Ancestral Genome Organization: an Alignment Approach

Patrick Holloway[1], Krister Swenson[2], David Ardell[3], and Nadia El-Mabrouk[4]

[1] Département d'Informatique et de Recherche Opérationnelle (DIRO), Université de Montréal, H3C 3J7, Canada, patrick.holloway@umontreal.ca
[2] DIRO and University of McGill Computer Science, swensonk@iro.umontreal.ca
[3] Center for Computational Biology, School of Natural Sciences, 5200 North Lake Road, University of California, Merced, CA 95343, dardell@ucmerced.edu
[4] DIRO, mabrouk@iro.umontreal.ca

Keywords: Comparative Genomics, Gene order, Duplication, Loss, Linear Programming, Alignment, Bacillus, tRNA.

Abstract. We present a comparative genomics approach for inferring ancestral genome organization and evolutionary scenarios, based on present-day genomes represented as ordered gene sequences with duplicates. We develop our methodology for a model of evolution restricted to duplication and loss, and then show how to extend it to other content-modifying operations, and to inversions. From a combinatorial point of view, the main consequence of ignoring rearrangements is the possibility of formulating the problem as an alignment problem. On the other hand, duplications and losses are asymmetric operations that are applicable to one of the two aligned sequences. Consequently, an ancestral genome can directly be inferred from a duplication-loss scenario attached to a given alignment. Although alignments are *a priori* simpler to handle than rearrangements, we show that a direct approach based on dynamic programming leads, at best, to an efficient heuristic. We present an exact pseudo-boolean linear programming algorithm to search for the optimal alignment along with an optimal scenario of duplications and losses. Although exponential in the worst case, we show low running times on real datasets as well as synthetic data. We apply our algorithm[5] in a phylogenetic context to the evolution of stable RNA (tRNA and rRNA) gene content and organization in *Bacillus* genomes. Our results lead to various biological insights, such as rates of ribosomal RNA proliferation among lineages, their role in altering tRNA gene content, and evidence of tRNA class conversion.

1 Introduction

During evolution, genomes continually accumulate mutations. In addition to base mutations and short insertions or deletions, genome-scale changes affect the overall gene content and organization of a genome. Evidence of these latter kinds of changes are observed by comparing the completely sequenced and annotated genomes of related species. Genome-scale changes can be subdivided into two categories: (1) the *rearrangement operations* that shuffle gene order such as inversion, and (2) the *content-modifying operations* that affect the number of gene copies, such as gene insertion, loss and duplication. In particular, gene duplication is a fundamental process in the evolution of species [34], especially in eukaryotes [10,16,20,24,31,46], where it is believed to play a leading role for the creation of novel gene function. In parallel, gene losses through pseudogenization and segmental deletions, appear generally to maintain a minimum number of functional gene copies [10,16,17,20,24,31,34]. Transfer RNAs (tRNAs) are typical examples of gene families that are continually duplicated and lost [7,37,43,47].

[5] The code is freely available upon request.

Indeed, tRNA clusters (or operons in microbial genomes) are highly dynamic and unstable genomic regions. In *Escherichia coli* for example, the rate of tRNA gene duplication/loss events has been estimated to be about one event every 1.5 million years [7,47].

One of the main goals of comparative genomics is to infer evolutionary histories of gene families, based on the comparison of the genomic organization of extant species. Having an evolutionary perspective of gene families is a key step towards answering many fundamental biological questions. For example, tRNAs are essential to establishing a direct link between codons and their translation into amino-acids. Understanding how the content and organization of tRNAs evolve is essential to the understanding of the translational machinery, and in particular, the variation in codon usage among species [19,28].

In the genome rearrangement approach to comparative genomics, a genome is modeled as one or many (in case of many chromosomes) linear or circular sequences of genes (or other building blocks of a genome). When each gene is present exactly once in a genome, sequences can be represented as permutations. In the most realistic version of the rearrangement problem, a sign is associated with a gene, representing its transcriptional orientation. The pioneering work of Hannenhalli and Pevzner in 1995 [25,26], has led to efficient algorithms for computing the inversion and/or translocation distance between two signed permutations. Since then, many other algorithms have been developed to compare permutations subject to various rearrangement operations and based on different distance measures. These algorithms have then been used from a phylogenetic perspective to infer ancestral permutations [11,14,32,33,40] and evolutionary scenarios on a species tree. An extra degree of difficulty is introduced in the case of sequences containing multiple copies of the same gene, as the one-to-one correspondence between copies is not established in advance. A review of the methods used for comparing two ordered gene sequences with duplicates can be found in [22,23]. They can be grouped into two main classes. The "Match-and-Prune" model aims at transforming strings into permutations, so as to minimize a rearrangement distance between the resulting permutations. On the other hand, the "Block Edit" model consists of performing the minimum number of "allowed" rearrangement and content-modifying operations required to transform one string into the other. Most studied distances and ancestral inference problems in this category are NP-complete [23].

In this paper, we focus on comparing two ordered gene sequences with duplicates. We develop our methodology for a model of evolution restricted to content-modifying operations, more specifically to duplication and loss, and then show how to extend it to other content-modifying operations, and to non-overlapping inversions. From a combinatorial point of view, the main consequence of ignoring rearrangement operations is the fact that gene organization is preserved, which allows us to reformulate the problem of comparing two gene orders as an alignment problem. Notice however that only the most recent events that have not been obscured by subsequent events are visible in such an alignment. Another advantage of a model restricted to duplication and loss is that a unique ancestral genome can directly be inferred from a duplication-loss scenario attached to a given alignment, as duplications and losses are asymmetrical operations that are applicable to one of the two aligned sequences.

Although alignments are *a priori* simpler to handle than rearrangements, there is no direct way of inferring optimal alignments together with a related duplication-loss scenario for two gene orders, as detailed in Section 4. Even our simpler goal of finding an alignment is fraught with difficulty as a naive branch-and-bound approach to compute such an alignment

is non-trivial; trying all possible alignments with all possible duplication and loss scenarios for each alignment is hardly practicable. As it is not even clear how, given an alignment, we can assign duplications and losses in a parsimonious manner, we present in Section 4.1 a pseudo-boolean linear programming (PBLP) approach to search for the optimal alignment along with an optimal scenario of duplications and losses. The disadvantage of the approach is that, in the worst case, an exponential number of steps could be used by our algorithm. On the other hand, we show in Section 6.2 that for real data, and larger simulated genomes, the running times are quite reasonable. Further, the PBLP is flexible in that a multitude of weighting schemes for losses and duplications could be employed to, for example, favor certain duplications over others, or allow for gene conversion.

In Section 5, we extend our initial model of evolution to handle additional content-modifying operations, as well as non-overlapping inversions, and we show how to relax some constraints regarding the visible (*i.e.* non-intersecting) nature of operations. In Section 6.1, we apply our algorithm in a phylogenetic context to infer the evolution of stable RNA (tRNA and rRNA) gene content and organization in various genomes from the genus *Bacillus*, a so-called "low G+C" gram-positive clade of Firmicutes that includes the model bacterium *B. subtilis* as well as the agent of anthrax. Stable RNA operon organization in this group is interesting because it has relatively fewer operons that are much larger and contain more segmental duplicates than other bacterial groups. We obtained results leading to various biological insights, such as more accurate quantification of ribosomal RNA operon proliferation, their role in altering tRNA gene content, and evidence of tRNA gene class conversion.

2 Research context

The evolution of g genomes is often represented by a phylogenetic (or species) tree T, binary or not, with exactly g leaves, each representing a different genome. When such a species tree T is known for a set of species, then we can use the gene order information of the present-day genomes to infer gene order information of ancestral genomes identified with each of the internal nodes of the tree. This problem is known in the literature as the "small" phylogeny problem, in contrast to the "large" phylogeny problem which is one of finding the actual phylogenetic tree T.

Although our methods may be extended to arbitrary genomes, we consider single chromosomal (circular or linear) genomes, represented as gene orders with duplicates. More precisely, given an alphabet Σ where each character represents a specific gene family, a **genome** or **string** is a sequence of characters from Σ where each character may appear many times. To simplify our explanation we ignore the orientation (sign) of the genes. However, our methodology is easily extendable to inclusion of this information. For example, given $\Sigma = \{a, b, c, d, e\}$, $A =$"*ababcd*" is a genome containing two gene copies from the gene family identified by a, two genes from the gene family b, and a single gene from each family c and d. A gene in a genome A is a **singleton** if it appears exactly once in A (for example c and d in A), and a **duplicate** otherwise (a and b in A above).

Let $\mathcal{O}$ be a set of "allowed" evolutionary operations. The set $\mathcal{O}$ may include organizational operations such as Reversals (R) and Transpositions (T), and content-modifying operations such as Duplications (D), Losses (L) or Insertions (I). For example, $\mathcal{O} = \{R, D, L\}$ is the set

of operations in an evolutionary model involving reversals, duplications and losses. In the next section, we will formally define the operations involved in our model of evolution.

Given a genome A, a **mutation** on A is characterized by an operation O from $\mathcal{O}$, the substring of A that is affected by the mutation, as well as possibly other characteristics such as the position of the re-inserted removed (in case of transposition) or duplicated substring. For simplicity, consider a mutation $O(k)$ to be characterized solely by the operation O from $\mathcal{O}$, and the size k of the substring affected by the mutation. Consider $c(O(k))$ to be a cost function defined on mutations. Finally, given two genomes A and X, an **evolutionary history** $O_{A\to X}$ from A to X is a sequence of mutations (possible of length 0) transforming A into X.

Let A, X be two strings on Σ with A being a **potential ancestor** of X, meaning that there is at least one evolutionary history $O_{A\to X} = \{O_1(k_1), \cdots, O_l(k_l)\}$ from A to X. Then the cost of $O_{A\to X}$ is:

$$C(O_{A\to X}) = \sum_{i=1}^{l} c(O_i(k_i))$$

Now let $\mathcal{O}_{A\to X}$ be the set of possible histories transforming A into X. Then we define:

$$C(A \to X) = \min_{O_{A\to X}\in\mathcal{O}_{A\to X}} C(O_{A\to X})$$

Then, the small phylogeny problem can be formulated as one of finding strings at internal nodes of a given tree T that minimize the total cost:

$$C(T) = \sum_{\text{all branches } b_i \text{ of } T} C(X_{i,1} \to X_{i,2})$$

where $X_{i,1}, X_{i,2}$ are the strings labeling the nodes of T adjacent to the branch b_i, with the node labeled $X_{i,1}$ being the parent of the node labeled $X_{i,2}$.

For most restrictions on genome structure and models of evolution, the simplest version of the small phylogeny problem — the median of three genomes — is NP-hard [12,35,44]. When duplicate genes are present in the genomes, even finding minimum distances between two genomes is almost always an NP-Hard task [27]. In this paper, we focus on **cherries of a species tree** (*i.e.* on subtrees with two leaves). The optimization problem we consider can be formulated as follows:

Two Species Small Phylogeny Problem:
INPUT: Two genomes X and Y.
OUTPUT: A potential common ancestor A of X and Y minimizing $C(A \to X) + C(A \to Y)$.

Solving the TWO SPECIES SMALL PHYLOGENY PROBLEM (2-SPP) can be seen as a first step towards solving the problem on a given phylogenetic tree T. The most natural heuristic to the Small Phylogeny Problem, that we will call the SPP-HEURISTIC, is to traverse T depth-first, and to compute successive ancestors of pairs of nodes. Such a heuristic can be used as the initialization step of the *steinerization* method for SPP [40,8]. The sets of all

optimal solutions output by an algorithm for the 2-SPP applied to all pairs of nodes of T (in a depth-first traversal) can alternatively be used in an iterative local optimization method, such as the dynamic programming method developed in [29].

3 The Duplication and Loss Model of Evolution

Our evolutionary model accounts for two operations, Duplication (denoted D) and Loss (denoted L). In other words $\mathcal{O} = \{D, L\}$, where D and L are defined as follows. Let $X[i \dots i+k]$ denote the substring $X_i X_{i+1} \cdots X_{i+k}$ of X.

- D: A **Duplication** of size $k+1$ on $X = X_1 \cdots X_i \cdots X_{i+k} \cdots X_j X_{j+1} \cdots X_n$ is an operation that copies a substring $X[i \dots i+k]$ to a location j of X outside the interval $[i, i+k]$ (*i.e.* preceding i or following $i+k$). In the latter case, D transforms X into

$$X' = X_1 \cdots \underline{X_i \cdots X_{i+k}} \cdots X_{j-1} \underline{X_i \cdots X_{i+k}} X_{j+1} \cdots X_n$$

  We call the original copy $X[i \dots i+k]$ the **origin**, and the copied string the **product** of the duplication D.
- L: A **Loss** of size k is an operation that removes a substring of size k from X.

Notice that gene insertions could be considered in our model as well. In particular, our linear programming solution is applicable to an evolutionary model involving insertions, in addition to duplications and losses. We ignore insertions for two main reasons: (1) insertions and losses are two symmetrical operations that can be interchanged in an evolutionary scenario. Distinguishing between insertions and losses may be possible on a phylogeny, but cannot be done by comparing two genomes; (2) gene insertions are usually due to lateral gene transfer, which may be rare events compared to nucleotide-level mutations that eventually transform a gene into a pseudogene.

As duplication and loss are content-modifying operations that do not shuffle gene order, the Two Species Small Phylogeny Problem can be posed as an alignment problem. However, the only operations that are "visible" on an alignment are the events on an evolutionary history that are not obscured by subsequent events. Moreover, as duplications and losses are asymmetrical operations, an alignment of two genomes X and Y does not reflect an evolutionary path from X to Y (as operations going back to a common ancestor are not defined), but rather two paths going from a common ancestor to both X and Y. A precise definition follows.

Definition 1. *Let X and Y be two genomes. A* **visible history** *of X and Y is a triplet $(A, O_{A\to X}, O_{A\to Y})$ where A is a potential ancestor of both X and Y, and $O_{A\to X}$ (respectively $O_{A\to Y}$) are evolutionary histories from A to X (respectively from A to Y) verifying the following property: Let D be a duplication in $O_{A\to X}$ or $O_{A\to Y}$ copying a substring S. Let S_1 be the origin and S_2 be the product of D. Then D is not followed by any other operation inserting (by duplication) genes inside S_1 or S_2, or removing (by loss) genes from S_1 or S_2. We call a* **visible ancestor** *of X and Y a genome A belonging to a visible history $(A, O_{A\to X}, O_{A\to Y})$ of X and Y.*

We now define an alignment of two genomes.

Definition 2. *Let X be a string on Σ, and let Σ^- be the alphabet Σ augmented with an additional character "$-$". An* **extension of** *A is a string A^- on Σ^- such that removing all occurrences of the character "$-$" from A^- leads to the string A.*

Definition 3. *Let X and Y be two strings on Σ. An* **alignment** *of size α of X and Y is a pair (X^-, Y^-) extending (X, Y) such that $|X^-| = |Y^-| = \alpha$, and for each i, $1 \leq i \leq \alpha$, the two following properties hold:*

- *If $X_i^- \neq$ "$-$" and $Y_i^- \neq$ "$-$" then $X_i^- = Y_i^-$;*
- *X_i^- and Y_i^- cannot be both equal to "$-$".*

Let $\mathcal{A} = (X^-, Y^-)$ be an alignment of X and Y of size α. It can be seen as a $2 \times \alpha$ matrix, where the ith column $\mathcal{A}_i$ of the alignment is just the ith column of the matrix. A column is a **match** iff it does not contain the character '-', and a **gap** otherwise. A gap $\begin{bmatrix} X_i \\ - \end{bmatrix}$ is either part of a loss in Y, or part of a duplication in X (only possible if the character X_i is a duplicate in X). The same holds for the column $\begin{bmatrix} - \\ Y_j \end{bmatrix}$. An interpretation of $\mathcal{A}$ as a sequence of duplications and losses is called a **labeling** of $\mathcal{A}$. The **cost of a labeled alignment** is the sum of costs of all underlying operations.

As duplications and losses are asymmetric operations that are applied explicitly to one of the two strings, each labeled alignment $\mathcal{A}$ of X and Y leads to a unique common ancestor A for X and Y. The following theorem shows that this, and the converse is true. See Figure 1 for an example.

Theorem 1. *Given two genomes X and Y, there is a one-to-one correspondence between labeled alignments of X and Y and visible ancestors of X and Y.*

Proof. Let A be a visible ancestor of X and Y and $(A, O_{A\to X}, O_{A\to Y})$ be a visible history of X and Y. Then construct a labeled alignment $\mathcal{A}$ of X and Y as follows:

1. *Initialization*: Define the two strings $X^- = Y^- = A$ on Σ^-, and define $\mathcal{A}$ as an alignment with all matches between X^- and Y^- (*i.e.* self-alignment of A).
2. Consider each operation of $O_{A\to X}$ in order.
   - If it is a duplication, then add the inserted string at the appropriate position in X^-, and add gaps ("$-$" characters) at the corresponding positions in Y^-. Label the inserted columns of $\mathcal{A}$ as a duplication in X, coming from the columns of the alignment representing the origin of the duplication.
   - If it is a loss, then replace the lost characters in X^- by gaps. Label the modified columns as a loss in X.
3. Consider each operation of $O_{A\to Y}$ and proceed in a symmetrical way.

As $(A, O_{A\to X}, O_{A\to Y})$ is a visible history of X and Y, by definition the origins and products of duplications remain unchanged by subsequent operations on each of $O_{A\to X}$ and $O_{A\to Y}$. Therefore, all intermediate labellings remain valid in the final alignment. Therefore, the constructive method described above leads to a labeled alignment of X and Y.

On the other hand, a substring $X_i \cdots X_j$ (resp. $Y_i \cdots Y_j$) that is labeled as a duplication in X (resp. Y) should not be present in A, as it is duplicated on the branch from A to X (resp. from A to Y). Also, a substring $X_i \cdots X_j$ (resp. $Y_i \cdots Y_j$) that is labeled as a loss in Y (resp. X) should be present in A, as it is lost on the branch from A to Y (resp. from A to X). This implies an obvious algorithm to reconstruct a unique ancestor. □

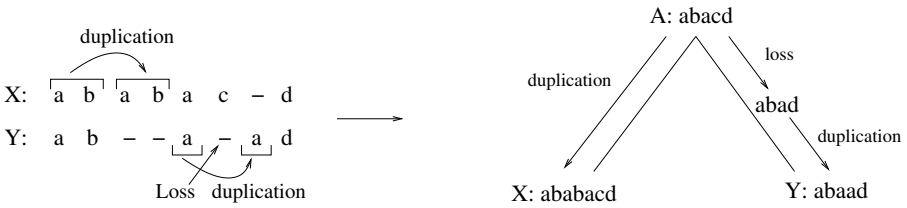


Fig. 1: Left: a labeled alignment between two strings $X = ababacd$ and $Y = abaad$. Right: the ancestor A and two histories respectively from A to X and from A to Y obtained from this alignment. The order of operations in the history from A to Y is arbitrary.

In other words, Theorem 1 states that the Two Species Small Phylogeny Problem reduces in the case of the Duplication-Loss model of evolution to the following optimization problem.

Duplication-Loss Alignment Problem:
Input: Two genomes X and Y on Σ.
Output: A labeled alignment of X and Y of minimum cost.

4 Method

Although alignments are *a priori* simpler to handle than rearrangements, a straightforward way to solve the Duplication-Loss Alignment Problem is not known. We show in the following paragraphs, that a direct approach based on dynamic programming leads, at best, to an efficient heuristic, with no guarantee of optimality.

Let X be a genome of size n and Y be a genome of size m. Denote by $X[1 \dots i]$ the prefix of size i of X, and by $Y[1 \dots j]$ the prefix of size j of Y. Let $C(i, j)$ be the minimum cost of a labeled alignment of $X[1 \dots i]$ and $Y[1 \dots j]$. Then the problem is to compute $C(m, n)$.

DP: A natural idea would be to consider a dynamic programming approach (DP), computing $C(i, j)$, for all $1 \leq i \leq n$ and all $1 \leq j \leq m$. Consider the variables $M(i, j)$, $D_X(i, j)$, $D_Y(i, j)$, $L_X(i, j)$ and $L_Y(i, j)$ which reflect the minimum cost of an alignment $\mathcal{A}_{i,j}$ of $X[1 \dots i]$ and $Y[1 \dots j]$ satisfying respectively, the constraint that the last column of $\mathcal{A}_{i,j}$ is a match, a duplication in X, a duplication in Y, a loss in X, or a loss in Y. Consider the following recursive formulae.

- $M(i,j) = \begin{cases} C(i-1,j-1) \text{ if } X[i] = Y[j] \\ +\infty \qquad\qquad\quad \text{otherwise} \end{cases}$
- $L_X(i,j) = \min_{0 \le k \le i-1}[C(k,j) + c(L(i-k))]$
  (the corresponding formula holds for $L_Y(i,j)$)
- $D_X(i,j) = \begin{cases} +\infty & \text{if } X[i] \text{ is a singleton} \\ \min_{l \le k \le i-1}[C(k,j) + c(D(i-k))] & \text{otherwise,} \end{cases}$
  where $X[l \dots i]$ is the longest suffix of $X[1 \dots i]$ that is a duplication
  (the corresponding formula holds for $D_Y(i,j)$).

The recursions for D_X and D_Y imply that duplicated segments are always inserted to the right of the origin. Unfortunately, such an assumption cannot be made while maintaining optimality of the alignment. For example, given the cost $c(D(k)) = 1$ and $c(L(k)) = k$, the optimal labeled alignment of $S_1 = abxabxab$ and $S_2 = xabx$ aligns ab of S_2 with the second ab of S_1, leading to an optimal history with two duplications inserting the second ab of S_1 to its left and to its right. Such an optimal scenario cannot be recovered by **DP**.

DP-2WAY: As a consequence of the last paragraph, consider the two-way dynamic programming approach DP-2WAY that computes $D_X(i,j)$ (resp. $D_Y(i,j)$) by looking for the longest suffix of $X[1 \dots i]$ (resp. $Y[1 \dots j]$) that is a duplication in the whole genome X (resp. Y). Unfortunately, DP-2WAY may lead to invalid cyclic evolutionary scenarios, as the same scenario may involve two duplications: one with origin S_1 and product S_2, and one with origin S_2 and product S_1, where S_1 and S_2 are two duplicated strings. This is described in more detail in Section 4.1.

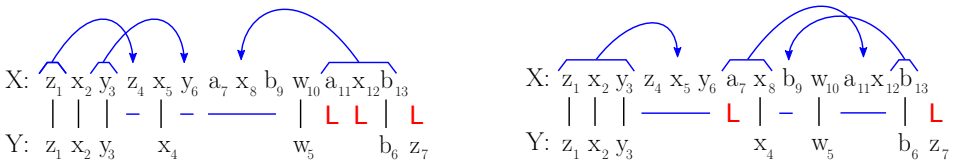
DP-2WAY-UNLABELED: The problem mentioned above with the output of DP-2WAY is not necessarily the alignment itself, but rather the label of the alignment. As a consequence, one may think about a method, DP-2WAY-UNLABELED, that would consider the unlabeled alignment output by DP-2WAY, and label it in an optimal way (*e.g.* find an evolutionary scenario of minimum cost that is in agreement with the alignment). Notice first that the problem of finding a most parsimonious labeling of a given alignment, is not *a priori* an easy problem, and there is no direct and simple way to do it. Moreover, although DP-2WAY-UNLABELED is likely to be a good heuristic algorithm to the Duplication-Loss Alignment Problem, it would not be an exact algorithm, as an optimal cyclic alignment is not guaranteed to have a valid labeling leading to an optimal labeled alignment. Figure 2 shows such an example; an optimal cyclic duplication and loss scenario can be achieved by both alignments (5 operations), while the optimal acyclic scenario can only be achieved by the alignment of Figure 2b.

4.1 The Pseudo-Boolean Linear Program

Consider genome X of length n and genome Y of length m. We show how to compute a labeled alignment of X and Y by use of pseudo-boolean linear programming (PBLP). The alignment that we compute is guaranteed to be optimal. While in the worst case our program could take an exponential number (in the length of the strings) of steps to find the alignment, our formulation has a cubic number of equations variables, and is far more efficient

than scoring all possible alignments along with all possible duplication/loss scenarios. We show that practical running times can be achieved on real data in Section 6.2.

For any alignment, an element of the string X could be considered a loss (this corresponds to gaps in the alignment), a match with an element of Y, or a duplication from another element in X (these also appear as gaps in the alignment). Thus, in a feasible solution, every element must be "covered" by one of those three possibilities. The same holds for elements of Y. Figure 2 shows two possible alignments for a given pair of strings, along with the



(a) 3 duplications and 3 losses (b) 3 duplications and 2 loss

Fig. 2: Alignments for strings X = "zxyzxyaxbwaxb" and Y = "zxyxwb". We consider the following cost: $c(D(k)) = 1$ and $c(L(k)) = k$ for any integer k. Matches are denoted by a vertical bar, losses denoted by an "L", and duplications denoted by bars, brackets, and arrows. Alignment (a) yields 6 operations and implies ancestral sequence "zxyxwaxbz", while (b) yields 5 operations and implies ancestral sequence "zxyaxwbz".

corresponding set of duplications and losses. In the alignment of Figure 2a, character x_8 is covered by the duplication of x_{12}, character a_{11} is covered by a loss, and character x_5 is covered by a match with character x_4 in Y.

Let M^i_j signify the match of character X_i to character Y_j in the alignment. Say character X_i could be covered by matches $M^i_1, M^i_2, \ldots, M^i_{p_i}$ or by duplications $DX^i_1, DX^i_2, \ldots, DX^i_{s_i}$. If we consider each of those to be a binary variable (can take value 0 or 1) and take the binary variable LX^i as corresponding to the possibility that X_i is a loss, then we have the following equation to ensure that character X_i is covered by exactly one operation:

$$LX^i + M^i_1 + M^i_2 + \cdots + M^i_{p_i} + DX^i_1 + DX^i_2 + \cdots + DX^i_{s_i} = 1, \tag{1}$$

where p_i and s_i are the number of matches and duplications that could cover character X_i. A potential duplication in X (DX^i_l for some l) corresponds to a pair of distinct, but identical, substrings in X. Each pair of such substrings yields two potential duplications (substring A was duplicated from substring B or substring B was duplicated from substring A). Each of the possible $O(n^3)$ duplications gets a variable. Each position in Y gets a similar equation to Equation 1.

The order of the matches with respect to the order of the strings must be enforced. For example, in Figure 2 it is impossible to simultaneously match w_{10} and x_{12} from X with w_5 and x_4 of string Y; the assignment of variables corresponding to this case must be forbidden in the program. Thus, we introduce equations enforcing the order of the matches. Recall that M^i_j is the variable corresponding to the match of the ith character from X with the jth character from Y. The existence of match M^i_j implies that any match M^k_l where $k \geq i$

and $l \leq j$ (or $k \leq i$ and $l \geq j$) is impossible, so must be forbidden by the program. The constraints can be written as:

$$M_j^i + M_{l_1}^{k_1} \leq 1,\ M_j^i + M_{l_2}^{k_2} \leq 1, \ldots,\ M_j^i + M_{l_{t_i}}^{k_{t_i}} \leq 1 \tag{2}$$

where t_i is the number of matches conflicting with M_j^i, and for any $M_{l_u}^{k_u}$ we have either $k_u \geq i$ and $l_u \leq j$, or $k_u \leq i$ and $l_u \geq j$. There are at most a linear number of inequalities for each of the possible $O(n^2)$ matches.

Equality 1 ensures that each character will be covered by exactly one D, M, or L. Our objective function minimizes some linear combination of all Ls and all Ds:

$$min\ \ c_1 LX^1 + \cdots + c_n LX^n + c_{n+1} LY^1 + \cdots + c_{n+m} LY^m + c_{n+m+1} D_1 + \cdots + c_{n+m+q} D_q \tag{3}$$

where c_l is a cost of the lth operation and q is the total number of duplications for all positions of X and Y (*i.e.* $D_l = DX_s^i$ or $D_l = DY_r^j$ for some i, j, s, and r). The full PBLP (excluding the trivial integrality constraints) is then:

$$\begin{aligned}
\min\quad & c_1 LX^1 + \cdots + c_n LX^n + c_{n+1} LY^1 + \cdots + c_{n+m} LY^m + \\
& c_{n+m+1} D_1 + c_{n+m+2} D_2 + \cdots + c_{n+m+q} D_q \\
\text{s.t.}\quad & LX^i + M_1^i + M_2^i + \cdots + M_{p_i}^i + DX_1^i + \cdots + DX_{s_i}^i = 1, \quad 0 \leq i \leq n \\
& LY^j + M_j^1 + M_j^2 + \cdots + M_j^{q_j} + DY_1^j + \cdots + DY_{r_j}^j = 1, \quad 0 \leq j \leq m \\
& M_j^i + M_{l_1}^{k_1} \leq 1,\ M_j^i + M_{l_2}^{k_2} \leq 1, \ldots, M_j^i + M_{l_{t_i}}^{k_{t_i}} \leq 1 \quad , \quad \forall i, j \text{ s.t. } X_i = Y_j \text{ and} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (k_m \geq i \text{ and } l_m \leq j), \text{or} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (k_m \leq i \text{ and } l_m \geq j)
\end{aligned}$$

In the example illustrated in Figure 2 — where X = "zxyzxyaxbwaxb" and Y = "zxyxwb" — there are 17 variables corresponding to matches, 20 variables corresponding to losses, and 24 variables corresponding to duplications.

Cyclic Duplications Recall the definition of the product of a duplication; in Figure 2b, the product of the leftmost duplication is z_4, x_5, and y_6. Consider a sequence of duplications $D_1, D_2, \ldots, D_l$ and characters $a_1, a_2, \ldots, a_l$ such that character a_i is in the product of D_i and is the duplication of a character in the product of D_{i-1} (a_1 is the duplication of some character a_0). We call this set of duplications **cyclic** if $D_1 = D_l$. Consider the set of duplications $\{D_1, D_2\}$ where D_1 duplicates the substring X_1X_2 to produce the substring X_3X_4 and D_2 duplicates the substring X_4X_5 to produce the substring X_1X_2. This implies the sequence of characters X_2, X_4, X_1, X_3 corresponding to the cyclic duplication D_1, D_2, D_1.

Theorem 2. *A solution to the PBLP of Section 4.1 that has no cyclic set of duplications is an optimal solution to the Duplication-Loss Alignment problem.*

Proof. Equation 1 ensures that each character of X is either aligned to a character of Y, aligned to a loss in Y, or the product of a duplication. The similar holds for each character of Y. Since there exists no cyclic set of duplications, then the solution given by the PBLP is a feasible solution to the Duplication-Loss Alignment problem. The minimization of Formula 3 guarantees optimality. □

However, if there does exist a cyclic duplication set, the solution given by the PBLP is not a feasible solution since the cycle implies a scenario that is impossible; the cycle implies a character that does not exist in the ancestor but does appear in X. A cyclic duplication set $\{D_1, D_2, \ldots, D_l\}$ can be forbidden from a solution of the PBLP by the inclusion of the following inequality:

$$D_1 + D_2 + \cdots + D_l \leq l - 1. \tag{4}$$

The following algorithm guarantees an acyclic solution to the PBLP. It simply runs the PBLP and each time it finds a cyclic set of duplications, it adds the corresponding constraint to forbid the set and reruns the PBLP. It is clear that the algorithm of Figure 1 guarantees an acyclic solution:

Theorem 3. *Algorithm 1 returns an optimal solution to the Duplication-Loss Alignment problem.*

Note that the constraints to forbid all possible cyclic sets of duplications, given a particular X and Y, could be added to the PBLP from the start, but in the worst case there is an exponential number of such constraints. We will see in Section 6.2 that in practice we do not have to rerun the PBLP many times to find an acyclic solution.

5 Extended models of evolution

This section shows how the PBLP may be adapted to handle additional content-modifying operations, as well as some rearrangement events. We will also consider the possibility of applying general distance methods to specific substrings for which we may want to consider less visible events. For any considered set $\mathcal{O}$ of allowed operations, the goal will be to add the appropriate equations to the PBLP in order to output an alignment reflecting a most parsimonious visible history.

Before describing our new operations, we need to define the notion of a visible history in a more general context, as Definition 1 is restricted to visible duplications. Definition 4 given bellow is general. Recall that a duplication operation has an origin and a product. In the case of rearrangement operations and losses, we consider the origin to be the empty string, and the product to be the resulting string. So in the case of inversions, the product is the inverted string, and in the case of losses both the origin and product are empty.

Definition 4. *Let X and Y be two genomes. A* **visible history** *of X and Y is a triplet $(A, O_{A\to X}, O_{A\to Y})$ where A is a potential ancestor of both X and Y, and $O_{A\to X}$ (respectively $O_{A\to Y}$) are evolutionary histories from A to X (respectively from A to Y) verifying the following property: Let O be an operation in $O_{A\to X}$. Let S_1 be the origin and S_2 be the product of O. Then O is not followed by any other operation modifying (by insertion, deletion, substitution or rearrangement) S_1 or S_2.*

5.1 Substitution

For the sake of generality, in addition to duplication, loss, and insertion (horizontal gene transfer discussed in Section 3), it seems natural to include substitution which is another

content-modifying operation representing the conversion of a gene from one function to another.

- S: A **Substitution** is an operation that replaces a character X_i at a given position i of a string X by another character Y_i.

Interestingly, applying our algorithm to the tRNA gene content in *Bacillus* (see Section 6) revealed misaligned positions that are likely to be substitutions representing tRNA functional shift.

The PBLP naturally generalizes to substitutions. In this case, a character X_i can be covered by any one of the characters of Y (*i.e.* there are exactly m match variables in Equation 1). However, the cost attached to a variable M_j^i has to be positive in case of a substitution ($X_i \neq Y_j$). Instead of attributing a constant cost for substitution it seems reasonable to allow for a cost which is dependent upon the aligned characters, corresponding to the likelihood of the particular gene conversion. It follows that the objective function of the PBLP has to be augmented with $m \times n$ terms of the form $c_{ij}M_j^i$.

Notice that the convenient property of asymmetry that holds for the Duplication and Loss model of evolution, which allows for the one-to-one correspondence between labeled alignments and visible ancestors (Theorem 1), does not hold anymore for the generalized model with substitution. Indeed, an alignment of character X_i with Y_j can be either interpreted as a substitution from X_i to Y_j or from Y_j to X_i, leading respectively to an ancestor with character X_i or Y_j.

5.2 Inversion

Although, in general, comparing two genomes based on genome rearrangement events can not be done with an alignment approach, non-overlapping inversions in a context of a visible history can be handled by our approach. Notice however that, similar to substitutions, inversions are symmetrical operations that can be applied to either one of the two genomes being considered; we lose the one-to-one correspondence between labeled alignments and visible ancestors (Theorem 1).

- I: An **Inversion** is an operation that transforms a proper substring $X[i+1 \dots i+k] = X[i+1]X[i+2] \cdots X[i+k]$ of X into its **reverse substring** $X[i+1 \dots i+k] = X[i+k] \cdots X[i+2]X[i+1]$.
- ID: An **Inverted Duplication** of size k is an operation that copies the reverse of a substring $X[i+1 \dots i+k]$ to a location j of X outside the interval $[i+1, i+k]$.

In order to handle inversions, Equation 1 is augmented with variables $IX_1^i, IX_2^i, \cdots IX_{u_i}^i$. A potential inversion (IX_l^i for some l) corresponds to a pair (X', Y'), where X' is a substring of X covering position i, and Y' is a substring of Y such that Y' is the reverse of X'. There are $O(n^3)$ such possible pairs, and thus $O(n^3)$ variables corresponding to an inversion operation for each i.

In order to handle inverted duplication, Equation 1 is augmented with variables IDX_l^i corresponding to all pairs (X', X'') of disjoint substrings of X such that X' covers position i and X'' is the reverse of X'. There are also $O(n^3)$ such pairs.

5.3 Relaxing the visibility criterion

The main restriction of our methodology imposed by the alignment strategy is the fact that only the most recent operations in a history, namely those that have not been obscured by subsequent operations, can be detected as visible in the alignment. More precisely, a duplication or inversion that is followed in the history by an operation affecting its origin or product is not detectable by our base method. In this section, we show that some relaxation of this restriction is possible.

The idea is that any substring of X can be covered by another substring X_o of X by a duplication of X_o followed by a series of operations. Alternatively, any substring of X can be covered by any substring of Y through some series of operations. For example, the substring ac of $S3 = acabc$ could be covered by the duplication of abc before a loss of b. Alternatively, ac from $S3$ could be covered by the inversion of the substring ca in $S4 = caabc$.

In general we could create $O(n^4)$ variables, each corresponding to covering a substring of X with another substring, the coefficient of the variable in the objective function being the cost of such a scenario. In practice, only certain pairs of substrings under certain sets of operations may be interesting. In this section we outline a few of the possible cases.

The DupLoss operation Consider a duplication that has been followed by a sequence of losses reducing the number of copied genes. We introduce a new operation representing such a sequence of events.

- DL: A **DupLoss** $DL(k, l)$ of duplication size k and loss size l on a string X, is an operation that copies a substring $X[i+1 \dots i+k]$ of X to a location j of X outside the interval $[i+1, i+k]$, and removes $l < k$ characters from the copied substring.

The most natural way to compute the cost for a DupLoss $DL(k, l)$ is to sum up the cost of the duplication with that of the loss events. For example, in case of a size independent loss cost, $c(DL(k, l)) = c(D(k)) + l \times c(L)$.

The generalization of the PBLP requires additional binary variables corresponding to the possibility that a character is covered by a DupLoss. More precisely, Equation 1 should be augmented with variables $DLX_1^i, DLX_2^i, \dots, DLX_{t_i}^i$. A potential DupLoss in X (DLX_l^i for some l) corresponds to origin A and product B where B is a subsequence of A, and B spans the index i. In this case, there are $O(n^4)$ possible pairs, and thus $O(n^4)$ variables corresponding to a DupLoss operation for each i.

The Inversion operation The previous section allows us to align genomes in the presence of non-overlapping inversions where the content of the inverted substring is identical in the two genomes. In this section, we introduce a relaxation on this constraint by allowing the detection of a consecutive sequence of possibly intersecting inversions.

- IB: An **Inversion Bloc** of size k on a string X is a sequence of k consecutive inversions acting on X .
- IDB: An **Inverted Duplication Bloc** of size k is a duplication D followed by an inversion bloc of size k acting on the product of D.

The most natural cost for an inversion bloc is simply the number of inversions in the bloc, while the most natural cost for an inverted duplication bloc is the cost of the duplication plus the number of inversions.

We use the notion of common interval considered in the genome rearrangement literature [5,6,30]. A pair of *common intervals* (A, B) is a pair of strings with the exact same gene content (same alphabet with the same number of copies for each character). Clearly, given two strings A and B, there is an inversion bloc transforming A into B if and only if (A, B) is a pair of common intervals.

The idea will be to interpret the alignment between a pair of common intervals (X', Y'), where X' is a substring of genome X and Y' a substring of genome Y, as resulting from an inversion bloc, with cost corresponding to the inversion distance (*i.e.* the minimum number of inversions transforming X' into Y'). Similarly, we will interpret a pair of common intervals (X', X''), where X' and X'' are two disjoint substrings of X, as an inverted duplication bloc.

In the case of X' and Y' being two permutations, computing the inversion distance between the two strings can be done in linear time [4]. Although the problem is NP-complete in the general case of substrings with multiple gene copies [2,9,13], many heuristics exist for approximating the inversion distance (see for example [15,39,41,42]).

6 Applications

6.1 Evolution of stable RNA gene content and organization in *Bacillus*

The stable RNAs are chiefly transfer RNAs (tRNAs) and ribosomal RNAs (rRNAs), which are essential in the process of translating messenger RNAs (mRNAs) into protein sequences. They are usually grouped in the genome within clusters (or operons in the case of microbial genomes), representing highly repetitive regions, causing genomic instability through illegitimate homologous recombination. In consequence, stable RNA families are rapidly evolving by duplication and loss [7,37,47].

We applied Algorithm 1 (without the extensions of Section 5) in a phylogenetic context, using the SPP-HEURISTIC described at the end of Section 2, to analyze the stable RNA content and organization of 5 *Bacillus* lineages: *Bacillus cereus ATCC 14579 (NC4722)*, *Bacillus cereus E33L (NC6274)*, *Bacillus anthracis (NC7530)*, *Bacillus licheniformis ATCC 14580 (NC6322)* and *Bacillus subtilis (NC964)*. The overall number of represented RNA families in these genomes is around 40, and the total number of RNAs in each genome is around 120. Our PBLP algorithm processes each pair of these genomes in a few seconds. We used the following cost for duplications and losses: $c(D(k)) = 1$ and $c(L(k)) = k$, for any integer k representing the size of an operation. The phylogeny in Figure 3 reflects the NCBI taxonomy. Each leaf is labeled by a block representation of the corresponding genome. Details on each colored block is given in Figure 4 of the appendix.

The costs and evolutionary scenarios given in Figure 3 are those output by our algorithm after interpretation. In particular, the five *Bacillus* genomes all show a large inverted segment in the region to the left of the origin of replication (the right part of each linearized representation in Figure 3). As the algorithm without extensions has been used, we preprocessed the genomes by inverting this segment. The genome representations given in Figure 3 are, however, the true ones. Signs given below the red bars represent their true orientations.

Consequently, the duplication of the right-most red bar to the left-most position should be interpreted as an inverted duplication that occurred around the origin of replication. On the other hand, some duplications of the red bar have been reported by our algorithm as two separate duplications of two segments separated by a single gene. After careful consideration (see Figure 5 of the appendix), these pairs of duplications are more likely a single duplication obscured by subsequent substitution, functional shift or loss of a single tRNA gene. Also, when appropriate (*e.g.* when a lone gene is positioned in a lone genome), we interpreted some of the losses in our alignments as insertions.

The ancestral genomes given in Figure 3 are those output by our algorithm. They reflect two separate inverted duplications that would have occurred independently in each of the two groups (*cereus*, *anthracis*) and (*licheniformis*, *subtilis*). We could alternatively infer that the ancestral *Bacillus* genome already contained both the origin and product of the inverted duplication. The consequence would be the simultaneous loss of the leftmost red bar in three of the five considered genomes. Moreover, nucleotide sequence alignment of the red bars in *subtilis* and *cereus ATCC* reveal a higher conservation of pairs of paralogous bars versus orthologous ones, which may indicate that inverted duplications are recent. Whatever the situation is, our inference of a duplication around the origin of replication is in agreement with the observation that has been largely reported in the literature that bacterial genomes have a tendency to preserve a symmetry around the replication origin and terminus [21,45,1]. The results also show independent proliferation of ribosomal RNA gene-containing operons in *Bacillus*, which has been associated to selection for increased growth rate [3]. They also show that in *Bacillus*, growth-selection on ribosomal RNA operon expansions may significantly alter tRNA gene content as well. The results given in Figure 5 of the appendix also suggest that some tRNA genes may have been affected by substitutions leading to conversions of function. Such tRNA functional shifts have been detected in metazoan mitochondrial genomes [36] and bacteria [38].

6.2 Execution time

Running times were recorded using a 12-core AMD 2.1GHZ processor, with 256GB of RAM, and the (multithreaded) IBM CPLEX solver under the default settings. Note that a significant amount of memory ($>$ 1GB) was required only for sequences of several thousand genes; all tests reported here could be run on a standard laptop with 2 GB of memory. Alignments of all pairs of genomes for each of three sets of stable RNA gene orders were computed. The average computation time for the *Bacillus* pairs was under thirty seconds. The average computation time for pairs from 13 *Staphylococcus* was under a second. Pairs from a dataset of *Vibrionaceae* which had a very high number of paralogs and a large number of rearrangements took a couple of days.

6.3 Simulations

Simulations were run in order to explore the limits of our method (full results not shown due to space limitations). A random sequence R was drawn from the set of all sequences of length n and alphabet size a. l moves were then applied to R to obtain the ancestral sequence A. To obtain the extant sequences X and Y, l more moves were applied to A for each. The set

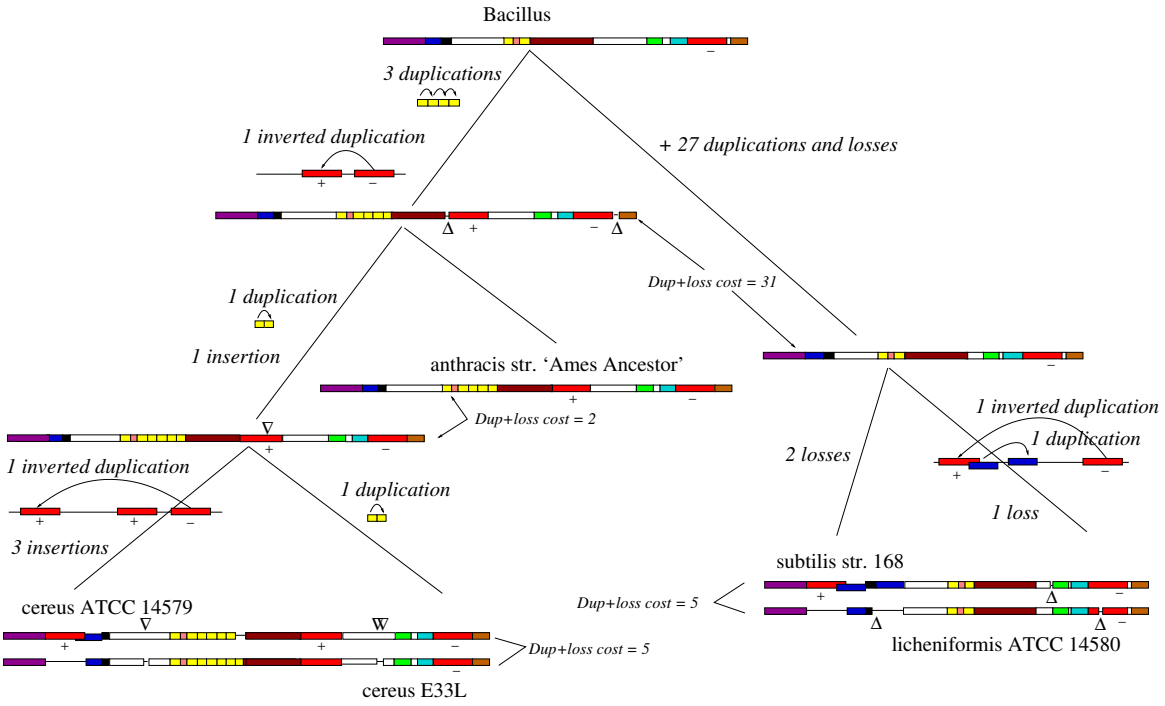


Fig. 3: An inferred evolutionary history for the five *Bacillus* lineages identified with each of the five leaves of the tree. Circular bacterial genomes have been linearized according to their origin of replication (*e.g.* the endpoints of each genome is its origin of replication). Bar length is proportional to the number of genes in the corresponding cluster. A key for the bars is given in Figure 4, except for white bars that represent regions that are perfectly aligned inside the two groups (*cereus*, *anthracis*) and (*licheniformis*, *subtilis*), but not between the two groups. More specifically, the 27 duplications and losses reported on the top of the tree are obtained from the alignment of these white regions. Finally, each Δ represents a loss and each ∇ is an insertion.

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 observations on *Bacillus* and *Staphylococcus*. Average running times for sequences with a fixed ratio of $2l/n = 1/5$ and $a/n = 1/2$ (statistics similar to those observed in *Bacillus*) were always below 6 minutes for $n < 800$. Sequences of length 2000 took less than 2 hours and sequences of length 5000 took a couple of days. When varying l, n, and a the most telling factor for running time was the ratio a/n. This explains the high running times for the set of *Vibrionaceae* which had, on average, nearly 100 moves for a sequence of length 140.

The distance of our computed ancestor to the simulated ancestor was found by computing an alignment between the two. For values of $n = 120$, $a = 40$, and $l = 15$ (values that mimic the statistics of more distant pairs of the *Bacillus* data) we compute ancestors that are, on average, 5 moves away from the true ancestor. In general, for sequences with ratios $2l/n = 1/5$ and $a/n = 1/2$, the average distance to the true ancestor stays at about 15% of l.

7 Conclusion

We have considered the two species small phylogeny problem for an evolutionary model reduced to non-overlapping (visible) content-modifying operations. We provided some extensions to the model by incorporating overlapping operations, and inversions. Although exponential running times are possible in the worst case, our pseudo-boolean linear programming algorithm turns out to be fast on real datasets, such as the RNA gene repertoire of bacterial genomes. We have also explored avenues for developing efficient non-optimal heuristics. As described in Section 4, a dynamic programming approach can be used to infer a reasonable, though not necessarily optimal unlabeled alignment of two genomes, or a labeled but possibly cyclic alignment. A recent investigation on the complexity of these problems has revealed that the minimum labeling alignment problem is APX-hard [18]. In future work the apparently easy "core" of finding a possibly cyclic alignment may be leveraged to use the Lagrangian relaxation technique — in this case each constraint against a cyclic duplication set would correspond to a modification of the objective function in our dynamic program.

Application to the RNA gene content of five *Bacillus* lineages has pointed out a number of interesting biological mechanisms that have to be further investigated. Probably the most interesting observations are the substitutions occurring among the tRNAs. Indeed, as illustrated in Figure 5, some positions indicate a shift of the anticodon, and thus potentially a shift of the original function of the tRNA. Alternatively, such divergence may just be an indication that the tRNA is in the process of losing its function and becoming a pseudogene. Taking into consideration not only the anticodon but rather the entire tRNA sequence, and comparing with additional lineages, may allow to conclude to one or the other alternative. Also other investigations on sequence alignment are required to test our hypothesis that two inverted duplications around the origin of replication have occurred independently on each group (*cereus*, *anthracis*) and (*licheniformis*, *subtilis*), rather than a single duplication preceding the *Bacillus* ancestor, followed by losses of the same regions in each of the two monophyletic groups.

References

1. Y. Ajana, J.F. Lefebvre, E. Tillier, and N. El-Mabrouk. Exploring the set of all minimal sequences of reversals - an application to test the replication-directed reversal hypothesis. In *LNCS*, volume 2452 of *WABI*, pages 300–315, 2002.
2. S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. A general framework for computing rearrangement distances between genomes with duplicates. *Journal of Computational Biology*, 14:379–393, 2007.
3. D.H. Ardell and L.A. Kirsebom. The genomic pattern of tDNA operon expression in *e. coli*. *PLoS Comp. Biol.*, 1(1:e12), 2005.
4. D.A. Bader, B.M.E. Moret, and M. Yan. A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comput. Biol.*, 8(5):483–491, 2001.
5. A. Bergeron, C. Chauve, and Y. Gingras. Formal models of gene clusters. In I. Mandoiu and A. Zelikovsky, editors, *Bioinformatics algorithms: techniques and applications*, chapter 8. Wiley, 2008.
6. A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. *Journal of Computational Biology*, 13:1340–1354, 2003.
7. C. Bermudez-Santana, C. S. Attolini, T. Kirsten, J. Engelhardt, S.J. Prohaska, S. Steigele, and P. Stadler. Genomic organization of eukaryotic tRNAs. *BMC Genomics*, 11(270), 2010.
8. M. Blanchette, G. Bourque, and D. Sankoff. Breakpoint phylogenies. Genome Informatics Workshop (GIW), pages 25- 34, 1997.

9. G. Blin, C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Comparing genomes with duplications: a computational complexity point of view. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:523–534, 2007.
10. T. Blomme, K. Vandepoele, S. De Bodt, C. Silmillion, S. Maere, and Y. van de Peer. The gain and loss of genes during 600 millions years of vertebrate evolution. *Genome Biology*, 7:R43, 2006.
11. G. Bourque and P.A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Research*, 12:26 – 36, 2002.
12. A. Caprara. Formulations and hardness of multiple sorting by reversals. In *RECOMB*, pages 84–94, 1999.
13. C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Genomes containing duplicates are hard to compare. In *Computational Science (ICCS 2006)*, volume 3992 of *Lecture Notes in Computer Science*, pages 783–790, 2006.
14. C. Chauve and E. Tannier. A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. *PloS Computational Biology*, 4:e1000234, 2008.
15. Z. Chen, B. Fu, and B. Zhu. The approximability of the exemplar breakpoint distance problem. In *AAIM*, volume 4041 of *Lecture Notes in Computer Science*, pages 291- 302, 2006.
16. J.A. Cotton and R.D.M. Page. Rates and patterns of gene duplication and loss in the human genome. *Proceedings of the Royal Society of London. Series B*, 272:277–283, 2005.
17. J.P. Demuth, T. De Bie, J. Stajich, N. Cristianini, and M.W. Hahn. The evolution of mammalian gene families. *PLoS ONE*, 1:e85, 2006.
18. R. Dondi and N. El-Mabrouk. On the complexity of minimum labeling alignment of two genomes. Arxiv, 2012.
19. H. Dong, L. Nilsson, and C. G. Kurland. Co-variation of tRNA abundance and codon usage in Escherichia coli at different growth rates. *Journal of Molecular Biology*, 260:649-663, 2006.
20. E.E. Eichler and D. Sankoff. Structural dynamics of eukaryotic chromosome evolution. *Science*, 301:793–797, 2003.
21. J.A. Eisen, J.F. Heidelberg, O. White, and S.L. Salzberg. Evidence for symmetric chromosomal inversions around the relication origin in bacteria. *Genome Bioloy*, 1(6), 2000.
22. N. El-Mabrouk. *Mathematics of Evolution and Phylogeny*, chapter Genome rearrangement with gene families, pages 291- 320. Oxford University Press, 2005.
23. G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of genome rearrangements*. The MIT Press, Cambridge, Massachusetts and London, England, 2009.
24. M.W. Hahn, M.V. Han, and S.-G. Han. Gene family evolution across 12 *drosophilia* genomes. *PLoS Genetics*, 3:e197, 2007.
25. S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, pages 581–592, 1995.
26. S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Journal of the ACM*, 48:1–27, 1999.
27. Minghui Jiang. The zero exemplar distance problem. In *RECOMB-CG*, pages 74–82, 2010.
28. S. Kanaya, Y. Yamada, Y. Kudo, and T. Ikemura. Studies of codon usage and tRNA genes of 18 unicellular organisms and quantification of Bacillus subtilis tRNAs: Gene expression level and species-specific diversity of codon usage based on multivariate analysis. *Gene*, 238:143-155, 1999.
29. J. Kovac, B. Brejova, and T. Vinar. A practical algorithm for ancestral rearrangement reconstruction. In *LNBI*, volume 6833 of *WABI*, pages 163- 174, 2011.
30. G.M. Landau, L. Parida, and O. Weimann. Gene proximity analysis across whole genomes via PQ trees. *Journal of Computational Biology*, 12:1289– 1306, 2005.
31. M. Lynch and J.S. Conery. The evolutionary fate and consequences of duplicate genes. *Science*, 290:1151-1155, 2000.
32. J. Ma, L. Zhang, B.B. Suh, B.J. Raney, R.C. Burhans, W.J. Kent, M. Blanchette, D. Haussler, and W. Miller. Reconstructing contiguous regions of an ancestral genome. *Genome Research*, 16:1557 – 1565, 2007.
33. B. Moret, L. Wang, T. Warnow, and S. Wyman. New approaches for reconstructing phylogenies from gene order data. *Bioinformatics*, 17:S165–S173, 2001.
34. S. Ohno. *Evolution by gene duplication*. Springer, Berlin, 1970.
35. I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Elec. Colloq. on Comput. Complexity*, 71, 1998.
36. T.A. Rawlings, T.M. Collins, and R. Bieler. Changing identities: trna duplication and remolding within animal mitochondrial genomes. *Proceedings of the National Academy of Sciences USA*, 100:15700-15705, 2003.
37. H.H. Rogers, C.M. Bergman, and S. Griffiths-Jones. The evolution of tRNA genes in Drosophila. *Genome Biol. Evol*, 2:467- 477, 2010.
38. M.E. Saks and J.S. Conery. Anticodon-dependent conservation of bacterial tRNA gene sequences. *RNA*, 13(5):651- 660, 2007.

39. D. Sankoff. Genome rearrangements with gene families. *Bioinformatics*, 15:909–917, 1999.
40. D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In T. Jiang and D.T. Lee, editors, *Computing and Combinatorics, Proceeedings of COCOON '97*, number 1276 in Lecture Notes in Computer Science, pages 251–263, Berlin, 1997. Springer.
41. G. Shi, L. Zhang, and T. Jiang. MSOAR 2.0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement. *BMC Bioinformatics*, 11:10, 2010.
42. K.M. Swenson, M. Marron, J.V. Earnest-DeYoung, and B.M.E. Moret. Approximating the true evolutionary distance between two genomes. In *Proc. 7th SIAM Workshop on Algorithm Engineering & Experiments (ALENEX'05)*, pages 121–129. SIAM Press, Philadelphia, 2005.
43. D.T. Tang, E.A. Glazov, S.M. McWilliam, W.C. Barris, and B.P. Dalrymple. Analysis of the complement and molecular evolution of tRNA genes in cow. *BMC Genomics*, 10(188), 2009.
44. E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, 10, 2009.
45. E.R.M. Tillier and R.A. Collins. Genome rearrangement by replication-directed translocation. *Nature Genetics*, 26, 2000.
46. I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev. Natural history and evolutionary principles of gene duplication in fungi. *Nature*, 449:54–61, 2007.
47. M. Withers, L. Wernisch, and M. Dos Reis. Archaeology and evolution of transfer RNA genes in the *escherichia coli* genome. *Bioinformatics*, 12:933-942, 2006.

8 Alignment Details

Ribosomal RNAs are grouped in bacteria into three families: the 16S, 5S and 23S rRNAs. As the major role of tRNAs is to serve as adapters between codons along the mRNA and the corresponding amino acids, we group them according to their anticodon. More precisely, the four-letter designation starts with one letter indicating functional class (either an IUPAC one-letter code for a charged amino acid, "X" for initiator or "J" for a special class of isoleucine tRNA) followed by an anticodon sequence in a DNA alphabet. The full alignment as given by our program can be retrieved at: http://lcbb4.epfl.ch/web/BacillusAlignements/.

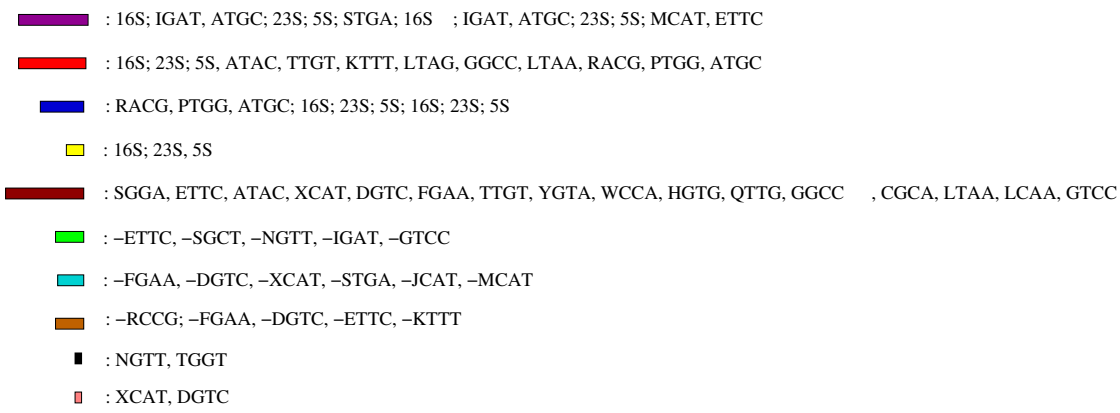


Fig. 4: The RNA gene clusters represented by each colored bar of Figure 3. Ribosomal RNAs are identified by their families' names: 16S, 23S, 5S. Each tRNA is identified by its anticodon preceded by the letter corresponding to the corresponding amino-acid. The symbol ',' (respectively ';') separates two genes that are inside (resp. not inside) the same operon.

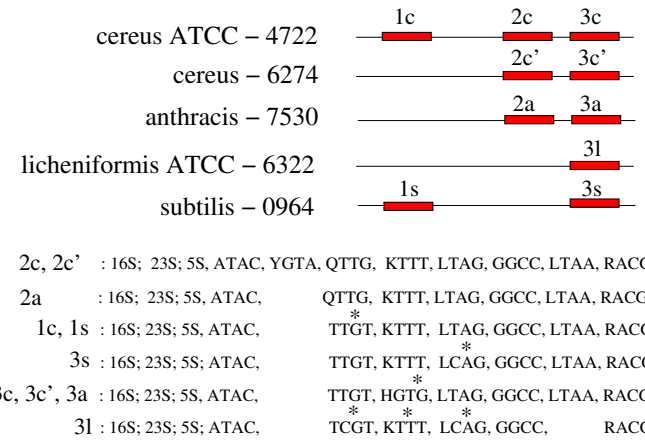


Fig. 5: (a) Genomes of Figure 3 restricted to their red bars; (b) An alignment of all red bars, reflecting one gene insertion (or alternatively 5 gene losses) and substitutions, indicated by '*'.

Algorithm 1 Pairwise-Alignment(PBLP)

get solution S to the PBLP
while S has a cycle **do**
  for each cycle $D_1 + D_2 + \cdots + D_l$ in S **do**
    PBLP $\leftarrow$ PBLP plus constraint $D_1 + D_2 + \cdots + D_l \leq l - 1$
  end for
  get solution S to the PBLP
end while
return S