Comparing genomes with rearrangements and segmental duplications

Mingfu Shao* and Bernard M.E. Moret*

School of Computer and Communication Sciences, EPFL, CH-1015, Lausanne, Switzerland

*To whom correspondence should be addressed.

Abstract

Motivation: Large-scale evolutionary events such as genomic rearrangements and segmental duplications form an important part of the evolution of genomes and are widely studied from both biological and computational perspectives. A basic computational problem is to infer these events in the evolutionary history for given modern genomes, a task for which many algorithms have been proposed under various constraints. Algorithms that can handle both rearrangements and content-modifying events such as duplications and losses remain few and limited in their applicability.

Results: We study the comparison of two genomes under a model including general rearrangements (through double-cut-and-join) and segmental duplications. We formulate the comparison as an optimization problem and describe an exact algorithm to solve it by using an integer linear program. We also devise a sufficient condition and an efficient algorithm to identify optimal substructures, which can simplify the problem while preserving optimality. Using the optimal substructures with the integer linear program (ILP) formulation yields a practical and exact algorithm to solve the problem. We then apply our algorithm to assign in-paralogs and orthologs (a necessary step in handling duplications) and compare its performance with that of the state-of-the-art method MSOAR, using both simulations and real data. On simulated datasets, our method outperforms MSOAR by a significant margin, and on five well-annotated species, MSOAR achieves high accuracy, yet our method performs slightly better on each of the 10 pairwise comparisons.

Availability and implementation: http://lcbb.epfl.ch/softwares/coser.

Contact: mingfu.shao@epfl.ch or bernard.moret@epfl.ch

1 Introduction

In addition to the point mutations (single base-pair substitutions, insertions and deletions), in the course of evolution, genomes also undergo many large-scale events, which are usually divided into two categories, rearrangements and content-modifying events. Genome rearrangements include inversions, transpositions, circularizations and linearizations, all of which act on a single chromosome, and translocations, chromosomal fusions and fissions, which act on two chromosomes. Rearrangements can shuffle the order and switch the transcriptional orientations of the genes on chromosomes but cannot change the number of gene copies. On the other hand, the content-modifying events, which include segmental duplications, tandem duplications, gene insertions and losses, can affect the copy number of the genes. These two types of large-scale events are ubiquitous in the tree of life and have been shown playing a very important role in the variations of the individual traits. The molecular mechanisms behind them, although have been widely studied, are still very diverse [see Gu et al. (2008) for a review].

One basic task of comparative genomics is to infer the events that took place in the evolutionary history for the extant species. Many combinatorial optimization problems aiming to compute the most parsimonious number of events between two given genomes (i.e. the edit distance) are formulated, and many algorithms, heuristics or exact ones, are proposed for them. When only rearrangement events are considered, Hannenhalli and Pevzner (1995) gave the first polynomial-time algorithm to compute the inversion distance, which was later improved to linear time (Bader et al., 2001). Yancopoulos et al. (2005) proposed a universal operation, called double-cut-and-join (DCJ), which can unite most of the rearrangement events. Under the DCJ model, the edit distance can also be computed in linear time, but in a more simple and elegant way (Bergeron et al., 2006). Because of its simplicity, DCJ model has formed the basis for the following algorithmic research on rearrangements (Bergeron et al., 2009; Chen, 2010; Moret et al., 2013).

All of the above efficient algorithms assume that genomes do not contain duplicated genes. In the presence of duplicated genes, most
of the edit distance problems are NP-hard. For two genomes with duplicated genes, Chen et al. (2005) proposed an efficient heuristic to compute the inversion distance by decomposing the problem into two new optimization problems. Shao et al. (2014) devised an exact algorithm to compute the DCJ distance by formulating the problem as an integer linear program. Both of the methods output a one-to-one correspondence between the homologous genes and thus can be applied to assign orthologs.

When only content-modifying events are considered, Kahn and Raphael (2008) devised an efficient dynamic programming algorithm to compute the duplication distance, which was later extended by introducing likelihood techniques and then applied to reconstruct the evolutionary history of the segmental duplications in human genome (Kahn et al., 2010). Holloway et al. (2013) proposed an alignment approach to reconstruct the ancestral genome for two genomes with segmental duplications and gene losses and applied it in a phylogenetic context to infer the evolution of the stable RNA gene content and organization in various genomes.

When both rearrangements and content-modifying events are considered, El-Mabrouk (2001) proposed an efficient algorithm to compute the edit distance for inversions and deletions. Braga et al. (2010, 2011) gave a linear time algorithm to compute the edit distance for DCJs, insertions and deletions. Notice that these algorithms also assume that the given genomes do not contain duplicated genes. Shao and Lin (2012) gave a 1.5-approximation algorithm to compute the edit distance for two genomes in the presence of duplicated genes under a model that includes DCJs, single-gene insertions and single-gene deletions. Fu et al. (2007) extended the heuristics in Chen et al. (2005) to unite rearrangements and single-gene duplications as a new software package, called MSAOAR, which can be applied to detect in-paralogs in addition to orthologs.

In this article, we compare two genomes in the presence of duplicated genes with DCJs and segmental duplications. Formally, the problem is to compute a set of segmental duplications in each genome and a bijection between the nonduplicated genes, such that the total cost of the segmental duplications and the DCJs induced by the bijection is minimized. We propose an exact algorithm for this problem by formulating it as an integer linear program. Based on studying the underlying structure of problem, we then devise an efficient preprocessing algorithm to simplify the problem while keeping the optimality. We also discuss and propose a reasonable way to balance the costs between DCJs and segmental duplications. Finally, we apply our method to assign in-paralogs and orthologs and compare its performance with MSAOAR on both simulated and biological datasets.

2 Problem statement

We model each genome as a set of chromosomes and model each chromosome as a linear or circular list of genes. Each gene is represented by a signed (+ or −) symbol, where the sign indicates the transcriptional direction of this gene. Homologous genes are grouped into gene families. For a genome X, we use A(X) to denote all the gene families in X and use F(X, f) to denote the set of genes in X that come from gene family f.

We say consecutive genes on one chromosome form a segment. The length of a segment s is defined as the number of genes in s, denoted by |s|. We say two segments in the same genome are independent if they do not contain the same gene. We say genes s = (a1, a2, · · · , a1) and t = (b1, b2, · · · , b1) are homologous if a1 and b1 are homologous and have the same sign for all 1 ≤ i ≤ n or a1 and b1+1−i are homologous and have the opposite sign for all 1 ≤ i ≤ n.

We say segment s is possibly duplicated, if there exists segment t in the same genome such that s and t are independent and homologous. For a genome X, we use S(X) to denote the set of all the possibly duplicated segments in X (Fig. 1a). We say a subset S ⊂ S(X) is independent if every two segments in S are independent. For an independent subset S ⊂ S(X), we use X \ S to denote the new genome after removing all genes appearing in the segments in S from X. Given two genomes X and Y, we say two independent subsets S ⊂ S(X) and T ⊂ S(Y) are consistent if X \ S and Y \ T have the same gene content, i.e. for each gene family f ∈ A(X) ∪ A(Y), we have |F(X \ S, f)| = |F(Y \ T, f)| (Fig. 1a). In this article, we assume that the given two genomes X and Y satisfy that A(X) = A(Y); otherwise we modify them by removing all the genes that are not in A(X) ∩ A(Y). With this assumption, there always exist two independent subsets S ∈ S(X) and T ∈ S(Y) that are consistent.

Suppose we are given two independent consistent subsets S ∈ S(X) and T ∈ S(Y). We denote by B(X \ S, Y \ T) the set of bijections that map each gene in X \ S to a homologous gene in Y \ T. If X \ S and Y \ T contain only singletons, i.e. we have |F(X \ S, f)| = |F(Y \ T, f)| = 1 for all f ∈ A(X) ∪ A(Y), then we have |B(X \ S, Y \ T)| = 1, and the DCJ distance between X \ S and Y \ T is well defined and can be computed in linear time (Bergeron et al., 2006). Once a bijection B ∈ B(X \ S, Y \ T) is given, we can relabel X \ S and Y \ T by assigning each pair of genes in B with a distinct gene family and thus results in two new genomes with only singletons. We denote by d(B) the DCJ distance between these two new genomes induced by bijection B.

In this article, we study the following problem: given two genomes X and Y satisfying A(X) = A(Y), and a cost function c(·), which maps each segment in S(X) ∪ S(Y) to a positive value, compute a triple Q = (S, T, B), where S ⊂ S(X) and T ⊂ S(Y) are two independent consistent subsets and B ∈ B(X \ S, Y \ T), such that the total cost of Q, i.e. Q = \sum s∈S,T \ c(s) + d(B), is minimized.

3 ILP formulation

We now formulate the above problem as an integer linear program. To achieve that, we first introduce the adjacency graph in Section 2.
3.1, which is the essential data structure to compute the DCJ distance. We also propose a new extension of the adjacency graph, called the extended adjacency graph, which can incorporate duplicated genes and thus forms the basis for the following ILP formulation. We then describe a capping method to remove the telomeres, in Section 3.2, which allows us only to count the number of cycles when computing the DCJ distance. On the basis of them, we finally give the ILP formulation in Section 3.3.

3.1 Adjacency graph

We first introduce some notations. The two ends of a gene $a$ are called extremities. The head is denoted by $ah$, and the tail is denoted by $at$. The set of all extremities in genome $X$ is called the extremity set of $X$, denoted by $E(X)$. If genes $a$ and $b$ are homologous, we also say the two corresponding extremity pairs, $ah_a$ and $at_a$, $ah_b$ and $at_b$, are homologous. Two consecutive genes $a$ and $b$ form one adjacency, which is represented by a set of two extremities. Thus, each adjacency comes in one of the four types: $\{ah_a, at_b\}$, $\{ah_b, at_a\}$, $\{ah_a, at_a\}$, and $\{ah_b, at_b\}$. If a gene $a$ lies at one end of a linear chromosome, then this end can be represented by a set of one extremity, $\{ah_a\}$ or $\{at_b\}$, called a telomere.

Suppose that we are given a triple $Q = (S, T, B)$, where $S \subset S(X), T \subset S(Y)$ are two independent consistent subsets and $B \in B(X \setminus S, Y \setminus T)$. We can build the adjacency graph w.r.t. $Q$, denoted by $G(Q)$, as follows. We first build $X \setminus S$ and $Y \setminus T$ through removing all genes in $S \cup T$ and take all the extremities in them, i.e. $E(X \setminus S) \cup E(Y \setminus T)$, as the vertices of $G(Q)$. Then for each adjacency in $X \setminus S$ and $Y \setminus T$, we add one black edge to connect the two extremities in it. Finally, for each pair of homologous extremities specified by $B$ (each homologous gene pair in $B$ specifies two pairs of homologous extremities), we add one black edge to connect them (Fig. 1a–c). Clearly, in $G(Q)$, the degree of each vertex is at most 2, and thus it consists of a set of vertex-disjoint cycles and paths. The length of a cycle (or a path) is defined as the number of black edges in it. Let $c$ be the number of cycles and $o$ be the number of odd-length paths in $G(Q)$. We have that the DCJ distance induced by $B$ can then be computed as $d(B) = n - c - o/2$, where $n$ is the number of genes in $X \setminus S$ (Bergeron et al., 2006).

Given a triple $Q = (S, T, B)$ defined above, we propose an equivalent form of $G(Q)$, called the extended adjacency graph w.r.t. $Q$, denoted by $G^\ast(Q)$. The set of vertices of $G^\ast(Q)$ includes all the extremities in $X$ and $Y$, i.e. $E(X) \cup E(Y)$. For each adjacency in $X$ and $Y$, there is one gray edge connecting the two extremities in it. For each pair of homologous extremities specified by $B$, there are two extremities connecting them. For each gene contained in some segment in $S \cup T$, there is one internal edge connecting the two extremities in this gene (Fig. 1d). The difference between $G^\ast(Q)$ and $G(Q)$ is that, the latter one explicitly removes those extremities in the genes in $S \cup T$, whereas the former one keeps them but adds internal edges connecting the two extremities in those genes. Clearly, $G^\ast(Q)$ also consists of a set of vertex-disjoint cycles and paths, and there is a one-to-one correspondence between the connected components in $G(Q)$ and that in $G^\ast(Q)$. Thus, the DCJ distance induced by $B$ can also be computed as $d(B) = n - c' - o'/2$, where $c'$ is the number of cycles and $o'$ is the number of odd-length paths in $G^\ast(Q)$, and $n$ is the number of genes in $X \setminus S$. As we will see later, this extended adjacency graph is the key point in devising the ILP formulation.

3.2 Add capping genes

In Shao and Lin (2012), we described a method to remove telomeres by introducing capping genes. A capping gene contains only one extremity, which combines with the adjacent telomere (or another capping gene) to form one adjacency. All capping genes are homologous to each other, forming a distinct gene family, denoted by $f_t$. Given two genomes $X$ and $Y$ with $I_X$ and $I_Y$ linear chromosomes, respectively (without loss of generality, we assume that $I_X \geq I_Y$), we first add one capping gene to each end of all the linear chromosomes in $X$ and $Y$; then we add $(I_X - I_Y)$ dummy chromosomes, each of which contains only a pair of capping genes, to genome $Y$ (Fig. 2). We denote by $X$ and $Y$ the two new genomes after adding capping genes for $X$ and $Y$. Clearly, we have $|F \setminus X, f_t| = |F \setminus Y, f_t|$. Thus, given a pair of independent consistent subsets $S \subset S(X)$ and $T \subset S(Y)$, we know that $X \setminus S$ and $Y \setminus T$ also have the same gene content. Using the same argument as in Shao et al. (2014), we can prove that

$$\min_{B \in B(X \setminus S, Y \setminus T)} d(B) = \min_{B \in B(X \setminus S, Y \setminus T)} d(B),$$

and the optimal $B$ can be recovered from the optimal $\hat{B}$ through discarding the pairs with capping genes. This statement allows us to add capping genes to remove telomeres on the two given genomes without affecting the optimal bijection. Since the two new genomes $X$ and $\hat{Y}$ do not contain telomeres, we have that for any triple $Q = (S, T, \hat{B})$, where $S \subset S(X), T \subset S(\hat{Y})$ and $\hat{B} \in B(X \setminus S, Y \setminus T)$, both $G(Q)$ and $G^\ast(Q)$ contain only cycles (Fig. 2). This property allows us only to count the number of cycles when computing the DCJ distance, which simplifies the following ILP formulation.

3.3 ILP formulation

Let $X$ and $Y$ be two given genomes after adding capping genes. Let $Q^* = (S', T', B')$ be the optimal triple minimizing $\sum_{s \in S'} e(s) + d(B')$, where we have $S' \subset S(X), T' \subset S(Y)$ and $B' \in B(X \setminus S', Y \setminus T')$. (Notice that here $X$ and $Y$ may contain capping genes, but we define $S(X)$ and $S(Y)$ in terms of the original genomes, which do not contain segments with capping genes.) To facilitate our description, we use $a \in X$ to denote that gene $a$ is contained in genome $X$. We use $a \in s$ to denote that gene $a$ is contained in segment $s$. We denote by $f_t$ the gene family to which gene $a$..
belongs. We say gene $a$ is duplicated in $Q^*$, if there exists one segment $s \in S' \cup T'$ such that $a \in s$ and nonduplicated otherwise.

We now give the ILP formulation to compute $Q^*$. For each segment $s \in S(X) \cup S(Y)$, we have one binary variable $x_s$ to indicate whether $s \in S' \cup T'$. For each gene $a \in X \cup Y$, we have one binary variable $y_a$ to indicate whether $a$ is duplicated in $Q^*$. We use the following two sets of constraints to guarantee that $y_a = 1$ if and only if there exists one segment $s \in S' \cup T'$ such that $a \in s$:

\[
y_a \geq x_s, \quad \forall s \in S(X) \cup S(Y) \text{ and } \forall a \in s;
\]

\[
y_a \leq \sum_{s : \exists a \in S(X) \cup S(Y) \text{ with } a \in s} x_s, \quad \forall a \in X \cup Y.
\]

We require that these segments in $S' \cup T'$ are independent, i.e. there do not exist two of them that contain the same gene:

\[
\sum_{a \in S(X) \cup S(Y) \text{ with } a \in s} x_s \leq 1, \quad \forall a \in X \cup Y.
\]

We also require that $X \setminus S'$ and $Y \setminus T'$ have the same gene content, i.e. for each gene family there must be an equal number of nonduplicated genes in $Q^*$ in this family in each genome:

\[
\sum_{a \in S(X)} (1 - y_a) = \sum_{b \in S(Y)} (1 - y_b), \quad \forall f \in A(X).
\]

And for each gene family, at least one gene is nonduplicated in $Q^*$:

\[
\sum_{a \in S(X)} (1 - y_a) \geq 1, \quad \forall f \in A(X);
\]

\[
\sum_{b \in S(Y)} (1 - y_b) \geq 1, \quad \forall f \in A(Y).
\]

For each pair of homologous genes $a \in X$ and $b \in Y$, we add one binary variable $z_{a,b}$ to indicate whether $B^*$ contains this pair. We require that for each gene in $X \cup Y$, it is mapped to exactly one homologous gene in the opposite genome if and only if it is nonduplicated in $Q^*$:

\[
\sum_{b \in S(Y)} z_{a,b} = 1 - y_a, \quad \forall a \in X;
\]

\[
\sum_{a \in S(X)} z_{a,b} = 1 - y_b, \quad \forall b \in Y.
\]

These constraints guarantee that these pairs in $B^*$ form a valid bijection between the genes in $X \setminus S'$ and those in $Y \setminus T'$. To compute $d(B^*)$, we need to count the number of cycles in $G'(Q^*)$. We add a variable $l_e$ for each extremity $e \in E(X) \cup E(Y)$ to represent the label of $e$. We then assign a distinct upper bound for $l_e$, denoted by $U_e$ (for example, we can just sort all the extremities in $E(X) \cup E(Y)$ in an arbitrary order and assign $U_e$ as the index of $e$ in the sorted list):

\[
0 \leq l_e \leq U_e, \quad \forall e \in E(X) \cup E(Y).
\]

We then require that all the extremities in the same cycle in $G'(Q^*)$ have the same label. This can be achieved by forcing that the two extremities connected by any edge in $G'(Q^*)$ have the same label. To guarantee this, we add the following three groups of constraints, each of which corresponds to one type of edges. First, we require that the two extremities in each adjacency have the same label (these constraints correspond to the gray edges):

\[
l_e = l_{e'}, \quad \forall \{e, e'\} \text{ form an adjacency in } X \text{ or } Y.
\]

Second, we require that each pair of extremities specified by $B^*$ have the same label (these constraints correspond to the black edges). To achieve that, we add the following four constraints for each pair of homologous genes $a \in X$ and $b \in Y$ (if $a$ and $b$ are capping genes, then we have $a_0 = a$ and $b_0 = b$, and thus the following four constraints degenerate into two):

\[
l_a \leq l_a + (1 - z_{a,b}) \cdot U_{a},
\]

\[
l_a \leq l_b + (1 - z_{a,b}) \cdot U_{b},
\]

\[
l_b \leq l_a + (1 - z_{a,b}) \cdot U_a,
\]

\[
l_b \leq l_b + (1 - z_{a,b}) \cdot U_b.
\]

Third, we require that the two extremities in each duplicated gene have the same label (these constraints correspond to the internal edges):

\[
l_a \leq l_a + (1 - y_a) \cdot U_a, \quad \forall a \in X \cup Y;
\]

\[
l_b \leq l_b + (1 - y_b) \cdot U_b, \quad \forall a \in X \cup Y.
\]

We then add a binary variable $w_e$ for extremity $e$ to indicate whether $l_e$ reaches its upper bound:

\[
w_e \leq l_e, \quad \forall e \in E(X) \cup E(Y).
\]

Since all the extremities in the same cycle in $G'(Q^*)$ are forced to have the same label, and all label variables have distinct upper bounds, we know that for each cycle in $G'(Q^*)$ at most one extremity can have its label reaching its upper bound. Thus, we have that

\[
\sum_{e \in E(X) \cup E(Y)} w_e
\]

is exactly the number of cycles in $G'(Q^*)$. And $d(B^*)$ can then be computed by

\[
|X| - \sum_{a \in X} y_a - \sum_{e \in E(X) \cup E(Y)} w_e,
\]

where the first two items give the number of genes in $X \setminus S'$.

Finally, we set the objective function of the ILP as

\[
\min \sum_{a \in S(X) \cup S(Y)} c(s) \cdot x_a + |X| - \sum_{a \in X} y_a - \sum_{e \in E(X) \cup E(Y)} w_e.
\]

4 Identify optimal substructures

Given two genomes $X$ and $Y$ after adding capping genes, we say two homologous segments $s$ in $X$ and $t$ in $Y$ form a pair of shared segments, denoted by $(s,t)$. Intuitively, shared segments are more likely to be nonduplicated and mapped to each other. Below, we give one sufficient condition and one algorithm to decide whether a pair of shared segments is in some optimal solution, i.e. in this optimal solution, $a_i$ and $b_i$ are nonduplicated and $a_n$ is mapped to $b_n$ for all $1 \leq i \leq n$. From now on, we assume that the cost function only depends on the length of the segments, i.e. we assume that if $|s| = |t|$ then we have $c(t) = c(s)$.

4.1 A sufficient condition

We say gene $a$ in genome $X$ is isolated, if there does not exist any segment $s \in S(X)$ such that $a \in s$ and $|s| \geq 2$. The following theorem gives a sufficient condition to decide whether a pair of shared segments of length two is an optimal substructure.

Theorem 1: Let $p = (a^1, b^1), (a^2, b^2)$. If we have $a^1$ and $a^2$ are singletons, and $b^1$ and $b^2$ are isolated, then $p$ is in some optimal solution.
Proof: Let \( Q = (S, T, B) \) be an arbitrary triple such that either \( b^1 \) or \( b^2 \) is duplicated in \( Q_0 \), or \( B \) does not contain \( (b^1, b^2) \). Below, we will show that we can always build a new triple \( Q' = (S', T', B') \) in which both \( b^1 \) and \( b^2 \) are nonduplicated and \( B' \) contains \( (b^1, b^2) \) and also verify that \( \gamma(Q') \leq \gamma(Q) \). Since \( Q \) is arbitrary, this proves the theorem.

First, assume that in \( Q \) both \( b^1 \) and \( b^2 \) are duplicated. Let \( s \in S \) and \( t \in T \) be the segments containing \( b^1 \) and \( b^2 \) respectively. Since both \( b^1 \) and \( b^2 \) are isolated, we know that \( |s| = |t| = 1 \).

We then insert \( a \) to \( s \) and \( t \) to \( T \). We have that \( \gamma(Q') \leq \gamma(Q) \). Since \( Q \) is arbitrary, this proves the theorem.

Keep reading the document...
that corresponding to \( \langle a, b \rangle \) and then replace the internal edges corresponding to \( a \) with that corresponding to \( a' \).

In summary, once we know \( R(Q_p), p \), we can then construct \( R(Q_p'), p \) and compare the number of cycles in them. If the number of cycles in \( R(Q_p'), p \) is no less than that in \( R(Q_p), p \), then \( p \) is in some optimal solution. However, the problem is that we do not know \( R(Q_p'), p \). Our strategy is to enumerate all the possibilities of \( R(Q_p'), p \). The vertices of \( R(Q_p'), p \), i.e. all the core vertices and all the boundary vertices w.r.t. \( p \), can be computed in advance very easily. All the genes of \( a_i \), \( 1 \leq i \leq n \), are duplicated in \( R(Q_p'), p \) by definition, and thus the two extremities in \( a_i \) are always connected by one internal edge in \( R(Q_p'), p \). All genes in \( F(X, f_a) \setminus \{a_i\} \) are possibly mapped to \( b_i \) in \( R(Q_p'), p \). For any two boundary vertices (maybe in the same genome), we need to check whether they can be connected by one reduced edge in \( R(Q_p'), p \), i.e. whether there exists one possible path connecting them that does not go through any other core vertices or boundary vertices. Notice that this path must be alternating, i.e. the edges with odd indices must be either black edges or internal edges and the edges with even indices must be gray edges (Fig. 4c and d).

There exists a linear time algorithm to decide the existence of an alternating path between two given vertices (Bang-Jensen and Gutin, 1998). We now adapt it for our use. Given a PHFSS \( p \) and two

![Fig. 4.](image-url) Fig. 4. \( X = \langle a^1, b^1, c^1, d^1, e^1 \rangle, \ Y = \langle d^2, b^2, c^2, d^2, e^2 \rangle, p = \langle \langle a^1, b^1 \rangle, \langle d^2, b^2 \rangle \rangle \). The four subgraphs show \( G(Q_p), G(Q_p'), R(Q_p), p \) and \( R(Q_p'), p \), respectively. The core vertices are shown as solid patterns, and the boundary vertices are shown as patterns with one inner point. Reduced edges are shown as dashed lines.

![Fig. 5.](image-url) Fig. 5. The underlying graph used to decide the existence of an alternating path connecting \( c^i \) and \( c^j \) w.r.t. \( p \) for the same instance in Figure 4. All bridging edges are shown as dotted lines.

boundary vertices \( x \) and \( y \), the algorithm first build a graph with \( V_1 \cup V_2 \cup \{x, y\} \) as its vertices, where \( V_1 \) is the set of all extremities except all the core vertices and boundary vertices and \( V_2 \) is a copy of \( V_1 \). Two extremities in \( V_1 \) are connected by one gray edge if they form one adjacency. Two homologous extremities in \( V_2 \) in different genomes are connected by one black edge, and the two extremities in \( V_2 \) in a possibly duplicated gene are connected by one internal edge. We connect \( x \) (resp. \( y \)) to its all homologous extremities in \( V_2 \) in the opposite genome by black edges. Finally, all the counterparts between \( V_1 \) and \( V_2 \) are connected by bridging edges (Fig. 5).

Clearly, all the bridging edges form a matching of size \( |V_1| \), denoted by \( M \). The algorithm then computes an augmenting path w.r.t. \( M \) using the Blossom algorithm, which takes linear time. We claim that such an augmenting path exists if and only if there exists one alternating path connecting \( x \) and \( y \) without going through any core vertices or boundary vertices. In fact, if such an augmenting path exists, then the two ends of this path must be \( x \) and \( y \), since they are the only two unmatched vertices. We claim that the edges in the augmenting path that are not in \( M \) form an alternating path connecting \( x \) and \( y \). This is because edges in \( M \) are spanning \( V_1 \) and \( V_2 \), whereas gray edges are all inside in \( V_1 \) and black edges and internal edges are inside in \( V_2 \). The opposite side of statement can be reasoned in a similar way.

The algorithm to decide whether a given PHFSS \( p \) is in some optimal solution proceeds as follows. The first phase of the algorithm is to compute the core vertices and the boundary vertices w.r.t. \( p \) and then for each pair of boundary vertices, to check whether they can be connected by a reduced edge. If the total number of edges (reduced edges plus those among core vertices) is larger than \( \log n \), the algorithm terminates. Otherwise, the algorithm comes to the second phase. It enumerates all the possibilities of \( R(Q_p'), p \) for each possible valid combination of the reduced edges (i.e. they form a matching that covers all the boundary vertices), it enumerates all the possible valid mappings for the genes in \( A(p) \) (\( a_i \) cannot be mapped to \( b_i \) by the definition of \( R(Q_p'), p \)) and the mapping that yields the maximum number of cycles, plus the current combination of the reduced edges, forms one possibility of \( R(Q_p'), p \). After that, for each possibility of \( R(Q_p'), p \), it then builds \( R(Q_p'), p \) and compares the number of cycles between them. If the number of cycles in \( R(Q_p'), p \) is always no less than that in \( R(Q_p), p \) for all the possibilities, then the algorithm concludes that \( p \) is in some optimal solution.

The above algorithm runs in polynomial time. In fact, the first phase runs in polynomial-time, since we can decide the existence of a reduced edge for each pair of boundary vertices in linear time. In the second phase, the number of edges is in logarithmic-size, which implies that the number of possibilities of \( R(Q_p'), p \) is in polynomial size. Thus, the second phase also runs in polynomial time.

We remark that usually not all pairs of boundary vertices can be connected by a reduced edge (Fig. 6). In fact, if this is not the case,
then there always exists one possibility such that \( R(Q^*_p, p) \) contains more cycles than \( R(Q^*_p, p) \), in which case the algorithm fails. In other words, the first phase to identify possible reduced edges is very essential, which not only decreases the number of possibilities but more importantly makes the algorithm capable of identifying optimal substructures. We also remark that this algorithm is a sufficient test, i.e. if it returns ‘yes’, then \( p \) is guaranteed in some optimal solution. However, if it returns ‘no’, then it is still possible that \( p \) is in some optimal solution. This is because two reduced edges in \( R(Q^*_p, p) \) might not be able to coexist in \( G^*(Q^*_p) \).

We can apply the theorem in Section 4.1 and the algorithm in Section 4.2 on all shared segments to verify their optimality. If such an optimal substructure is identified, we immediately fix it and update the genomes through assigning each pair of genes in it a distinct gene family. We can iteratively repeat this process until no such optimal substructure can be found. This serves as a preprocessing algorithm to simplify the problem before calling the ILP solver. The performance of this preprocessing algorithm on real genomes is analyzed in Table 4.

5 Set the cost

Under a parsimonious model, it is natural to set a unit cost for all segmental duplications (as we do for all DCJs). However, in this case, two segmental duplications, one in each genome, that create a pair of shared segments can be always explained as two DCJs with the same total cost. Consider the example in Figure 7a, for which we have two optimal solutions with total cost of 2: one is to regard \( a^2 \) and \( a^4 \) as duplicated genes, and the other uses two DCJs, which first cut \( a^2 \) out from X and then insert it back between \( c^1 \) and \( d^1 \). The scenario in the second case (two DCJs using one circular chromosome as intermediate) requires three inversions to explain, and therefore it is much less unlikely to happen comparing with the first scenario. Thus, to avoid the second case we set \( c(\cdot) < 1 \).

On the other hand, if we have \( c(\cdot) \leq 0.5 \), then every DCJ that inverts a possibly duplicated segment can be always explained by two segmental duplications with the same or even better total cost. Consider the example in Figure 7b, for which one solution is to use only one DCJ to invert the segment \( (a^2, b^2) \) on X. However, if we have \( c(\cdot) \leq 0.5 \), then we can regard \( (a^2, b^2) \) and \( (-b^4, -a^4) \) as duplicated segments, whose total cost is at most 1. Thus, to avoid the second case, we need to set \( c(\cdot) > 0.5 \).

Combining the above two facts, in the following experiments, we set \( c(\cdot) = 0.75 \).

6 Infer in-paralogs and orthologs

Under a most parsimonious evolutionary scenario, the duplicated genes in the optimal triple infer the in-paralogs in each genome, whereas the bijection between the nonduplicated genes in the two genomes infers a subset of the orthology pairs [more specifically, positional orthologs (Dewey, 2011)]. In the following, we apply our method to infer in-paralogs and orthologs on both simulated datasets and biological datasets and compare its performance with MSOAR.

6.1 Results on simulated datasets

We simulate a pair of genomes as follows. We start from an ancestor genome with only one linear chromosome consisting of \( N = 5000 \) singletons (we also test \( N = 1000 \) and \( N = 2000 \); the results are not presented since they agree with \( N = 5000 \)). We then perform \( S_1 \) segmental duplications on the ancestor genome to make some gene families contain more than one copy. A segmental duplication randomly chooses a segment of length \( L \) and inserts its copy to another random position. The two extant genomes then speciate independently from this ancestor genome. The speciation process on each branch includes randomly mixed \( S_2 \) segmental duplications and \( D \) DCJs. A DCJ randomly chooses two positions in the genome and then reverses the segment in between. We make sure that the expected number of genes per gene family in each extant genome is 1.5 (this number is comparable to that in human genome, which is 1.46), therefore we have that \( S_1 + S_2 = 0.5 \cdot N / L \). We further fix \( S_1 = 0.2 \cdot N / L \) and \( S_2 = 0.3 \cdot N / L \) (we also test \( S_1 = 0 \) and \( S_2 = 0.5 \cdot N / L \), and the results are almost the same). Thus, a simulation configuration is determined by parameters \( L \) and \( D \).

For each pair of simulated genomes \( X \) and \( Y \), we take them as input to run MSOAR and our method. For MSOAR, we run its binary version downloaded from http://msoar.cs.ucr.edu. For our method, we first apply the preprocessing algorithm described in Section 4 and then formulate the simplified problem as an ILP instance, which is solved using the Gurobi solver. We set the time limit to 2 h for each instance, i.e. if the ILP solver does not return the optimal solution in 2 h, we terminate it and return the current sub-optimal solution.

Both methods return triples \((S, T, B)\), where \( S \) and \( T \) infers the in-paralogs in the two extant genomes, respectively, and \( B \) infers the orthology pairs. We now give the measures to evaluate them. First, we regard the problem to infer in-paralogs as a standard binary classification problem: those genes that are generated by segmental duplications in the speciation process are considered as gold standard positive in-paralogs and those genes that are in the segments in \( S \cup T \) are considered as predicted positive in-paralogs. Thus, we use the sensitivity and specificity to measure \((S, T)\). To evaluate the performance of \( B \), we refer to those gene pairs in the two extant genomes that correspond to the same gene in the ancestor genome as the true orthology pairs. We therefore use the following way to evaluate \( B \): we say a pair in \( B \) is assessable, if at least one of its two genes can be found in some true orthology pair, and the accuracy of
Fig. 8. Sensitivity of the inferred in-paralogs. The solid lines and dashed lines track our method and MSOAR, respectively. The circles, triangles and diamonds track \( L = 1, 2 \) and \( L = 5 \), respectively.

Fig. 9. Specificity of the inferred in-paralogs

B is then defined as the ratio between the number of true orthology pairs in B and the number of assessable pairs in B.

For each parameter configuration, we simulate 10 instances and compute the average sensitivity, specificity and accuracy for both methods. The performance of the two methods is shown in Figures 8–10, where the parameters \( L \in \{1, 2, 5\} \) and \( D \) ranges from 250 to 2000. First, we can observe that both methods get very high sensitivity (above 90% on all configurations). However, MSOAR gets relatively low specificity. One reason for this is that MSOAR uses unit cost for both rearrangements and single-gene duplications. According to the discussion in Section 5, unit cost for all operations might misclassify in-paralogs. Second, as \( D \) increases, the performance of both methods decreases. This is because the number of DCJs is highly positively correlated to the difficulty of the problem. When \( D \leq 500 \), i.e. roughly 10% of the size of the simulated genome (which is usually the case for real genomes, see Table 2 columns \( d(B) \) for some examples), we can see that our method almost gets perfect performance. Third, observe that MSOAR is very sensitive to \( L \) even when \( D \) is very small. This might be because the evolutionary model for in-paralogs in MSOAR is single-gene duplication, which creates trouble when genomes contain long segmental duplications. Finally, our method outperforms MSOAR on all the configurations.

6.2 Results on biological datasets

We compare both methods on five mammalian species, human (H.s.), gorilla (G.g.), orangutan (P.a.), mouse (M.m.) and rat (R.n.). For each species, we collect all the protein-coding genes and download their positions on the chromosomes and the Ensembl gene family names from Ensembl (http://www.ensembl.org). Two genes are considered as homologous if they have the same Ensembl gene family name. Since the tandemly arrayed genes (TAGs) have a different evolutionary model from segmental duplications, we merge each group of TAGs into only one gene through only keeping the first gene in the group while removing all the following ones.

According to the discussion in Section 5, unit cost for all operations might misclassify in-paralogs. Second, as \( D \) increases, the performance of both methods decreases. This is because the number of DCJs is highly positively correlated to the difficulty of the problem. When \( D \leq 500 \), i.e. roughly 10% of the size of the simulated genome (which is usually the case for real genomes, see Table 2 columns \( d(B) \) for some examples), we can see that our method almost gets perfect performance. Third, observe that MSOAR is very sensitive to \( L \) even when \( D \) is very small. This might be because the evolutionary model for in-paralogs in MSOAR is single-gene duplication, which creates trouble when genomes contain long segmental duplications. Finally, our method outperforms MSOAR on all the configurations.

For each species, we collect all the protein-coding genes and download their positions on the chromosomes and the Ensembl gene family names from Ensembl (http://www.ensembl.org). Two genes are considered as homologous if they have the same Ensembl gene family name. Since the tandemly arrayed genes (TAGs) have a different evolutionary model from segmental duplications, we merge each group of TAGs into only one gene through only keeping the first gene in the group while removing all the following ones.

We do the pairwise comparison for all five species, and for each pair of species, we run both methods to obtain triples \((S, T, B)\). We use the same accuracy defined in Section 6.1 to evaluate \( B \). To compute the accuracy, we use the gene symbols (HGNC symbols for primary genes, MGI symbols for mouse genes and RGD symbols for rat genes, downloaded from Ensembl) to define true orthology pairs: those gene pairs that have the same gene symbol form the set of true orthology pairs for each pair of species. We do not have annotation data to serve as gold standard positive in-paralogs (we cannot just regard those genes that are not in the true orthology pairs as gold standard positive in-paralogs, since many genes have not yet been assigned a valid gene symbol). Thus, we are not able to compute the sensitivity and specificity of \((S, T)\).
Species pairs  | S1 | S2 | S3 | | T1 | T2 | T3 | | [S] | [T] |
---|---|---|---|---|---|---|---|---|---|
G.g. and H.s.  | 98.7 | 1.2 | 0.0 | 1347 | 95.2 | 4.3 | 0.3 | 615 |
G.g. and M.m.  | 98.5 | 1.3 | 0.1 | 1421 | 96.7 | 3.2 | 0.3 | 948 |
G.g. and P.a.  | 97.9 | 1.8 | 0.3 | 1579 | 98.8 | 1.1 | 0.0 | 680 |
G.g. and R.n.  | 98.1 | 1.6 | 0.2 | 1377 | 94.9 | 3.7 | 1.2 | 2243 |
H.s. and M.m.  | 94.8 | 4.9 | 0.1 | 563 | 96.2 | 3.5 | 0.2 | 818 |
H.s. and P.a.  | 93.9 | 4.8 | 1.2 | 807 | 99.3 | 0.4 | 0.1 | 637 |
H.s. and R.n.  | 94.9 | 4.2 | 0.7 | 631 | 95.3 | 3.2 | 1.3 | 2254 |
M.m. and P.a.  | 96.0 | 3.3 | 0.6 | 1109 | 99.1 | 0.5 | 0.2 | 716 |
M.m. and R.n.  | 95.5 | 3.5 | 0.9 | 648 | 94.5 | 3.5 | 1.8 | 2091 |
R.n. and P.a.  | 95.2 | 3.5 | 1.2 | 2472 | 99.0 | 0.6 | 0.2 | 736 |

S1 (respectively, T1) contains the percentage of the segments of length k in S (respectively, T).

Table 4. Composition of B

| Species pairs | Trivial (%) | Predetermined (%) | Remaining (%) | | [B] |
---|---|---|---|---|---|
G.g. and H.s.  | 51.5 | 47.9 | 0.5 | 16 213 |
G.g. and M.m.  | 48.7 | 49.5 | 1.6 | 15 015 |
G.g. and P.a.  | 50.7 | 48.5 | 0.7 | 15 271 |
G.g. and R.n.  | 46.3 | 50.5 | 3.0 | 14 983 |
H.s. and M.m.  | 51.0 | 47.5 | 1.3 | 15 572 |
H.s. and P.a.  | 52.3 | 47.1 | 0.4 | 14 481 |
H.s. and R.n.  | 48.5 | 49.5 | 1.8 | 15 379 |
M.m. and P.a.  | 50.0 | 48.6 | 1.2 | 14 620 |
M.m. and R.n.  | 48.9 | 49.4 | 1.6 | 16 347 |
R.n. and P.a.  | 47.7 | 50.1 | 2.0 | 14 534 |

The comparison on accuracy is shown in Table 1. We can observe that both methods have very high accuracy, indicating that the inferred orthology pairs from gene order data mostly agree with the annotations. On the other hand, our method gets higher accuracy than MSOAR on orthology pairs from gene order data.

In Table 2, we compare the number of operations and total score inferred by the two methods to evaluate their ability as an optimizer. First, we can see that our method gets more segmental duplications and many fewer DCJs than MSOAR. One reason for this is that we use smaller weight for segmental duplications. Second, our method gets smaller total cost on all the 10 pairs. This shows the advantage of our exact algorithm over the heuristic applied in MSOAR. Notice that both methods have very high accuracy, indicating that the inferred segments and many isolated genes (because most of the segmental duplications are single-gene duplications), and thus there are many optimal substrutures that can be identified by our algorithm.

7 Conclusion and discussion

We proposed an exact algorithm to compute a set of DCJs and segmental duplications with minimum total cost between two given genomes. As far as we know, this is the first exact algorithm to compare two genomes in the presence of duplicated genes with both rearrangements and content-modifying events. This algorithm can be applied to infer in-paralogs and orthologs, and the inferred results were showed highly agreeing with the annotations.

The algorithm described in Section 4.2 has potential to extend. For example, it can be directly used to test whether a general substructure, rather than a single PHF5, is optimal. Moreover, we made a strong assumption that all genes in the related gene families are isolated, which immediately makes the cost of the segmental duplications trivial to compare and thus allows us to focus on the number of cycles. In fact, we can relax this assumption, as long as we can guarantee that the segmental duplications induced by the substructure that is tested is optimal.

Although the evolutionary model used in our algorithm, i.e. DCJ plus segmental duplication, is already quite general, there are some other events, like tandem duplications, that cannot be explained by this model. We will extend our algorithm for more general models in the future.

Conflict of Interest: none declared.

References


