JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 22, Number 5, 2015
© Mary Ann Liebert, Inc.
Pp. 425–435
DOI: 10.1089/cmb.2014.0096

An Exact Algorithm to Compute the Double-Cut-and-Join Distance for Genomes with Duplicate Genes

MINGFU SHAO[1], YU LIN[2] and BERNARD M.E. MORET[1]

ABSTRACT

Computing the edit distance between two genomes is a basic problem in the study of genome evolution. The double-cut-and-join (DCJ) model has formed the basis for most algorithmic research on rearrangements over the last few years. The edit distance under the DCJ model can be computed in linear time for genomes without duplicate genes, while the problem becomes NP-hard in the presence of duplicate genes. In this article, we propose an integer linear programming (ILP) formulation to compute the DCJ distance between two genomes with duplicate genes. We also provide an efficient preprocessing approach to simplify the ILP formulation while preserving optimality. Comparison on simulated genomes demonstrates that our method outperforms MSOAR in computing the edit distance, especially when the genomes contain long duplicated segments. We also apply our method to assign orthologous gene pairs among human, mouse, and rat genomes, where once again our method outperforms MSOAR.

Key words: adjacency graph, DCJ distance, maximum cycle decomposition, orthology assignment.

1. INTRODUCTION

The combinatorics and algorithmics of genomic rearrangements have been the subject of much research since the problem was formulated in the 1990s (Fertin et al., 2009). The advent of whole-genome sequencing has provided us with masses of data on which to study genomic rearrangements. Genomic rearrangements include inversions, transpositions, circularizations, and linearizations, all of which act on a single chromosome, and translocations, fusions, and fissions, which act on two chromosomes. These operations can all be described in terms of the single double-cut-and-join (DCJ) operation (Bergeron et al., 2006; Yancopoulos et al., 2005), which has formed the basis for most algorithmic research on rearrangements over the last few years (Bergeron et al., 2009; Chen, 2010; Chen et al., 2011; Yancopoulos and Friedberg, 2008; Moret et al., 2013). A DCJ operation makes two cuts in the genome, either in the same chromosome or in two different chromosomes, producing four cut ends, then rejoins the four cut ends in a different order.

A basic problem in genome rearrangements is to compute the edit distance between two genomes, that is, the minimum number of operations needed to transform one genome into another. Under the inversion

[1]Laboratory for Computational Biology and Bioinformatics, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
[2]Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California.

model, Hannenhalli and Pevzner (1995) gave the first polynomial-time algorithm to compute the edit distance for unichromosomal genomes, which was later improved to linear time (Bader et al., 2001). As for the multichromosomal genomes, the edit distance under the Hannenhalli–Pevzner model (inversions and translocations) has been studied through a series of articles (Hannenhalli and Pevzner, 1995; Jean and Nikolski, 2007; Ozery-Flato and Shamir, 2003; Tesler, 2002), culminating in a fairly complex linear-time algorithm (Bergeron et al., 2009). Under the DCJ model, the edit distance can be computed in linear time for two multichromosomal genomes in a simple and elegant way (Bergeron et al., 2006).

All of these algorithms assume genomes contain no duplicate genes. However, gene duplications are widespread events and have long been recognized as a major driving force of evolution (Bailey and Eichler, 2006; Lynch, 2007). For example, in human genomes segmental duplications are hotspots for non-allelic homologous recombination leading to genomic disorders, copy-number polymorphisms, and gene and transcript innovations (Jiang et al., 2007). The problem of computing the inversion distance for genomes in the presence of duplicate genes has been proved NP-hard (Chen et al., 2005). Suksawatchon et al. (2007) proposed a heuristic for this problem using binary integer programming, which was later extended to handle gene deletion (Laohakiat et al., 2008). Chen et al. (2005) decomposed this problem into two new optimization problems, called the *minimum common partition* and the *maximum cycle decomposition*, for which efficient heuristics were designed. They packaged the whole algorithms into the SOAR software system and applied SOAR to assign orthologs on a genome-wide scale. Later, they extended SOAR to unite rearrangements and single-gene duplications as a new software package, called MSOAR, which can be applied to detect inparalogs in addition to orthologs (Fu et al., 2007). Recently, they incorporated tandem duplications into their model and demonstrated that the new system achieved a better sensitivity and specificity than MSOAR (Shi et al., 2010).

In this article, we focus on the problem of computing the edit distance for two genomes with duplicate genes under the DCJ model. This problem is also NP-hard, which can be proved by a reduction from the NP-hard problem of *breakpoint graph decomposition* (Kececioglu and Sankoff, 1995). We first reduce this problem to the problem of finding the optimal consistent decomposition of the corresponding adjacency graph, then formulate the latter problem as an integer linear program. We also provide an efficient preprocessing approach to reduce the ILP formulation while preserving optimality. Finally, we compare our method with MSOAR on both simulated and biological datasets.

2. PROBLEM STATEMENT

We model one genome as a set of chromosomes, and each chromosome as a linear or circular list of genes. Homologous genes are grouped into *gene families*. In this article, we study two genomes with the same gene content: each gene family has the same number of genes in both genomes. Assuming that two genomes G_1 and G_2 have the same gene content, we say a bijection between G_1 and G_2 is *valid* if it specifies n homologous gene pairs, where n is the number of genes in each genome. If G_1 and G_2 contain only singleton gene families (exactly one gene in each family in each genome), then there is a unique valid bijection between G_1 and G_2, and the DCJ distance between G_1 and G_2 can be computed in linear time (Bergeron et al., 2006). If G_1 and G_2 contain gene families with multiple genes in each genome, then there are many valid bijections between G_1 and G_2. Different valid bijections define different one-to-one correspondences between homologous genes, yielding possibly different DCJ distances between G_1 and G_2. In this article, we study the following *generalized DCJ distance problem*: given two genomes G_1 and G_2 with the same gene content, find a valid bijection between G_1 and G_2 that minimizes the DCJ distance. We denote the generalized DCJ distance between G_1 and G_2 as $d(G_1, G_2)$.

We use the notation introduced by Bergeron et al. (2006) for gene orders. The two ends of a gene g are called *extremities*, the head as g_h, and the tail as g_t. If genes f and g are homologous, its corresponding extremities (f_h and g_h, f_t and g_t) are also *homologous*. Two consecutive genes a and b can be connected by one *adjacency*, which is represented by a set of two extremities; thus adjacencies come in four types: $\{a_t, b_t\}$, $\{a_h, b_t\}$, $\{a_t, b_h\}$, and $\{a_h, b_h\}$. If gene g lies at one end of a linear chromosome, then this end can be represented by a set of one extremity, $\{g_t\}$ or $\{g_h\}$, called a *telomere*. The set of all extremities of a genome is called the *extremity set*.

Let G_1 and G_2 be two genomes with the same gene content, and let S_1 and S_2 be the extremity sets of G_1 and G_2, respectively. The *adjacency graph* with respect to G_1 and G_2 can be written as $AG=(V, E)$, with

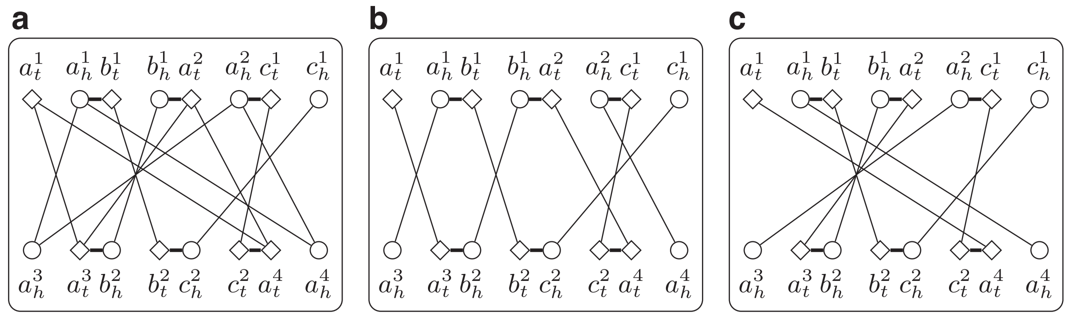
a
b
c
a_t^1 a_h^1 b_t^1 b_h^1 a_t^2 a_h^2 c_t^1 c_h^1
a_h^3 a_t^3 b_h^2 b_t^2 c_h^2 c_t^2 a_t^4 a_h^4

FIG. 1. An example of adjacency graph and its two consistent decompositions. Genome 1 contains one linear chromosome, (a^1, b^1, a^2, c^1), and genome 2 also contains one linear chromosome ($-a^3$, $-b^2$, $-c^2$, a^4). Genes in the same gene family are represented by the same label and distinguished by different superscripts. All black edges are represented by long thin lines, and all gray edges are represented by short thick lines. (**a**) The corresponding adjacency graph, in which head extremities are represented by circles, while tail extremities are represented by diamonds. (**b**) A consistent decomposition with two odd-length paths, whose corresponding valid bijection maps a^1 to a^3 and a^2 to a^4. (**c**) Another consistent decomposition with two odd-length paths and one cycle, whose corresponding valid bijection maps a^1 to a^4 and a^2 to a^3.

$V = S_1 \cup S_2$ and where E is composed of two types of edges, *black edges* and *gray edges*. Two extremities in different extremity sets (one is in S_1 and the other one is in S_2) are connected by one black edge if they are homologous, and two extremities in the same extremity set are connected by one gray edge if they form an existing adjacency. Figure 1a gives an example.

We say that a cycle (or path) in the adjacency graph is *alternating* if any two adjacent edges in this cycle (or path) consist of one black edge and one gray edge. The *length* of a cycle (or path) is defined as the number of its black edges. A *decomposition* of the adjacency graph is a set of vertex-disjoint alternating cycles and paths that cover all vertices and all gray edges. We say a decomposition is *consistent* if for any two homologous genes f and g, either both (f_h, g_h) and (f_t, g_t) are in this decomposition, or neither of them is in this decomposition. Figure 1b and c give two examples of consistent decompositions.

Given two genomes G_1 and G_2 with the same gene content, there is a natural one-to-one correspondence between the set of all possible valid bijections from G_1 to G_2 and the set of all possible consistent decompositions of the adjacency graph with respect to G_1 and G_2. In fact, if one valid bijection is given, which maps gene f in G_1 to a homologous gene g in G_2, then we can keep the black edges (f_h, g_h) and (f_t, g_t) in the decomposition. We do the same thing for every pair of genes specified by this valid bijection; this process culminates in a consistent decomposition. On the other hand, if we are given a consistent decomposition of the corresponding adjacency graph, we can collect all homologous gene pairs (f, g) indicated by black edges (f_h, g_h) and (f_t, g_t), which form a valid bijection from G_1 to G_2. Given a consistent decomposition with c cycles and o odd-length paths, exactly $(|V|/4 - c - o/2)$ DCJ operations are needed to transform G_1 into G_2 (Bergeron et al., 2006). Thus, we can write $d(G_1, G_2) = \min_{D \in \mathcal{D}} (|V|/4 - c_D - o_D/2) = |V|/4 - \max_{D \in \mathcal{D}} (c_D + o_D/2)$, where $\mathcal{D}$ is the space of all consistent decompositions, and c_D and o_D are the numbers of cycles and odd-length paths in a decomposition D, respectively. This formula transforms the generalized DCJ distance problem into the *maximum cycle decomposition problem*, which asks for a consistent decomposition of the adjacency graph such that the number of cycles plus half the number of odd-length paths in this decomposition is maximized.

3. ILP FOR THE MAXIMUM CYCLE DECOMPOSITION PROBLEM

Shao and Lin (2012) described a capping method to remove telomeres by introducing *null extremities*. All null extremities are homologous to each other, but none are homologous to any other extremity. Let $AG = (V = S_1 \cup S_2, E)$ be the adjacency graph with respect to two given genomes G_1 and G_2. Suppose that G_1 and G_2 contain $2 \cdot k_1$ and $2 \cdot k_2$ telomeres respectively. The "telomere removal" proceeds as follows (see Fig. 2, for example). For each extremity $u \in S_1$ coming from each telomere in G_1, we add one null extremity τ to S_1 and add one gray edge to E that connects u and τ. Similarly, for each extremity $v \in S_2$

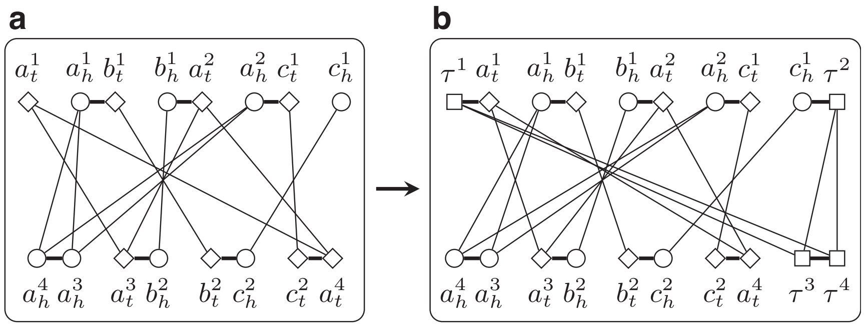
a
b
a_t^1 a_h^1 b_t^1 b_h^1 a_t^2 a_h^2 c_t^1 c_h^1
a_h^4 a_h^3 a_t^3 b_h^2 b_t^2 c_h^2 c_t^2 a_t^4
τ^1 a_t^1 a_h^1 b_t^1 b_h^1 a_t^2 a_h^2 c_t^1 c_h^1 τ^2
a_h^4 a_h^3 a_t^3 b_h^2 b_t^2 c_h^2 c_t^2 a_t^4 τ^3 τ^4

FIG. 2. An example of the telomere removal. Genome 1 contains one linear chromosome, (a^1, b^1, a^2, c^1), and genome 2 contains one circular chromosome $(-a^3, -b^2, -c^2, a^4)$. (**a**) The corresponding adjacency graph. (**b**) The adjacency graph after the telomere removal, in which null extremities are represented by squares.

coming from each telomere in G_2, we add one null extremity τ to S_2 and add one gray edge to E that connects v and τ. If we additionally have $k_1 < k_2$, we then add $(k_2 - k_1)$ pairs of null extremities to S_1, each of which is connected by one more gray edge added to E. We finally add black edges connecting all possible pairs of null extremities between S_1 and S_2. We can prove that this telomere removal process does not change $d(G_1, G_2)$ using the same argument as in Yancopoulos and Friedberg (2008) and Shao and Lin (2012). In the following we assume that each vertex is adjacent to exactly one gray edge in the adjacency graph, and that the consistent decompositions consist of only cycles.

Now we formulate the maximum cycle decomposition problem as an integer linear program. Let $AG = (V, E)$ be the adjacency graph with respect to two given genomes G_1 and G_2 with the same gene content. For each edge $e \in E$, we create binary variable x_e to indicate whether e will be in the final decomposition. First, we require that all gray edges be in the final decomposition:

$$x_e = 1, \quad \forall e \text{ that are grey}$$

Second, we require that the final decomposition be consistent:

$$x_{(f_h, g_h)} = x_{(f_t, g_t)}, \quad \forall f \in G_1 \text{ and } \forall g \in G_2 \text{ that are homologous}$$

Third, we require that for each vertex exactly one adjacent black edge adjacent be chosen:

$$\sum_{(u,v) \in E,\ v \in S_2} x_{(u,v)} = 1, \quad \forall u \in S_1$$
$$\sum_{(u,v) \in E,\ u \in S_1} x_{(u,v)} = 1, \quad \forall v \in S_2$$

These three groups of constraints guarantee that all selected edges form a consistent decomposition.

Now we count the number of cycles. We first index the vertices arbitrarily, $V = \{v_1, v_2, \cdots, v_{|V|}\}$. For each vertex v_i, we create variable y_i to indicate the *label* of v_i. We set a distinct positive bound i for each y_i:

$$0 \le y_i \le i, \quad 1 \le i \le |V|$$

We require that all vertices in the same cycle in the final decomposition have the same label, which can be guaranteed by requiring that, for each selected edge, the two adjacent vertices have the same label:

$$y_i \le y_j + i \cdot (1 - x_e), \quad \forall e = (v_i, v_j) \in E$$
$$y_j \le y_i + j \cdot (1 - x_e), \quad \forall e = (v_i, v_j) \in E$$

Then, for each vertex v_i, we create binary variable z_i to indicate whether y_i is equal to its upper bound i:

$$i \cdot z_i \le y_i, \quad 1 \le i \le |V|$$

Since all vertices in the same cycle have the same label and all upper bounds are distinct, there is exactly one vertex in each cycle whose label can be equal to its upper bound. Finally, we set the objective to

$$\max \sum_{1 \le i \le |V|} z_i,$$

which is equal to the number of cycles.

There are $O(|E|)$ variables and $O(|E|)$ constraints in this ILP formulation.

4. FIXING CYCLES OF LENGTH TWO

A cycle of length two in the adjacency graph indicates one shared adjacency. The following theorem gives a sufficient condition to fix this cycle while preserving optimality, which can be used to narrow the search for an optimal bijection.

Theorem 1. *Given an adjacency graph $AG=(V, E)$, if a length-two cycle C contains some vertex with total degree 2, then there exists an optimal consistent decomposition of AG that contains C.*

Proof. Let $\{a_h^1, b_h^1, a_h^2, b_h^2\}$ be the four vertices of C, where a_h^1 and b_h^2 form an adjacency in G_1 while a_h^2 and b_h^2 form an adjacency in G_2, and (a_h^1, a_h^2) and (b_h^1, b_h^2) are the two black edges of C. Let a_h^1 be the vertex of total degree 2; then the gene family of $\{a^1, a^2\}$ is a singleton family, and thus edge (a_h^1, a_h^2) appears in every consistent decomposition. Now we prove the theorem by contradiction. Suppose that edge (b_h^1, b_h^2) is not in any optimal consistent decomposition. Take any optimal consistent decomposition D, in which b_h^1 is linked to b_h^4 and b_h^2 is linked to b_h^3. Since D is consistent, we know that edges (b_t^1, b_t^4) and (b_t^2, b_t^3) are also in D. We now transform D into a new decomposition D'' that contains edge (b_h^1, b_h^2) by exchanging two pairs of edges. Figure 3 illustrates this process. First, we remove edges (b_h^1, b_h^4) and (b_h^3, b_h^2) from D and add edges (b_h^1, b_h^2) and (b_h^3, b_h^4); denote this inconsistent decomposition by D'. Since in this step one cycle is split into two small cycles, we have that $c_{D'} = c_D + 1$. Now, we remove edges (b_t^1, b_t^4) and (b_t^3, b_t^2) from D' and add edges (b_t^1, b_t^2) and (b_t^3, b_t^4) to obtain the consistent decomposition D''. This step involves at most two cycles of D', and merges these two cycles together in the worst case. Thus, we have $c_{D''} \ge c_{D'} - 1$. Overall, we have that $c_{D''} \ge c_D$, which means D'' is also an optimal consistent decomposition—the desired contradiction. ■

If all four vertices in a cycle of length two have degree larger than 2, then it is possible that this cycle is not part of any optimal consistent decomposition. Figure 4 gives such an example. Moreover, this example also shows that if a shared adjacency appears exactly once in each genome, it is still possible that the corresponding cycle of length two is not part of any optimal consistent decomposition.

5. EXPERIMENTAL RESULTS

We compare our method with MSOAR on both simulated and biological datasets. The input for both methods is two genomes with the same gene content, and the output is a bijection between the two

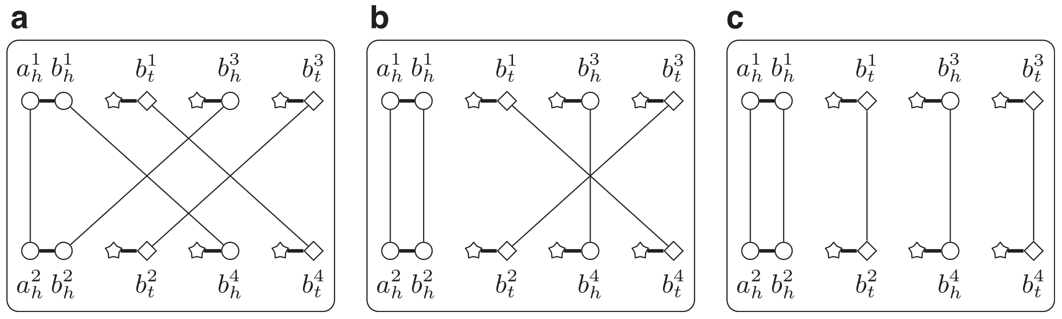
a
b
c
a_h^1 b_h^1 b_t^1 b_h^3 b_t^3
a_h^2 b_h^2 b_t^2 b_h^4 b_t^4

FIG. 3. The process of building a new optimal consistent decomposition that contains edge (b_h^1, b_h^2). (**a**) One optimal consistent decomposition D without edge (b_h^1, b_h^3). Stars represent unrelated extremities. (**b**) The inconsistent decomposition D'. (**c**) The consistent decomposition D''.

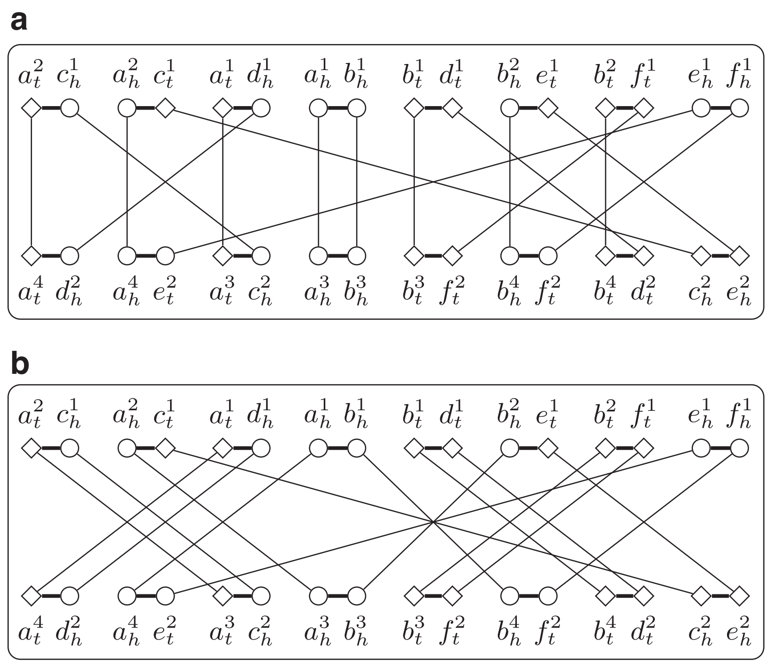
a
$a_t^2\ c_h^1$ $a_h^2\ c_t^1$ $a_t^1\ d_h^1$ $a_h^1\ b_h^1$ $b_t^1\ d_t^1$ $b_h^2\ e_t^1$ $b_t^2\ f_t^1$ $e_h^1\ f_h^1$
$a_t^4\ d_h^2$ $a_h^4\ e_t^2$ $a_t^3\ c_h^2$ $a_h^3\ b_h^3$ $b_t^3\ f_t^2$ $b_h^4\ f_t^2$ $b_t^4\ d_t^2$ $c_h^2\ e_h^2$
b
$a_t^2\ c_h^1$ $a_h^2\ c_t^1$ $a_t^1\ d_h^1$ $a_h^1\ b_h^1$ $b_t^1\ d_t^1$ $b_h^2\ e_t^1$ $b_t^2\ f_t^1$ $e_h^1\ f_h^1$
$a_t^4\ d_h^2$ $a_h^4\ e_t^2$ $a_t^3\ c_h^2$ $a_h^3\ b_h^3$ $b_t^3\ f_t^2$ $b_h^4\ f_t^2$ $b_t^4\ d_t^2$ $c_h^2\ e_h^2$

FIG. 4. An example of a cycle of length two that is not part of any optimal consistent decomposition. (**a**) A consistent decomposition with four cycles that contain the cycle of length two of $\{a_h^1, b_h^1, a_h^3, b_h^3\}$ (**b**) An optimal consistent decomposition with five cycles.

genomes, plus the DCJ distance calculated as $n - c - o/2$, where n is the number of genes in each genome, and c and o are the numbers of cycles and odd-length paths in the adjacency graph induced by the bijection. We use both the accuracy of the bijection, which is defined as the percentage of correct gene pairs (compared with a reference bijection), and the deviation from the true evolutionary distances, to evaluate the performance of the two methods.

For our method, given two genomes, we first build the adjacency graph and then employ the telomere removal technique to obtain a new adjacency graph without telomeres. Then we apply Theorem 1 to fix possible cycles of length two, and finally invoke GUROBI (Gurobi Optimization Inc., 2013) to solve the ILP formulation. Since the ILP solver might take a long time, we set a time limit of 2 hours for each instance in our experiments—the best solution will be returned if the ILP solver does not terminate in 2 hours. For MSOAR, we run its binary version downloaded from (MSOAR Group, 2006). We compare our method with MSOAR, rather than the latest version MSOAR 2.0, because we focus on genomes with the same gene content, which implicitly requires that, after the speciation event, only DCJ operations are involved. Compared with MSOAR, MSOAR 2.0 aims to identify tandem duplications of genes *after* the speciation. Thus, under our evolutionary model that does not contain postspeciation duplications, MSOAR and MSOAR 2.0 are equivalent.

5.1. Simulation results

We simulate artificial genomes under an evolutionary model including segmental duplications and DCJs. We introduce duplicated genes through segmental duplications. For each segmental duplication, we uniformly select a position to start duplicating a segment of the genome and place the new copy to a new position. Since the average copy number of each gene in human, mouse, and rat genomes are 1.46, 1.55, and 1.28, respectively, we set the average copy number to 1.5 in our simulation. From a genome of S distinct genes, we generate an ancestor genome with *1*.5 S genes by randomly performing $S/(2 \cdot L)$ segmental duplications of length L (in terms of the number of genes in the segment). We then simulate two extant genomes from the ancestor by randomly performing N DCJs (in terms of inversions) independently. Thus, the true evolutionary distance between the two extant genomes is $2 \cdot N$. The reference bijection consists of those gene pairs that correspond to the same gene in the ancestor.

We first set $S = 1000$ and test four different lengths for segmental duplications ($L = 1$, 2, 5, 10). The results illustrate the trends and capabilities of the two methods in handling genomes with duplicated segments. We also vary the number of DCJs over a broad range ($N = 200, 210, \cdots, 500$) that reaches beyond the saturation point. For each combination of L and N, we randomly simulate five independent instances, and calculate the average accuracy of the bijection and the average deviation from the true evolutionary distances over these five instances for both methods.

Figure 5 shows the deviation from the true evolutionary distances for both methods. The first observation is that saturation starts occurring for a true evolutionary distance of 720: the DCJ distance obtained from the reference bijection is smaller than the true evolutionary distance, and the gap increases along with the increase of the true evolutionary distance. Second, when the true evolutionary distance is less than 720, our method obtains very accurate DCJ distances while MSOAR usually overestimates the DCJ distance. The difference is particularly pronounced for $L \geq 2$: in such cases, there exist identical segments in each genome, a situation that creates problems when MSOAR tries to partition each genome into a minimum number of common segments (Chen et al., 2005). Figure 6 shows the the accuracy of the bijections for both methods. For $L = 1$, both methods can correctly identify most gene pairs. For $L \geq 2$, our method significantly outperforms MSOAR. For large L, the accuracy of our method decreases rapidly beyond saturation, but continues to dominate MSOAR.

We also simulate very large genomes by setting $S = 5000$. Again we test different segmental duplications ($L = 1$, 2, 5, 10) and different number of DCJs ($N = 1000$, 1100, $\cdots$, 2000). Figure 7 shows the average accuracy of the bijection. For large L and small N, our method can identify almost all correct gene pairs, while MSOAR outputs a significant portion of incorrect pairs. Similar to the case of $S = 1000$, when the true evolutionary is large, the accuracy of our method decreases quickly, but still outperforms MSOAR.

The running time of MSOAR grows slowly as the the true evolutionary distance increases. For the most complicated case of $S = 5000$, $L = 10$, and $N = 2000$, MSOAR needs roughly 1 hour to finish. Regarding our method, when the true evolutionary distance is relatively small (for example, $N \leq 320$ when $S = 1000$ and $L = 5$, $N \leq 1500$ when $S = 5000$ and $L = 5$), the preprocessing method can fix a considerable portion of the adjacency graph, leaving a small ILP instance that can be solved very quickly (it takes only a few seconds and is faster than MSOAR). When the true evolutionary distance is relatively large, the ILP solver cannot terminate in 2 hours and a suboptimal solution is obtained. Usually, this solution is equal or very close to the optimal solution, because the ILP solver can find the optimal solution very quickly, but must spend more time to verify that it is optimal. This observation is also verified by the very high accuracy before the saturation point shown in Figure 6.

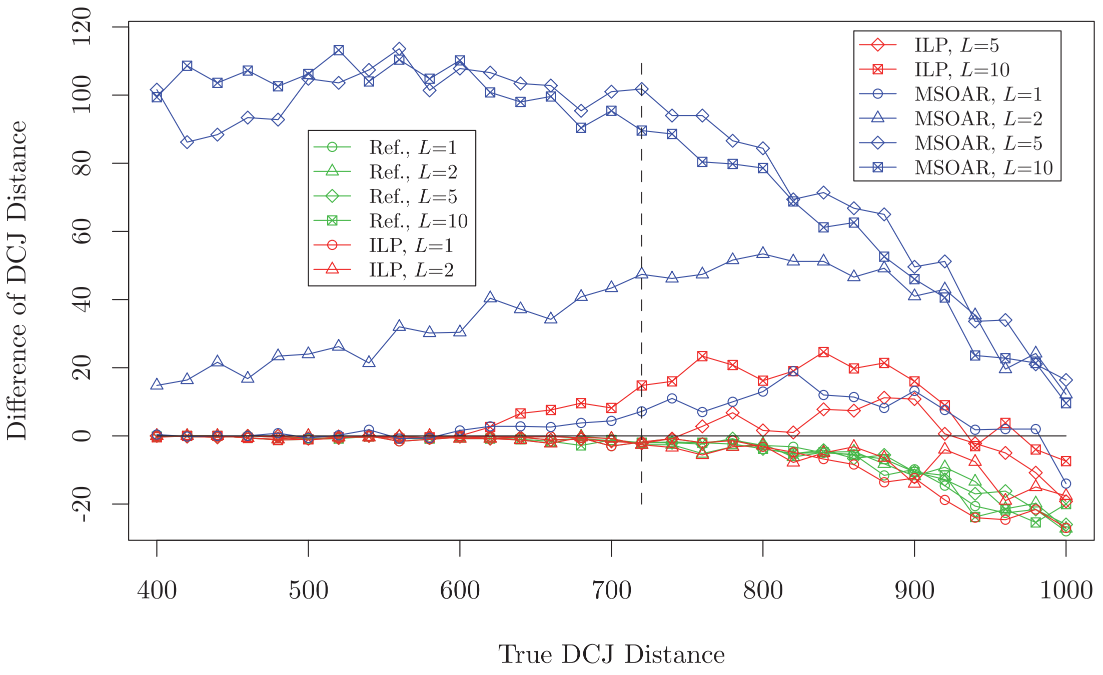

FIG. 5. Deviation from the true evolutionary distances for $S = 1000$. Green lines track reference bijection, red lines track our method, and blue lines track MSOAR.

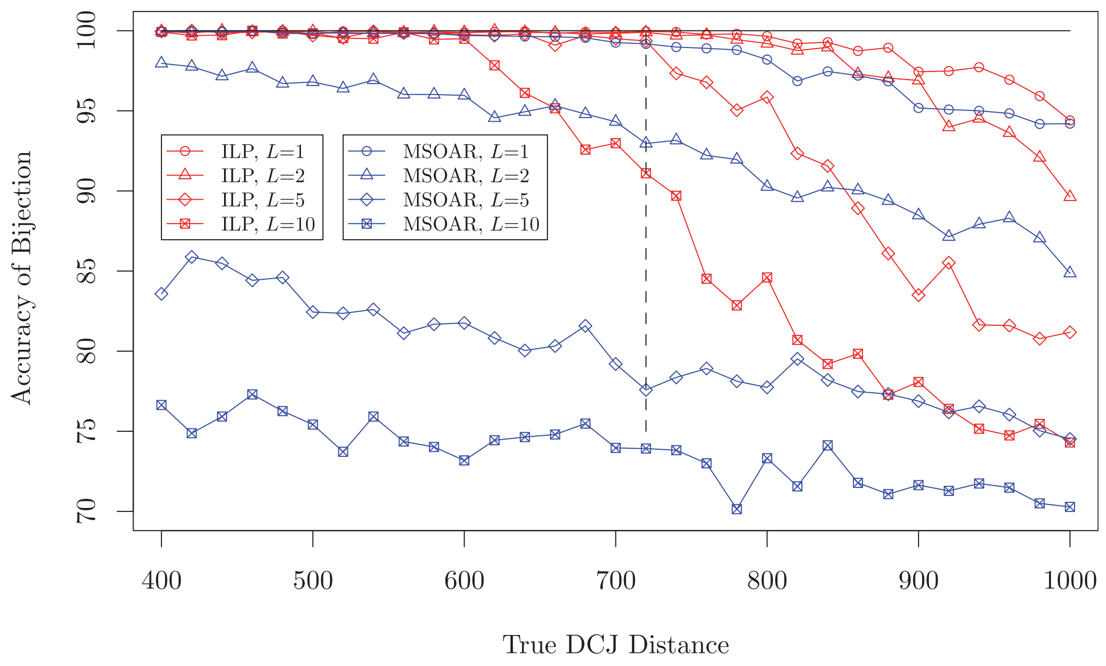

FIG. 6. The accuracy of the bijections for $S = 1000$. Red lines track our method, while blue lines track MSOAR.

5.2. *Application to orthology assignment*

Under a parsimonious evolutionary scenario, the optimal valid bijection between two genomes with the same gene content minimizes the number of DCJs after speciation, and thus infers the orthologous gene pairs (Chen et al., 2005). We test both methods for assigning orthologous genes between pairs of genomes. Human, mouse, and rat genomes are well annotated, so we chose them to evaluate the performance of the two methods. For each species, we downloaded the information for all protein-coding genes from Ensembl

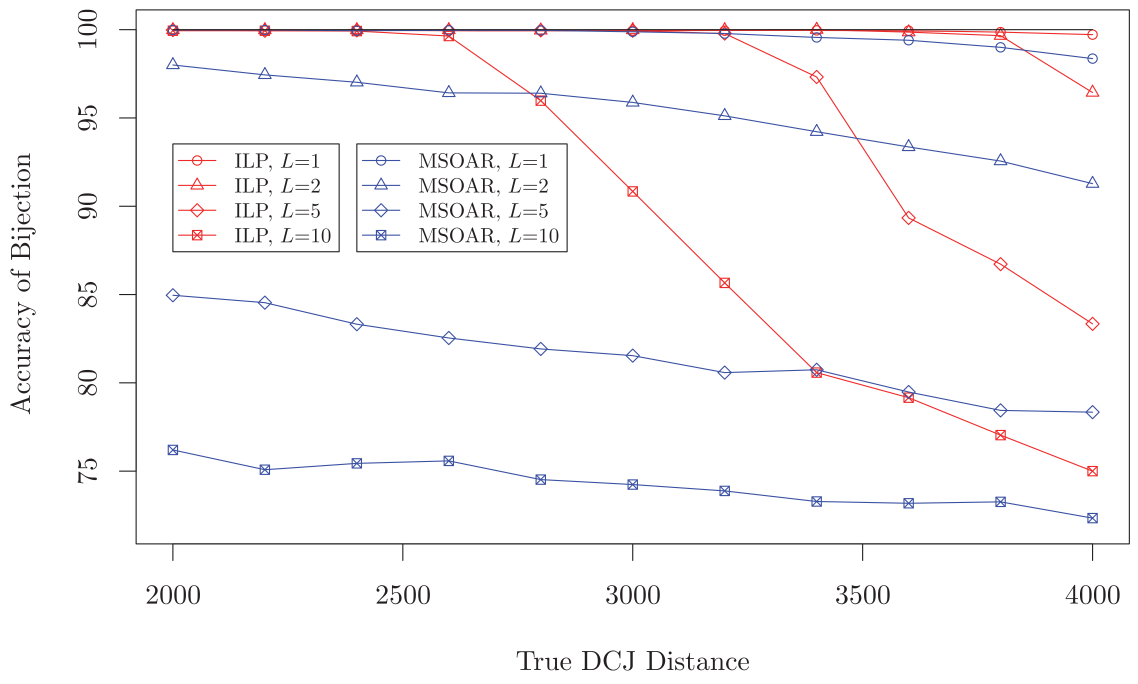

FIG. 7. The accuracy of the bijections for $S = 5000$. Red lines track our method, while blue lines track MSOAR.

Table 1. Comparison of Human, Mouse, and Rat Genomes

		Accuracy of bijection (%)		*DCJ distance*	
Species pairs	*Gene pairs*	*MSOAR*	*Our method*	*MSOAR*	*Our method*
Human mouse	14876	98.63	99.18	933	894
Human rat	12971	98.79	99.28	1320	1294
Mouse rat	13525	98.60	99.26	968	916

including gene family names, positions on the chromosomes, and gene symbols. If a gene has multiple alternative products, we keep its longest isoform. Two genes are considered homologous if they have the same Ensembl gene family name; they are considered orthologous if they have the same gene symbol. (Note that two orthologous genes are necessarily homologous, but two homologous genes need not be orthologous.) For a pair of genomes, we keep only orthologous gene pairs, thereby obtaining two genomes with the same gene content; our reference bijection is then defined by these orthologous gene pairs. For both methods, we use gene family and position information to infer orthologous relationships and compare them to the reference bijection.

The results of comparing these three genomes are shown in Table 1. Both methods mostly agree with annotation, indicating that the parsimonious model is appropriate when comparing these genomes; our method obtains slightly better accuracy. On human and mouse for example, our bijection has 122 different gene pairs compared with the reference bijection. Among these pairs, 34 of them can be explained by a simple structure, illustrated in Figure 8. For two identical segments, our method outputs a sequential bijection for which no DCJ operation is needed, while the reference bijection contains a crossover, for which at least two DCJ operations are needed. The other 87 pairs can be explained by 32 pairs of segments, for each of which our bijection needs fewer DCJ operations than the reference bijection. On the comparison of the DCJ distance, our method gets fewer DCJ operations than MSOAR in all three pairs.

6. CONCLUSION

We formulated the maximum cycle decomposition problem as an integer linear program. We proved a theorem that can be used to reduce the complexity while preserving optimality. The combination of the two gives a practical method to compute the exact DCJ distance for genomes with duplicate genes. Such a method is crucial for comparative genomics, since duplicate genes are commonly observed in most species.

The ILP formulation can be extended in various ways. First, we can use the relaxed LP (linear programming) techniques to design possible approximation algorithms. Second, when we apply it to do

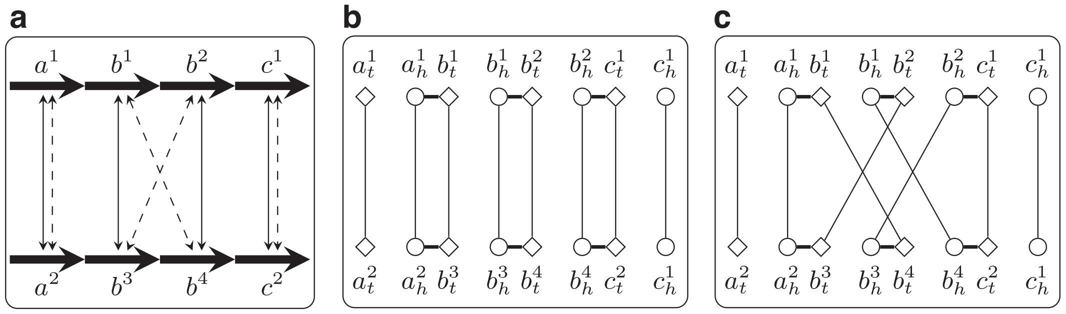
a
a^1 b^1 b^2 c^1
a^2 b^3 b^4 c^2
b
a_t^1 a_h^1 b_t^1 b_h^1 b_t^2 b_h^2 c_t^1 c_h^1
a_t^2 a_h^2 b_t^3 b_h^3 b_t^4 b_h^4 c_t^2 c_h^1
c
a_t^1 a_h^1 b_t^1 b_h^1 b_t^2 b_h^2 c_t^1 c_h^1
a_t^2 a_h^2 b_t^3 b_h^3 b_t^4 b_h^4 c_t^2 c_h^1

FIG. 8. Comparison of the reference bijection with our bijection. (**a**) Two identical segments. Our bijection is shown by solid lines while reference bijection is shown by dashed lines. (**b**) The adjacency graph corresponding to the our bijection, in which there are three cycles. (**c**) The adjacency graph corresponding to the reference bijection, in which there is only one cycle.

orthology assignment, we can also take the sequence similarity information into account by adding a term of the form $\sum_{e \in E} w_e \cdot x_e$ to the objective function, where w_e can be set to the similarity of the two genes. How to combine sequence similarity and DCJ distances remains an unexplored problem, but our ILP formulation provides a first step by allowing us to study linear combinations of the two.

We assumed that, after a speciation event, only DCJ operations are involved. This assumption is clearly unrealistic—it was made to simplify the problem and enable us to devise a first exact solution. However, now that our ILP method has proved successful, we can combine it with our previous work (Shao and Lin, 2012) to include single-gene deletion and single-gene insertion in the model.

AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Bader, D., Moret, B., and Yan, M. 2001. A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comput. Biol.* 8, 483–491.

Bailey, J., and Eichler, E. 2006. Primate segmental duplications: Crucibles of evolution, diversity and disease. *Nature Reviews Genetics* 7, 552–564.

Bergeron, A., Mixtacki, J., and Stoye, J. 2006. A unifying view of genome rearrangements. *Lecture Notes in Comp. Sci.* 4175, 163–173.

Bergeron, A., Mixtacki, J., and Stoye, J. 2009. A new linear-time algorithm to compute the genomic distance via the double cut and join distance. *Theor. Comput. Sci.* 410, 5300–5316.

Chen, X. 2010. On sorting permutations by double-cut-and-joins. *Lecture Notes in Comp. Sci.* 6196, 439–448.

Chen, X., Sun, R., and Yu, J. 2011. Approximating the double-cut-and-join distance between unsigned genomes. *BMC Bioinformatics* 12, S17.

Chen, X., Zheng, J., Fu, Z., et al. 2005. Assignment of orthologous genes via genome rearrangement. *ACM/IEEE Trans. on Comput. Bio. & Bioinf.* 2, 302–315.

Fertin, G., Labarre, A., Rusu, I., et al. 2009. *Combinatorics of Genome Rearrangements*. MIT Press, Inc., Cambridge, Massachuseetts.

Fu, Z., Chen, X., Vacic, V., et al. 2007. MSOAR: a high-throughput ortholog assignment system based on genome rearrangement. *J. Comput. Biol.* 14, 1160–1175.

Gurobi Optimization Inc. 2013. Gurobi optimizer reference manual. Houston, Texas.

Hannenhalli, S., and Pevzner, P. 1995. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Ann. ACM Symp. Theory of Comput. (STOC'95)*, pages 178–189.

Jean, G., and Nikolski, M. 2007. Genome rearrangements: a correct algorithm for optimal capping. *Inf. Proc. Letters* 104, 14–20.

Jiang, Z., Tang, H., Ventura, M., et al. 2007. Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. *Nature Genetics* 39, 1361–1368.

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

Laohakiat, S., Lursinsap, C., and Suksawatchon, J. 2008. Duplicated genes reversal distance under gene deletion constraint by integer programming. In *Proc. 8th IEEE Symp. on Bioinformatics and Bioengineering (BIBE'09)*, pages 527–530.

Lynch, M. 2007. *The Origins of Genome Architecture*. Sinauer Assoc., Sunderland, MA.

Moret, B., Lin, Y., and Tang, J. 2013. Rearrangements in phylogenetic inference: Compare, model, or encode?, 147–172. *In* Chauve, C., El-Mabrouk, N., and Tannier, E., eds. *Models and Algorithms for Genome Evolution*, *Computational Biology Series*, Vol. 19. Springer Verlag, Berlin.

MSOAR Group, 2006. http://msoar.cs.ucr.edu/.

Ozery-Flato, M., and Shamir, R. 2003. Two notes on genome rearrangement. *J. Bioinf. Comp. Bio.* 1, 71–94.

Shao, M., and Lin, Y. 2012. Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. *BMC Bioinformatics* 13, S13.

Shi, G., Zhang, L., and Jiang, T. 2010. MSOAR 2.0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement. *BMC Bioinformatics* 11, 10.

Suksawatchon, J., Lursinsap, C., and Bodén, M. 2007. Computing the reversal distance between genomes in the presence of multi-gene families via binary integer programming. *J. Bioinf. Comp. Bio.* 5, 117–133.

Tesler, G. 2002. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.* 65, 587–609.

Yancopoulos, S., Attie, O., and Friedberg, R. 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346.

Yancopoulos, S., and Friedberg, R. 2008. Sorting genomes with insertions, deletions and duplications by DCJ. *Lecture Notes in Comp. Sci.* 5267, 170–183.

Address correspondence to:
Mr. Mingfu Shao
EPFL IC IIF LCBB, INJ 211
Station 14, CH1015
Lausanne 1015
Switzerland

E-mail: mingfu.shao@epfl.ch