

# Scaffold Filling under the Breakpoint Distance

Haitao Jiang<sup>1,2</sup>, Chunfang Zheng<sup>3</sup>, David Sankoff<sup>4</sup>, and Binhai Zhu<sup>1</sup>

<sup>1</sup> Department of Computer Science, Montana State University,  
Bozeman, MT 59717-3880, USA

htjiang, bhz@cs.montana.edu

<sup>2</sup> School of Computer Science and Technology, Shandong University, Jinan, China

<sup>3</sup> Département d'informatique et de recherche opérationnelle,

Université de Montréal, Montréal, Canada H3C 3J7

chunfang313@gmail.com

<sup>4</sup> Department of Mathematics and Statistics, University of Ottawa, Ottawa, Canada K1N 6N5

sankoff@uottawa.ca

**Abstract.** Motivated by the trend of genome sequencing without completing the sequence of the whole genomes, Muñoz et al. recently studied the problem of filling an incomplete multichromosomal genome (or *scaffold*)  $I$  with respect to a complete target genome  $G$  such that the resulting genomic distance between  $I'$  and  $G$  is minimized, where  $I'$  is the corresponding filled scaffold. We call this problem the *one-sided* scaffold filling problem. In this paper, we follow Muñoz et al. to investigate the scaffold filling problem under the breakpoint distance for the simplest unichromosomal genomes. When the input genome contains no gene repetition (i.e., is a fragment of a permutation), we show that the *two-sided* scaffold filling problem is polynomially solvable. However, when the input genome contains some genes which appear twice, even the one-sided scaffold filling problem becomes NP-complete. Finally, using the ideas for solving the two-sided scaffold filling problem under the breakpoint distance we show that the two-sided scaffold filling problem under the genomic/rearrangement distance is also polynomially solvable.

## 1 Introduction

Due to the advancement of genome sequencing technology, it is possible to sequence more organisms for genomic analysis. (Throughout this paper, a multichromosomal genome is represented as sequences of genes, while a unichromosomal genome is just represented as a sequence of genes.) Interesting and somehow contradicting, the cost of finishing genome sequencing has not decreased at the same rate compared with the cost of random sequencing [1]. This means that many genomes released are not completely finished. It would be unsuitable to use these incomplete genomes (scaffolds) for genomic analysis, simply due to the errors they could introduce.

Therefore, a natural problem is to fill the missing genes into scaffolds, with combinatorial algorithms. As one must find a biologically meaningful way of filling scaffolds, it makes sense to make use of some complete genomes (from some close species). Muñoz et al. [10] recently carried out this idea on filling an incomplete multichromosomal scaffold  $I$  to have  $I'$ , such that the genomic distance [13] between  $I'$  and a given (complete)

genome  $G$  is minimized. (The genomic distance is also called *rearrangement* distance, which is the minimum number of allowed rearrangement operations transforming one genome into the other.) We call this the *one-sided* scaffold filling problem. Basically, the one-sided scaffold filling can be solved in polynomial time; in fact, linear time when the breakpoint graph on  $I$  and  $G$  is constructed [10].

In [10], much effort has been put on several practical issues. For instance, what if the missing genes can only be inserted in certain locations? What if some missing genes in  $I$  are not really missing (i.e., they should not appear in  $G$ )? However, the corresponding two-sided problem is not tackled in [10].

In this paper, we follow Muñoz et al. to investigate the scaffold filling problem under the breakpoint distance for the simplest unichromosomal genomes. When the input genome contains no gene repetition (i.e., is a fragment of a permutation), we show that the *two-sided* scaffold filling problem is polynomially solvable. However, when the input genome contains some genes which appear twice, even the one-sided scaffold filling problem becomes NP-complete. The latter problem has a close connection with the Minimum Common String Partition (MCSP) problem [2,3,4,7,8,9].

This paper is organized as follows. In Section 2, we give necessary definitions. In Section 3, we present the polynomial time algorithm for the scaffold filling problem. In Section 4, we show the NP-completeness proof for the one-sided scaffold filling problem when gene duplications are allowed. In Section 5, we show how to adapt our ideas in Section 3 to solve the two-sided scaffold filling problem under the rearrangement distance (i.e., for multichromosomal genomes) in polynomial time. In Section 6, we conclude the paper.

## 2 Preliminaries

We first present some necessary definitions.

Given alphabet  $\Sigma$ , a string  $S$  is called a *permutation* if each element in  $\Sigma$  appears exactly once in  $S$ . We also use  $c(S) = \Sigma$  to denote the set of elements in permutation  $S$ . An (unsigned) unichromosomal genome is just a permutation over  $\Sigma$ .

A *scaffold* is an incomplete permutation, i.e., with some missing elements. We use  $+$  to denote permutation scaffold filling, e.g., for a permutation  $A$  and an element set  $X$  such that  $c(A) \cap X = \emptyset$ , if  $A^*$  is a resulting permutation after filling all the elements in  $X$  into  $A$ , then  $A^* = A + X$ . Similarly, we use  $-$  to denote element elimination from the permutation. Given two permutations  $A$  and  $B$ , if  $c(A) = c(B)$ , then  $A$  and  $B$  are *related*. Given two related permutations  $A$  and  $B$ , two consecutive elements  $a_i$  and  $a_{i+1}$  in  $A$  form an *adjacency* if they are also consecutive in  $B$  (i.e., as  $a_i a_{i+1}$  or  $a_{i+1} a_i$ ), otherwise they form a *breakpoint*. The number of breakpoints in  $A$ , which is equal to that of  $B$ , is the breakpoint distance between  $A$  and  $B$ , denoted as  $bd(A, B)$ . Note that our breakpoint definition and the corresponding results all work when the letters (or genes) are possibly *signed*.

The (*two-sided*) scaffold filling problem is defined as follows.

### **Scaffold Filling under the Minimum Permutation Breakpoint Distance (SF-PBD)**

**Input:** two incomplete permutations  $A$  and  $B$  and two sets of elements  $X$  and  $Y$ , where  $X = c(B) - c(A)$  and  $Y = c(A) - c(B)$ .

**Question:** minimize  $bd(A + X, B + Y)$ .

In the above definition, when either  $X$  or  $Y$  is empty, we have the *one-sided* scaffold filling problem. Note that if  $A$  and  $B$  were related (i.e.,  $c(A) = c(B)$ ), then we would have  $X = Y = \emptyset$ .

In practice, sometimes we need to deal with genomes with orthologous (duplicated) genes. Let  $C(S)$  be a multiset to denote all the appearances of all the elements. We still use  $+$  to denote string scaffold filling.

### Scaffold Filling under the Minimum String Breakpoint Distance (SF-SBD)

**Input:** two strings  $A$  and  $B$  and two multisets of elements  $X$  and  $Y$  where  $X = c(B) - c(A)$  and  $Y = c(A) - c(B)$ .

**Question:** minimize  $bd(A + X, B + Y)$ .

For this problem, when some of the genes can appear more than once, we will show that even the one-sided scaffold filling problem is NP-complete. This problem has a close connection with the Minimum Common String Partition problem. We present the details of our results under the breakpoint distance in the next two sections.

## 3 Polynomial Algorithm for SF-PBD

In this section we present a polynomial algorithm for scaffold filling under the permutation breakpoint distance.

**Lemma 1.** *Given two incomplete permutations  $A$  and  $B$ , let  $X = c(B) - c(A)$  and  $Y = c(A) - c(B)$  be the sets of elements to be filled into  $A$  and  $B$  respectively. If there is an adjacency  $a_i a_{i+1}$  in the two related permutations  $A - Y$  and  $B - X$ , then there exists a scaffold filling such that every two consecutive elements between  $a_i$  and  $a_{i+1}$  (in  $A + X$  and  $B + Y$ ) form an adjacency.*

*Proof.* To obtain  $A + X$  and  $B + Y$ , we just need to insert the elements in  $X$  into  $A$  and insert the elements in  $Y$  into  $B$  respectively. As  $a_i a_{i+1}$  is an adjacency in  $A - Y$  and  $B - X$ , we have the full freedom to insert the respective elements in  $X + Y$  in between  $a_i$  and  $a_{i+1}$  such that they all form adjacencies in  $A + X$  and  $B + Y$ .  $\square$

**Lemma 2.** *Given two permutations  $A$  and  $B$ , let  $X = c(B) - c(A)$  and  $Y = c(A) - c(B)$  be the sets of elements to be filled into  $A$  and  $B$  respectively. If there is a breakpoint  $b_i b_{i+1}$  in  $A - Y$ , then in any scaffold filling, there is at least one breakpoint between  $b_i$  and  $b_{i+1}$  in  $A + X$ .*

*Proof.* As shown in the previous lemma, we insert the respective elements in  $X + Y$  into  $A - Y$  to obtain  $A + X$ . When we insert some respective elements in between  $b_i$  and  $b_{i+1}$ , if these inserted elements contain a breakpoint then the lemma is proven. If the inserted elements contain no breakpoint, then they must introduce at least a breakpoint right before  $b_{i+1}$  or right after  $b_i$ .  $\square$

**Lemma 3.** *Given two permutations  $A$  and  $B$ , let  $X = c(B) - c(A)$  and  $Y = c(A) - c(B)$  be the sets of elements to be filled into  $A$  and  $B$  respectively. If there is a breakpoint  $b_i b_{i+1}$  in  $A - Y$ , then there exists a scaffold filling such that there is only one breakpoint between  $b_i$  and  $b_{i+1}$  in  $A + X$ .*

*Proof.* As shown in the previous lemma, we just insert the respective elements in  $X + Y$  in between  $b_i$  and  $b_{i+1}$ . If the breakpoint in  $B - X$  has one letter in agreement with  $b_i$  or  $b_{i+1}$ , say  $b_i b'$ , then we just make sure that the inserted elements contain no breakpoint, the only breakpoint hence introduced is right before  $b_{i+1}$  (or right after  $b_i$  in  $A + X$ , if the corresponding breakpoint in  $B - X$  has the form of  $b' b_{i+1}$ ).  $\square$

**Theorem 1.** *Given two permutations  $A$  and  $B$ , let  $X = c(B) - c(A)$  and  $Y = c(A) - c(B)$  be the sets of elements to be filled into  $A$  and  $B$  respectively. Then  $bd(A - Y, B - X) = bd(A + X, B + Y)$  and the corresponding  $A + X$  and  $B + Y$  can be computed in  $O(n^2)$  time.*

*Proof.* Following the previous lemmas, it is easy to see that  $bd(A - Y, B - X) = bd(A + X, B + Y)$ . The corresponding algorithm is as follows. First, we identify  $A - Y$  and  $B - X$ , and compute the breakpoint distance  $bd(A - Y, B - X)$ . Second, we insert the respective elements from  $Y$  into  $A - Y$  and  $B - X$  such that the number of breakpoints (in  $A$  and  $B - X + Y$ ) does not change. Then, we insert the respective elements from  $X$  into  $A$  and  $B - X + Y$ , still maintaining the number of breakpoints. Eventually, we obtain  $A + X$  and  $B + Y$  accordingly. The quadratic running time is dominated by finding the correspondence between the identical characters in  $A$  and  $B$ .  $\square$

An example of the above algorithm is as follows.

$$\begin{aligned} A &= acefh, B = adbhge \\ X &= \{d, b, g\}, Y = \{c, f\} \\ A - Y &= aeh, B - X = ahe, bd(A - Y, B - X) = 1 \\ A + X &= acdbegfh, B + Y = acdbh fge, bd(A + X, B + Y) = 1. \end{aligned}$$

We comment that our algorithm also works when in  $A$  and  $B$  there are predefined adjacencies one could not break, as long as these adjacencies have no conflict. For example, we have  $A = \dots \boxed{ac} \dots \boxed{gh} \dots$  and  $B = \dots \boxed{ea} \dots \boxed{fg} \dots$ , with predefined adjacencies in boxes. When we fill  $e, f$  into  $A$  and  $c, g$  into  $B$ , the algorithm still works as the predefined adjacencies are not in conflict. However, if  $A = \dots \boxed{bac} \dots \boxed{ghf} \dots$  and  $B = \dots \boxed{bae} \dots \boxed{gfh} \dots$ , then our result does not hold anymore. In [10], this is related to insert missing genes only in between contigs (i.e., not anywhere), which is a problem needing further study.

## 4 Hardness of SF-SBD

In this section, we prove that SF-SBD is NP-complete; in fact, even the one-sided scaffold filling problem is NP-complete, when each gene is allowed to appear at most twice.

We make a reduction from the maximum independent set problem on cubic graphs (3-MIS). The main idea of this proof is from the NP-hardness proof by Goldstein *et al.* for 2-MCSP (Minimum Common String Partition with each letter appears at most

twice in an input string) [7]. We try to add more separators and adapt the proof for our purposes. For completeness, we describe the MCSP problem as follows.

Define that two strings  $A$  and  $B$  are related if each element appears the same number of times in  $A$  and  $B$ . A partition of a string  $A$  is a sequence  $P = (P_1, P_2, \dots, P_m)$  of strings, called the blocks, whose concatenation is equal to  $A$ . Given a partition  $P$  of a string  $A$  and a partition  $Q$  of a string  $B$ , we say that the pair  $(P, Q)$  is a common partition of  $A$  and  $B$  if  $Q$  is a permutation of  $P$ . A minimum common string partition is a common partition of two strings  $A$  and  $B$  with the minimum number of blocks. Two related strings always have a minimum common string partition. Note that the breakpoint distance of two related strings is equal to the size of minimum common string partition minus one. An example is as follows.  $A = 01101110111$ ,  $B = 01110011111$ , the three optimal blocks are 011, 01110, 111.

Given a cubic graph  $G = (V, E)$  as an input for 3-MIS, for each vertex  $v \in V$ , we create two substrings  $A_v$  and  $B_v$ .

$A_v =$

$d_v x_{v_1} y_{v_2} a_v b_v x_{v_2} y_{v_3} c_v d_v e_v x_{v_3} y_{v_4} b_v e_v z_v f_v g_v x_{v_4} y_{v_5} f_v h_v k_v x_{v_5} y_{v_6} g_v l_v x_{v_6} y_{v_1} h_v$

$B_v =$

$b_v y_{v_1} x_{v_1} c_v d_v y_{v_2} x_{v_2} a_v b_v e_v y_{v_3} x_{v_3} d_v e_v f_v h_v y_{v_4} x_{v_4} f_v g_v l_v y_{v_5} x_{v_5} h_v k_v y_{v_6} x_{v_6} g_v$

In both  $A_v$  and  $B_v$ ,  $x_{v_i} y_{v_{(i+1) \bmod 6}}$  for  $1 \leq i \leq 5$ ,  $x_{v_6} y_{v_1}$ , and  $y_{v_i} x_{v_i}$  for  $1 \leq i \leq 6$  are separators and are not adjacencies following our definition of breakpoints.  $A_v$  and  $B_v$  contain 29 and 28 letters respectively, with  $z_v$  not appearing in  $B_v$ . Among other letters,  $a_v$ ,  $c_v$ ,  $k_v$  and  $l_v$  only appear once in  $A_v$ .

For each edge  $(u, v) \in E$ , we locally modify  $A_u$ ,  $B_u$ ,  $A_v$  and  $B_v$ . Before that a few terms will be needed. The letters  $a_v$  and  $c_v$  in  $A_v$  are called the left sockets of  $A_v$  and the letters  $k_v$  and  $l_v$  in  $A_v$  are the right sockets. Initially, all sockets are free.

As in [7], we orient the edges of  $G$  in such a way that each vertex has at most two incoming edges and at most two outgoing edges. This can be done as follows: find a maximal set (with respect to inclusion) of edge-disjoint cycles in  $G$ , and in each cycle, orient the edges to form a directed cycle. The remaining edges form a forest. For each tree in the forest, choose one of its nodes of degree one to be the root, and orient all edges in the tree away from the root. This orientation will clearly satisfy the desired properties.

Consider an edge  $(\overrightarrow{u, v})$  and a free right socket  $s_u$  of  $A_u$  and a free left socket  $s_v$  of  $A_v$ . Define  $R = f_u h_u$  if  $s_u = k_u$ , and  $R = g_u$  if  $s_u = l_u$ ;  $S = d_v e_v$  if  $s_v = c_v$ , and  $S = b_v$  if  $s_v = a_v$ . We do the following modifications:

(1) insert  $S$  to the immediate right of  $s_u$  in  $A_u$ ; in other words, change  $R s_u$  into  $R s_u S$ ;

(2) replace  $s_v S$  with  $s_u$  in  $A_v$ ; in other words, change  $s_v S$  into  $s_u$ ;

(3) replace  $s_v$  with  $s_u$  in  $B_v$ .

(4)  $B_u$  is unchanged.

After this modification process, all relevant sockets will become *non-free* or *used*.

The final instance of SF-SBD is composed of strings  $A$ ,  $B$  and a set  $Z$ , where  $A = \bigcup_{v \in V} A_v p_v q_v r_v w_v$ ,  $B = \bigcup_{v \in V} B_v r_v p_v w_v q_v$ , and  $Z = \{z_v | v \in V\}$ . More precisely,

$A$  and  $B$  can be considered as a concatenation of  $A_v p_v q_v r_v w_v$ 's and  $B_v r_v p_v w_v q_v$ 's with orders insignificant. We need to insert the elements in  $Z$  into  $B$ .

The reduction can be completed with the following lemmas.

**Lemma 4.** *If there is an edge  $(u, v) \in E$ , after the above modification,  $A_u$  and  $A_v$  cannot both contain 6 adjacencies with respect to  $B$ .*

*Proof.* If  $A_u$  contains 6 adjacencies with respect to  $B$ , they must be

$$\{(a_u b_u), (c_u d_u), (e_u z_u), (z_u f_u), (h_u k_u), (g_u l_u)\}.$$

If  $R = g_u$  and  $S = b_v$ , after the modification  $(g_u l_u)$  and  $(l_u b_v)$  cannot be an adjacency at the same time. (The claim is also true when  $R = g_u$  and  $S = d_v e_v$ .) Symmetrically, if  $R = f_u h_u$  and  $S = b_v$ , after the modification  $(h_u k_u)$  and  $(k_u b_v)$  cannot be an adjacency at the same time. (The claim is also true when  $R = f_u h_u$  and  $S = d_v e_v$ .) Therefore, as long as there is an edge between  $u$  and  $v$ ,  $A_u$  and  $A_v$  cannot both contain 6 adjacencies with respect to  $B$ .  $\square$

**Lemma 5.** *Let  $G$  be a cubic graph on  $n$  vertices. There exists an independent set  $I$  of size  $k$  in  $G$  if and only if there exists a scaffold filling such that there are  $5n+k$  adjacencies between  $A$  and  $B + Z$ .*

*Proof.* For a vertex  $v \in V$ , if  $v \in I$  then there can be 6 adjacencies in  $A_v$ . This can be done as follows: insert  $z_v$  between  $e_v$  and  $f_v$  in  $B_v$ , i.e.,

$$B_v + \{z_v\} =$$

$$b_v y_{v1} x_{v1} c_v d_v y_{v2} x_{v2} a_v b_v e_v y_{v3} x_{v3} d_v e_v z_v f_v h_v y_{v4} x_{v4} f_v g_v l_v y_{v5} x_{v5} h_v k_v y_{v6} x_{v6} g_v.$$

Otherwise, if  $v$  is not in  $I$ , there are at most 5 adjacencies in  $A_v$ , following Lemma 4. All the separators  $x_{v_i}$ ,  $y_{v_i}$  and  $p_v$ ,  $q_v$ ,  $r_v$  and  $w_v$  cannot form an adjacency. Therefore, we have  $6k + 5(n - k) = 5n + k$  adjacencies between  $A$  and  $B + Z$ .

The converse can be proved using a similar argument. If there are  $5n + k$  adjacencies between  $A$  and  $B + Z$ , following Lemma 4, exactly  $6k$  adjacencies are from some  $A_v$  where  $v$  is in an independent set. Consequently, there is an independent set of  $G$  with size  $k$ .  $\square$

As it is obvious that SF-SBD is in NP, with Lemma 5, we hence have the following theorem.

**Theorem 2.** *SF-SBD is NP-complete.*

## 5 Two-Sided Scaffold Filling under the Rearrangement Distance

In this section, we show how to adapt the ideas in Section 3 to solve the two-sided scaffold filling problem under the rearrangement distance, when the genomes are multichromosomal and the genes are signed.

## Rearrangement Distance

The *rearrangement distance* or *genomic distance*  $D(G_1, G_2)$  is a metric counting the number of genomic rearrangement operations necessary to transform one signed multichromosomal genome  $G_1$  containing  $n$  distinct genes into another,  $G_2$ . The + or – sign indicates the “reading direction” of a gene, left-to-right or right-to-left.

For a comprehensive repertoire of operations, which we need not elaborate here, Yanopoulos *et al.* [13] showed that  $D$  could be calculated efficiently using the *breakpoint graph* [11] of  $G_1$  and  $G_2$  as follows:

1. Replace each positively-signed gene  $g$  on a chromosome by the vertex pair  $g_t, g_h$ ; replace a negative  $-g$  by  $g_h, g_t$ .
2. Each pair of successive genes in the genome defines one edge connecting the pair of vertices that are adjacent in the vertex order. E.g., if  $i\ j - k$  are neighboring genes on a chromosome then the two edges they define are  $\{i_h, j_t\}$  and  $\{j_h, k_h\}$ . This leaves two unconnected vertices at the ends of each chromosome. Define an edge incident to each such vertex in genome  $G_1$  and  $G_2$  connecting it to a new vertex, all labelled  $T_1$  in  $G_1$  and  $T_2$  in  $G_2$ .
3. Color the edges of  $G_1$  and  $G_2$  blue and red, respectively.
4. Identify (i.e., superimpose) each vertex in  $G_1$  with the identically labelled vertex in  $G_2$ .
5. Make a cycle of any path ending in two  $T_1$  or two  $T_2$  vertices, connecting them by a red or blue edge, respectively, while for a path ending in a  $T_1$  and a  $T_2$ , collapse them to form one  $T$  vertex.
6. Each vertex is now incident to one blue and one red edge. This bicolored graph, the breakpoint graph, decomposes uniquely into  $\kappa$  alternating cycles. If  $n'$  is the number of blue edges,  $D(G_1, G_2) = n' - \kappa$  (see [13]) and the optimizing rearrangements are rapidly recovered by operations on the graph.

## Bundles

Now consider the case where the genes in  $G_2$  are a subset of the genes in  $G_1$ . We say some genes are *missing* from  $G_2$ . The one-sided scaffold filling problem is to insert the missing genes in  $G_2$ , thus forming  $\bar{G}_2$ , in such a way as to minimize  $D(G_1, \bar{G}_2)$ . In [10], it was shown that the one-sided scaffold filling problem under the rearrangement distance can be solved in polynomial time, in fact, in linear time once the breakpoint graph is constructed.

We can still construct the breakpoint graph, except that some vertices, called *free ends*, will only be incident to a blue edge and thus paths in the graph can end not only in  $T$  vertices but also in free ends. When this happens, step 5 in the breakpoint graph construction cannot be completed, and the decomposition and calculation in step 6 are blocked.

A *bundle* is a subset of the paths in this partial breakpoint graph of  $G_1$  and  $G_2$ . (Partial because some paths are not cycles nor do they end in a  $T$ .) Each bundle is associated with one or more of the missing genes. The vertices corresponding to each missing gene, its free ends, must be in the same bundle and must be endpoints of one

or two paths. To simplify the exposition, we assume that no bundle consists entirely of blue edges, i.e., no chromosome in  $G_1$  has all its genes absent from  $G_2$ . This case is easily handled separately, and does not affect the distance calculations.

To construct a bundle, we initiate it with any path not already in any bundle and ending with a free end. Then if a path containing free end  $g_t$  is in a bundle  $B$ , then we also include the path with  $g_h$  as a free end, and vice versa. There can be zero or two  $T$  vertices in a bundle. We now present the details on the two-sided scaffold filling problem under the rearrangement distance. It is not hard to show the following lemma.

**Lemma 6.** *If genomes  $G_1$  and  $G_2$  both contain the same  $n$  genes, and if  $m \geq 1$  genes are inserted anywhere into both  $G_1$  and  $G_2$  to get  $\bar{G}_1$  and  $\bar{G}_2$ , then  $D(\bar{G}_1, \bar{G}_2) \geq D(G_1, G_2)$ .*

In a one-sided scaffold filling problem for  $G_1$  and  $G_2$ , suppose there are  $r$  cycles in the partial breakpoint graph and suppose this graph determines  $\beta$  bundles. Let  $\underline{G}_1$  be the genome formed by deleting from  $G_1$  the genes already missing from  $G_2$ .

**Lemma 7.** *Let  $\kappa$  be the number of cycles in the breakpoint graph of  $\underline{G}_1$  and  $G_2$ . Then  $\beta = \kappa + r$ .*

*Proof.* If  $a_t$  and  $a_h$  are a pair of free ends in a bundle, incident to blue edges  $(a_t, x)$  and  $(a_h, y)$ , remove  $a_t$  and  $a_h$  and replace  $(a_t, x)$  and  $(a_h, y)$  by  $(x, y)$ . Repeat until there are no more free ends in the bundle. This process converts a bundle into a cycle. Repeated across all bundles it also removes all the missing genes. Therefore the number of cycles in the partial breakpoint graph plus the number of bundles determined by this graph equals the number of cycles in the breakpoint graph of  $\underline{G}_1$  and  $G_2$ .  $\square$

It was shown in [10] how the one-sided scaffold filling problem can be solved by *completing* each bundle separately, i.e., by inserting the missing genes or drawing the red edges between free ends within each bundle. It turns out that we have the following theorem [10].

**Theorem 3.** *For a bundle with  $\nu$  paths, there are  $\nu + 1$  cycles produced by completing the bundle, while  $\nu$  genes inserted in  $G_2$ .*

The following corollary follows immediately.

**Corollary 1.** *After completing all the bundles with  $(\nu_1, \nu_2, \dots, \nu_\beta)$  paths, there are  $m = \nu_1 + \nu_2 + \dots + \nu_\beta$  genes inserted and the number of cycles is  $m + \beta$ .*

Therefore, we have the following main result.

**Theorem 4.**  $D(G_1, \bar{G}_2) = D(\underline{G}_1, G_2)$ .

*Proof.* The breakpoint graph of  $G_1$  and  $\bar{G}_2$  has  $m$  more blue edges than the breakpoint graph of  $\underline{G}_1$  and  $G_2$ , corresponding to the insertion of the  $m$  missing genes. But since it had  $r$  cycles before bundle completion, the breakpoint graph of  $G_1$  and  $\bar{G}_2$  has  $r + m + \beta$  cycles, from Corollary 1. This is  $m$  more than the  $r + \beta$  cycles in the breakpoint graph of  $\underline{G}_1$  and  $G_2$  (Lemma 7). By definition of the distance  $D$ , we have  $D(G_1, \bar{G}_2) = D(\underline{G}_1, G_2)$   $\square$

For two-sided scaffold filling, we thus have the following exact algorithm for the case when  $G_1$  contains genes  $G + X$  and  $G_2$  contains genes  $G + Y$ , where  $G$ ,  $X$  and  $Y$  are disjoint sets of genes.

1. Remove all the genes in set  $X$  from  $G_1$  to get  $G'_1$ , containing only the genes in  $G$ .
2. Apply one-sided scaffold filling to  $G_2$  and  $G'_1$ , inserting all the genes in  $Y$  into  $G'_1$ , producing genome  $G''_1$ .
3. Restore genes in set  $X$  to  $G''_1$ . The insertion points are only constrained by the gene order in  $G_1$ . The new genome,  $G'''_1$  contains all the genes in  $G$ ,  $X$ , and  $Y$ .
4. Apply one-sided scaffold filling to  $G'''_1$  and  $G_2$ , inserting all the genes in  $X$  into  $G_2$ .

Because the distance between the two genomes reduced to the genes in  $G$  is a lower bound for the distance between any two genomes enlarged through gene insertion, by Lemma 6, and because steps 2 and 4 do not increase this distance, by Theorem 4, and because step 3 is as general as possible while respecting the gene order of  $G_1$ , the correctness of the algorithm is verified.

As for the one-sided scaffold filling, once the breakpoint graph is constructed, the remaining steps run in linear time.

## 6 Concluding Remarks

In this paper, we investigate the scaffold filling problems under both the breakpoint and rearrangement distances. A very interesting open problem is when the missing genes can be only inserted in between contigs (i.e., in some predefined locations); our current method cannot generate any result with some performance guarantee. Another problem is dealing with the cases when gene duplications are allowed. Our NP-completeness proof implies that this is closely related to the Minimum Common String Partition problem, for which the existence of an FPT algorithm [5] is not known yet. In [4,8], several special cases were shown to admit exact algorithms, but when using the optimal number of blocks in the final solution as the only parameter, we do not know whether an FPT algorithm exists or not.

## Acknowledgments

This research is partially supported by NSF under grant DMS-0918034, by NSF of China under grant 60928006 and by NSERC of Canada. We also thank anonymous reviewers for several useful comments.

## References

1. Chain, P., Grafham, D., Fulton, R., FitzGerald, M., Hostetler, J., Muzny, D., Ali, J., et al.: Genome project standards in a new era of sequencing. *Science* 326, 236–237 (2009)
2. Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., Jiang, T.: Computing the assignment of orthologous genes via genome rearrangement. In: Proc. of the 3rd Asia-Pacific Bioinformatics Conf. (APBC 2005), pp. 363–378 (2005)

3. Chrobak, M., Kolman, P., Sgall, J.: The greedy algorithm for the minimum common string partition problem. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) *RANDOM 2004 and APPROX 2004*. LNCS, vol. 3122, pp. 84–95. Springer, Heidelberg (2004)
4. Damaschke, P.: Minimum Common String Partition Parameterized. In: Crandall, K.A., Lagergren, J. (eds.) *WABI 2008*. LNCS (LNBI), vol. 5251, pp. 87–98. Springer, Heidelberg (2008)
5. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer, Heidelberg (1999)
6. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York (1979)
7. Goldstein, A., Kolman, P., Zheng, J.: Minimum common string partitioning problem: Hardness and approximations. In: Fleischer, R., Trippen, G. (eds.) *ISAAC 2004*. LNCS, vol. 3341, pp. 473–484. Springer, Heidelberg (2004); also in: *The Electronic Journal of Combinatorics* 12, paper R50 (2005)
8. Jiang, H., Zhu, B., Zhu, D., Zhu, H.: Minimum common string partition revisited. In: Lee, D.-T., Chen, D.Z., Ying, S. (eds.) *FAW 2010*. LNCS, vol. 6213, pp. 45–52. Springer, Heidelberg (2010)
9. Kaplan, H., Shafir, N.: The greedy algorithm for edit distance with moves. *Inf. Process. Lett.* 97(1), 23–27 (2006)
10. Muñoz, A., Zheng, C., Zhu, Q., Albert, V., Rounsley, S., Sankoff, D.: Scaffold filling, contig fusion and gene order comparison. *BMC Bioinformatics* 11, 304 (2010)
11. Tesler, G.: Efficient algorithms for multichromosomal genome rearrangements. *J. Computer and System Sciences* 65, 587–609 (2002)
12. Watterson, G., Ewens, W., Hall, T., Morgan, A.: The chromosome inversion problem. *J. Theoretical Biology* 99, 1–7 (1982)
13. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346 (2005)