BIOINFORMATICS ORIGINAL PAPER

Vol. 27 no. 3 2011, pages 311–316
doi:10.1093/bioinformatics/btq674

Genome analysis

Advance Access publication December 6, 2010

Algorithms for sorting unsigned linear genomes by the DCJ operations

Haitao Jiang[1,2], Binhai Zhu[1,*] and Daming Zhu[2]

[1]Department of Computer Science, Montana State University, Bozeman, MT 59717, USA and [2]School of Computer Science and Technology, Shandong University, Jinan, China

Associate Editor: Martin Bishop

ABSTRACT

Motivation: The double cut and join operation (abbreviated as DCJ) has been extensively used for genomic rearrangement. Although the DCJ distance between signed genomes with both linear and circular (uni- and multi-) chromosomes is well studied, the only known result for the NP-complete unsigned DCJ distance problem is an approximation algorithm for unsigned linear unichromosomal genomes. In this article, we study the problem of computing the DCJ distance on two unsigned linear multichromosomal genomes (abbreviated as UDCJ).

Results: We devise a 1.5-approximation algorithm for UDCJ by exploiting the distance formula for signed genomes. In addition, we show that UDCJ admits a weak kernel of size $2k$ and hence an FPT algorithm running in $O(2^{2k}n)$ time.

Contact: bhz@cs.montana.edu

Received on September 27, 2010; revised on December 1, 2010; accepted on December 2, 2010

1 INTRODUCTION

Computing genomic distance on gene order is a fundamental problem in computational biology. In the last two decades, a variety of biological operations, such as reversals, translocations, fusions, fissions, transpositions and block-interchanges, have been proposed to handle gene order. The *double cut and join* operation, introduced by Yancopoulos *et al*., 2005, unifies all the classical operations. In the past, the rearrangement distance for signed genomes is well studied for single operations, like reversals (Hannenhalli and Pevzner, 1999), combinations of operations (reversals, translocations, fusions and fissions) (Hannenhalli and Pevzner, 1995) and universal operations (double cut and join) (Bergeron *et al*., 2006; Yancopoulos *et al*., 2005).

Unfortunately, as for unsigned genomes, most of these problems seem to be NP-hard. Then it is natural to devise relevant approximation algorithms. A 1.5-approximation algorithm was devised for sorting by unsigned reversals (Christie, 1998), and the approximation factor was improved to 1.375 by Berman *et al*., 2002. The problem of sorting by unsigned translocations was investigated by Cui *et al*., 2008, and an algorithm with an approximation factor of $1.5+\varepsilon$ was proposed. Transposition, though occurring much less than reversal and translocation, is an indispensable operation in the evolutionary events. The problem of sorting by transpositions was first studied by Bafna and Pevzner, 1998, who devised a 1.5-approximation algorithm running in quadratic time. Later, the approximation factor was improved to 1.375 by Elias and Hartman, 2006. The problem of sorting by short block-moves, a special but more practical case of transpositions, was studied by Jiang and Zhu, 2011, and they obtained an 14/11-approximation algorithm. The design of FPT algorithms for genome rearrangement problems was started very recently, with the help of *weak kernels*. (Intuitively, an FPT algorithm is an exact algorithm which runs in polynomial time when the problem solution size, like the number of unsigned reversals to sort a sequence, is bounded by a constant. The relevant formal definitions will be given in the next section.) Both sorting by unsigned reversals and sorting by unsigned translocations admit small weak kernels, hence are in FTP (Jiang *et al*., 2010).

As far as we know, the only known positive result for sorting unsigned genomes by minimum DCJ operations (or interchangeably, the unsigned DCJ distance problem) is a factor-1.416 approximation for the case of linear unichromosomal genomes (Chen, 2010). Of course, even in this case the problem involves computing a maximum alternating-cycle decomposition (MAX-ACD) of the breakpoint graph, which is NP-complete (Caprara, 1999); therefore, it is not surprising that the unsigned DCJ distance problem is NP-complete, even for linear unichromosomal genomes (Chen, 2010). Prior to our current work, there has been no FPT algorithm known for the unsigned DCJ distance problem.

Our contributions: In this article, we introduce DCJ operations on unsigned linear multichromosomal genomes to compute the corresponding genomic distance. We devise a 1.5-approximation algorithm for linear multichromosomal genomes in Section 3. In Section 4, we obtain a weak kernel of size $2k$ for UDCJ; moreover, we present an FPT algorithm running in $O(2^{2k}n)$ time.

2 PRELIMINARIES

Gene, chromosome and genome: An unsigned gene is a sequence of DNA, usually denoted by a positive integer. A chromosome can be viewed as a sequence of genes and denoted by a permutation, while a genome is a set of chromosomes. A gene that lies at the end of some linear chromosome is called an *end-gene*. Gene i and j form an *adjacency* if they are consecutive in some chromosome. An adjacency (g_i, g_{i+1}) is *perfect* if it satisfies $|g_{i+1}-g_i|=1$. A chromosome is perfect if every adjacency is perfect. A genome is perfect if all its chromosomes are perfect. As a convention, we always list the genes in a perfect genome in increasing order. For instance, a perfect genome with 3 chromosomes and 10 genes can be listed as (1, 2, 3, 4), (5, 6, 7) and (8, 9, 10). We study unsigned linear

*To whom correspondence should be addressed.

multichromosomal (multilinear or simply linear, for short) genomes in this article.

Breakpoint graph: Above all, we recall the well-known tool for computing the genomic rearrangement distance, the Breakpoint Graph (Bafna and Pevzner, 1998). Given two unsigned genomes A and B on the same set of n genes, the *Breakpoint Graph* $BG(A,B)=(V,E_b\cup E_g)$, where $|V|=n$ and each vertex in V corresponds to a gene, every adjacency in A forms a black edge belonging to E_b and every adjacency in B forms a gray edge belonging to E_g. It is known that in this case computing a maximum alternating-cycle decomposition in $BG(A,B)$ is NP-complete (Caprara, 1999).

As for signed genomes F and H, the breakpoint graph $BG_s(F,H)$ is a bit different. Due to the sign, each gene has one head and one tail corresponding to two vertices in the breakpoint graph. Consequently, the head has only one adjacency in F and H respectively, so does the tail. Then each vertex in the breakpoint graph has degree at most two, which means that the breakpoint graph is composed of cycles and paths, and the black edges and gray edges appear alternatively in the cycles or paths. So the maximum alternating-cycle decomposition is easy in this case. A cycle that contains l black edges is called an l-cycle.

The double cut and join operations: The Double Cut and Join operation (abbreviated as DCJ) unifies all the traditional genome rearrangement operations such as reversal, translocation, fusion, fission, transposition and block interchange, as well as excision, integration, circularization and linearization. The formal definition of a DCJ operation on the breakpoint graph is as follows.

DEFINITION 1. *The double cut and join operation acts on the breakpoint graph in the following four ways (Fig. 1):*

(1) *For two black edges* $b_1=(g_i,g_{i+1})$ *and* $b_2=(g_j,g_{j+1})$*, cut them, and either form two new black edges* $b'_1=(g_i,g_{j+1})$ *and* $b'_2=(g_j,g_{i+1})$ *or form two new black edges* $b'_1=(g_i,g_j)$ *and* $b'_2=(g_{i+1},g_{j+1})$.

(2) *For a black edge* $b=(g_i,g_{i+1})$ *and an end-gene* g_j*, cut the black edge, and either form a new black edge* $b'=(g_i,g_j)$ *and a new end-gene* g_{i+1} *or form a new black edge* $b'=(g_j,g_{i+1})$ *and a new end-gene* g_i.

(3) *For two end-genes* g_i *and* g_j*, join them with a black edge* (g_i,g_j).

(4) *For a black edge* $b=(g_i,g_{i+1})$*, cut it into two end-genes* g_i *and* g_{i+1}.

Note that the black edges and end-genes involved in one DCJ operation can be in the same chromosome, then a circular chromosome may form after some DCJ operations.

Problem statement: We now formally formulate the problem to be investigated in this article.

Sorting unsigned genomes by the DCJ operations (UDCJ):
Input: Two unsigned linear genomes A and B and an integer k.
Question: Can A be converted into B by a series of k DCJ operations $\rho_1,\rho_2,\ldots,\rho_k$?

The minimum k is the unsigned *DCJ distance* between A and B. Following the results in (Caprara, 1999; Chen, 2010), UDCJ is also NP-complete.

W.L.O.G, assume that B is perfect. Let l_A and l_B be the number of linear chromosomes in A and B, respectively, we can also assume

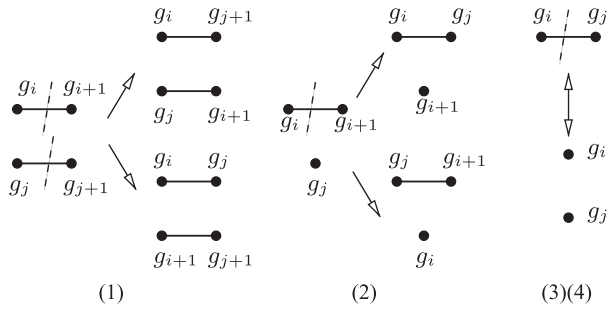

Fig. 1. The DCJ operation.

that $l_A\geq l_B$, since all the DCJ operations are reversible, which means that if there exists consecutive DCJ operations $\rho_1\rho_2\ldots\rho_m$ that convert A into B, then we can also convert B into A by $\rho_m^{-1}\rho_{m-1}^{-1}\ldots\rho_1^{-1}$, where ρ_i^{-1} is the reversed operation of ρ_i.

FPT and weak kernel: Basically, a fixed parameter tractable (FPT) algorithm for a decision problem Π with solution value k is an algorithm, which solves the problem in $O(f(k)n^c)=O^*(f(k))$ time, where f is any function only on k, n is the input size and c is some fixed constant not related to k. FPT also stands for the set of problems that admit such an algorithm (Downey and Fellows, 1999; Flum and Grohe, 2006). Weak kernel is a relatively new concept; intuitively, it refers to the direct or indirect 'search space' to solve a search problem. For a search problem in NP, if it admits a weak kernel of size $g(k)$, then it is in FPT (Jiang *et al.*, 2010). We comment that weak kernel is different from the traditional kernel in which the problem instance size is reduced (to a function of k), while a weak kernel only implies that the direct or indirect solution search space is reduced (to a function of k). More details can be found in (Jiang *et al.*, 2010).

3 A 1.5-APPROXIMATION ALGORITHM

In this section, we present a 1.5-approximation algorithm for double cut and join distance on unsigned multilinear genomes. We first comment that the method by Chen (2010) cannot be converted to solve our problem, as with multilinear genomes the underlying breakpoint graph is more complex (i.e. possibly with many paths). Given an original genome A with l_A chromosomes and a target perfect genome B with l_B chromosomes, our goal is to convert A into B by a series of DCJ operations so that the number of DCJ operations is as few as possible. To design an approximation algorithm, we first need the structure properties of UDCJ, which in fact can be obtained from the corresponding *signed* genomes.

3.1 Structure properties of UDCJ

For an unsigned genome A, a *signed-version* of A is obtained by assigning '$+$' or '$-$' to each gene in A, with '$+$' signs usually omitted. Obviously, every genome of n genes has exponential, i.e. 2^n, signed versions. Given two signed genomes F,H, we use $DCJ_s(F,H)$ to denote their signed DCJ distance.

THEOREM 1. *Given two unsigned linear multichromosomal genomes* A *and* B*, let the minimum DCJ distance between* A,B *be* $DCJ(A,B)$.

Then $DCJ(A,B)=DCJ_s(A^*,B^+)$, where A^* is some signed version of A, and B^+ is a special signed version of B with all signs being positive.

PROOF. Notice that, loosely speaking, we can take $B=B^+$.

($\Rightarrow$) Assume that there exists a series of consecutive DCJ operations $\rho_1\rho_2\ldots\rho_m$ that convert A into B. We say that a DCJ operation ρ changes the sign of a gene g if ρ involves reversing a segment of genes including g. For each gene g in A, let T_g denote the number of times that the sign of g is changed if we trace all the m DCJ operations. g is assigned '$-$', if T_g is odd; and '$+$', if T_g is even. Then we obtain a signed version of A, A^*, which can be converted into B^+ by the m equivalent signed DCJ operations. Thus, $DCJ(A,B)\geq DCJ_s(A^*,B^+)$.

($\Leftarrow$) If there exists a signed version A^* of A that can be converted into B^+ by m signed DCJ operations $\rho_1\rho_2\ldots\rho_m$, then we can also use these m (signed) DCJ operations to convert A into B, ignoring the gene signs. Thus $DCJ(A,B)\leq DCJ_s(A^*,B^+)$. ■

We now proceed to obtain the necessary properties of the optimal solution. First of all, in order to avoid distinct end points of chromosomes in A and B, we add unlabeled caps to both ends of each linear chromosome in genomes A and B, respectively, then connect the A-cap and its adjacent end-gene with a black edge and the B-cap and its adjacent end-gene with a gray edge in $BG(A,B)$. The above preprocess is called *capping*. Note that each gene in $BG(A,B)$ has degree 4 after capping, i.e. with two black edges and two gray edges. After capping, genomes A and B become $\bar{A}$ and $\bar{B}$, respectively. We denote the resulting graph by $BG(\bar{A},\bar{B})$.

As it seems to be hard to extract the properties of the optimal solution from $BG(\bar{A},\bar{B})$ directly, we take a detour. We notice that, for signed genomes F and H, after capping each vertex in the breakpoint graph $BG_s(F,H)$ has degree two and each cap has degree one, which means that all the paths end with caps. A path with an A-cap end and a B-cap end (respectively, two A-cap ends, two B-cap ends) is an AB-path (respectively, AA-path, BB-path).

There are three ways to construct cycles from $BG_s(F,H)$in the breakpoint graph $BG_s(F,H)$ of signed genomes F and H, after capping .

(1) *single-identifying*: identify the two caps of each AB-path, close the path into a cycle containing just one A-cap (with the B-cap eliminated).

(2) *double-identifying*: identify each B-cap of a BB-path and each A-cap of an AA-path, join an AA-path and a BB-path into a cycle containing two A-caps (with the two B-caps eliminated).

(3) *joining*: connect the two A-caps of an AA-path with a gray edge.

Let $BG_s(\bar{F},\bar{H})$ denote the resulting breakpoint graph after constructing cycles from $BG_s(F,H)$ following the above three ways. Then the signed DCJ distance between the signed genomes $\bar{F}$ and $\bar{H}$, $DCJ_s(\bar{F},\bar{H})=b-c$, where b is the number of black edges and c is the number of cycles in $BG_s(\bar{F},\bar{H})$ (Yancopoulos *et al.*, 2005).

In Figure 2, we show an example of $\bar{F},\bar{H}$ and $BG_s(\bar{F},\bar{H})$, before the identifying and joining operations are performed. In the figure, an empty round (respectively, square) node is an A-cap (respectively, B-cap); moreover, in $BG_s(\bar{F},\bar{H})$, a signed gene $+i$ (respectively, $-i$) is already converted to $(2i-1,2i)$ [respectively, $(2i,2i-1)$]. After two single-identifying operations are performed, we have two new

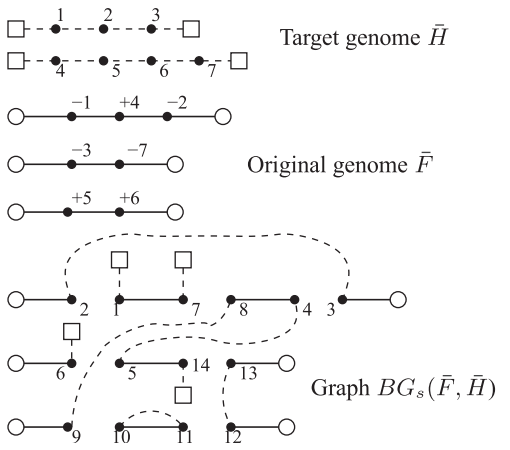

Fig. 2. The breakpoint graph $BG_s(\bar{F},\bar{H})$, before the identifying and joining operations are performed.

cycles (6) and (9, 8, 4, 5, 14). After a double-identifying operation is performed, we have a new cycle (2, 3, 7, 1). After a joining operation is performed, we have a new cycle (12, 13).

It is worth mentioning that this distance formula is equivalent to that of Bergeron *et al.*, 2006, i.e. $DCJ_s(F,H)=n-C-\lfloor I/2\rfloor$, where n is the number of genes, C is the number of cycles and I is the number of odd paths in their corresponding adjacency graph. To see this, note that I also equals to the number of AB-paths in the breakpoint graph; in addition, we have $b=n+l_A$, $c=C+I+\lfloor(2l_A-I)/2\rfloor$. So $DCJ_s(\bar{F},\bar{H})=DCJ_s(F,H)$.

COROLLARY 1. *Given two unsigned linear multichromosomal genomes A and B, let A^* and B^+ be defined as in Theorem 1. Then $DCJ(A,B)=DCJ_s(\bar{A}^*,\bar{B}^+)$, where $\bar{A}^*$ (respectively, $\bar{B}^+$) is a capping of A^* (respectively, B^+).*

PROOF. It follows from Theorem 1 that $DCJ(A,B)=DCJ_s(A^*,B^+)$. The statements in the last paragraph show that $DCJ_s(A^*,B^+)=DCJ_s(\bar{A}^*,\bar{B}^+)$. Then the corollary follows. ■

Notice that computing an alternating-cycle decomposition of $BG(\bar{A},\bar{B})$ is equivalent to finding a signed version of $\bar{A}$. To extract the properties of the optimal solution, we first try to make use of the breakpoint graph $BG(\bar{A},\bar{B})$ instead of $BG(A,B)$. Following Corollary 1, we can now make use of the breakpoint graph $BG_s(\bar{A}^*,\bar{B}^+)$. From the way $BG_s(\bar{A}^*,\bar{B}^+)$ is constructed, we only need to find an optimal $\bar{A}^*$ such that the number of disjoint alternating-cycles in $BG_s(\bar{A}^*,\bar{B}^+)$ is maximized. The reason is that the number of black edges in $BG(\bar{A},\bar{B})$ is fixed.

Then we have $d^{opt}=DCJ(A,B)=DCJ_s(\bar{A}^*,\bar{B}^+)=b-c_1-c_2-c_3'$, where b is the number of black edges in $BG_s(\bar{A}^*,\bar{B}^+)$, c_1 and c_2 are the number of 1-cycles and 2-cycles in $BG_s(\bar{A}^*,\bar{B}^+)$, respectively, and c_3' is the number of cycles with three or more black edges in $BG_s(\bar{A}^*,\bar{B}^+)$. Obviously, $c_3'\leq(b-c_1-2c_2)/3$, thus we have the following formula:

$$\begin{aligned} d^{opt} &= b-c_1-c_2-c_3' \\ &\geq b-c_1-c_2-(b-c_1-2c_2)/3 \end{aligned}$$

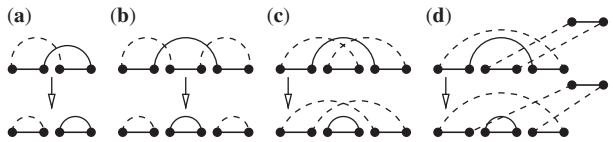

Fig. 3. 1-cycle containing two genes.

$$= 2(b-c_1)/3-c_2/3$$
$$= \frac{2}{3}\cdot(b-c_1-c_2/2).$$

The above formula implies that, if we can convert A into B by at most $b-c_1-c_2/2$ DCJ operations, then we obtain a 1.5-approximation algorithm for UDCJ.

3.2 The algorithm

The idea of our approximation algorithm is as follows. We compute $BG(\bar{A},\bar{B})$ and try to first keep all the 1-cycles in it. Then we compute many 2-cycles from $BG(\bar{A},\bar{B})$ (in fact, at least $c_2/2$ such 2-cycles). We comment that a similar idea was used by Christie (1998) on sorting by unsigned reversals. On the other hand, the LP-relaxation algorithm by Chen, 2010 cannot handle paths (and caps) so it cannot be immediately generalized to solve our problem.

The following lemma, which involves handling paths and caps, shows that keeping all the 1-cycles in $BG(\bar{A},\bar{B})$ is a good strategy to obtain some optimal alternating-cycle decomposition of it.

LEMMA 1. *There is some maximum alternating-cycle decomposition of $BG(\bar{A},\bar{B})$ in which all c_1' 1-cycles in $BG(\bar{A},\bar{B})$ are kept.*

PROOF. We modify the optimal alternating-cycle decomposition in $BG(\bar{A},\bar{B})$ in such a way: if two genes, say g_i and g_{i+1}, are connected by a black edge and a gray edge, then we reassign the signs of these two genes to obtain a 1-cycle; if a gene, say g_i, is connected to an A-cap by a black edge and to a B-cap by a gray edge, then we reassign the sign of the gene and identify the two caps to obtain a 1-cycle. If the newly obtained 1-cycle contains two genes, then there are two cases.

Case (I): Only one of the signs of g_i and g_{i+1} is changed. W.L.O.G, assume that the sign of g_i is changed, see Figure 3a. The number of cycles is increased by one.

Case (II): Both of the signs of g_i and g_{i+1} are changed, see Figure 3b–d. The number of cycles is increased by two or one or unchanged, respectively.

If the newly obtained 1-cycle contains one gene and a cap (which is identified by an A-cap a and a B-cap b), then there are four cases. Note that b must be identified with some A-cap a''.

Case (1): The A-cap a joins with another A-cap a'. The number of cycles is unchanged. See Figure 4a.

Case (2): The A-cap a is identified with a B-cap b' and a,b belongs to distinct cycles. The number of cycles is unchanged. See Figure 4b.

Case (3): The A-cap a is identified with a B-cap b' and a,b belongs to the same cycle. The number of cycles is increased by one. See Figure 4c.

Case (4): The A-cap a is identified with the B-cap b but the cycle containing a,b also contains two identified caps a' and b'. The number of cycles is increased by one. See Figure 4d. ■

Following Lemma 1, we know that keeping all the 1-cycles in in $BG(\bar{A},\bar{B})$ will not affect the value of some optimal alternating-cycle decomposition of it. Therefore, from now on we only focus on the optimal alternating-cycle decomposition of $BG(\bar{A},\bar{B})$, which always keeps all the 1-cycles. Consequently, in order to approximate the optimal DCJ distance, we just need to find out as many as at least half of the 2-cycles in an optimal alternating-cycle decomposition of $BG(\bar{A},\bar{B})$ (which keeps all 1-cycles). Now we present the algorithm *2-Cycle Decomposition* to compute such 2-cycles. In this algorithm, we first construct a graph G_1 whose vertices are the black edges (not in any 1-cycle) in $BG(\bar{A},\bar{B})$ and M is a maximum matching in G_1.

Note that the maximum matching M can be computed in polynomial time (Galil *et al.*, 1986); moreover, each edge in M results in a candidate 2-cycle. In order to bound the cardinality of S, we need the following lemmas.

LEMMA 2. *Let M be a maximum matching in G_1, then $|M|\geq c_2$.*

PROOF. Following the discussion in Section 3.1, c_2 corresponds to the number of 2-cycles in an optimal alternating-cycle decomposition of $BG(\bar{A},\bar{B})$. These 2-cycles clearly form a matching in G_1. By the maximality of M, we have $|M|\geq c_2$. ■

Algorithm *2-Cycle Decomposition*
Input: $BG(\bar{A},\bar{B})$
Output: *A set of edge-disjoint 2-cycles*

1 Construct a graph $G_1=(P,E_1)$ as follows:
  1.1 Each black edge in $BG(\bar{A},\bar{B})$, not contained in any 1-cycle, corresponds to a vertex in P.
  1.2 For each pair of vertices $u=(g_i,g_{i+1})$ and $v=(g_j,g_{j+1})$ corresponding to black edges between two genes, $(u,v)\in E_1$ iff there exist two gray edges which can form a 2-cycle together with these two black edges.
  1.3 For each pair of vertices $u=(g_i,a_i)$ and $v=(g_j,a_j)$ corresponding to black edges between a gene and a A-cap, $(u,v)\in E_1$ iff there exist a gray edge (g_i,g_j) in $BG(\bar{A},\bar{B})$.
  1.4 For a 2-gene vertex $u=(g_i,g_{i+1})$ and a 1-gene-1-cap vertex $v=(g_j,a_j)$, $(u,v)\in E_1$ iff there exist two gray edges (g_i,g_j) and (g_{i+1},b_j) or two gray edges (g_{i+1},g_j) and (g_i,b_{j+1}) in $BG(\bar{A},\bar{B})$, where b_j and b_{j+1} are B-caps.
2 Compute a maximum matching M in G_1.
3 Construct a graph $G_2=(Q,E_2)$ as follows:
  3.1 Each 2-cycle computed at Step 2 corresponds to a vertex in Q.
  3.2 Two vertices in Q form an edge in E_2 iff their corresponding cycles share a gray edge.
4 Compute a maximum independent set S of G_2.
5 Return S which is a set of edge-disjoint 2-cycles.

LEMMA 3. *The graph G_2 is composed of simple paths and isolated vertices.*

PROOF. All 2-cycles computed at Step 2 cannot share black edges. Since each gray edge is connected to at most four black edges, at most two 2-cycles which do not share black edges can share this gray edge. Equivalently, each gray edge can belong to at most two cycles computed from M. Each 2-cycle has two gray edges, so each vertex in G_2 has degree at most two.

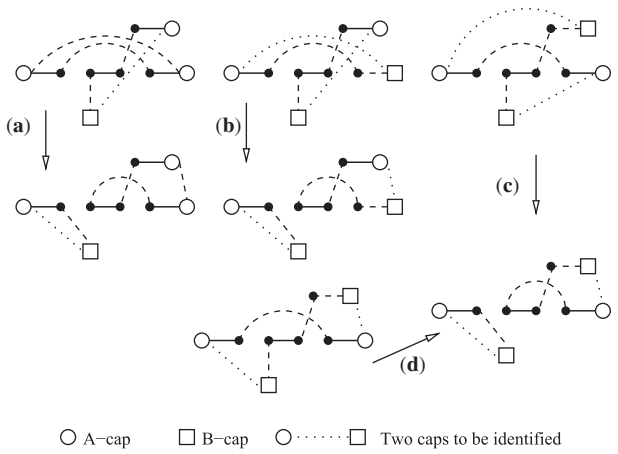

Fig. 4. 1-cycle containing one gene and one cap.

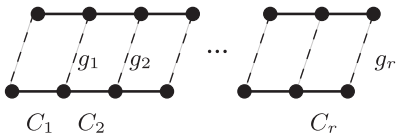

Fig. 5. An example of 2-cycles sharing gray edges.

It is sufficient to prove that G_2 does not contain cycles. Assume to the contrary that 2-cycles $C_1C_2\ldots C_r$ form a cycle in G_2, where C_i shares gray edge g_i with C_{i+1}, $1\leq i\leq r-1$, and C_r shares g_r with C_1. Then C_1 contains two gray edges g_1 and g_r, but the end points of g_1 and g_r cannot form two black edges (otherwise these two black edges will force into some black cycle—which implies that the input genome contains some circular chromosome). See Figure 5. ■

It is obvious that every 2-cycle containing caps has degree at most one in G_2, because the gray edge containing caps cannot be shared by two 2-cycles computed from M. The property we just proved in Lemma 3 is important for us to compute a maximum independent set in G_2 (without this property, the computation of a maximum independent set might be intractable). Lemma 3 immediately implies the next lemma.

LEMMA 4. *Let S be a maximum independent set in G_2, then $|S|\geq \lceil\frac{|M|}{2}\rceil$.*

Note that if a gene is contained in some 1-cycle, then its sign can be fixed easily, i.e. if the black edge reads from (left to right) like $(i,i+1)$ then both genes i and $i+1$ will be given positive signs, otherwise they will be given negative signs. If a gene is contained in some 2-cycle, its sign is fixed similarly. For instance, if in a 2-cycle the two black edges read like $(i,j),(i+1,j+1)$ (from left to right), then the signing should be $+i,-j,-(i+1),+(j+1)$. The other cases, e.g., when the directions of these black edges are possibly changed, are very much symmetric hence omitted. To complete the cycle decomposition, we arbitrarily assign signs to the remaining genes, then properly identify and join the remaining caps in the corresponding breakpoint graph. The complete *Whole-Cycle Decomposition* algorithm is presented as follows.

Algorithm *Whole-Cycle Decomposition*
Input: $BG(\bar{A},\bar{B})$
Output: $BG_s(\bar{A}',\bar{B}^+)$
1 Keep all 1-cycles and assign proper signs to genes involved in the 1-cycles in $BG(\bar{A},\bar{B})$.
2 Call 2-Cycle Decomposition, and assign proper signs to genes involved in the resulting 2-cycles.
3 Assign arbitrary signs to the remaining genes to have a signed genome $\bar{A}'$.
4 Construct $BG_s(\bar{A}',\bar{B}^+)$ by identifying and joining caps with single-identifying, double-identifying and joining operations.

Notice that once we have $BG_s(\bar{A}',\bar{B}^+)$ it is straightforward to compute the signed DCJ distance $d^{wcd}=DCJ_s(\bar{A}',\bar{B}^+)$ in linear time (Bergeron *et al.*, 2006; Yancopoulos *et al.*, 2005).

THEOREM 2. *Algorithm* Whole-Cycle Decomposition *approximates the DCJ distance between two unsigned linear multichromosomal genomes with a factor of 1.5.*

PROOF. From Lemma 4, we know that $|S|\geq\lceil c_2/2\rceil$; it follows from Lemma 1 that $c'_1=c_1$. The distance computed by Algorithm *Whole-Cycle Decomposition* is $d^{wcd}\leq b-c'_1-|S|$. The optimal distance d^{opt} satisfies that $d^{opt}\geq 2(b-c'_1-c_2/2)/3$. Thus, $d^{wcd}\leq (b-c'_1-\lceil c_2/2\rceil)\leq 1.5d^{opt}$. ■

4 A WEAK KERNEL AND AN FPT ALGORITHM

Similar to the problem of sorting by unsigned reversals and sorting by unsigned translocations (Jiang *et al.*, 2010), the UDCJ problem also possesses a (small and indirect) weak kernel.

Let k be the minimum number of DCJ operations converting A into B. A *weak kernel* for UDCJ is a set of genes in A whose signs cannot be fixed after the genes involved in all 1-cycles have been properly signed (following Lemma 1). Before computing the size of the weak kernel, we state the following lemma, which is simple but critical.

LEMMA 5. *Each DCJ operation can generate at most two 1-cycles.*

PROOF. Each DCJ operation cuts two black edges and forms at most two new black edges. Each new black edge can form at most one 1-cycle. ■

THEOREM 3. *The UDCJ problem has a weak kernel of size $2k$, hence can be solved in $O^*(2^{2k})$ time.*

PROOF. The $2k$ weak kernel is straightforward from Lemma 1 and Lemma 5. For any optimal alternating-cycle decomposition of $BG(\bar{A},\bar{B})$ which contains all possible number of c'_1 1-cycles, we have $k=b-c'_1-c'_2$ and $c'_2\leq(b-c'_1)/2$, where c'_2 is the number of cycles of length at least 2 in the optimal alternating-cycle decomposition of $BG(\bar{A},\bar{B})$. Thus, $k\geq(b-c'_1)/2$, equivalently, $(b-c'_1)\leq 2k$. Following Lemma 1, we can assign signs to all genes involved in 1-cycles. So each of the remaining gene is connected to two black edges, and each black edge has at most two unsigned genes as its end points, which means that the number of unsigned genes N is bounded by the number of black edges not involved in any 1-cycle, e.g. $N\leq b-c'_1\leq 2k$. Hence, the problem admits a weak kernel of size $2k$.

In other words, if the DCJ distance is equal to or smaller than k, there are at most 2^{2k} signed versions of A among which there must be an optimal one (e.g. A^* in Theorem 1). For each signed version of A, we can exploit the algorithm in Bergeron *et al*. (2006); Yancopoulos *et al*. (2005) to check whether it can be converted into B^+ by k or few DCJ operations. If so, we can compute the corresponding k unsigned DCJ operations to convert A into B. If no valid solution is found, we report NO. This algorithm clearly runs in $O(2^{2k}n) = O^*(2^{2k})$ time. ■

5 DISCUSSION

In this article, we devise the first approximation algorithm with a factor of 1.5 and an FPT algorithm running in $O(2^{2k}n)$ time for the NP-complete problem of sorting linear multichromosomal genomes under unsigned DCJ distance. It is interesting to improve the approximation factor as well as the running time of the FPT algorithm. For genomes containing circular chromosomes, our approximation algorithm cannot achieve the same performance as linear genomes, so it is also meaningful to handle the problem of sorting mixed genomes (i.e. with both linear and circular chromosomes) under unsigned DCJ distance.

ACKNOWLEDGEMENT

We thank anonymous reviewers for their valuable comments.

Funding: NSF grant (DMS-0918034); NSF of China under grant (60928006 and 61070019 in part).

Conflict of Interest: none declared.

REFERENCES

Bafna,V. and Pevzner,P. (1998) Sorting by Transpositions. *SIAM J. Discrete Math.*, **11**, 224–240.

Bergeron,A. *et al.* (2006) A unifying view of genome rearrangements. In *Proceedings of the 6th International Workshop on Algorithms in Bioinformatics (WABI'06)*, Springer, Germany, pp. 163–173.

Berman,P. *et al.* (2002) 1.375-Approximation algorithm for sorting by reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'02)*, Springer, Germany, pp. 200–210.

Caprara,A. (1999) Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM J. Discrete Math.*, **12**, 91–110.

Chen,X. (2010) On sorting permutations by double-cut-and-joins. In *Proceedings of the 16th International Conf. on Computing and Combinatorics (COCOON'10)*, Springer, Germany, pp. 439–448.

Christie,D. (1998) A 3/2-Approximation algorithm for sorting by reversals. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, ACM, New York, pp. 244–252.

Cui,Y. *et al.* (2008) A (1.5 + ϵ)-Approximation algorithm for unsigned translocation distance. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **5**, 56–66.

Downey,D. and Fellows,M. (1999) *Parameterized Complexity*, Springer.

Elias,I. and Hartman,T. (2006) A 1.375-Approximation algorithm for sorting by transpositions. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **3**, 369–379.

Flum,J. and Grohe,M. (2006) *Parameterized Complexity Theory.* Springer, Germany.

Galil,Z. *et al.* (1986) An O(EVlog V) algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput.*, **15**, 120–130.

Hannenhalli,S. and Pevzner,P. (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS'95)*, IEEE Computer Society, pp. 581–589.

Hannenhalli,S. and Pevzner,P. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, **46**, 1–27.

Jiang,H. *et al.* (2010) Weak kernels. *ECCC Report, TR10-005*.

Jiang,H. and Zhu,D. (2011) A 14/11-Approximation algorithm for sorting by short block-moves. To appear in *Science in China Series F*.

Yancopoulos,S. *et al.* (2005) Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, **21**, 3340–3346.