A Randomized FPT Approximation Algorithm for Maximum Alternating-Cycle Decomposition with Applications

Haitao Jiang[1], Lianrong Pu[1], Letu Qingge[2], David Sankoff[3], and Binhai Zhu[2(✉)]

[1] School of Computer Science and Technology, Shandong University, Jinan, China
htjiang@sdu.edu.cn
[2] Gianforte School of Computing, Montana State University, Bozeman, MT 59717-3880, USA
letu.qingge@msu.montana.edu, bhz@montana.edu
[3] Department of Mathematics and Statistics, University of Ottawa, Ottawa, Ont K1N 6N5, Canada
sankoff@uottawa.ca

Abstract. Comparing genomes in terms of gene order is a classical combinatorial optimization problem in computational biology. Some of the popular distances include translocation, reversal, and double-cut-and-join (abbreviated as DCJ), which have been extensively used while comparing two genomes. Let d_x, $x \in \{\text{translocation, reversal, DCJ}\}$, be the distance between two genomes such that one can be sorted/converted into the other using the minimum number of x-operations. All these problems are NP-hard when the genomes are unsigned. Computing d_x, $x \in \{\text{translocation, reversal, DCJ}\}$, between two unsigned genomes involves computing a proper alternating cycle decomposition of its breakpoint graph, which becomes the bottleneck for computing the genomic distance under almost all types of genome rearrangement operations and prohibits to obtain approximation factors better than 1.375 in polynomial time. In this paper, we devise an FPT (fixed-parameter tractable) approximation algorithm for computing the DCJ and translocation distances with an approximation factor $4/3+\varepsilon$, and the running time is $O^*(2^{d^*})$, where d^* represents the optimal DCJ or translocation distance. The algorithm is randomized and it succeeds with a high probability. This technique is based on a new randomized method to generate approximate maximum alternating cycle decomposition.

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,

L. Wang and D. Zhu (Eds.): COCOON 2018, LNCS 10976, pp. 26–38, 2018.
https://doi.org/10.1007/978-3-319-94776-1_3

have been proposed to handle gene order. The *double-cut-and-join* operation, introduced by Yancopoulos *et al.* [22], unifies all the classical operations. In the past, the rearrangement distance for signed genomes is well studied by single operations (like reversals) [15], combinations of operations (reversals, translocations, fusions and fissions) [16] and universal operations (double-cut-and-join) [2,22].

Unfortunately, as for unsigned genomes, almost all these problems are NP-hard. Then people resort to approximation algorithms. Christie devised a factor-1.5 approximation algorithm for sorting unsigned genomes by reversals [8], and the approximation factor was improved to 1.375 by Berman *et al.* in 2002 [4]. Cui *et al.* investigated the problem of sorting by unsigned translocations and proposed an algorithm with an approximation factor $1.5+\varepsilon$ [9]. This bound was improved to $1.408+\epsilon$ [17] and recently further to 1.375 [21]. The problem of Sorting by Transpositions was first studied by Bafna and Pevzner [1], who devised an 1.5-approximation algorithm which runs in quadratic time. The bound was improved to 1.375 by Elias and Hartman in 2006 [11]. As far as we know, the best polynomial-time approximation algorithms for the unsigned DCJ distance problem has a factor $1.408+\epsilon$ [7]. Among almost all these problems, a bottleneck to break the 1.375 barrier seems to be on decomposing the breakpoint graph (to be defined formally) into maximum (edge-disjoint) alternating-cycles. We make fundamental contributions in this paper on using FPT approximation algorithms. The design of FPT algorithms for genome rearrangement problems was started very recently. With the help of weak kernels, sorting unsigned genomes by either reversals, translocations or DCJs admits small weak kernels, hence are in FPT with a running time $O^*(4^k)$, where k is the solution value [18,19]. However, this algorithm is only practical for k bounded from above by around 20 to 25.

In this paper, we devise a new randomized algorithm for maximum alternating-cycle decomposition. Consequently, we design an FPT approximation algorithm for sorting unsigned genomes by DCJ operations (resp. by translocations), the approximation factor reaches $4/3+\varepsilon$, and the running time is $O^*(2^{d^*})$, where d^* represents the optimal DCJ distance (resp. translocation distance). The algorithm is randomized and it succeeds with a high probability.

2 Preliminaries

We first define the basics regarding gene, chromosome and genome. An unsigned gene is a sequence of DNA, which is 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 *ending-gene*. Gene g_i and g_j form an *adjacency* if they are consecutive in some chromosome. An adjacency (g_i, g_{i+1}) is *trivial* if it satisfies $|g_{i+1} - g_i| = 1$. A chromosome is *trivial* if every adjacency is trivial. A genome is *trivial* if all its chromosomes are trivial.

In the context of sorting genomes, the comparative order of the genes in the same chromosome does matter, but not the order of chromosomes and the direction of a whole chromosome, which implies that each chromosome can be viewed

in both directions. In the case of signed genomes, a chromosome $\langle g_i, g_{i+1}, \cdots, g_j \rangle$ is equivalent to $\langle -g_j, \cdots, -g_{i+1}, -g_i \rangle$; and in the case of unsigned genomes, a chromosome $\langle g_i, g_{i+1}, \cdots, g_j \rangle$ is equivalent to $\langle g_j, \cdots, g_{i+1}, g_i \rangle$.

2.1 Breakpoint Graph of Signed and Unsigned Genomes

Now, we recall the well-known tool for computing the genomic rearrangement distance, the *Breakpoint Graph*. Given signed genomes X and Y over the same set of genes (WLOG, assume that Y is trivial), the breakpoint graph $G_s(X, Y)$ can be obtained as follows: for each chromosome $S = [x_1, x_2, \ldots, x_{n_i}]$ of X, replace each x_i with an ordered pair $(l(x_i), r(x_i))$ of vertices. If x_i is positive, then $(l(x_i), r(x_i)) = (x_i^t, x_i^h)$; and if x_i is negative, then $(l(x_i), r(x_i)) = (x_i^h, x_i^t)$. If the genes x_i and x_{i+1} are adjacent in X, then we connect $r(x_i)$ and $l(x_{i+1})$ by a black edge in $G_s(X, Y)$. If the genes x_i and x_{i+1} are adjacent in Y, then we connect $r(x_i)$ and $l(x_{i+1})$ by a gray edge in $G_s(X, Y)$. Every vertex (except the ones at the two ends of a chromosome) in $G_s(X, Y)$ is incident to one black and one gray edge. Therefore, $G_s(X, Y)$ can be uniquely decomposed into cycles, on which the black edges and gray edges appear consecutively. A cycle containing exactly i black (gray) edges is called an *i-cycle*.

As for unsigned genomes, the breakpoint graph is a bit different. Given two unsigned genomes X and Y on the same set of n genes, the *Breakpoint Graph* $G_u(X, Y) = (V, E_b \cup E_g)$, where $|V| = n$ and each vertex in V corresponds to a gene, every adjacency in X forms a black edge belonging to E_b and every adjacency in Y forms a gray edge belonging to E_g. The breakpoint graph $G_u(X, Y)$ can be decomposed into a set of edge-disjoint cycles, denoted as $\mathbf{D}$, and on each cycle, the black edges and gray edges appear alternatively.

2.2 The Signed DCJ Distance Formula

Let b (resp. c) be the number of black edges (resp. cycles) in $G_s(X, Y)$. Yancopoulos *et al.* proved the following theorem [22].

Theorem 1. *Let $d_s(X, Y)$ be the (optimal) signed DCJ distance between X and Y. Then $d_s(X, Y) = b - c$.*

2.3 The UDCJ Problem

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 the DCJ operation on the breakpoint graph (for both signed and unsigned genomes) is as follows.

Definition 1. *The Double-Cut-and-Join operation acts on the Breakpoint Graph in the following four ways (Fig. 1):*

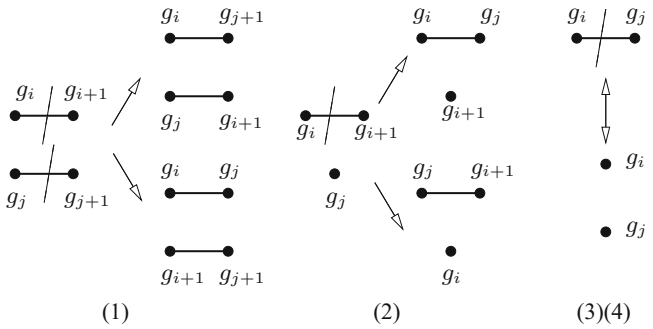

Fig. 1. The DCJ Operation.

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}.

We now formally formulate the problem to be investigated in this paper.

Sorting Unsigned Genomes by the DCJs (UDCJ):
Input: Two unsigned linear genomes X and Y, Y is trivial, and an integer k.
Question: Can X be converted into Y by a series of k DCJs $\rho_1, \rho_2, \cdots, \rho_k$.

The minimum k is the unsigned *DCJ distance* between X and Y.

Throughout this paper, we assume that the ending-gene sets of X and Y are the same, since the details to handle genomes with different ending-gene sets is not the main purpose of this paper.

Coming back to the technical details, since each ending-gene of X and Y is incident to only one black edge and one gray edge; and each of the rest genes is incident to exactly two black edges and two gray edges, the ways to decompose $G_u(X, Y)$ into cycles might not be unique. Caprara showed that computing a maximum alternating-cycle decomposition (MAX-ACD) of the breakpoint graph is NP-hard [5], which implies that UDCJ is also NP-hard. We comment that the best polynomial-time approximation for MAX-ACD only has a factor $1.4193 + \epsilon$ [20].

2.4 Converting Unsigned Genomes into Signed Ones

A natural way to solve UDCJ is to convert the unsigned genome into a signed one, then resort to the algorithm for computing the signed DCJ distance. But, how to convert an unsigned genome into a 'good' signed genome, which would result in a smaller DCJ distance? Once we have a cycle-decomposition $\mathbf{D}$ of

$G_u(X, Y)$, we can obtain two signed genomes $\bar{X}$ and $\bar{Y}$ by assigning a sign to each gene in X and Y such that $G_s(\bar{X}, \bar{Y}) = \mathbf{D}$.

As Y is trivial, we arrange all its chromosomes monotonously increasing, then assign all its genes positive. Therefore, all gray edges in $G_s(\bar{X}, \bar{Y})$ have the form $((x_i)^h, (x_i + 1)^t)$.

Next, we show how to assign a proper sign to each gene in X (to obtain $\bar{X}$). An ending-gene is positive if it lies at the same (i.e., both left or both right) ends of some chromosome in X and some chromosome in Y; otherwise, it is negative in X. For a non-ending gene x_i, according to the two gray edges, $((x_i)^h, (x_i + 1)^t)$ and $((x_i - 1)^h, (x_i)^t)$ in the cycle decomposition, we assign x_i positive if $((x_i - 1)^h, l(x_i))$ is a gray edge in the given cycle decomposition; if $((x_i - 1)^h, r(x_i))$ is a gray edge in the given cycle decomposition, then x_i is assigned a negative sign. See Fig. 2 for an example.

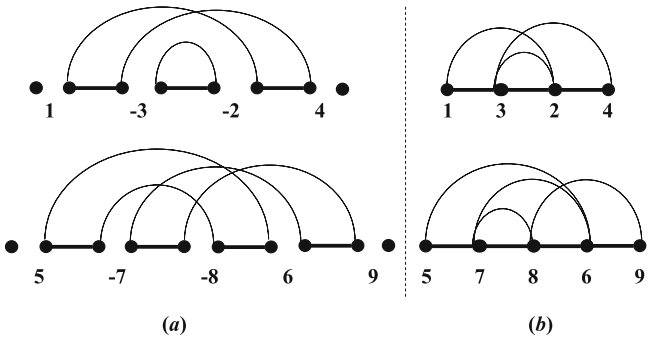

Fig. 2. (a) The breakpoint graph $G_s(\bar{X}, \bar{Y})$ for the signed case, where $\bar{X}$={$[1, -3, -2, 4]$, $[5, -7, -8, 6, 9]$} and $\bar{Y}$={[1,2,3,4], [5,6,7,8,9]}. (b) The breakpoint graph $G(X, Y)$ for the unsigned case, where X={[1,3,2,4], [5,7,8,6,9]} and Y={[1,2,3,4], [5,6,7,8,9]}. (a) is a cycle decomposition of (b).

Finally, a fixed-parameter tractable (FPT) algorithm for a decision problem Π with solution value k is an algorithm which solves the problem in $O(f(k)n^{O(1)}) = O^*(f(k))$ time, where f is any function only on k, n is the input size. FPT also stands for the set of problems which admit such an algorithm [10,12].

In summary, to solve UDCJ efficiently, we need to find a proper cycle decomposition of the breakpoint graph. We handle this NP-hard problem by designing an FPT approximation algorithm. This is the main content in the next section.

3 An FPT-time Approximation Algorithm

In this section, we present a factor-$(4/3 + \epsilon)$ FPT-approximation algorithm for UDCJ, which runs in $O^*(2^{d^*})$ time, where d^* is the optimal DCJ distance. We

try to decompose the breakpoint graph into enough number of small cycles (i.e., cycles containing 1, 2, and 3 black edges, formally called *1-cycles, 2-cycles* and *3-cycles* henceforth), with a new randomized method. The algorithm is randomized and it succeeds with a high probability.

3.1 General Sketch

Jiang *et al.* [19] showed that all the possible 1-cycles could be kept in the cycle decomposition by the following lemma.

Lemma 1. *There exists some optimal cycle decomposition containing all the existing 1-cycles.*

Hence, our cycle-decomposition will keep all the 1-cycles, and is always compared with an optimal cycle-decomposition which also keeps all the 1-cycles.

Let c_i^* be the number of i-cycles in some optimal cycle-decomposition, from the signed DCJ distance formula $d_s(X,Y) = b - \sum_{i\geq 1} c_i^*$. Since $b = \sum_{i\geq 1} i * c_i^*$, we have $\sum_{i\geq 3} c_i^* \leq \frac{1}{3}(b - c_1^* - 2c_2^*)$, that results in, $d_s(X,Y) \geq b - c_1^* - c_2^* - \frac{1}{3}(b - c_1^* - 2c_2^*) = \frac{2}{3}(b - c_1^*) - \frac{1}{3}c_2^* = \frac{2}{3}(b - c_1^* - \frac{1}{2}c_2^*)$, which implies that, to achieve an approximation factor of 1.5, it is sufficient to find a half number of 2-cycles of the optimal cycle-decomposition.

Moreover, if we can find an α portion of 2-cycles and a β portion of 2-cycles and 3-cycles, then use the bound, $d_s(X,Y) = b - \sum_{i\geq 1} c_i^* \geq b - c_1^* - c_2^* - c_3^* - \frac{1}{4}(b - c_1^* - 2c_2^* - 3c_3^*)$, together with the conditions $c_2 \geq \alpha * c_2^*$, $c_2 + c_3 \geq \beta * (c_2^* + c_3^*)$, and $d_{alg}(X,Y) \leq b - c_1^* - c_2 - c_3$, the approximation factor could become $max\{\frac{4}{3}, 2-\alpha, \frac{3-\beta}{2}, \frac{3\alpha-\beta-\alpha\beta}{2\alpha-\beta}\}$ [6].

Our idea is to find an α portion of 2-cycles, and a β portion of 2-cycles as well as a γ portion of 3-cycles. We will show that, ignoring some small constant ϵ, $\alpha \geq \frac{5}{6}$, $\beta \geq \frac{3}{5}$, and $\gamma \geq \frac{3}{5} * \frac{57}{64}$, which leads to an approximation 4/3.

Now we give the details of our algorithm. By cycle decomposition, we aim at finding small cycles, but searching cycles directly from the breakpoint graph will not guarantee enough number, that is because cycles in the breakpoint graph could possibly share some edges, and it is necessary to compute an independent set from them. Our main idea is to fix the sign of some genes, then find cycles from the partly decomposed breakpoint graph, so that we can obtain more cycles.

3.2 Finding 2-Cycles

Let V_2 be the set of vertices, each of which is involved in at least two 2-cycles in the breakpoint graph. (Following [6], the intersection graph of the 2-cycles has a maximum degree of 6. By the following random selection procedure, together with the enumeration of the signs of the selected vertices, we show that the intersection graph of these 'partial' 2-cycles has a maximum degree of 3.) We randomly choose $|V_2|/2$ vertices and enumerate all possible combinations of signs for them. Under each combination of sign assignment, the breakpoint graph could be partly decomposed. A *candidate* 2-cycle is a 2-cycle which could exist

subject to the current sign assignment, and at least one vertex has a sign fixed. Nonetheless, in the partly decomposed breakpoint graph, some of the candidate 2-cycles could share edges and could not co-exist in any cycle decomposition. Let C_{c2} be the set of candidate 2-cycles. Construct a conflict graph $G_{c2} = (C_{c2}, E_{c2})$, where each 2-cycle of C_{c2} corresponds to a vertex, and there is an edge between two vertices if and only if their corresponding 2-cycles share edges in the partly decomposed breakpoint graph. Thus, an independent set of G_{c2} represents the conflict-free 2-cycles we find.

Algorithm 1. Finding 2-cycles

1: Identify vertices of V_2.
2: Choose $|V_2|/2$ vertices randomly.
3: **for** each combination of sign assignment for these chosen vertices **do**
4: Construct the conflict graph G_{c2}.
5: Compute an approximate independent set I_{c2} of G_{c2}.
6: **end for**
7: Keep the 2-cycles corresponding to the maximum I_{c2}.

Lemma 2. *There exists an approximation algorithm with ratio* $\frac{5}{r+3} - \varepsilon$*, for any* $\varepsilon > 0$*, for the maximum independent set problem on a graph with maximum degree* r *(see reference* [3]*).*

3.3 Finding 2,3-Cycles

Let a 2,3-cycle be either a 2-cycle or a 3-cycle in the breakpoint graph. Let V_{23} be the set of vertices, each of which is involved in at least two 2,3-cycles in the breakpoint graph. We randomly choose $|V_{23}|/2$ vertices, enumerate all possible combinations of signs for them. Under each combination of sign assignment, the breakpoint graph has been partly decomposed. A *candidate* 2,3-cycle is a 2-cycle or a 3-cycle which could exist under the current sign assignment, and at least two vertices have their signs fixed. In the partly decomposed breakpoint graph, some of the 2-cycles and candidate 3-cycles could share edges and could not co-exist in any cycle decomposition. Each 2-cycle and candidate 3-cycles is composed of paths, where each path is composed of black edges, or gray edges, or composed of black edges and gray edges appearing alternatively. A candidate 2,3-cycles is composed of at most four such paths. We view paths as elements, and candidate 2,3-cycles as sets of elements. We can construct a set packing system S_{c23}, whose basic elements are the paths and each candidate 2,3-cycles is a subset of at most four elements. Thus, a set packing could be the 2,3-cycles we find.

Lemma 3. *There is a ratio* $\frac{3}{p+1} - \varepsilon$ *approximation algorithm, for any* $\varepsilon > 0$*, for the maximum set packing problem with set size at most* p *and set degree bounded (see reference* [13]*).*

Algorithm 2. Finding 2,3-cycles

1: Identify vertices of V_{23}.
2: Choose $|V_{23}|/2$ vertices randomly.
3: **for** each combination of sign assignment for these chosen vertices **do**
4: Construct the set packing system S_{c23}.
5: Compute an approximate set packing P_{c23}.
6: **end for**
7: Keep the 2,3-cycles corresponding to the maximum P_{c23}.

Algorithm 3. UDCJ-Final

1: Run Algorithm 1 n times, among the computed I_{c2}'s, pick the largest one $I_{c2}^{\max}$.
2: Run Algorithm 2 n times, among the computed P_{c23}'s, pick the largest one $P_{c23}^{\max}$.
3: If $|I_{c2}^{\max}| \geq |P_{c23}^{\max}|$, then keep the 2-cycles corresponding to $I_{c2}^{\max}$,
4: Otherwise, keep the 2,3-cycles corresponding to $P_{c23}^{\max}$.
5: Arbitrarily assign signs to the rest of genes so that every gene has a sign.
6: Compute the DCJ distance between the signed genomes.
7: Simulate the signed DCJ sorting process to the original unsigned genomes.

4 Performance Analysis

4.1 The Approximation Factor

As aforementioned, the approximation factor performance is determined by the number of 2-cycles and 3-cycles we have found.

Lemma 4. *G_{c2} is a graph with maximum degree 3.*

Proof. Omitted due to space constraint. □

Corollary 1. *I_{c2} is an $\frac{5}{6} - \varepsilon$ approximation for the maximum independent set of G_{c2}.*

Lemma 5. *S_{c23} is a set packing system with set size at most 4 and set degree bounded.*

Proof. Each 3-cycle has exactly 6 vertices, since at least two of them have a fixed sign, each fixed-sign vertex brings a fixed connection of a black edge and a gray edge. Then each 3-cycle has at most four undetermined connections, i.e., it is composed of at most four paths.

Now we show that each path can be shared by at most 4 such 2,3-cycles. Note that if three paths of such a 3-cycle are fixed, then the 2,3-cycle is obtained. Standing on an ending vertex of a path, we have at most two choices. After any choice, the path is extended, i.e., two paths are connected together. To form 2,3-cycles, we have twice opportunities to make choices, which would result in at most four 2,3-cycles. Thus, each subset could share elements with at most $4\times(4-1)=12$ other subsets. The set degree is bounded. □

Corollary 2. *P_{c23} is an $\frac{3}{5} - \varepsilon$ approximation for the maximum set packing of S_{c23}.*

Next, we bound the number of vertices in G_{c2} and the number of subsets in S_{c23}. Let c_i^* be the number of i-cycles in the optimal cycle decomposition.

Lemma 6. *With probability $1 - \frac{1}{e^{O(n)}}$, the maximum independent set of one G_{c2} at Step 1 in Algorithm 3 has a size greater than $(1-\delta)\frac{15}{16}c_2^*$ for any $0 < \delta < 1$.*

Proof. We show that each 2-cycle of the optimal cycle decomposition has a probability of $\frac{15}{16}$ to fall into G_{c2}. If the 2-cycle has a vertex with a fixed sign, then it will surely become a candidate 2-cycle. If all its vertices do not have a fixed sign, then they are all in V_2. The probability that none of them is chosen is $\frac{C_{|v_2|-4}^{|v_2|/2}}{C_{|v_2|}^{|v_2|/2}} \leq \frac{1}{16}$. Hence, we can conclude that, with probability $\frac{15}{16}$, we have $\frac{15}{16}$ portion of 2-cycles of the optimal cycle decomposition in G_{c2}; moreover, they form an independent set of G_{c2}. If we view X_i as a random variable to put a 2-cycle of the optimal cycle decomposition into G_{c2} and define $X = \sum_i X_i$, then $E[X] = \mu = \frac{15}{16}c_2^*$.

By Chernoff bounds, for any $0 < \delta < 1$,

$$P[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2\mu}{2}},$$

which means $P[X \leq (1-\delta)\mu] \leq \frac{1}{e^{O(1)}}$ when μ is only a constant. For Algorithm 3, as we repeat Algorithm 1 n times, the probablity that the MIS of all the n G_{c2}'s has a size at most $(1-\delta)\frac{15}{16}c_2^*$ is at most $(\frac{1}{e^{O(1)}})^n = \frac{1}{e^{O(n)}}$.

Therefore, with probability $1 - \frac{1}{e^{O(n)}}$, the MIS of one of the G_{c2} at Step 1 in Algorithm 3, has a size greater than $(1-\delta)\frac{15}{16}c_2^*$ for any $0 < \delta < 1$. □

As δ could be arbitrarily small, we would use this size as $\frac{15}{16}c_2^*$ in the proof of Theorem 2.

Lemma 7. *With probability $1 - \frac{1}{e^{O(n)}}$, the maximum set packing of one of the S_{c23} at Step 2 in Algorithm 3 has a size greater than $(1-\delta)(c_2^* + \frac{57}{64}c_3^*)$ for any $0 < \delta < 1$.*

Proof. We show that each 3-cycle of the optimal cycle decomposition has a probability of $\frac{57}{64}$ to be a candidate 3-cycle. If the 3-cycle has two vertices whose signs are fixed, then it will surely become a candidate 3-cycle. If the 3-cycle has exactly one vertex whose sign is fixed, then the other five vertices do not have a fixed sign, so they are all in V_{23}. The probability that none of them is chosen is $\frac{C_{|v_{23}|-5}^{|v_{23}|/2}}{C_{|v_{23}|}^{|v_{23}|/2}} \leq \frac{1}{32}$. If all its vertices do not have a fixed sign, then they are all in V_{23}. The probability that none of them is chosen is $\frac{C_{|v_{23}|-6}^{|v_{23}|/2}}{C_{|v_{23}|}^{|v_{23}|/2}} \leq \frac{1}{64}$. The probability that exact one of them is chosen is $6 \times \frac{C_{|v_{23}|-6}^{|v_{23}|/2}}{C_{|v_{23}|}^{|v_{23}|/2}} \leq \frac{6}{64}$. Hence, we can

conclude that an expected $1 - \frac{1}{64} - \frac{6}{64} = \frac{57}{64}$ portion of 3-cycles of the optimal cycle decomposition are in S_{c23}; moreover, together with the c_2^* 2-cycles of the optimal cycle decomposition, they form a set packing of S_{c23}. Note that a 2-cycle has only 4 edges, by Lemma 5, all of the c_2^* 2-cycles will be in the set packing solution.

By an argument similar to Lemma 6, we conclude that, with probability $1 - \frac{1}{e^{O(n)}}$, the maximum set packing one of the S_{c23}'s at Step 2 of Algorithm 3, has size greater than $(1-\delta)(c_2^* + \frac{57}{64}c_3^*)$ for any $0 < \delta < 1$. □

Again, as δ could be arbitrarily small, we would use this size as $c_2^* + \frac{57}{64}c_3^*$ in the proof of Theorem 2.

Theorem 2. *With probability* $1 - \frac{1}{e^{O(n)}}$, *Algorithm 3 approximates the DCJ distance within a factor 4/3+ε.*

Proof. Let c_i be the number of i-cycles computed by Algorithm 3, c_i^* be the number of i-cycles in the optimal cycle decomposition. Let d^* and d be the optimal DCJ distance and approximated DCJ distance respectively. Let the approximation factor be ρ, where $\frac{4}{3} \leq \rho \leq \frac{3}{2}$.

We have, $d^* = b - c_1^* - c_2^* - c_3^* - \sum_{i \geq 4} c_i^*$ Since, $\sum_{i \geq 4} c_i^* \leq (b - c_1^* - 2c_2^* - 3c_3^*)/4$, and $b - c_1^* \geq 2c_2^* + 3c_3^*$, then,

$$\begin{aligned} d^* &\geq b - c_1^* - c_2^* - c_3^* - (b - c_1^* - 2c_2^* - 3c_3^*)/4 \\ &= \frac{3}{4}(b - c_1^*) - \frac{1}{2}c_2^* - \frac{1}{4}c_3^* \\ &= \frac{1}{\rho}((b - c_1^*) + (\frac{3}{4}\rho - 1)(b - c_1^*) - \frac{1}{2}\rho c_2^* - \frac{1}{4}\rho c_3^*) \\ &\geq \frac{1}{\rho}((b - c_1^*) + (\frac{3}{4}\rho - 1)(2c_2^* + 3c_3^*) - \frac{1}{2}\rho c_2^* - \frac{1}{4}\rho c_3^*) \\ &= \frac{1}{\rho}(b - c_1^* - (2-\rho)c_2^* - (3-2\rho)c_3^*) \end{aligned} \tag{1}$$

This implies that, if the number of cycles we found satisfy $c_2 \geq (2-\rho)c_2^*$ and $c_3 \geq (3-2\rho)c_3^*$, then the approximated DCJ distance computed by our algorithm satisfies $d \leq b - c_1^* - c_2 - c_3$; consequently, the approximation factor reaches to ρ.

From Corollary 1 and Lemma 6, with high probability, we have

$$|I_{c2}^{\max}| \geq (\frac{5}{6} - \varepsilon) \times \frac{15}{16}c_2^*. \tag{2}$$

From Corollary 2 and Lemma 7, with high probability, we have

$$|P_{c23}^{\max}| \geq (\frac{3}{5} - \varepsilon') \times (c_2^* + \frac{57}{64}c_3^*). \tag{3}$$

Now we show that, by a balanced analysis, at least one of $|I_{c2}^{\max}|$ and $|P_{c23}^{\max}|$ are greater than $(2-\rho)c_2^* + (3-2\rho)c_3^*$.

We have two cases: either (I) $(\frac{5}{6}-\varepsilon)\times\frac{15}{16}c_2^* \geq (\frac{3}{5}-\varepsilon')\times(c_2^*+\frac{57}{64}c_3^*)$ or (II) not.

In case (I): $(\frac{5}{6}-\varepsilon)\times\frac{15}{16}c_2^* \geq (\frac{3}{5}-\varepsilon')\times(c_2^*+\frac{57}{64}c_3^*)$. Since ε and ε' are very small comparing with the other constants, for the sake of computation simplification, we ignore them here. Then, $(\frac{25}{32}-\frac{3}{5})c_2^* \geq \frac{3}{5}\times\frac{57}{64}c_3^*$; that is, $c_2^* \geq \frac{171}{58}c_3^*$. Therefore, if $\frac{25}{32}c_2^* \geq (2-\rho)c_2^*+(3-2\rho)c_3^*$, we are done. Solving this inequality,

$$(2-\rho)c_2^*+(3-2\rho)c_3^* \leq (2-\rho)c_2^*+(3-2\rho)\times\frac{58}{171}c_2^* \leq \frac{25}{32}c_2^*,$$

it holds provided that $\rho \geq 1.332$. Together with the constraint that $\rho \geq \frac{4}{3}$, we choose $\rho = \frac{4}{3}$.

In case (II): $(\frac{5}{6}-\varepsilon)\times\frac{15}{16}c_2^* < (\frac{3}{5}-\varepsilon')\times(c_2^*+\frac{57}{64}c_3^*)$. Similarly, we have $c_2^* < \frac{171}{58}c_3^*$. If $\frac{3}{5}\times(c_2^*+\frac{57}{64}c_3^*) \geq (2-\rho)c_2^*+(3-2\rho)c_3^*$, then we are done. Hence we need to prove,

$$(\frac{3}{5}\times\frac{57}{64}-3+2\rho)c_3^* \geq (2-\frac{3}{5}-\rho)c_2^*.$$

Again, this inequality holds when $\rho \geq 1.332$. Together with the constraint that $\rho \geq \frac{4}{3}$, we have $\rho = \frac{4}{3}$. □

4.2 The Time Complexity

Theorem 3. *The time complexity of Algorithm 3 is $O^*(2^{d^*})$, where d^* is the optimal DCJ distance.*

Proof. The most time-consuming parts are enumerating all possible sign combination of V_2 and V_{23}. $d^* \geq b-c_1^*-c_2^*-c_3^*$ and $c_2^*+c_3^* \leq (b-c_1^*)/2$, then, $d^* \geq (b-c_1^*)/2$. Each vertex of V_2 or V_{23} is connected to two black edges, while each black edge has at most two unsigned genes as its endpoints, which means that the number of unsigned genes of V_2 or V_{23} is smaller than that of black edges, e.g., $|V_2| \leq b-c_1^* \leq 2d^*$ and $|V_{23}| \leq b-c_1^* \leq 2d^*$.

Hence while we choose a half of $|V_2|$ vertices or a half of $|V_{23}|$ vertices, and enumerating all their combination of signs, the time complexity is at most $O^*(2^{d^*})$. □

In fact, we could extend our method to Sorting by Translocations [9,14,17,21]. We summarize the result as follows. The details will be given in the full version.

Theorem 4. *With probability $1-\frac{1}{e^{O(n)}}$, there is an FPT algorithm which approximates the translocation distance within a factor $\alpha = 4/3+\varepsilon$.*

5 Concluding Remarks

We design a factor $4/3+\epsilon$ FPT-approximation algorithm for the DCJ distance, improving the previous (polynomial-time approximation) factor of $1.408+\epsilon$. The

algorithm is randomized and it succeeds with a high probability. The running time is bounded by $O^*(2^k)$; in fact, by $O^*(2^x)$ where $x \leq k$, which makes it practical for k at least as large as 40–50. The exact FPT algorithm for the same problem takes $O^*(4^k)$ time, which is only practical for k bounded by 20–25 from above. Our algorithm involves a new randomized method to decompose the breakpoint graph into the maximum number of alternating-cycles and can be used to improve the approximation factor for Sorting by Translocations — again in a similar FPT time, which admits a factor-1.375 (polynomial-time) approximation and uses maximum alternating-cycle decomposition as a subroutine.

For Sorting by Reversals, note that special care must be taken as in the optimal solution the 1-cycles might not be all kept. For instance, for the sequence $S = \langle 3, 4, 1, 2\rangle$, if we keep the 1-cycles (3,4) and (1,2) then three reversals are needed to sort S into $\langle 1, 2, 3, 4\rangle$. On the other hand we could sort $S = \langle 3, 4, 1, 2\rangle$ into $\langle 1, 2, 3, 4\rangle$ using two reversals: $\langle 3, \underline{4, 1, 2}\rangle \rightarrow \langle \underline{3, 2, 1}, 4\rangle \rightarrow \langle 1, 2, 3, 4\rangle$.

A related open question is whether one can design an FPT-approximation algorithm with a factor better than 1.375 for the problem of Sorting by Transpositions. Note that the technique of giving signs to some genes does not seem to work for this problem.

Acknowledgments. This research is partially supported by NSF of China under project 61472222 and 61628207.

References

1. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM J. Discret. Math. **11**(2), 224–240 (1998)
2. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS, vol. 4175, pp. 163–173. Springer, Heidelberg (2006). https://doi.org/10.1007/11851561_16
3. Berman, P., Fürer, M.: Approximating maximum independent set in bounded degree graphs. In: Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1994), pp. 365–371 (1994)
4. Berman, P., Hannenhalli, S., Karpinski, M.: 1.375-approximation algorithm for sorting by reversals. In: Möhring, R., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 200–210. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45749-6_21
5. Caprara, A.: Sorting permutations by reversals and Eulerian cycle decompositions. SIAM J. Discret. Math. **12**(1), 91–110 (1999)
6. Caprara, A., Rizzi, R.: Improved approximation for breakpoint graph decomposition and sorting by reversals. J. Comb. Optim. **6**(2), 157–182 (2002)
7. Chen, X., Sun, R., Yu, J.: Approximating the double-cut-and-join distance between unsigned genomes. BMC Bioinform. **12**(S-9), S17 (2011)
8. Christie, D.A.: A 3/2-approximation algorithm for sorting by reversals. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998), pp. 244–252 (1998)

9. Cui, Y., Wang, L., Zhu, D., Liu, X.: A (1.5 + ϵ)-approximation algorithm for unsigned translocation distance. IEEE/ACM Trans. Comput. Biol. Bioinform. **5**(1), 56–66 (2008)
10. Downey, R., Fellows, M.: Parameterized Complexity. Springer, Heidelberg (1999). https://doi.org/10.1007/978-1-4612-0515-9
11. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**(4), 369–379 (2006)
12. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29953-X
13. Halldórsson, M.M.: Approximating discrete collections via local improvements. In: Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1995), pp. 160–169 (1995)
14. Hannenhalli, S.: Polynomial-time algorithm for computing translocation distance between genomes. Discret. Appl. Math. **71**(1–3), 137–151 (1996)
15. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM **46**(1), 1–27 (1999)
16. Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1995), pp. 581–589 (1995)
17. Jiang, H., Wang, L., Zhu, B., Zhu, D.: A factor-(1.408 + ϵ) approximation for sorting unsigned genomes by reciprocal translocations. Theor. Comput. Sci. **607**, 166–180 (2015)
18. Jiang, H., Zhang, C., Zhu, B.: Weak Kernels. ECCC Report, TR10-005, September 2010
19. Jiang, H., Zhu, B., Zhu, D.: Algorithms for sorting unsigned linear genomes by the DCJ operations. Bioinformatics **27**(3), 311–316 (2011)
20. Lin, G., Jiang, T.: A further improved approximation algorithm for breakpoint graph decomposition. J. Comb. Optim. **8**(2), 183–194 (2004)
21. Pu, L., Zhu, D., Jiang, H.: A new approximation algorithm for unsigned translocation sorting. In: Frith, M., Storm Pedersen, C.N. (eds.) WABI 2016. LNCS, vol. 9838, pp. 269–280. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43681-4_22
22. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**, 3340–3346 (2005)