

Genome analysis

Divide-and-conquer approach for the exemplar breakpoint distance

C. Thach Nguyen¹, Y. C. Tay^{1,2} and Louxin Zhang^{2,*}¹School of Computing and ²Department of Mathematics, National University of Singapore, Singapore 117543

Received on December 3, 2004; revised on February 5, 2005; accepted on February 10, 2005

Advance Access publication February 15, 2005

ABSTRACT

Motivation: A one-to-one correspondence between the sets of genes in the two genomes being compared is necessary for the notions of breakpoint and reversal distances. To compare genomes where there are paralogous genes, Sankoff formulated the exemplar distance problem as a general version of the genome rearrangement problem. Unfortunately, the problem is NP-hard even for the breakpoint distance.

Results: This paper proposes a divide-and-conquer approach for calculating the exemplar breakpoint distance between two genomes with multiple gene families. The combination of our approach and Sankoff's branch-and-bound technique leads to a practical program to answer this question. Tests with both simulated and real datasets show that our program is much more efficient than the existing program that is based only on the branch-and-bound technique.

Availability: Code for the program is available from the authors.

Contact: matzlx@nus.edu.sg

1 INTRODUCTION

In the theory of genome rearrangement, a genome is usually modeled as a permutation (or equivalently a linear order) on a set of genes and the breakpoint or reversal distances are used for evolutionary inference (Nadeau and Taylor, 1984; Watterson *et al.*, 1982; International Mouse Genome Consortium, 2002; Moret *et al.*, 2005). The breakpoint and signed reversal distances were introduced by Watterson *et al.* (1982) and Sankoff (1989), respectively. Since then, efficient algorithms for these and other measures have been studied extensively [see Kececioğlu and Sankoff (1994); Sankoff and Blanchette (1998) and El-Mabrouk (2005) for a complete bibliography]. The breakpoint distance is easily calculated in linear time. However, the permutations is polynomial-time computable only for the signed reversal distance (Hannenhalli and Pevzner, 1999; Caprara, 1997; Kaplan *et al.*, 2000).

A one-to-one correspondence between the sets of genes in the two genomes is necessary for the notions of both breakpoint and reversal distances. While this may be appropriate for small viruses and mitochondria genomes, it becomes problematic when applied to eukaryotic genomes where paralogous genes exist. Hence, several approaches have been proposed for comparison of genomic sequences with gene families. In inferring phylogeny from genome rearrangement data, Tang and Moret (2003) assumed that the size of

each gene family is the same in each genome and simply treated each gene member in a gene family as a distinct gene. Similarly, Chen *et al.* (2005) also assumed that each gene family has the same size in each genome by removing from a gene family those members that were the least homologous to the members of the counterpart family. Under the assumption of equal content, they proposed a graph-based heuristic algorithm for computing the assignment of orthologous genes that minimizes the reversal distance between two genomes. Finally, a more generalized version of the genome rearrangement problem was formulated by Sankoff (1999) where the restriction of the identical size of each gene family is removed. His idea is to delete all but one member of a gene family in each of the two genomes, so as to minimize the breakpoint or reversal distance between the two reduced genomes, called exemplars. Such an approach is clearly relevant in the phylogenetic context where the object is to reconstruct the ancestral gene order and the size of each gene family had been changed due to gene duplication/deletion (Sankoff and El-Mabrouk, 2000; El-Mabrouk, 2005).

In the current paper, we study efficient algorithms for the exemplar problem for genomes with gene families. Formally, the exemplar problem with the breakpoint distance is, given two genomes \mathcal{G} and \mathcal{H} , to find the minimum breakpoint distance between g and h overall choices of exemplars g and h of \mathcal{G} and \mathcal{H} respectively. Even though it is almost trivial to calculate the breakpoint distance between two genomes with single-gene families, the exemplar breakpoint distance problem is not only NP-hard (Bryant, 2000), but also unlikely to have polynomial-time constant-ratio approximation algorithm unless NP = P (C.T.Nguyen, Unpublished manuscript). By observing the monotonicity of the exemplar distances, Sankoff proposed a branch-and-bound method to tackle this problem. The idea is to work on one gene family separately, choosing the pair of exemplars that least increases the distance when inserted into the partial exemplar genomes already constructed.

Motivated by the above idea, we propose to calculate the exemplar breakpoint distance by partitioning the gene families into disjoint subsets such that two gene families in different subsets are 'independent'. Then, we find the pair of exemplars of gene families in each subset at a time, and finally merge all the exemplars together to obtain two optimal exemplars of the given genomes. Tests with both simulated and real datasets show that our divide-and-conquer approach is much more efficient than the branch-and-bound approach.

The paper is organized into six parts. Section 2 provides the needed definitions and notations. Section 3 introduces the concept

*To whom correspondence should be addressed.

of independence of gene families and proves a theorem that characterizes the relationship between the independence of gene families and the breakpoint distance. Section 4 presents our divide-and-conquer approach that is based on the characterization theorem established in Section 3, and Section 5 reports our test results. Section 6 concludes with a few remarks on future research.

2 THE EXEMPLAR DISTANCE PROBLEM

In the study of genome rearrangement with gene duplication, a genome \mathcal{G} is modeled as a string of signed (+ or -) symbols from an alphabet A , which represents the gene families. For a symbol $a \in A$, all its occurrences in genome \mathcal{G} constitute a gene family; and the number $k_{\mathcal{G}}(a)$ of its occurrences is the size of the gene family.

If $k_{\mathcal{G}}(a) = 1$ for all $a \in A$, then \mathcal{G} is a ‘permutation’ on the symbols in A . Consider two permutation genomes $G' = g_1, g_2, \dots, g_n$ and $H' = h_1, h_2, \dots, h_n$. We say g_i precedes g_{i+1} in G' and h_i precedes h_{i+1} in H' for $i \leq n-1$. If gene a precedes gene b in G' , but neither a precedes b nor $-b$ precedes $-a$ in H' , they determine a ‘breakpoint’ in G' . Additional breakpoints are posited if $g_1 \neq h_1$ and if $g_n \neq h_n$. The number of breakpoints, $d_B(G', H')$, in G' is called the breakpoint distance between G' and H' since $d_B(G', H') = d_B(H', G')$.

A reversal operation on a segment $ab \dots cd$ of a genome $\dots ab \dots cd \dots$ transforms the genome to $\dots -d -c \dots -b -a \dots$. For example, genome 3214 can be transformed into 4-3-1-2 in two reversals as follows: 3214 \rightarrow 3-4-1-2 \rightarrow 4-3-1-2. The reversal distance between G' and H' , denoted by $d_R(G', H')$, is the minimum number of reversals necessary to transform G' into H' or vice versa.

Given two genomes \mathcal{G} and \mathcal{H} (on alphabet A) satisfying $k_{\mathcal{G}}(a) > 0$ and $k_{\mathcal{H}}(a) > 0$ for all $a \in A$. An exemplar of a genome \mathcal{G} is obtained by deleting all but one occurrence of each gene family. Note that an exemplar string is a permutation. Let $E(\mathcal{G})$ denote the set of all exemplars of \mathcal{G} . Then, we define the exemplar breakpoint distance (EBD) between \mathcal{G} and \mathcal{H} as

$$d_{EB}(\mathcal{G}, \mathcal{H}) = \min_{G' \in E(\mathcal{G}), H' \in E(\mathcal{H})} d_B(G', H').$$

Similarly, the exemplar reversal distance (ERD) is

$$d_{ER}(\mathcal{G}, \mathcal{H}) = \min_{G' \in E(\mathcal{G}), H' \in E(\mathcal{H})} d_R(G', H').$$

In the rest of the paper, we shall focus on deriving an efficient algorithm for finding $d_{EB}(\mathcal{G}, \mathcal{H})$ given \mathcal{G} and \mathcal{H} over an alphabet.

3 INDEPENDENCE OF GENE FAMILIES

Let \mathcal{G} and \mathcal{H} be two genome strings with gene family set A . For gene family a , each occurrence $a_{\mathcal{G}}$ in \mathcal{G} and each occurrence $a_{\mathcal{H}}$ in \mathcal{H} form an occurrence pair. If $k_{\mathcal{G}}(a) = k_{\mathcal{H}}(a) = 1$, a is called a single-pair family; otherwise, it is called a multi-pair family. For example, in genomes

$$\begin{aligned} \mathcal{G} : & \quad lxax - bcy - dzyer, \\ \mathcal{H} : & \quad lb - x - a - e - zdz - c - z - yr, \end{aligned}$$

there are three multi-pair families: x , y and z . We shall number the occurrences of a gene family g from left to right in a genome and use $g_{\mathcal{G}}^i$ and $g_{\mathcal{H}}^i$ to denote the i -th copy of g in \mathcal{G} and \mathcal{H} respectively.

A permutation \mathcal{E} on a subset A' of gene families is called a partial exemplar of \mathcal{G} if it can be obtained by deleting all the occurrences of each gene family in $A \setminus A'$ and deleting all but one occurrence of each family in A' . For instance, in the above example, the six single-pair gene families a, b, c, d, e, l and r form a partial exemplar to $la - bc - der$ of \mathcal{G} and $lb - a - ed - cr$ of \mathcal{H} . For an occurrence $a_{\mathcal{G}}$ of a gene $a \in A \setminus A'$ in \mathcal{G} , we use $\mathcal{E}(a_{\mathcal{G}})$ to denote the resulting partial exemplar of \mathcal{G} after $a_{\mathcal{G}}$ is inserted into \mathcal{E} . Similarly, we define $\mathcal{E}(S)$ for a set S of occurrences of different gene families. We let $\mathcal{E}(a_{\mathcal{G}}) = \mathcal{E}$ if $a \in A'$. In the rest of this section, we assume that \mathcal{G}' and \mathcal{H}' are partial exemplars of \mathcal{G} and \mathcal{H} , respectively.

For two gene families a and b , we say their consecutive occurrences $a_{\mathcal{G}}$ and $b_{\mathcal{G}}$ in \mathcal{G}' are adjacent in \mathcal{G}' and \mathcal{H}' if and only if they do not form a breakpoint in \mathcal{G}' and \mathcal{H}' .

DEFINITION 1. Assume a and b are two gene families at least one of which does not occur in \mathcal{G}' and \mathcal{H}' .

- Two occurrence pairs $p = (a_{\mathcal{G}}, a_{\mathcal{H}})$ and $q = (b_{\mathcal{G}}, b_{\mathcal{H}})$ in \mathcal{G}' and \mathcal{H}' are said to be possibly adjacent with respect to \mathcal{G}' and \mathcal{H}' if and only if $a_{\mathcal{G}}$ and $b_{\mathcal{G}}$ are adjacent in $\mathcal{G}'(\{a_{\mathcal{G}}, b_{\mathcal{G}}\})$ and $a_{\mathcal{H}}$ and $b_{\mathcal{H}}$ are adjacent in $\mathcal{H}'(\{a_{\mathcal{H}}, b_{\mathcal{H}}\})$.
- Let c and d be an adjacent pair in \mathcal{G}' and \mathcal{H}' . An occurrence pair $p = (a_{\mathcal{G}}, a_{\mathcal{H}})$ of a kills (c, d) if and only if (c, d) is no longer adjacent in $\mathcal{G}'(a_{\mathcal{G}})$ and $\mathcal{H}'(a_{\mathcal{H}})$.
- Let r and s be two possibly adjacent pairs with respect to \mathcal{G}' and \mathcal{H}' . A pair $p = (a_{\mathcal{G}}, a_{\mathcal{H}})$ blocks r and s with respect to \mathcal{G}' and \mathcal{H}' if and only if r and s are not possibly adjacent with respect to exemplars $\mathcal{G}'(a_{\mathcal{G}})$ and $\mathcal{H}'(a_{\mathcal{H}})$.

Take the partial exemplars $\mathcal{G}' = l a - b c - d e r$ of \mathcal{G} and $\mathcal{H}' = l b - a - e d - c r$ of \mathcal{H} as an example. Pairs $(y_{\mathcal{G}}^2, y_{\mathcal{H}}^1)$ and $(z_{\mathcal{G}}^1, z_{\mathcal{H}}^2)$ are possibly adjacent with respect to \mathcal{G}' and \mathcal{H}' ; $(y_{\mathcal{G}}^1, y_{\mathcal{H}}^1)$ and $(z_{\mathcal{G}}^1, z_{\mathcal{H}}^2)$ kill adjacent pair $(c, -d)$ in \mathcal{G}' ; and $(y_{\mathcal{G}}^2, y_{\mathcal{H}}^1)$ blocks $(z_{\mathcal{G}}^1, z_{\mathcal{H}}^1)$ and $(-d, d)$ with respect to \mathcal{G}' and \mathcal{H}' .

DEFINITION 2. Assume a and b are two different gene families that do not occur in \mathcal{G}' and \mathcal{H}' .

- Two occurrence pairs of a and b are independent with respect to \mathcal{G}' and \mathcal{H}' if and only if (1) they are not possibly adjacent with respect to \mathcal{G}' and \mathcal{H}' , (2) they do not block each other from forming an adjacency with another pair with respect to \mathcal{G}' and \mathcal{H}' and (3) they do not kill the same adjacent pair in \mathcal{G}' and \mathcal{H}' .
- Two gene families a and b are independent with respect to \mathcal{G}' and \mathcal{H}' if and only if any occurrence pair of a and occurrence pair of b are independent with respect to \mathcal{G}' and \mathcal{H}' .

For a gene pair $p = (a_{\mathcal{G}}, a_{\mathcal{H}})$ of a gene family a that does not occur in \mathcal{G}' and \mathcal{H}' , we use $\mathcal{G}'(p)$ and $\mathcal{H}'(p)$ to denote $\mathcal{G}'(a_{\mathcal{G}})$ and $\mathcal{H}'(a_{\mathcal{H}})$, respectively. Similarly, we can define $\mathcal{G}'(P)$ and $\mathcal{H}'(P)$ for set P of occurrence pairs that are from different gene families.

LEMMA 1. Let P and Q be sets of occurrence pairs of different gene families that are not in \mathcal{G}' and \mathcal{H}' . If any $p \in P$ and $q \in Q$ are independent with respect to \mathcal{G}' and \mathcal{H}' , then

$$\Delta(P \cup Q) = \Delta(P) + \Delta(Q),$$

where $\Delta(P) = d_B(\mathcal{G}'(P), \mathcal{H}'(P)) - d_B(\mathcal{G}', \mathcal{H}')$, i.e. it denotes the change in the number of breakpoints after all the occurrence pairs in P are inserted.

PROOF. Let a and b be two consecutive signed symbols in \mathcal{G}' . Let $\delta_{ab}(X)$ denote the change in the number of breakpoints between a and b after all the occurrence pairs in X are inserted in \mathcal{G}' and \mathcal{H}' for $X = P, Q, P \cup Q$. Obviously, for each X , $\Delta(X) = \sum_{(a,b)} \delta_{ab}(X)$. Thus we only need to prove the following equation:

$$\delta_{ab}(P \cup Q) = \delta_{ab}(P) + \delta_{ab}(Q). \quad (1)$$

for any consecutive symbols a and b in \mathcal{G}' . To this end, we consider the following two cases.

Case 1. The pair (a, b) is adjacent in \mathcal{G}' . After inserting all pairs in $P \cup Q$, (a, b) is still adjacent in both \mathcal{G}' and \mathcal{H}' or separated by some occurrence pairs in $\mathcal{G}'(P \cup Q)$ or $\mathcal{H}'(P \cup Q)$. In the former case, we have $\delta_{ab}(P \cup Q) = \delta_{ab}(P) = \delta_{ab}(Q) = 0$ and hence Equation (1) holds. In the latter case, we have the following facts.

CLAIM 1. a and b can only be separated by either some pairs in P or some pairs in Q , but not both in $\mathcal{G}'(P \cup Q)$ and $\mathcal{H}'(P \cup Q)$.

PROOF. Assume that a and b are separated by some pairs in P and some pairs in Q . Suppose $p = (c_G, c_{\mathcal{H}}) \in P$ and $q = (d_G, d_{\mathcal{H}}) \in Q$ such that c_G and d_G are inserted between a and b in $\mathcal{G}'(P \cup Q)$ in the following configuration: $a \cdots c_G \cdots d_G \cdots b$. Then, both p and q kill the same adjacency (a, b) in \mathcal{G}' and \mathcal{H}' , contradicting the independence hypothesis.

Suppose $p = (c_G, c_{\mathcal{H}}) \in P$ and $q = (d_G, d_{\mathcal{H}}) \in Q$ such that only c_G is inserted between a and b in $\mathcal{G}'(P \cup Q)$ and $d_{\mathcal{H}}$ is inserted between a and b (or $-b$ and $-a$) in $\mathcal{H}'(P \cup Q)$. Again, both p and q kill the same adjacency (a, b) in \mathcal{G}' , contradicting the independence hypothesis. This finishes the proof of Claim 1.

We may assume that a and b are separated by k pairs $(c_G^i, c_{\mathcal{H}}^i)$ ($1 \leq i \leq k$) in P after $P \cup Q$ are inserted: $a c_G^1 c_{\mathcal{H}}^2 \cdots c_G^k b$ is a substring of $\mathcal{G}'(P \cup Q)$. For simplicity, we define $c_G^0 = a$ and $c_{\mathcal{H}}^{k+1} = b$. Then, we have the following.

CLAIM 2. c_G^i and $c_{\mathcal{H}}^{i+1}$ are adjacent in $\mathcal{G}'(P \cup Q)$ and $\mathcal{H}'(P \cup Q)$ if and only if they are adjacent in $\mathcal{G}'(P)$ and $\mathcal{H}'(P)$.

PROOF. Assume that c_G^i and $c_{\mathcal{H}}^{i+1}$ are adjacent in $\mathcal{G}'(P)$ and $\mathcal{H}'(P)$. If they are not adjacent in $\mathcal{G}'(P \cup Q)$ and $\mathcal{H}'(P \cup Q)$, there must be an occurrence pair $(e_G, e_{\mathcal{H}})$ in Q such that $e_{\mathcal{H}}$ separates $c_{\mathcal{H}}^i$ and $c_{\mathcal{H}}^{i+1}$. In other words, $(e_G, e_{\mathcal{H}})$ blocks $(c_G^i, c_{\mathcal{H}}^i)$ and $(c_G^{i+1}, c_{\mathcal{H}}^{i+1})$. Since at least one of $(c_G^i, c_{\mathcal{H}}^i)$ and $(c_G^{i+1}, c_{\mathcal{H}}^{i+1})$ is in P , this contradicts the independence hypothesis. Thus c_G^i and $c_{\mathcal{H}}^{i+1}$ are adjacent in $\mathcal{G}'(P \cup Q)$ and $\mathcal{H}'(P \cup Q)$.

On the other hand, it is obvious that if c_G^i and $c_{\mathcal{H}}^{i+1}$ are adjacent in $\mathcal{G}'(P \cup Q)$ and $\mathcal{H}'(P \cup Q)$, they are adjacent in $\mathcal{G}'(P)$ and $\mathcal{H}'(P)$. This finishes the proof of Claim 2.

By Claim 2, $\delta_{ab}(Q) = 0$ and $\delta_{ab}(P \cup Q) = \delta_{ab}(P)$. Hence, Equation (1) holds.

Case 2. The pair (a, b) is a breakpoint in \mathcal{G}' . Denote by t_X the number of gene occurrences between a and b in $\mathcal{G}'(X)$, and by u_X the number of adjacent pairs between a and b in $\mathcal{G}'(X)$. Obviously, $t_{PQ} = t_P + t_Q$. Moreover, $u_{PQ} = u_P + u_Q$ since a pair in P can neither block a pair in Q to form adjacency with other pairs nor form

an adjacency with a pair in Q and vice versa. Therefore, we have that $\delta_{ab}(X) = (t_X + 1 - u_X) - 1 = t_X - u_X$, and thus

$$\begin{aligned} \delta_{ab}(PQ) &= t_{PQ} - u_{PQ} \\ &= (t_P - u_P) + (t_Q - u_Q) \\ &= \delta_{ab}(P) + \delta_{ab}(Q). \end{aligned}$$

This concludes the proof.

Using the above lemma, we obtain the following.

THEOREM 1. Let A' and A'' be sets of gene families that do not occur in \mathcal{G}' and \mathcal{H}' such that any $a' \in A'$ and $a'' \in A''$ are independent with respect to \mathcal{G}' and \mathcal{H}' . If $\{p_i \mid i \leq |A'|\}$ and $\{q_i \mid i \leq |A''|\}$ are occurrence pairs of gene families in A' and A'' such that $d_B(\mathcal{G}'(\{p_i\}), \mathcal{H}'(\{p_i\}))$ and $d_B(\mathcal{G}'(\{q_i\}), \mathcal{H}'(\{q_i\}))$ reach the minimum values over all the choices of occurrence pairs, then $d_B(\mathcal{G}'(\{p_i\} \cup \{q_i\}), \mathcal{H}'(\{p_i\} \cup \{q_i\}))$ has the minimum value over all the occurrence pairs of A' and A'' .

The occurrence pairs p_i are called the *optimal representative pairs* of A' with respect to \mathcal{G}' and \mathcal{H}' .

4 ALGORITHMS

Divide-and-Conquer approach. Following the discussion in the previous section, we propose to calculate $d_{EB}(\mathcal{G}, \mathcal{H})$ in two steps:

- (1) Form partial exemplars \mathcal{G}_s and \mathcal{H}_s with all the single-pair gene families, and then divide multi-pair gene families into disjoint subsets that are independent with respect to \mathcal{G}_s and \mathcal{H}_s .
- (2) For each gene family subset found in step (1), find its optimal representative pairs. Obtain the optimal exemplars of \mathcal{G} and \mathcal{H} by taking the union of all the optimal representative pairs.

To partition the gene families into disjoint subsets that are independent, we define a graph $G(\mathcal{G}, \mathcal{H})$, called the dependence graph of the gene families. Each vertex in $G(\mathcal{G}, \mathcal{H})$ represents a multi-pair family. Two vertices are joined by an edge if and only if they are not independent with respect to \mathcal{G}_s and \mathcal{H}_s . By Definition 2, one can efficiently determine whether two families are 'independent' or not. Noting that gene families in different connected components are independent, we just partition the gene families according to the connected components of $G(\mathcal{G}, \mathcal{H})$ in Step (1).

In Step (2), we use Sankoff's program (1999) to find optimal exemplar occurrence pairs for each gene family subset and call it *SimplBB*.

Approximating algorithm. Dividing multigene families into disjoint subsets does not always solve the problem. For large genomes, these subsets are still too large and finding their optimal representative pairs is forbidding.

To speed up the procedure, we delete a small subset X of vertices to split each connected component into smaller ones with an appropriate size in the graph. By these deletions, we trade accuracy for efficient computation. Hence, we only obtain tight lower and upper bounds of $d_{EB}(\mathcal{G}, \mathcal{H})$. Assume that $\mathcal{G}_1, \mathcal{H}_1$ are the partial genomes obtained by removing the gene families corresponding to vertices in X . Since the breakpoint distance is monotone (Sankoff, 1999), $d_{EB}(\mathcal{G}_1, \mathcal{H}_1)$ is a lower bound of $d_{EB}(\mathcal{G}, \mathcal{H})$. On the other hand, we also obtain an upper bound of $d_{EB}(\mathcal{G}, \mathcal{H})$ by finding the breakpoint distance between

$\mathcal{G}'_1(X)$ and $\mathcal{H}'_1(X)$, where \mathcal{G}'_1 and \mathcal{H}'_1 are the optimal exemplars of \mathcal{G}_1 and \mathcal{H}_1 , respectively. When X is small, these bounds are tight.

The deletion can be done solely on $G(\mathcal{G}, \mathcal{H})$ since removing a vertex (or a family) does not affect the others. However, the problem of deleting the smallest subset of vertices to reduce all connected components to a desired size is NP-hard since it contains the independent set problem as a special case. Hence, we simply delete the vertices with the largest degree until the size of each component is small enough for SimplBB to run fast. On the other hand, the size should not be too small to make the gap between the upper and lower bounds large. Because of this, we determine the value of the size parameter according to the input dataset.

5 EXPERIMENTAL TESTS

In Sankoff (1999), the computational cost of SimplBB is measured in the number of calls to subroutine BD which calculates the breakpoint distance between two partial exemplar strings. In our program, most of the computation is done by calling SimplBB on each independent subset of gene families. Thus, we also measure the computational cost of our program FastEBD in the number of calls to BD for comparing it with SimplBB. We implemented both SimplBB and FastEBD in C++ and ran them on a PC with 1.6 GHz Pentium processor and 512 MB RAM.

5.1 Results on simulated genomes

We use the simulation method proposed by Sankoff (1999) to generate genome pairs for our validation test. Each genome pair is characterized by two parameters: the number of (both single-pair and multi-pair) gene families and their configuration, which indicates the number of copies of each multi-pair gene family in each of the two genomes. In the configurations given in Table 1, each row corresponds to a genome, each column corresponds to a multi-pair gene family and each entry gives the number of genes of the corresponding gene family in the corresponding genome. In each genome pair generated according to the configuration 'threes', a genome contains roughly about three copies of each multi-pair family. However, in each genome pair generated according to the second configuration, the number of gene copies in each multi-pair family varies. Some multi-pair families contain as many as 10 gene copies in each genome while others contain only one or two copies in each genome. Hence, the second configuration is named unbalanced.

We run both SimplBB and FastEBD to find the exemplar breakpoint distance between two genomes with the same configuration described in Table 1. In Figure 1, we plot the numbers of calls to the subroutine BD made by SimplBB and FastEBD in log scale against the length of exemplars. The graph indicates that FastEBD is much more efficient than SimplBB.

We also note that when the number of single-pair gene families increases, the running cost of SimplBB increases while the cost of FastEBD increases for a narrow range of length and then decreases. The reason for FastEBD running faster when there is a large number of gene families is that when more single-pair gene families are included, the multi-pair gene families become less dependent on each other and hence the large set of gene families could be partitioned into small unrelated subsets.

Another property of our program FastEBD is that it allows a trade-off between accuracy and computational efficiency. This is vital when working with real genome datasets that contain hundreds or

Table 1. The two configurations from Sankoff (1999) used to compare the cost of SimplBB and FastEBD

Label	Size of families
Threes	5 2 3 3 3 3 3 3 3 3
Unbalanced	10 5 5 2 2 2 2 2 2 1 1 1 1
	10 5 5 2 2 2 2 1 1 1 1 2 2 2 2

For 'threes', 5 2 3 3 3 3 3 3 3 3 denotes a genome \mathcal{G} , 2 5 3 3 3 3 3 3 3 3 denotes a genome \mathcal{H} , with 10 multi-pair gene families each; the first family has five copies in \mathcal{G} and two copies in \mathcal{H} , etc.

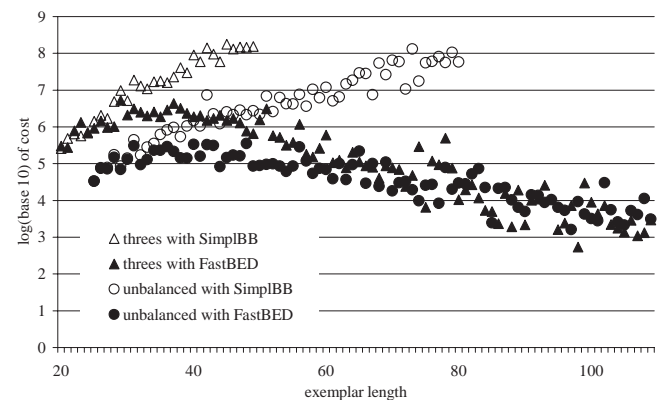


Fig. 1. Calling cost of SimplBB and FastEBD on genomes with each of the two configurations given in Table 1. Each point represents the average number of calls of 100 tests on different genome pairs generated with the same parameters.

thousands of genes. Thus we conducted experiments to examine the accuracy of the approximating version of our program. By setting the threshold T on the number of calls to BD, our algorithm outputs the lower and upper bounds on the exemplar breakpoint distance between the input genomes. If T is small, we have to remove more gene families so that the remaining gene families can be partitioned into 'independent' subsets on which SimplBB can finish by calling BD at most T times. Symmetrically, if T is large, we just need to remove less gene families and hence obtain better lower and upper bounds on the exemplar breakpoint distance between the input genomes. We measured accuracy using the ratio of the difference of the lower and upper bounds to the upper bound of the exemplar breakpoint distance output from the program.

To generate genomic datasets that are close to biological data, we adopt a common approach in benchmarking algorithms for calculating genomic distance. Starting with two identical genomes, we let them undergo a series of mutations such as duplication, transposition and reversal with certain probabilities. Since duplication, transposition and reversal of long segments happen less frequently than those of short segments (Pinter and Skiena, 2002), we use a Poisson distribution $P_\lambda(k)$ to calculate the probabilities of mutations on a segment of length k with the mean λ as a parameter.

We generated 30 genome pairs in which each genome contains 3000 genes and undergoes 500 cycles of mutations. The mean of probabilities for each type of mutation is 0.1. The resulting genome

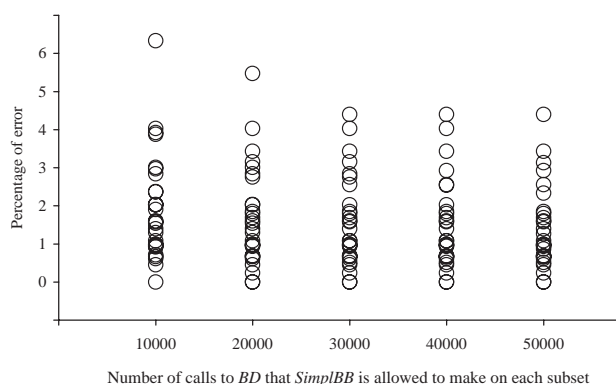


Fig. 2. Error of the results returned by FastEBD against number of calls SimplBB is allowed to make.

pairs contain about 3–8 copies in each gene family, giving rise to about 10^{250} – 10^{350} exemplar pairs. We ran FastEBD on each pair of genomes with T taking values 10^4 , 2×10^4 , 3×10^4 , 4×10^4 and 5×10^4 . The accuracy ratios in these tests are summarized in Figure 2. From this graph, it is clear that accuracy is improved when T is bigger. It also shows that even with $T = 10^4$, the error is $<5\%$ in most of our experiments.

5.2 Bacterial genomes

We ran the algorithms on five bacteria genome pairs studied in Earnst-De Young *et al.* (2004): *Buchnera aphidicola* and *W. brevipalpis* (*Baphi-Wigg*), *Pasteurella multocida* and *Hamophilus influenzae* (*Pmult-Hinfl*), *Escherichia coli* and *Salmonella typhimurium* (*Ecoli-Styphi*), *Xanthomonas axonopodis* and *Xanthomonas campestris* (*Xaxo-Xcamp*) and *Yersinia pestis-KIM* and *Yersinia pestis-CO92* (*Ypes*).

Each genome in the five genome pairs above contains gene families that are not in the other. There are two ways to deal with these genes for our purpose. The easy way is to delete all such genes and consider the reduced genomes. Alternatively, we first put a ‘pivot’ gene x at the right end of both genomes. Then, we construct two genomes from \mathcal{G} and \mathcal{H} by doing the following: (1) for each block B of consecutive occurrences of gene families that are in \mathcal{G} but not in \mathcal{H} , we introduce a new gene family g_B , replace B with a copy of g_B and append a copy at the right end of \mathcal{H} . (2) We deal with the gene families that appear in \mathcal{H} , but not in \mathcal{G} , in the same way. The purpose of this preprocessing method is to take gene deletions into accounts. But it may introduce redundant hypothetical gene deletion events. Our computation results show that we might underestimate the exemplar breakpoint distance between two given genomes by using the first method and overestimate it by using the second method. By using these two methods to deal with gene losses, we obtain two genomic datasets from each of the five bacteria genome pairs, denoted by *Baphi-Wigg1*, *Baphi-Wigg2*, *Pmult-Hinfl1*, *Pmult-Hinfl2*, *Ecoli-Styphi1*, *Ecoli-Styphi2*, *Xaxo-Xcamp1*, *Xaxo-Xcamp2*, *Ypes1* and *Ypes2* respectively.

We ran both SimplBB and FastEBD on these datasets. SimplBB could finish only on *Baphi-Wigg1* and made about 10^5 calls to BD. FastEBD with $T = 10^4$ finished on all the genome pairs, returning either an exact EBD or lower and upper bounds. (See Section 5.1 for threshold T .) In the latter case, the bounds differ by only 1. Furthermore, we are able to obtain the exact EBD for all pairs by

Table 2. Experimental results on 10 pairs of bacteria genomes

Genome pair	N_g	N_p	d_{EB}	Cost of FastEBD
<i>Baphi-Wigg1</i>	386	$10^{13.5}$	177	374
<i>Baphi-Wigg2</i>	605	$10^{13.5}$	427	390
<i>Pmult-Hinfl1</i>	1268	$10^{84.61}$	503	3482
<i>Pmult-Hinfl2</i>	1487	$10^{84.61}$	733	3492
<i>Ecoli-Styphi1</i>	2646	$10^{308.39}$	153	20787
<i>Ecoli-Styphi2</i>	2921	$10^{308.39}$	494	20766
<i>Xaxo-Xcamp1</i>	3020	$10^{228.96}$	118–119	60032
<i>Xaxo-Xcamp2</i>	3126	$10^{228.96}$	271–272	59832
<i>Ypes1</i>	2829	$10^{244.45}$	55–56	144335
<i>Ypes2</i>	2925	$10^{244.45}$	228–229	124202

The total numbers of calls made by FastEBD to BD shown in the fifth column are significantly smaller when compared with the numbers of exemplar pairs listed in the third column. Here, N_g and N_p denote the number of genes and exemplar pairs respectively.

increasing T . For example, with $T = 3 \times 10^6$, FastEBD output 119 given *Xaxo-Xcamp1* as input. The results are summarized in Table 2.

6 CONCLUSION

In this paper, we have proposed an efficient algorithm for calculating the exemplar breakpoint distance between two genomes with gene families. It is not only fast, but also outputs accurate breakpoint distances. Our idea can be incorporated into the existing methods to derive an efficient algorithm for computing the exemplar reversal distance because of the close relation of the reversal distance and the breakpoint distance. Future research topics also include how to incorporate our idea into the existing methods such as the branch-and-bound approach to derive efficient algorithms for the exemplar problem with other distance measures such as translocation distance (Hannenhalli and Pevzner, 1995) or syntenic distance (Feretti *et al.*, 1996).

As for applications, investigating whether our approach can be extended into an efficient method for genomic comparison or for reconstructing phylogeny from gene-content data is also interesting.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for the careful reading of the manuscript and the useful suggestions for revising it. They would also like to thank D. Sankoff for the helpful discussions on the exemplar problem, E. Lerat for sending us her genomic datasets, and J. Tang for the comments on the first draft of this manuscript. L.Z. was partially supported by BMRC Research Grant BMRC01/1/21/19/140.

REFERENCES

- Bryant, D. (2000) The complexity of calculating exemplar distances. In Sankoff, D. and Nadeau, J.H. (eds), *Comparative Genomics*. Kluwer Academic Publishers, Dordrecht, pp. 207–212.
- Caprara, A. (1997) Sorting by reversals is difficult. In *Proceedings of the First International Conference on Computational Molecular Biology (RECOM97)*. ACM Press, pp. 75–83.
- Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S. and Jiang, T. (2005) Computing the assignment of orthologous genes via genome rearrangement. In *Proceedings of the Third Asia-Pacific Bioinformatics Conference*, Imperial College Press, pp. 363–378.

- Earnst-De Young,J., Lerat,E. and Moret,B. (2004) Reversing gene erosion—reconstructing ancestral bacterial genomes from gene content and order data. In *Proceedings of the 4th Annual Workshop on Algorithms in Bioinformatics (WABI'04)*. Springer, Berlin, pp. 1–13.
- El-Mabrouk,N. (2005) Genome rearrangements with gene families. In Gascuel,O. (ed.), *Mathematics of Evolution and Phylogeny*. Oxford University Press (In press).
- Feretti,V., Nadeau,J.H. and Sankoff,D. (1996) Original syntenies. In *Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, Lecture Notes in Computer Science, Vol. 1075, Springer, Berlin, pp. 78–87.
- Hannenhalli,S. and Pevzner,P. (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of IEEE 36th Annual Symposium on Foundations of Computer Science*, The IEEE Computer Society, pp. 581–592.
- Hannenhalli,S. and Pevzner,P. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the Association for Computing Machinery*, **46**, 1–27.
- International Mouse Genome Consortium (2002) Initial sequencing and comparative analysis of the mouse genome. *Nature*, **420**, 520–562.
- Kaplan,H. et al. (2000) Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.*, **29**, 880–892.
- Kececioğlu,J. and Sankoff,D. (1994) Efficient bounds for oriented chromosome-inversion distance. In *Proceedings of the Fifth Symposium Combinatorial Pattern Matching*, Lecture Notes in Computer Science, Vol. 807. Springer, Berlin, pp. 307–325.
- Moret,B. et al. (2005) Reconstructing phylogenies from gene-content and gene-order data. In Gascuel,O. (ed.), *Mathematics Of Evolution and Phylogeny*. Oxford University Press.
- Nadeau,J.H. and Taylor,B.A. (1984) Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc. Natl. Acad. Sci. USA*, **81**, 814–818.
- Pinter,R.Y. and Skiena,S. (2002) Genomic sorting with length-weighted reversals. In *Proceedings of the Thirteenth International Conference on Genomics Informatics*, Universal Academy Press, Tokyo, Japan, pp. 103–111.
- Sankoff,D. (1989) Mechanisms of genome evolution: models and inference. *Bull. Int. Stat. Institut.*, **47**, 461–475.
- Sankoff,D. and Blanchette,M. (1998) Multiple genome rearrangement and breakpoint phylogeny. *J. Comput. Biol.*, **5**, 555–570.
- Sankoff,D. and El-Mabrouk,N. (2000) Duplication, rearrangement and reconciliation. In Sankoff,D. and Nadeau,J.H. (eds), *Comparative Genomics*. Kluwer Academic Press, Dordrecht, pp. 537–550.
- Sankoff,D. (1999) Genome rearrangement with gene families. *Bioinformatics*, **15**, 909–917.
- Tang,J. and Moret,B. (2003) Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. In *Proc. 8th Workshop on Algorithms and Data Structures (WADS'03)*, Lecture Notes in Computer Science, vol. 2748, Springer Verlag, Berlin, pp. 37–46.
- Watterson,G.A. et al. (1982) The chromosome inversion problem. *J. Theor. Biol.*, **99**, 1–7.