source of the duplication.
• **Step 2.** Minimum Labeling Alignment: Resolve each duplication cycle $\mathcal{D}$ of $(\mathcal{L}(\mathcal{X}), \mathcal{L}(\mathcal{Y}))$, by interpreting the shortest overlapping string of $\mathcal{D}$ as a loss rather than a duplication (see Examples (i) and (ii) in Figure 1).

*Complexity:* For simplicity, suppose $|X| = |Y| = n$. From the recurrences detailed in [11], each $C(i, j)$, for $1 \leq i, j \leq n$, can be computed in time $O(n)$ which leads to an $O(n^3)$ algorithm for Step 1. As for Step 2, it requires constructing a graph for $X$ ($Y$ respectively): for each duplication, add two vertices corresponding to its source and target, and one edge from source to target. Constructing the graphs, findings the cycles and resolving them can be done in time $O(n^3)$, which leads to an $O(n^3)$ worst-time complexity for the whole heuristic.

### 5.1 Simulations

A random string $R$ was drawn from the set of all strings of length $n$ on an alphabet of size $a$, and $l$ moves were then applied to $R$ to obtain an ancestral genome $A$. To obtain the extant genomes $X$ and $Y$, $l$ more moves were applied to $A$ for each. The set of moves were segmental duplications and single gene losses. The length of a duplication was drawn from a Gaussian distribution with mean 5 and standard deviation 2; these lengths were consistent with those observed for the tRNA repertoire in *Bacillus* lineages [11].

*Execution time:* With $2l/n = 1/5$ and $a/n = 1/2$, statistics similar to those observed for the tRNA repertoire in *Bacillus*, strings of length 5000 took a couple of days to be processed by the linear programming algorithm on a standard PC workstation with 4 GB of memory. In comparison, the same data have been processed by DLAlign on the same computer in less than two seconds.

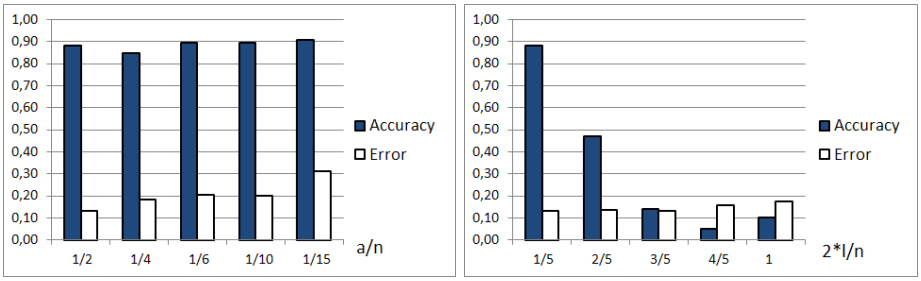


**Fig. 3.** The score returned by DLAlign compared to the optimal one returned by the linear programming algorithm, for datasets of size up to $n = 200$. The left diagram is obtained by varying the alphabet size $a$ (x-axis is $a/n$), and the right diagram by varying the number of moves $l$ (x-axis is $2 * l/n$). See text for more details

*Accuracy:* We compare $Res$, the alignment cost returned by DLAlign, with the optimal cost $Opt$ obtained by running the linear programming algorithm. Due to the exponential-time complexity of the later, we had to restrict ourselves to relatively small values of $n$, $a$ and $l$. Results of Figure 3 are averaged over up to $Total = 1000$ simulations. White bars refer to $Error = \frac{Res-Opt}{Res}$, and blue ones to $Accuracy = \frac{NbOpt}{Total}$, where $NbOpt$ is the number of simulations among $Total$ for which DLAlign outputs the optimal alignment (i.e. $Error = 0$).

With ratios $2l/n = 1/5$ and $a/n = 1/2$, DLAlign returns the optimal alignment cost for more than 85% of the simulations. This accuracy rate remains stable for decreasing alphabet size, i.e. increasing number of gene copies (left diagram in Figure 3), but quickly drops with increasing number $l$ of moves (right diagram). Notice however that, even for a number of moves being equal to the size of the strings, the error rate $Error$ always remains lower than 0.16.

## 6 Conclusion

In this paper, we investigated the problem of aligning two genomes, based on a duplication and loss model of evolution. We developed a heuristic in two steps: first use dynamic programming to output a best candidate solution, then consider MLA to compute a feasible solution. The heuristic exhibited a high degree of accuracy on simulated datasets. Moreover, it is a thousands of times faster than the previously developed linear programming algorithm, which makes possible its application to large genomes, and allows generalization to multiple genome alignment in a phylogenetic context. From a theoretical point of view, we showed that the MLA problem is APX-hard even when each gene has at most five occurrences in a genome. Interesting future work will be to investigate the approximation and parametrized complexity of MLA.

## Acknowledgements

We thank Krister M. Swenson for advices and help with simulations.

## References

1. Alimonti, P., Kann, V.: Some APX-completeness results for cubic graphs. Theoretical Computer Science 237(1–2), 123–134 (2000)
2. Ausiello, G., Crescenzi, P., Gambosi, G. Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer-Verlag, Heidelberg (1999)
3. Bergeron, A.: A very elementary presentation of the hannenhalli-pevzner theory. In: Amir, A., Landau, G.M. (eds.) CPM 2001. LNCS, vol. 2089, pp. 106–117. Springer (2001)
4. Bourque, G., Pevzner, P.: Genome-scale evolution: Reconstructing gene orders in the ancestral species. Genome Research 12, 26 – 36 (2002)
5. Canzar, S., Andreotti, S.: A branch-and-cut algorithm for the 2-species duplication-loss phylogeny problem. CoRR abs/1208.2698 (2012)
6. El-Mabrouk, N.: Mathematics of Evolution and Phylogeny, chap. Genome rearrangement with gene families, pp. 291- 320. Oxford University Press, Oxford (2005)
7. El-Mabrouk, N., Sankoff, D.: Evolutionary genomics: statistical and computational methods, chap. Analysis of Gene Order Evolution beyond Single-Copy Genes. Methods in Molecular Biology, Springer (Humana), New York (2012)
8. El-Mabrouk, N.: Genome rearrangement by reversals and insertions/deletions of contiguous segments. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. Lecture Notes in Computer Science, vol. 1848, pp. 222–234. Heidelberg (2000)
9. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of genome rearrangements. The MIT Press, Cambridge, Massachusetts and London, England (2009)
10. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). Journal of the ACM 48, 1–27 (1999)
11. Holloway, P., Swenson, K.M., Ardell, D.H., El-Mabrouk, N.: Evolution of genome organization by duplication and loss: An alignment approach. In: Chor, B. (ed.) RECOMB 2012. pp. 94–112. Springer, Heidelberg (2012)
12. Ma, J., Zhang, L., Suh, B., Raney, B., Burhans, R., Kent, W., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome Research 16, 1557 – 1565 (2007)
13. Marron, M., Swenson, K.M., Moret, B.M.E.: Genomic distances under deletions and insertions. In: Warnow, T., Zhu, B. (eds.) COCOON 2003. LNCS, vol. 2697, pp. 537–547. Springer, Heidelberg (2003)
14. Moret, B., Wang, L., Warnow, T., Wyman, S.: New approaches for reconstructing phylogenies from gene order data. Bioinformatics 17, S165–S173 (2001)
15. Sankoff, D., Blanchette, M.: The median problem for breakpoints in comparative genomics. In: Jiang, T., Lee, D.T. (eds.) COCOON 1997. LNCS, vol. 1276, pp. 251–264. Springer, Heidelberg (1997)