JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 24, Number 6, 2017

Pp. 571–580
DOI: 10.1089/cmb.2016.0149

On Computing Breakpoint Distances for Genomes with Duplicate Genes

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

ABSTRACT

A fundamental problem in comparative genomics is to compute the distance between two genomes in terms of its higher level organization (given by genes or syntenic blocks). For two genomes without duplicate genes, we can easily define (and almost always efficiently compute) a variety of distance measures, but the problem is NP-hard under most models when genomes contain duplicate genes. To tackle duplicate genes, three formulations (exemplar, maximum matching, and any matching) have been proposed, all of which aim to build a matching between homologous genes so as to minimize some distance measure. Of the many distance measures, the breakpoint distance (the number of nonconserved adjacencies) was the first one to be studied and remains of significant interest because of its simplicity and model-free property. The three breakpoint distance problems corresponding to the three formulations have been widely studied. Although we provided last year a solution for the exemplar problem that runs very fast on full genomes, computing optimal solutions for the other two problems has remained challenging. In this article, we describe very fast, exact algorithms for these two problems. Our algorithms rely on a compact integer-linear program that we further simplify by developing an algorithm to remove variables, based on new results on the structure of adjacencies and matchings. Through extensive experiments using both simulations and biological data sets, we show that our algorithms run very fast (in seconds) on mammalian genomes and scale well beyond. We also apply these algorithms (as well as the classic orthology tool MSOAR) to create orthology assignment, then compare their quality in terms of both accuracy and coverage. We find that our algorithm for the "any matching" formulation significantly outperforms other methods in terms of accuracy while achieving nearly maximum coverage.

Keywords: breakpoint distance, exemplar, gene family, ILP, intermediate, maximum matching, orthology assignment.

1. INTRODUCTION

The combinatorics and algorithmics of genomic rearrangements have been the subject of much research in comparative genomics since the problem was formulated in the 1990s (Fertin et al., 2009). Perhaps the most fundamental problem is the computation of some distance measure between two genomes. When the two genomes being compared have no duplicate genes (or synteny blocks, since the basic unit of

[1]Laboratory for Computational Biology and Bioinformatics, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
[2]Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

description need not be restricted to genes), we have linear-time algorithms for most of these distance problems, such as the breakpoint distance, the inversion distance (Bader et al., 2001), and the DCJ (double-cut-and-join) distance (Yancopoulos et al., 2005; Bergeron et al., 2006).

However, gene duplications are widespread events and have long been recognized as a major driving force of evolution (Bailey and Eichler, 2006; Lynch and Walsh, 2007). To define the distance in the presence of duplicate genes, three formulations have been proposed, all based on building a matching between duplicate genes and discarding copies not in the matching. This matching leads to treating each matched pair as a new gene family and thus removes duplicates, reducing the distance problem to its simplest version. In all formulations, the goal is to return that matching (from among all those obeying stated constraints), which minimizes the chosen distance. The first formulation is because of Sankoff (1999), who proposed the *exemplar* model (E-model): select exactly one matched pair in each gene family. Several years later, Blin et al. (2004) proposed the *maximum matching* model (M-model): use as many matched pairs as possible—until all genes in the genome with the smaller number of copies in each gene family are matched. Finally, Angibaud et al. (2007) proposed what we shall term the *intermediate* model (I-model): the matching must contain at least one matched pair per gene family—so that the focus is clearly on minimizing the distance, not on meeting constraints on the matching. Figure 1 illustrates these three formulations. Unfortunately, for almost all distance measures, the corresponding distance problems under the three formulations are NP-hard (Blin et al., 2007).

In this article, we focus on the distance problems in the presence of duplicate genes under the breakpoint distance measure. We refer to the three corresponding problems under the three formulations as E-BDP (short for exemplar breakpoint distance problem), M-BDP (maximum matching BDP) and 1-BDP (intermediate BPD), respectively. Many algorithms have been proposed for these problems. For E-BDP, Sankoff (1999) gave a first branch-and-bound algorithm in his original article. Nguyen et al. (2005) gave a much faster divide-and-conquer approach, whereas Angibaud et al. (2007) gave an exact algorithm by formulating it as an integer-linear program (ILP). In Shao and Moret (2015), we gave an exact solution through combining ILP and novel preprocessing algorithms and constraint-generating algorithms, which runs very fast (in a few seconds) and scales up to the largest genomes analyzed on both simulations and biological data sets. For M-BDP, Blin et al. (2004) had defined (but not tested) a branch-and-bound algorithm, whereas Swenson et al. (2005) gave an approximation algorithm. Angibaud et al. (2008) gave exact ILP formulations as well as several heuristics for both M-BDP and I-BDP—the heuristics being needed as the ILP formulations did not scale well. The ILP formulations in Angibaud et al. (2008) were tested on 12 bacterial genomes, with number of genes varying from a few hundreds to 3000; among the 66 pairwise comparisons, their ILP for I-BDP took 3 minutes on 63 of them, but could not terminate on the other 3 within a few hours. It thus remained a challenge to design fast and exact algorithms for I-BDP (and the simpler M-BDP) that could scale easily to mammalian genome sizes and beyond.

From the perspective of a biologist, neither the E-model nor the M-model is satisfactory. In real gene families, we expect to find multiple orthologs, not just a single pair, so the E-model is an oversimplification and provides only a small amount of information, which in turn could lead to distortions in the pairwise distances. The M-model suffers from an even more fundamental problem: forcing all genes in the genome with the smaller number of copies to be matched effectively amounts to stating that every one of these genes has an ortholog in the other genome, something that clearly need not always be true. In terms of evolutionary events, this requirement also gives a much larger weight to duplications and losses of genes than to genomic rearrangements, thus implying that duplication and loss events are much less likely than rearrangement events. In other words, the M-model loses the model-free characteristic of the breakpoint distance. In contrast, the I-model retains the model-free property of the breakpoint distance while making full use of the information present in the gene families.

In this article, we describe fast and exact algorithms for M-BDP and I-BDP. These algorithms employ ILP formulations and use a novel algorithm to reduce the number of key variables in the ILP; based on new

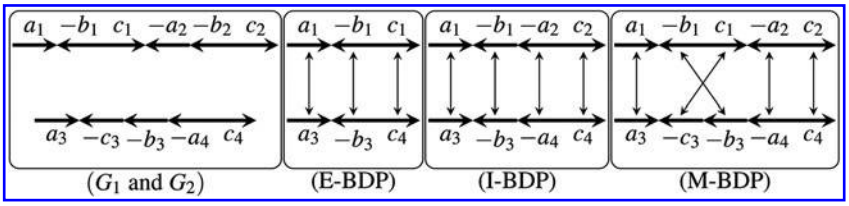

FIG. 1. Examples for E-BDP, I-BDP, and M-BDP. Optimal matchings are shown within each model.

results we prove about matchings and adjacencies. We evaluate these algorithms on simulated genomes and on several mammalian genomes. Our results show that our algorithms easily scale beyond the size of mammalian genomes: in all of our testing, almost all instances took a few seconds and none more that 70 seconds. They also demonstrate that our new algorithm to reduce the number of variables is a crucial contributor to this speed, especially for the more challenging I-BDP. Thus, with our previous (and equally fast) algorithm for E-BDP (Shao and Moret, 2015), we now have fast and exact algorithms to compute the breakpoint distance under all three formulations.

We also apply our algorithms to infer orthologs among five mammalian genomes and compare the quality of these assignments both among the three formulations and with the classic orthology tool MSOAR (Chen et al., 2005; Fu et al., 2007; Shi et al., 2010). (We use MSOAR as it is one of the few orthology tools based on genome rearrangements and matching and because it has been extensively tested by its authors against other orthology tools.) The results demonstrate that the I-model substantially outperforms all other methods in terms of accuracy while giving very high coverage—close to the M-model (which defines the maximum possible number).

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. Given a chromosome, we can reverse the list of symbols and switch all the signs, which will result in the same chromosome; for instance, $g_1 g_2 \cdots g_{n-1} g_n$ and $-g_n - g_{n-1} \cdots - g_2 - g_1$ represent the same linear chromosome. Homologous genes among the given genomes are grouped into *gene families*. In this article, we assume that the given genomes have the same set of gene families, denoted by $\mathcal{F}$. For a gene family $f \in \mathcal{F}$ and a genome G, we use $F(G, f)$ to denote the set of genes in G that come from f. We say a gene family $f \in \mathcal{F}$ is a *singleton* in G if $|F(G, f)| = 1$; otherwise we say f is a *multigene family*.

Two consecutive genes g and h on the same chromosome, with g ahead of h along the chromosome, form an *adjacency*, written as gh. Given two genomes G_1 and G_2, we say two adjacencies $g_1h_1 \in G_1$ and $g_2h_2 \in G_2$ form a *pair of shared adjacencies* or a *PSA*, written as $\langle g_1h_1, g_2h_2 \rangle$, if g_1 and g_2 (and also h_1 and h_2) have the same sign and come from the same gene family, or g_1 and h_2 (and also h_1 and g_2) have opposite signs and come from the same gene family. If two given genomes G_1 and G_2 contain only singletons (have at most one gene each per gene family), then, for each adjacency in G_1 (respectively, G_2), there exists at most one adjacency in G_2 (respectively, G_1) shared with it.

Given two genomes G_1 and G_2 that may contain multigene families, we define a *matching* between them as a one-to-one correspondence between a subset of genes in G_1 and a subset of genes in G_2, such that each element of the matching is a pair of homologous genes. We denote by $\mathcal{M}$ the set of all possible matchings between G_1 and G_2. For a matching $M \in \mathcal{M}$ and a gene family $f \in \mathcal{F}$, we use $M(f)$ to denote the set of gene pairs in M that come from f. We say a gene is *covered* by M if it appears in some pair in M. Given $M \in \mathcal{M}$, we can modify G_1 and G_2 as follows: we first remove all genes that are not covered by M, then set up a new distinct gene family for each pair of genes in M. Clearly, these two new genomes contain only singletons; we denote by $\mathcal{S}(M)$ the set of PSAs between such two new genomes induced by M.

Given two genomes, the *breakpoint distance problem* is to compute a matching (satisfying some requirements depending on certain models) such that the number of shared adjacencies between the new genomes induced by this matching is maximized. Define the following sets of matchings:

$$M_e = \{M \in \mathcal{M} : |M(f)| = 1, \forall f \in \mathcal{F}\};$$
$$M_i = \{M \in \mathcal{M} : |M(f)| \geq 1, \forall f \in \mathcal{F}\};$$
$$M_m = \{M \in \mathcal{M} : |M(f)| = \min\{|F(G_1, f)|, |F(G_2, f)|\}, \forall f \in \mathcal{F}\}.$$

Then the three problems corresponding to the three formulations can be written as

$$\max_{M \in \mathcal{M}_e} |\mathcal{S}(M)| \qquad (\mathrm{E-BDP})$$
$$\max_{M \in \mathcal{M}_i} |\mathcal{S}(M)| \qquad (\mathrm{I-BDP})$$
$$\max_{M \in \mathcal{M}_m} |\mathcal{S}(M)| \qquad (\mathrm{M-BDP})$$

Notice that our formulations are essentially based on *conserved adjacencies*, not breakpoints. For the E-model and the M-model, the two are equivalent; for the I-model, they are different (although clearly similar). This is because of the fact that the number of breakpoints equals the total number of adjacencies minus the number of conserved adjacencies, and that for the E-model and the M-model, the number of adjacencies is a fixed value, whereas for the I-model it is variant. Using conserved adjacencies rather than breakpoints has a significant advantage when the size of the matching is not fixed, as in the I-model the measure rewards both increased coverage and increased structural similarity.

3. ALGORITHMS

We described a fast and exact algorithm for E-BDP in previous work (Shao and Moret, 2015). In this section, we describe fast and exact algorithms for I-BDP and M-BDP. Both algorithms use the same framework, consisting of an ILP, described in Section 3.1, and an algorithm to reduce the number of variables in the ILP, described in Section 3.2. In each of them, we first describe the formulation or algorithm for I-BDP, then we state them for M-BDP, mainly focusing on clarifying their differences.

3.1. ILP formulations

We first generalize the definition of adjacency. For two genes g and h on the same chromosome (g is ahead of h), we use $[g, h]$ to represent the genes from g to h along the chromosome (including g and h), and use (g, h) to represent the genes between g and h (excluding g and h). We say $[g, h]$ forms a *potential adjacency* for I-BDP, if we can remove all genes in (g, h) such that at least one gene remains in each gene family (as required by I-BDP). We say two potential adjacencies $[g_1, h_1] \in G_1$ and $[g_2, h_2] \in G_2$ form a *pair of shared potential adjacencies* (PSPA), written as $\langle [g_1, h_1], [g_2, h_2] \rangle$, if g_1 and g_2 (and also h_1 and h_2) have the same sign and come from the same gene family (case 1), or g_1 and h_2 (and also h_1 and g_2) have opposite signs and come from the same gene family (case 2). Without loss of generality, in the following we always assume that a PSPA belongs to case 1—all corresponding results for PSPAs belonging to case 2 can be derived directly in a symmetric way. We denote by $\mathcal{P}_i$ the set of all PSPAs between G_1 and G_2 for I-BDP. Given a matching $M \in \mathcal{M}_i$, we say a PSPA $p = \langle [g_1, h_1], [g_2, h_2] \rangle \in \mathcal{P}_i$ *survives w.r.t. M* if we have $\langle g_1 h_1, g_2 h_2 \rangle \in \mathcal{S}(M)$.

Let $M_i^* \in \mathcal{M}_i$ be an optimal matching for I-BDP. Our ILP formulation to compute M_i^* has three types of variables. First, for every gene g in the two given genomes, we use one binary variable x_g to indicate whether g is covered by M_i^*. Second, for each pair of homologous genes $g_1 \in G_1$ and $g_2 \in G_2$, we use one binary variable y_{g_1,g_2} to indicate whether pair $\langle g_1, g_2 \rangle$ is in M_i^*. Third, for every PSPA $p \in \mathcal{P}_i$, we use one binary variable z_p to indicate whether p survives *w.r.t.* M_i^*.

Our ILP to compute M_i^* has three types of constraints. First, we require that for each gene family in each genome, at least one gene is covered by M_i^*:

$$\begin{aligned} \sum_{g \in F(G_1, f)} x_g \geq 1, \quad \forall f \in \mathcal{F}; \\ \sum_{g \in F(G_2, f)} x_g \geq 1, \quad \forall f \in \mathcal{F}. \end{aligned} \tag{1}$$

Second, we use the following constraints to guarantee that gene g is covered by M_i^* if and only if there exists a pair in M_i^* that includes g:

$$\begin{aligned} \sum_{g_2 \in F(G_2, f)} y_{g_1,g_2} = x_{g_1}, \quad \forall g_1 \in F(G_1, f), \forall f \in \mathcal{F}; \\ \sum_{g_1 \in F(G_1, f)} y_{g_1,g_2} = x_{g_2}, \quad \forall g_2 \in F(G_2, f), \forall f \in \mathcal{F}. \end{aligned}$$

Third, for each PSPA $p = \langle [g_1, h_1], [g_2, h_2] \rangle \in \mathcal{P}_i$, we require that, if p survives *w.r.t.* M_i^*, then we must have $\langle g_1, g_2 \rangle \in M_i^*$, $\langle h_1, h_2 \rangle \in M_i^*$, and that all genes in $(g_1, h_1) \cup (g_2, h_2)$ cannot be covered by M_i^*:

$$\begin{aligned} y_{g_1,g_2}, y_{h_1,h_2} \geq z_p, \quad \forall p = \langle [g_1, h_1], [g_2, h_2] \rangle \in \mathcal{P}_i; \\ 1 - z_p \geq x_g, \quad \forall g \in (g_1, h_1) \cup (g_2, h_2). \end{aligned} \tag{2}$$

The objective of the ILP is to maximize the sum of the variables for PSPAs:

$$\max \sum_{p \in \mathcal{P}_i} z_p.$$

A few modifications to this ILP for I-BDP provide an ILP for M-BDP. We say that $[g, h]$ is a potential adjacency for M-BDP if at least $\min\{|F(G_1,f)|, |F(G_2,f)|\}$ genes remain in gene family $f \in \mathcal{F}$ after genes in (g, h) are removed. Let $\mathcal{P}_m$ be the set of PSPAs for M-BDP and let $M_m^* \in \mathcal{P}_m$ be one optimal matching. We can construct the ILP to compute M_m^* from the mentioned one to compute M_i^* by replacing M_i^* with M_m^*, $\mathcal{P}_i$ with $\mathcal{P}_m$, and constraints (1) with the following constraints:

$$\sum_{g \in F(G_1,f)} x_g = \min\{|F(G_1,f)|, |F(G_2,f)|\}, \quad \forall f \in \mathcal{F};$$
$$\sum_{g \in F(G_2,f)} x_g = \min\{|F(G_1,f)|, |F(G_2,f)|\}, \quad \forall f \in \mathcal{F}.$$

Our ILP formulations use the same ideas we used for the exact algorithm to solve E-BDP in Shao and Moret (2015). They are similar to those proposed in Angibaud et al. (2008), but have fewer variables. In fact, the variables in our formulations are a subset of those in the formulations of Angibaud et al. (2008): the authors use two additional binary variables, c_{g_1,h_1} and c_{g_2,h_2}, for each PSPA $\langle [g_1, h_1], [g_2, h_2] \rangle$ to indicate whether g_1h_1 and g_2h_2 form adjacencies in the resulting genomes. In our formulations, the functions of these variables are carried out by constraints (2). ILP solvers typically do better with fewer variables and do not suffer (and often benefit) from the addition of extra constraints.

3.2. Building a sufficient subset

The number of binary variables corresponding to the PSPAs is the most important factor affecting the efficiency of our ILP formulations. We now describe an algorithm to reduce the number of these variables while preserving the optimality of the ILP. Formally, we say $\mathcal{P} \subset \mathcal{P}_i$ is *sufficient* if there exists an optimal matching $M_i^* \in \mathcal{M}_i$ such that for any PSA $\langle g_1h_1, g_2h_2 \rangle \in \mathcal{S}(M_i^*)$, its corresponding PSPA $\langle [g_1, h_1], [g_2, h_2] \rangle$ is in $\mathcal{P}$. Clearly, if we replace $\mathcal{P}_i$ by a sufficient subset $\mathcal{P}$ in the ILP formulation described in Section 3.1, an optimal matching can be still obtained. In the following we first prove two conditions to reduce a single PSPA, then use these results to devise an iterative algorithm to compute a sufficient subset of reduced cardinality.

Intuitively, the first lemma states that we can remove a PSPA without breaking sufficiency if it can be split into two PSPAs.

Lemma 1. *Let $\mathcal{P} \subset \mathcal{P}_i$ be a sufficient subset. Let $p = \langle [g_1, h_1], [g_2, h_2] \rangle \in \mathcal{P}$. If there exist gene $x_1 \in (g_1, h_1)$ and gene $x_2 \in (g_2, h_2)$ that have the same sign and come from the same gene family, then we have that $\mathcal{P} \backslash \{p\}$ is sufficient.*

Proof. We prove this lemma by contradiction. Suppose that $\mathcal{P} \backslash \{p\}$ is not sufficient. Let M_i^* be any optimal matching. Since $\mathcal{P}$ is sufficient but $\mathcal{P} \backslash \{p\}$ is not, p survives *w.r.t.* M_i^*, that is, we have $\langle g_1h_1, g_2h_2 \rangle \in \mathcal{S}(M_i^*)$. Let $M_i' = M_i^* \cup \{\langle x_1, x_2 \rangle\}$. Clearly, we have that $\mathcal{S}(M_i') = \mathcal{S}(M_i^*) \backslash \{\langle g_1h_1, g_2h_2 \rangle\} \cup \{\langle g_1x_1, g_2x_2 \rangle, \langle x_1h_1, x_2h_2 \rangle\}$. Thus we have that $|\mathcal{S}(M_i')| = |\mathcal{S}(M_i^*)| + 1$, contradicting with the assumption that M_i^* is optimal. ■

We now give another condition to reduce the size of a sufficient subset. Intuitively, the following lemma states that we can remove a PSPA without breaking sufficiency if there exists a gene inside that can play the same role as one of its four boundary genes.

Lemma 2. *Let $\mathcal{P} \subset \mathcal{P}_i$ be a sufficient subset. Set $p = \langle [g_1, h_1], [g_2, h_2] \rangle \in \mathcal{P}$ and $X = \{\langle x_1, x_2 \rangle : \langle [h_1, x_1], [h_2, x_2] \rangle \in \mathcal{P}\}$. If there exists $h \in (g_1, h_1)$ satisfying $\langle [g_1, h], [g_2, h_2] \rangle \in \mathcal{P}$, such that for all $\langle x_1, x_2 \rangle \in X$, we have $\langle [h, x_1], [h_2, x_2] \rangle \in \mathcal{P}$, then $\mathcal{P} \backslash \{p\}$ is sufficient.*

Proof. Let M_i^* be an optimal matching. If p does not survive *w.r.t.* M_i^*, then we can conclude that $\mathcal{P} \backslash \{p\}$ is sufficient. Now consider the case where p survives *w.r.t.* M_i^*, that is, $\langle g_1h_1, g_2h_2 \rangle \in \mathcal{S}(M_i^*)$. Set $M_i' = M_i^* \backslash \{\langle h_1, h_2 \rangle\} \cup \{\langle h, h_2 \rangle\}$. Consider the difference between $\mathcal{S}(M_i^*)$ and $\mathcal{S}(M_i')$. Clearly, at most two PSAs in $\mathcal{S}(M_i^*)$ can be affected by the removal of $\langle h_1, h_2 \rangle$. One of them is $\langle g_1h_1, g_2h_2 \rangle$, while we have

$\langle g_1 h, g_2 h_2\rangle \in \mathcal{S}(M_i')$ in return. If we further have $\langle h_1 x_1, h_2 x_2\rangle \in \mathcal{S}(M_i^*)$ for some $\langle x_1, x_2\rangle$, we must also have $\langle h x_1, h_2 x_2\rangle \in \mathcal{S}(M_i')$ in return. Thus, we have $|\mathcal{S}(M_i')| = |\mathcal{S}(M_i^*)|$, which implies that M_i' is also optimal. In contrast, we can show that all PSAs in $\mathcal{S}(M_i')$ have their corresponding PSPAs in $\mathcal{P}\backslash\{p\}$. In fact, p does not survive *w.r.t.* M_i', and we have $\langle [g_1, h], [g_2, h_2]\rangle \in \mathcal{P}$ and $\langle [h, x_1], [h_2, x_2]\rangle \in \mathcal{P}$; finally, the validity of all other PSAs in $\mathcal{S}(M_i')$ is guaranteed by the sufficiency of $\mathcal{P}$. Thus, in addition to the optimality of M_i', we have that $\mathcal{P}\backslash\{p\}$ is sufficient. ■

Given a PSPA $\langle [g_1, h_1], [g_2, h_2]\rangle$, Lemma 2 only proves the condition related to h_1. We can derive the other three conditions for g_1, g_2, and h_2 analogously. If a PSPA passes any of these four conditions, it can be removed while keeping sufficiency.

Based on the mentioned two lemmas, our algorithm to build a sufficient subset (BSS) of reduced cardinality proceeds as follows. The algorithm initializes the current set of PSPAs $\mathcal{P}$ as $\mathcal{P}_t$, and then it iteratively removes redundant PSPAs in $\mathcal{P}$. In each iteration, the algorithm examines each PSPA $p \in \mathcal{P}$ using the mentioned two lemmas: if p passes either of the tests, then we update $\mathcal{P}$ as $\mathcal{P}\backslash\{p\}$ and start a new iteration. The algorithm terminates if no PSPA passes any tests in an iteration.

We give two examples to illustrate the effect of the BSS algorithm. First, suppose that two genomes contain a pair of nontrivial ($n > 1$) shared segments $\langle g^1 g^2 \cdots g^n, h^1 h^2 \cdots h^n\rangle$, that is, g^k and h^k have the same sign and come from the same gene family, for $1 \le k \le n$. Then there are $n \cdot (n-1)/2$ PSPAs generated from this shared segment, namely, $\langle [g^i, g^j], [h^i, h^j]\rangle$ for all $1 \le i < j \le n$. After running BSS on these PSPAs, we can get a sufficient subset with only $(n-1)$ PSPAs, namely, $\langle [g^k, g^{k+1}], [h^k, h^{k+1}]\rangle$, $1 \le k \le n-1$. The removal of PSPAs here is because of Lemma 1—Lemma 1 is very effective when genomes contain duplicate segments. Second, suppose that G_1 contains a segment $a_1 b_1 c_1$ and G_2 contains a segment $a_2 b_2 b_3 \cdots b_n c_2$. Here $2 \cdot (n-1)$ PSPAs can be generated from these two segments, namely, $\langle [a_1, b_1], [a_2, b_k]\rangle$ and $\langle [b_1, c_1], [b_k, c_2]\rangle$, $2 \le k \le n$. After applying BSS on these PSPAs, we can see that a sufficient subset with at most four PSPAs is returned, namely, $\{\langle [a_1, b_1], [a_2, b_2]\rangle, \langle [b_1, c_1], [b_n, c_2]\rangle\} \cup \{\langle [a_1, b_1], [a_2, b_k]\rangle, \langle [b_1, c_1], [b_k, c_2]\rangle\}$ for some k, $2 \le k \le n$, where the value of k depends on the order in which these PSPAs are tested in BSS. The removal of PSPAs here is because of Lemma 2—Lemma 2 is very effective when genomes contain locally duplicate genes.

Since M-BDP requires keeping the maximum number of pairs, we have that a PSPA $\langle [g_1, h_1], [g_2, h_2]\rangle \in \mathcal{P}_m$ cannot have a pair of genes $x_1 \in (g_1, h_1)$ and $x_2 \in (g_2, h_2)$ within the same gene family. In other words, the PSPAs that are tested for removal in Lemma 1 do not appear in $\mathcal{P}_\mathrm{m}$ by definition. In fact, it is clear that $\mathcal{P}_\mathrm{m}$ is a (usually very small) subset of $\mathcal{P}_t$, making M-BDP significantly easier to solve than I-BDP (as is also apparent from the results of Sections 4 and 5). Lemma 2 holds unchanged for M-BDP. Thus the BSS algorithm for M-BDP uses only Lemma 2.

4. SIMULATION RESULTS

We refer to the full algorithms (ILP plus BSS) for I-BDP and M-BDP as I-A1 and M-A1, respectively, and refer to the baseline algorithms (just the ILP) for I-BDP and M-BDP as I-A0 and M-A0, respectively. We use simulated data to evaluate these four algorithms to illustrate both the performance of the full algorithms and the effectiveness of the BSS reduction algorithm. We do not explicitly compare with the algorithms proposed in Angibaud et al. (2008) for I-BDP and M-BDP, because their performance is similar to and bounded by that of I-A0 and M-A0, respectively, as discussed in Section 3.1.

We simulate a pair of genomes as follows. We start from a genome with only one linear chromosome consisting of N singletons. We then perform S_1 segmental duplications to make some multigene families, which then form the ancestor genome. 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. After that on each branch, it happens randomly mixed I inversions and S_2 segmental duplications. An inversion 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 2 (the average copy number of each gene family in human, mouse, and rat genomes is 1.46, 1.55, and 1.28, respectively; thus in terms of duplicated genes, our simulated genomes are more complicated than typical mammalian genomes). Therefore, we have that $(S_1 + S_2) \cdot L = N$. Let $r_1 = S_1/(S_1 + S_2)$ be the percentage of segmental duplications before speciation for each genome. Let $r_2 = I/(I + S_2)$ be the percentage of inversions after speciation. We can calculate that $S_1 = r_1 \cdot N/L$, $S_2 = (1 - r_1) \cdot N/L$, and $I = r_2 \cdot (1 - r_1) \cdot N/(L \cdot (1 - r_2))$. Thus, a simulation configuration is determined by

TABLE 1. COMPARISON OF THE FOUR ALGORITHMS ON PARAMETERS ($N = 10,000, L, r_1 = 0, r_2$)

	$L=1$				$L=5$				$L=10$			
r_2	*I-A1*	*I-A0*	*M-A1*	*M-A0*	*I-A1*	*I-A0*	*M-A1*	*M-A0*	*I-A1*	*I-A0*	*M-A1*	*M-A0*
0.0	1	2	0	0	25	(0)	8	8	55	(0)	10	12
0.1	1	1	0	0	21	(0)	7	9	43	(0)	10	13
0.2	0	0	0	0	14	(0)	7	7	33	(0)	9	11
0.3	0	0	0	0	11	(0)	5	4	26	(0)	8	8
0.4	0	0	0	0	6	554	3	3	22	(0)	7	8
0.5	0	0	0	0	4	123	2	2	15	(0)	6	5
0.6	0	0	0	0	2	38	1	1	10	(0)	4	4
0.7	0	0	0	0	1	3	1	0	7	430	3	2
0.8	0	0	0	0	0	0	0	0	4	70	1	1
0.9	0	0	0	0	0	0	0	0	0	0	0	0

For each combination, if all 10 instances finish in 30 minutes, the average running time (in seconds) is recorded; otherwise, the number of finished instances is recorded in parentheses. All four programs were run on an 8-core (2.1 GHz) PC with 16 GB memory.

parameters (N, L, r_1, r_2). For each parameter combination, we randomly simulate 10 independent instances and run the 4 algorithms (I-A1, I-A0, M-A1, and M-A0) to calculate the average running time over these 10 instances. Since ILP might take very long time, we give a time limit of 30 minutes for each instance—an algorithm will be terminated when it exceeds such time limit.

In the following experiments, all our ILP instances are solved with GUROBI. We first test parameters $(N = 10,000, L, r_1 = 0, r_2)$, where $r_2 \in \{0.0, 0.1, \cdots, 0.9\}$ and $L \in \{1, 5, 10\}$. In this setting, all segmental duplications appear after speciation ($r_1 = 0$), and the expected number of genes in each extant genome is 20,000. The results on these parameters are shown in Table 1. First, we can observe that as L increases, all algorithms take more time, indicating that the simulated instances become harder with larger L. This is because longer shared segments create drastically more PSPAs [in order of $O(n^2)$, where n is the length of this shared segments]. Second, all the four algorithms take less time as r_2 increases. This is because larger r_2 means more inversions after speciation, which can destroy existing PSPAs. Third, we can see I-A1 can finish in <1 minute for all parameters, whereas I-A0 will exceed time limit for large L and small r_2—this proves that the BSS algorithm is very crucial for I-BDP. Fourth, we can see both M-A1 and M-A0 can finish in a very short time for all parameters, indicating that the ILP formulation for M-BDP is already very efficient. This is because the definition of M-BDP determines that the number of PSPAs is reasonably small, as we analyzed theoretically in Section 3.2. For the same reason, we can also observe that M-BDP is easier to solve than I-BDP.

We then test parameters $(N = 20,000, L, r_1 = 0.5, r_2)$, where $r_2 \in \{0.0, 0.1, \cdots, 0.9\}$ and $L \in \{1, 5, 10\}$. In this setting, the expected number of genes in the ancestor genome is 30,000, and the expected number of

TABLE 2. COMPARISON OF THE FOUR ALGORITHMS ON PARAMETERS ($N = 20,000, L, r_1 = 0.5, r_2$)

	$L=1$				$L=5$				$L=10$			
r_2	*I-A1*	*I-A0*	*M-A1*	*M-A0*	*I-A1*	*I-A0*	*M-A1*	*M-A0*	*I-A1*	*I-A0*	*M-A1*	*M-A0*
0.0	6	(0)	2	2	48	(0)	22	25	104	(0)	44	36
0.1	8	(0)	2	2	45	(0)	26	22	93	(0)	39	39
0.2	5	521	1	1	42	(0)	22	24	89	(0)	38	37
0.3	6	200	1	1	37	(0)	20	19	83	(0)	36	32
0.4	8	28	1	1	31	(0)	18	17	65	(0)	31	29
0.5	4	3	2	1	29	(0)	16	17	57	(0)	35	31
0.6	2	1	1	1	25	(0)	13	14	52	(0)	29	26
0.7	1	1	1	0	18	(0)	11	15	42	(0)	27	22
0.8	1	1	1	0	8	336	5	7	27	(0)	18	15
0.9	1	1	2	0	2	3	1	1	11	437	8	5

The setup is the same as in Table 1.

genes in each extant genome is 40,000. The results are shown in Table 2. Again, we can observe that the instances become harder to solve as L increases and r_2 decreases. Also observe that M-A1 and M-A0 take similar (and very small) amount of time, since the ILP formulation for M-BDP is already very efficient. We emphasize that I-A0 can only finish within the time limit for small L and large r_2, whereas I-A1 can get the optimal solution for all parameters very fast, showing that the BSS algorithm plays a key role in producing a fast algorithm for I-BDP. Notice that in these simulations, the size of extant genomes exceeds that of typical mammalian genomes and our full algorithms (I-A1 and M-A1) return optimal solutions in a very short time.

5. BIOLOGICAL RESULTS

We study five well-annotated genomes, 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 their Ensembl gene family names from Ensembl. Genes are grouped into the same gene family if they have the same Ensembl gene family name. For each species, we merge each group of tandemly arrayed genes into a single gene by keeping only the first gene and discarding the following genes in the group.

We perform pairwise comparisons among these five species. For each pair of species, we compare the running time for the four algorithms (I-A1, I-A0, M-A1, and M-A0). We also run another two algorithms for comparison, namely, the exact algorithm to solve E-BDP described in Shao and Moret (2015) (referred to as E-A1) and MSOAR (Fu et al., 2007), which uses heuristics to compute a matching such that the inversion distance induced by this matching is minimized. The results are shown in Table 3. Note that I-A0 cannot finish within the time limit for any pair, whereas I-A1 can finish in a very short time (<30 seconds except one taking 72 seconds) for all pairs, once again indicating that the BSS algorithm is indispensable for solving I-BDP. The BSS algorithm improves the solution for M-BDP as well, allowing M-A1 to finish within 2 seconds for all pairs. (MSOAR takes quite long time for these pairs, ranging from half an hour to a few hours.) Thus we now have exact and very fast algorithms (E-A1, I-A1, and M-A1) for all three formulations.

All of these algorithms give a matching between the homologous genes for each pair of species, a matching that defines a subset of orthologs under a parsimonious evolutionary assumption. We thus apply our full algorithms (E-A1, I-A1, and M-A1) together with MSOAR to infer orthologs among these five species. The quality of the inferred orthologs is evaluated using two measures, the coverage (the number of orthologous pairs identified) and the accuracy. To compute the accuracy of a matching, we use the gene symbols (HGNC symbols for primate genes, MGI symbols for mouse genes, and RGD symbols for rat genes, downloaded from Ensembl): those gene pairs that have the same gene symbol form the set of *true orthology pairs*. We say a pair in a matching is *trivial* if it consists of two singletons. Notice that by definition all trivial pairs must appear in the matchings returned by these four algorithms. We thus exclude trivial pairs for comparison. Among the nontrivial pairs in a matching, we say a pair is *assessable* if at least one gene in this pair is covered by some true orthology pair. Then the *accuracy* of a matching is defined as

Table 3. The Running Time (in Seconds) of the Algorithms on Comparing Five Genomes

Species pairs	*I-A1*	*I-A0*	*M-A1*	*M-A0*	*E-A1*	*MSOAR*
G.g. and *H.s.*	8	N/A	1	1	2	1770
G.g. and *M.m.*	20	N/A	1	59	3	5298
G.g. and *P.a.*	16	N/A	1	1	2	3191
G.g. and *R.n.*	22	N/A	1	4	3	12,555
H.s. and *M.m.*	23	N/A	1	2	2	3660
H.s. and *P.a.*	9	N/A	1	2	2	1585
H.s. and *R.n.*	22	N/A	1	6	2	7328
M.m. and *P.a.*	18	N/A	1	1	2	4287
M.m. and *R.n.*	72	N/A	2	66	2	5627
R.n. and *P.a.*	18	N/A	1	2	3	6009

I-A0 cannot finish in 30 minutes for any species pair.

Table 4. The Performance of the Algorithms on Inferring Orthologs Among the Five Genomes

		Nontrivial				*Accuracy (%)*			
Species pairs	*Trivial*	*E-A1*	*I-A1*	*M-A1*	*MSOAR*	*E-A1*	*I-A1*	*M-A1*	*MSOAR*
G.g. and *H.s.*	8331	3448	7830	8131	8051	97.51	98.18	97.56	97.60
G.g. and *M.m.*	7304	3478	7572	8025	7858	97.33	98.29	97.61	97.48
G.g. and *P.a.*	7737	3399	7466	7893	7720	97.26	98.10	97.37	97.17
G.g. and *R.n.*	6915	3787	7826	8610	8317	94.65	96.12	94.56	94.43
H.s. and *M.m.*	7932	3250	7546	7834	7722	97.89	98.71	98.16	98.20
H.s. and *P.a.*	8091	3223	7355	7585	7501	98.27	98.73	98.03	98.05
H.s. and *R.n.*	7436	3638	7808	8196	8072	94.64	96.10	94.94	94.93
M.m. and *P.a.*	7311	3276	7208	7537	7408	97.91	98.61	97.95	97.81
M.m. and *R.n.*	7953	3706	8376	8837	8671	94.82	96.52	95.63	95.20
R.n. and *P.a.*	6911	3610	7466	7987	7789	94.90	96.63	95.27	95.08

the ratio between the number of nontrivial true orthology pairs in this matching and the number of (nontrivial) assessable pairs in this matching.

The quality of the orthologs inferred by these four algorithms is shown in Table 4. First, we can observe that the coverage of I-A1 (by definition, it is between E-A1 and M-A1) is much closer to M-A1 (much higher than E-A1). Second, notice that the accuracy of I-A1 significantly outperforms the other three algorithms—it is 1.08% higher than E-A1, 0.89% higher than M-A1, and 1% higher than MSOAR, on average over the 10 pairs. Third, the quality of MSOAR is very close to that of M-A1, in terms of both the coverage and accuracy (the accuracy of M-A1 is 0.11% higher than that of MSOAR on average). Thus, we believe that I-A1 is an excellent choice for inferring orthologs, which outperforms in terms of both coverage and accuracy.

We do not evaluate the accuracy of these algorithms on simulation data. The reason is that orthology assignment requires a biologically credible model for generating simulation data—a model that needs to combine duplications, losses, rearrangements, and sequence mutations and indels. Such a model that takes into account all these evolutionary events in a biologically reasonable way is currently not available. Besides, we have high-quality fully assembled genomes with curated annotations to assess these methods, which leaves little necessity to perform comparison on simulation data.

6. CONCLUSION AND DISCUSSION

We have described exact and very fast algorithms for the three breakpoint distance problems, by formulating them as ILPs and designing additional algorithms to improve their efficiency through proving key properties of these problems. Using extensive experiments on both simulations and biological data sets, we have demonstrated that these algorithms scale beyond the size of mammalian genomes, and also achieve very high accuracy when applied to infer orthologs. We conclude that among those orthology assignment tools, I-A1 is the most suitable choice, since it gives highest accuracy and nearly highest coverage.

We also help understanding the structures of these problems through proving several properties. As we have already illustrated, these properties are crucial in designing efficient algorithms for the corresponding problems. Notice that some properties are common among the three problems, whereas some others only hold for one or two of them. In Section 3.2, we give some theoretical analysis between I-BDP and M-BDP through showing that Lemma 1 only applies for I-MDP, whereas Lemma 2 holds for both of them. Now we further state the relationship between E-BDP and I-BDP/M-BDP. We can easily prove that both lemmas in this article also apply to E-BDP. However, in Shao and Moret (2015), we proved a lemma for E-BDP stating that if one pair in a PSA is in an optimal matching, then the other pair can also be fixed optimally. However, this property does not hold for I-BDP and M-BDP. To see that, consider the example of $G_1 = a_1 b_1 c_1 b_2 d_1$ and $G_2 = a_2 b_3 d_2 b_4 c_2$. Notice that $\langle a_1, a_2 \rangle$ is a singleton pair and thus must be in any optimal matching. If we apply this property, then $\langle b_1, b_3 \rangle$ is also in some optimal matching. However, we can easily verify that for both I-BDP and M-BDP, $\langle b_1, b_3 \rangle$ is not in any optimal matching.

ACKNOWLEDGMENT

We thank Daniel Dörr for helpful discussions.

AUTHOR DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Angibaud, S., Fertin, G., Rusu, I., et al. 2007. A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes, 16–29. *In* Tesler, G., and Durand, D., eds. *Comparative Genomics, volume 4751 of Lecture Notes in Computer Science*.

Angibaud, S., Fertin, G., Rusu, I., et al. 2008. Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. *J. Comput. Biol.* 15, 1093–1115.

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. *Nat. Rev. Genet.* 7, 552–564.

Bergeron, A., Mixtacki, J., and Stoye, J. 2006. A unifying view of genome rearrangements, 163–173. *In Proceedings of the 6th Workshop on Algorithms in Bioinformatics (WABI'06), volume 4175 of Lecture Notes in Computer Science*.

Blin, G., Chauve, C., and Fertin, G. 2004. The breakpoint distance for signed sequences, 3–16. *In Proceedings of the 1st Conference on Algorithms and Computational Methods for Biochemical and Evolutionary Networks (CompBioNets'04), volume 3*.

Blin, G., Chauve, C., Fertin, G., et al. 2007. Comparing genomes with duplications: A computational complexity point of view. *ACM/IEEE Trans. Comput. Biol. Bioinf.* 4, 523–534.

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

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

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.

Lynch, M., and Walsh, B. 2007. *The Origins of Genome Architecture*, vol. 98. Sinauer Associates, Sunderland, MA.

Nguyen, C., Tay, Y., and Zhang, L. 2005. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics* 21, 2171–2176.

Sankoff, D. 1999. Genome rearrangement with gene families. *Bioinformatics* 15, 909–917.

Shao, M., and Moret, B. 2015. A fast and exact algorithm for the exemplar breakpoint distance, 309–322. *In Proceedings of the 19th International Conference on Computational Molecular Biology (RECOMB'15), volume 9029 of Lecture Notes in Computer Science*.

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.

Swenson, K., Marron, M., Earnest-DeYoung, J., et al. 2005. Approximating the true evolutionary distance between genomes, 121–129. *In Proceedings of the. 7th SIAM Workshop on Algorithm Engineering & Experiments (ALENEX'05).*

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

Address correspondence to:
Mingfu Shao
GHC 7401
5000 Forbes Ave.
Pittsburgh, PA 15206
USA

E-mail: mingfu.shao@cs.cmu.edu