Assembling contigs in draft genomes using reversals and block-interchanges

Chi-Long Li, Kun-Tze Chen, Chin Lung Lu*

From RECOMB-seq: Third Annual Recomb Satellite Workshop on Massively Parallel Sequencing
Beijing, China. 11-12 April 2013

Abstract

The techniques of next generation sequencing allow an increasing number of draft genomes to be produced rapidly in a decreasing cost. However, these draft genomes usually are just partially sequenced as collections of unassembled contigs, which cannot be used directly by currently existing algorithms for studying their genome rearrangements and phylogeny reconstruction. In this work, we study the one-sided block (or contig) ordering problem with weighted reversal and block-interchange distance. Given a partially assembled genome \( \pi \) and a completely assembled genome \( s \), the problem is to find an optimal ordering to assemble (i.e., order and orient) the contigs of \( \pi \) such that the rearrangement distance measured by reversals and block-interchanges (also called generalized transpositions) with the weight ratio 1:2 between the assembled contigs of \( \pi \) and \( s \) is minimized. In addition to genome rearrangements and phylogeny reconstruction, the one-sided block ordering problem particularly has a useful application in genome resequencing, because its algorithms can be used to assemble the contigs of a draft genome \( \pi \) based on a reference genome \( s \). By using permutation groups, we design an efficient algorithm to solve this one-sided block ordering problem in \( O(\delta n) \) time, where \( n \) is the number of genes or markers and \( \delta \) is the number of used reversals and block-interchanges. We also show that the assembly of the partially assembled genome can be done in \( O(n) \) time and its weighted rearrangement distance from the completely assembled genome can be calculated in advance in \( O(n) \) time. Finally, we have implemented our algorithm into a program and used some simulated datasets to compare its accuracy performance to a currently existing similar tool, called SIS that was implemented by a heuristic algorithm that considers only reversals, on assembling the contigs in draft genomes based on their reference genomes. Our experimental results have shown that the accuracy performance of our program is better than that of SIS, when the number of reversals and transpositions involved in the rearrangement events between the complete genomes of \( \pi \) and \( s \) is increased. In particular, if there are more transpositions involved in the rearrangement events, then the gap of accuracy performance between our program and SIS is increasing.

Background

The techniques of next generation sequencing have greatly advanced in the past decade [1-3], which allows an increasing number of draft genomes to be produced rapidly in a decreasing cost. Usually, these draft genomes are partially sequenced, leading to their published genomes as collections of unassembled contigs (short for contiguous fragments). These draft genomes in contig form, however, can not be used immediately in some bioinformatics applications, such as the study of genome rearrangements, which requires the completely assembled genomes to calculate their rearrangement distances [4]. To adequately address this issue, Gaul and Blanchette [5] introduced and studied the so-called block ordering problem defined as follows. Given two partially assembled genomes, with each representing as an unordered set of blocks, the block ordering problem is to assemble (i.e., order and orient) the blocks of the two genomes such that the distance of genome rearrangements between the two assembled genomes is minimized. The blocks mentioned above are the contigs,
each of which can be represented by an ordered list of genes or markers. In their work [5], Gaul and Blanchette proposed a linear-time algorithm to solve the block ordering problem if the problem is further simplified to maximize the number of cycles in the breakpoint graph corresponding to the assembled genomes. The rationale behind this modification is totally based on a result obtained by Bourque and Pevzner [6], showing that the reversal distance between two assembled genomes can be approximated well by maximizing the number of cycles in their corresponding breakpoint graph. Actually, in addition to the number of cycles, the number of hurdles, as well as the presence of a fortress or not, is also important and needed for determining the actual reversal distance [7]. Therefore, it is still a challenge to efficiently solve the block ordering problem by optimizing the true rearrangement distance.

In the literature, many different kinds of genome rearrangements have been extensively studied [4], such as reversal (also called inversion), transposition and block-interchange (also called generalized transposition), translocation, fusion and fission. Reversal affects a segment on a chromosome by reversing this segment as well as exchanging its strands. Transposition rearranges a chromosome by interchanging its two adjacent and nonoverlapping segments. Block-interchange is a generalized transposition that exchanges two nonoverlapping but not necessarily adjacent segments on a chromosome. Translocation acts on two chromosomes by exchanging their the end fragments. Fusion is a special translocation that joins two chromosomes into one and fission is also a special translocation that splits a chromosome into two. In this study, we consider a variant of the block ordering problem, in which one of the two input genomes is still partially assembled but the other is completely assembled, with optimizing the genome rearrangement distance measured by weighted reversals and block-interchanges, whose weights are 1 and 2, respectively. For distinguishing this special block ordering problem from the original one, we call it as one-sided block (or contig) ordering problem. In fact, an efficient algorithm to solve the one-sided block ordering problem has a useful application in genome resequencing [8,9], because the reference genome for resequencing organisms can serve as the completely assembled genome in the one-sided block ordering problem and the contigs of partially assembled resequencing genome can then be assembled together into one or several scaffolds based on the reference genome. From this respect, the one-sided block ordering problem can be considered as a kind of contig scaffolding (or assembly) problem that aims to use genome rearrangements to create contig scaffolds for a draft genome based on a reference genome.

Currently, several contig scaffolding tools based on the reference genomes have been developed, such as Projector 2 [10], OSLay [11], ABACAS [12], Mauve Aligner [13], fillScaffolds [14], r2cat [15] and SIS [16]. Among these contig scaffolding tools, both SIS and fillScaffolds use the concept of genome rearrangements to generate contig scaffolds for a draft genome. SIS deals with only reversals, while in addition to reversals, fillScaffolds considers other rearrangements, such as transpositions and translocations (including fissions and fusions). Basically, SIS was dedicated to creating the contig scaffolds for prokaryotic draft genomes by heuristically searching for their inversion signatures, where an inversion signature is a pair of adjacent genes or markers appearing along a contig such that they form a breakpoint and are also located in different transcriptional strands. As for fillScaffolds, it used the traditional technique of breakpoint graphs to assemble the contigs of draft genomes. In the study by Dias and colleagues [16], they have used real prokaryotic draft genomes to demonstrate that SIS had the best overall accuracy performance when compared to the other tools we mentioned above.

In this study, we utilize permutation groups in algebra, instead of the breakpoint graphs used by Gaul and Blanchette [5], to design an efficient algorithm, whose time complexity is $O(\delta n)$, for solving the one-sided block ordering problem with weighted reversal and block-interchange distance, where $n$ is the number of genes or markers and $\delta$ is the number of reversals and block-interchanges used to transform the assembly of the partially assembled genome (i.e., draft genome) into the completely assembled genome (i.e., reference genome). In particular, we also show that the assembly of the partially assembled genome can be done in $O(n)$ time and its weighted reversal and block-interchange distance from the completely assembled genome can be calculated in advance in $O(n)$ time. In addition, we have implemented our algorithm into a program and used some simulated datasets to compare its accuracy performance to SIS on assembling the contigs in the draft genomes based on their reference genomes. Our experimental results have shown that the averaged normalized contig mis-join errors of our program are lower than those of SIS, when the number of reversals and transpositions involved in the rearrangement events between the complete genomes of the partially and completely assembled organisms is increased. In particular, if there are more transpositions involved in the rearrangement events, then the gap of accuracy performance between our program and SIS is increasing.

**Preliminaries**

**One-sided block ordering problem**

In the following, we dedicate ourselves to linear, unichromosomal genomes. With a slight modification, however, our algorithmic result can still apply to circular,
uni-chromosomal genomes, or to multi-chromosomal genomes with linear or circular chromosomes in a chromosome-by-chromosome manner. Once completely assembled, a uni-chromosomal genome can be represented by a signed permutation of \( n \) integers between 1 and \( n \), with each integer representing a gene or marker and its associated sign indicating the strandedness of the corresponding gene or marker. If the genome is partially assembled, then it will be represented by an unordered set of blocks, where a block \( B \) of size \( k \), denoted by \( B = \{b_1, b_2, \ldots, b_k\} \), is an ordered list of \( k \) signed integers. Let \( B = [-b_k, -b_{k-1}, \ldots, -b_1] \) denote the reverse of \( B \). Given an unordered set of \( m \) blocks, say \( B = \{B_1, B_2, \ldots, B_m\} \), corresponding to a partially assembled genome, an ordering (or assembly) of \( B \) is an ordered list of \( m \) blocks in which each block \( B_i \) or its reverse \( \overline{B_i} \) appears exactly once, where \( 1 \leq i \leq m \). For instance, suppose that \( B = \{B_1, B_2, B_3\} = \{(1, 4), (5, 2), (5, 6)\} \). Then \( (B_1, B_2, B_3) = ((1, 4), [5, 6], [3, 2]) \) and \( (B_1, B_2, B_3) = ([1, 4], [-5, 6], [3, 2]) \) are two orderings of \( B \). Basically, each ordering of \( B \) induces (or defines) a signed permutation of size \( n \), which is obtained by concatenating the blocks in this ordered list. For instance, the ordering \( (B_1, B_2, B_3) \) in the above exemplified \( B \) induces the signed permutation \((1, 4, -5, 6, 3, 2)\), which simply is denoted by \( B_1 \otimes B_2 \otimes B_3 \). Clearly, the permutation induced by an ordering of \( B \) corresponds to an assembly of the blocks in \( B \). Now, the one-sided block ordering problem we study in this paper is formally defined as follows:

**One-sided block ordering problem with reversal and block-interchange distance**

**Input:** A partially assembled genome \( \pi \) and a completely assembled genome \( \sigma \).

**Output:** Find an ordering of \( \pi \) such that the rearrangement distance measured by reversals and block-interchanges with the weight ratio 1:2 between the permutation induced by the ordering of \( \pi \) and \( \sigma \) is minimized.

As discussed in our previous study [17], it is biologically meaningful to assign twice the weight to block-interchanges than to reversals, due to the observation from the biological data that transpositions occur with about half the frequency of reversals [18].

**Permutation groups**

Permutation groups have been proven to be a very useful tool in the studies of genome rearrangements [17]. Below, we recall some useful definitions, notations and properties borrowed form our previous work [17]. Basically, given a set \( E = \{1, 2, \ldots, n\} \), a permutation is defined to be a one-to-one function from \( E \) into itself and usually expressed as a product of cycles in the study of genome rearrangements. For instance, \( \pi = (1)(3, 2) \) is a product of two cycles to represent a permutation of \( E = \{1, 2, 3\} \) and means that \( \pi(1) = 1, \pi(2) = 3 \) and \( \pi(3) = 2 \). The elements in a cycle can be arranged in any cyclic order and hence the cycle \((3, 2)\) in the permutation \( \pi \) exemplified above can be rewritten as \((2, 3)\). Moreover, if the cycles in a permutation are all disjoint (i.e., no common element in any two cycles), then the product of these cycles is called the cycle decomposition of the permutation. In fact, a permutation in the cycle decomposition can be used to model a genome containing several circular chromosomes, with each disjoint cycle representing a circular chromosome. Notice that in the rest of this article, we say “cycle in a permutation” to mean “cycle in the cycle decomposition of this permutation” for simplicity, unless otherwise specified. A cycle with \( k \) elements is further called a \( k \)-cycle. In convention, the 1-cycles in a permutation are not written explicitly since their elements are fixed in the permutation. For instance, the above exemplified permutation \( \pi \) can be written as \( \pi = (2, 3) \). If the cycles in a permutation are all 1-cycles, then this permutation is called an identify permutation and denoted by \( 1 \).

Suppose that \( \alpha \) and \( \beta \) are two permutations of \( E \). Then their product \( \alpha \beta \), also called their composition, defines a permutation of \( E \) satisfying \( \alpha \beta(x) = \alpha(\beta(x)) \) for all \( x \in E \). If both \( \alpha \) and \( \beta \) are disjoint, then \( \alpha \beta = \beta \alpha \). If \( \alpha \beta = 1 \), then \( \alpha \) is called the inverse of \( \beta \), denoted by \( \beta^{-1} \), and vice versa.

Moreover, the conjugation of \( \beta \) by \( \alpha \), denoted by \( \alpha \cdot \beta \), is defined to be the permutation \( \alpha \beta \alpha^{-1} \). It can be verified that if \( y = \beta(x) \), then \( \alpha(y) = \alpha \beta \alpha^{-1} \alpha(x) = \alpha \cdot \beta \alpha \). Hence, \( \alpha \cdot \beta \) can be obtained from \( \beta \) by just changing its element \( x \) with \( \alpha(x) \). In other words, if \( \beta = \langle b_1, b_2, \ldots, b_k \rangle \), then \( \alpha \cdot \beta = \langle \alpha(b_1), \alpha(b_2), \ldots, \alpha(b_k) \rangle \).

It is a fact that every permutation can be expressed into a product of 2-cycles, in which 1-cycles are written implicitly. Given a permutation \( \alpha \) of \( E \), its norm, denoted by \( ||\alpha|| \), is defined to be the minimum number, say \( k \), such that \( \alpha \) can be expressed as a product of \( k \) 2-cycles. In the cycle decomposition of \( \alpha \), let \( n_k(\alpha) \) denote the number of its disjoint cycles, notably including the 1-cycles not written explicitly. Given two permutations \( \alpha \) and \( \beta \) of \( E \), \( \alpha \) is said to divide \( \beta \), denoted by \( \alpha | \beta \), if and only if \( ||\beta|-1|| \leq ||\beta||-||\alpha|| \). In our previous work [17], it has been shown that \( ||\alpha|| = |E| - n_k(\alpha) \) and for any \( k \) elements in \( E \), say \( a_1, a_2, \ldots, a_k \), they all appear in a cycle of \( \alpha \) in the ordering of \( a_1, a_2, \ldots, a_k \) if and only if \( \langle a_1, a_2, \ldots, a_k \rangle \mid |\alpha| \).

Let \( \alpha = \langle a_1, a_2 \rangle \) be a 2-cycle and \( \beta \) be an arbitrary permutation of \( E \). If \( \alpha \beta \), that is, both \( a_1 \) and \( a_2 \) appear in the same cycle of \( \beta \), then the composition \( \alpha \beta \), as well as \( \beta \alpha \), has the effect of fission by breaking this cycle into two smaller cycles. For instance, let \( \alpha = (1, 3) \) and \( \beta = (1, 2, 3, 4) \). Then \( \alpha \beta \), since both 1 and 3 are in the
cycle (1, 2, 3, 4), and $\alpha\beta = (1, 2)(3, 4)$ and $\beta\alpha = (4, 1)$ (2, 3). On the other hand, if $\alpha \not\perp \beta$, that is, $a_1$ and $a_2$ appear in different cycles of $\beta$, then $\alpha\beta$, as well as $\beta\alpha$, has the effect of fusion by joining the two cycles into a bigger cycle. For example, if $\alpha = (1, 3)$ and $\beta = (1, 2)(3, 4)$, then $\alpha \not\perp \beta$ and, as a result, $\alpha\beta = (1, 2, 3, 4)$ and $\beta\alpha = (2, 1, 4, 3)$.

A model for representing DNA molecules

As mentioned before, a permutation in the form of the cycle decomposition can be used to model a genome containing multiple chromosomes (or a chromosome with multiple contigs), with each cycle representing a chromosome (or contig). To facilitate modelling the rearrangement of reversals using the permutation groups, however, we need to use two cycles to represent a chromosome, with one cycle representing a strand of the chromosome and the other representing the complementary strand. For this purpose, we first let $E = \{-1, 1, -2, 2, ..., -n, n\}$ and $\Gamma = (1, -1)(2, -2)...(n, -n)$. We then use an admissible cycle, which is a cycle containing no $i$ and its opposite $-i$ simultaneously for some $i \in E$, to represent a chromosomal strand, say $\pi'$, and use $\pi'' = \Gamma \cdot (\pi')^{-1}$, which is the reverse complement of $\pi''$, to represent the opposite strand of $\pi'$. As demonstrated in our previous work [17], it is useful to represent a double stranded chromosome $\pi$ by the product of its two strands $\pi'$ and $\pi''$, that is, $\pi = \pi'\pi'' = \pi''\pi'$, because a reversal (respectively, block-interchange) acting on this DNA molecule can be mimicked by multiplying two (respectively, four) 2-cycles with $\pi$, as described in the following lemmas.

Lemma 1 ([17]) Let $\pi = \pi'\pi''$ denote a double stranded DNA and let $x$ and $y$ be two elements in $E$. If $(x, y) \not\perp \pi$, that is, $x$ and $y$ are in the different strands of $\pi$, then the effect of $(\pi')(y), (\pi'')(x), (x, y)$ is a reversal acting on $\pi$.

Lemma 2 ([17]) Let $\pi = \pi'\pi''$ denote a double stranded DNA and let $u, v, x$ and $y$ be four elements in $E$. If $(u, v, y, v)|\pi|$, that is, $u$, $v$, $x$, and $v$ appear in the same strand of $\pi''$ in this order, then the effect of $(\pi''(v), (\pi''(u)), (\pi''(x), (u, v)(x, y))$ is a block-interchange acting on $\pi$.

Moreover, as described in the following lemma, we have shown in [17] that given two different DNA molecules $\pi$ and $\sigma$, every cycle $\alpha$ in the cycle decomposition of $\sigma\pi^{-1}$ always has a mate cycle $(\pi')(\alpha')^{-1}$ that also appears in $\sigma\pi^{-1}$. In fact, $\alpha$ and $(\pi')(\alpha')^{-1}$ in $\sigma\pi^{-1}$ are each other’s mate cycle.

Lemma 3 ([17]) Let $\pi$ and $\sigma$ be two different double-stranded DNA molecules. If $\alpha$ is a cycle in $\pi\sigma^{-1}$, then $(\pi')(\alpha')^{-1}$ is also a cycle in $\sigma\pi^{-1}$.

An efficient algorithm for the one-sided block ordering problem

To clarify our algorithm, we start with defining some notations. Let $\tilde{\alpha}$ denote an arbitrary linear DNA molecule (or contig). As mentioned previously, it is represented by the product of its two strands $\alpha$ and $\tilde{\alpha}$, that is, $\alpha = \alpha\tilde{\alpha}$. If $\alpha$ contains $k$ genes (or markers), we also denote its $i$-th gene by $(\alpha[i], 1), (\alpha'[2], 2), ..., (\alpha'[k], k)$, where $\alpha'[i]$ is the $i$-th gene in $\alpha$, and its $\alpha'$ by $(\alpha'[1], 1), (\alpha'[2], 2), ..., (\alpha'[k], k)$. By convention, $\alpha'[1]$ and $\alpha'[1]$ are called as tails of $\alpha$. Let $\pi = \pi_1\pi_2 ... \pi_m$ be a linear, uni-chromosomal genome that is partially assembled into $m$ contigs $\pi_i, \pi_2, ..., \pi_m$, each with $n_i$ genes, and $\sigma = (1, 2, ..., n)$ be a linear, uni-chromosomal genome that is assembled completely. Let $C = \{c_k = n + k - 1: 0 \leq k \leq 2m - 1\} \cup \{-c_k = -n - k - 1: 0 \leq k \leq 2m - 1\}$ be a set of $4m$ distinct integers, called caps, which are different from those genes in $E$. Let $\tilde{\pi} = E \cup C$ and $\tilde{\pi} = (1, -1)(2, -2)...(n + 2m, -n - 2m)$. For the purpose of designing our algorithm later, we add four caps $c_{2(1) - 1}, c_{2(1) - 1}, c_{2(1) - 1}, c_{2(1) - 1}, c_{2(1) - 1}$ to the ends of each contig $\pi_i$, where $1 \leq i \leq m$, leading to a capturing contig $\tilde{\pi}_i$, with $\tilde{\pi}_i[1] = c_{2(1) - 1}, \tilde{\pi}[j] = \pi_i[j - 1], \tilde{\pi}^{-1}[i] = \pi_i^{-1}[1], for 2 \leq j \leq n_i + 1, \pi_i[n_i + 2] = c_{2(1) - 1}, \pi_i^{-1}[1] = \pi_i^{-1}[1], for 2 \leq j \leq n_i + 1, and \pi_i[n_i + 2] = \tilde{\pi}(c_{2(1) - 1})$. Moreover, we insert $m$-1 dummy contigs without any genes (i.e., null contigs) $\sigma_2, \sigma_3, ..., \sigma_m$ into $\sigma$, where the original contig in $\sigma$ becomes $\sigma_1$ now, and add four caps $c_{2(1) - 1}, c_{2(1) - 1}, c_{2(1) - 1}, c_{2(1) - 1}$ to the ends of each contig $\sigma_i$ to obtain a capturing contig $\tilde{\sigma}_i$, where $\tilde{\sigma}[1] = c_{2(1) - 1}, \tilde{\sigma}[j] = \sigma_i[j - 1], \tilde{\sigma}_i[1] = \pi_i^{-1}[1], for 2 \leq j \leq n_i + 1, \tilde{\sigma}_i[n_i + 2] = \tilde{\pi}(c_{2(1) - 1}), \tilde{\sigma}_i^{-1}[1] = \sigma_i^{-1}[j - 1], for 2 \leq j \leq n_i + 1, and \tilde{\sigma}_i[n_i + 2] = \tilde{\pi}(c_{2(1) - 1})$. Notice that the purpose of adding caps to the ends of the contigs is to serve as delimiters when we use permutation groups to model translocations of multiple contigs later. We denote the capturing $\pi$ and $\sigma$ by $\tilde{\pi}$ and $\tilde{\sigma}$, respectively. To distinguish the four caps in a capturing contig, say $\tilde{\sigma}_i$, we call the left caps $\tilde{\pi}_i^{-1}[1]$ and $\tilde{\sigma}_i^{-1}[1]$ as $5'$ caps and the right caps $\tilde{\pi}_i[1] + 2$ and $\tilde{\sigma}_i[1] + 2$ as $3'$ caps.

Given an integer $x$ in $\tilde{\pi}$ that is contained in a contig $\alpha = \alpha\tilde{\alpha}$ with $k$ genes (or markers), we define a function $\text{char}(x, \tilde{\alpha})$ below to represent the character of $x$ in the capturing contig $\tilde{\alpha}$ that contains $x$. For convenience, we extend the definitions above from the capturing contig to the capturing genome. For instance, given a capturing

In addition, we define $5\text{cap}(x, \tilde{\alpha})$ to be the $5'$ cap in the strand of $\tilde{\alpha}$ that contains $x$. For convenience, we extend the definitions above from the capturing contig to the capturing genome. For instance, given a capturing
genome, say \( \hat{x} \), \( \text{char}(x, \hat{x}) \) denotes the character of \( x \) in a capping contig \( \hat{x}_i \) of \( \hat{x} \) that contains \( x \), and \( 5\text{cap}(x, \hat{x}) \) denotes the 5th cap of the strand in \( \hat{x}_i \) containing \( x \), that is, \( \text{char}(x, \hat{x}) = \text{char}(x, \hat{x}_i) \) and \( 5\text{cap}(x, \hat{x}) = 5\text{cap}(x, \hat{x}_i) \). In our previous work [17], we have shown the following lemma.

**Lemma 4** ([17]) For a capping genome \( \hat{x} \) and \( x \in \hat{x} \), if \( \text{char}(x, \hat{x}) = C3 \) (respectively, \( T \)), then \( \text{char}(\hat{x} \hat{x}^T(x), \hat{x}) \) is \( T \) (respectively, \( C3 \)) and if \( \text{char}(x, \hat{x}) = O \) (respectively, \( N3 \) and \( C5 \)), then \( \text{char}(\hat{x} \hat{x}^T(x), \hat{x}) \) is \( O \) (respectively, \( N3 \) and \( C5 \)).

Basically, we design our algorithm to solve the one-sided block ordering problem by dealing with the contigs of the capping genome \( \hat{x} \) as if they were linear chromosomes. Let \( c_1 = (x, y) \) and \( c_2 = (u, v) \) be two 2-cycles with character pairs of (non-C5, non-C5) and (C5, C5), respectively, and let \( c'_1 = (\hat{x} \hat{x}^T(y), \hat{x} \hat{x}^T(x)) \) and \( c'_2 = (\hat{x} \hat{x}^T(v), \hat{x} \hat{x}^T(u)) \). Notice that the character pair of \( c'_2 \) is \( (C5, C5) \) by Lemma 4. In our previous study [17], we have proven that performing a translocation \( \tau \) on \( \hat{x} \) can be mimicked by the composition of \( c'_2 c'_1 c_2 \hat{x} \hat{x}^T \) (i.e., \( \tau = c'_2 c'_1 c_2 \hat{x} \hat{x}^T \)), if \( (x, u) \hat{x} (y, v) \hat{x} (x, y) \hat{x} \) and \( (x, \hat{x}^T(y)) \hat{x} (i.e., x, y, and u, as well as y and v, lie in the same contig stand in \( \hat{x} \), but \( x \) and \( y \) appear in the different contigs in \( \hat{x} \)). Moreover, if the character pair of \( c_1 \) is in \( \text{CEpair} = \{ (C3, C3), (C3, N3), (T, T), (T, N3), (N3, N3) \} \), then \( \tau \) acts on \( \hat{x} \) by exchanging the two caps of some contig in \( \hat{x} \) with the two caps of another contig and, as a result, leaves the original genome \( \pi \) unaffected. Notice that the character pair of \( c'_1 \) also belongs to \( \text{CEpair} \) and that of \( c'_2 \) is \( (C5, C5) \) according to Lemma 4. Furthermore, if \( c_1 \) is a 2-cycle of character pair \( (T, C3) \) (respectively, \( (O, N3) \)), then \( \tau \) becomes a fusion (respectively, fission) to act on \( \pi \). Hence, we have the following lemma, where it can be verified that \( \hat{x} \hat{x}^T(5\text{cap}(x, \hat{x})) = 5\text{cap}(\hat{x} \hat{x}^T(x), \hat{x}) \) and \( \hat{x} \hat{x}^T(5\text{cap}(y, \hat{x})) = 5\text{cap}(\hat{x} \hat{x}^T(y), \hat{x}) \).

**Lemma 5** ([17]) Let \( c_1 = (x, y) \) denote a 2-cycle with char \( (x, \hat{x}) = T \) and char \( (y, \hat{x}) = C3 \), and let \( c'_1 = (\hat{x} \hat{x}^T(y), \hat{x} \hat{x}^T(x)) \) and \( \hat{x} \hat{x}^T(5\text{cap}(x, \hat{x})) \). If \( (x, y) \hat{x} (x, \hat{x}^T(y)) \), then the effect of \( c'_2 c'_1 c_2 \hat{x} \hat{x}^T \) is a fusion that acts on \( \pi \) by concatenating the contig containing \( y \) with the contig containing \( x \).

It is not hard to see that the permutation induced by an ordering of the uncapped genome \( \pi \) can be considered as the result of applying consecutive \( m - 1 \) fusions to the \( m \) contigs in \( \pi \). Based on the above discussion, it can be realized that our purpose is to find \( m - 1 \) translocations to act on \( \hat{x} \) such that their rearrangement effects on the original \( \pi \) are \( m - 1 \) fusions and the genome rearrangement distance measured by weighted reversals and block-interchanges between the resulting assembly of the contigs in \( \pi \) and \( \sigma \) is minimum. In Algorithm 1 below, we describe our algorithm for efficiently solving the one-sided block ordering problem, where reversals are weighted one and block-interchanges are weighted two. Basically, we try to derive \( m - 1 \) fusions from \( \sigma \hat{x}^{-1} \) to act on \( \pi \) in Algorithm 1.

**Algorithm 1**

**Input:** A partially assembled, linear, uni-chromosomal genome \( \pi = \pi_1 \pi_2 \ldots \pi_m \) and a completely assembled, linear, uni-chromosomal genome \( \sigma = \sigma_1 \).

**Output:** An optimally assembled genome of \( \pi \), denoted by \( \text{assembly}(\pi) \), and the weighted reversal and block-interchange distance \( \Delta(\pi, \sigma) \) between \( \text{assembly}(\pi) \) and \( \sigma \).

1. Add \( m - 1 \) null contigs \( \sigma_2, \sigma_3, \ldots, \sigma_m \) into \( \sigma \) such that \( \sigma = \sigma_1 \sigma_2 \ldots \sigma_m \).

Obtain \( \hat{x} = \hat{x}_1 \hat{x}_2 \ldots \hat{x}_m \) and \( \hat{\sigma} = \hat{\sigma}_1 \hat{\sigma}_2 \ldots \hat{\sigma}_m \) by capping \( \pi \) and \( \sigma \).

2. Compute \( \hat{\sigma} \hat{x}^{-1} \) and \( \hat{x} \hat{x}^T \).

3. /* To perform cap exchanges */

Let \( i = 0 \).
while there are \( x \) and \( y \) in a cycle of \( \hat{x} \hat{x}^{-1} \) such that \( (\text{char}(x, \hat{x}), \text{char}(y, \hat{x})) \) \( \in \) \( \text{CEpair} \) do

Let \( i = i + 1 \).
Find \( x \) and \( y \) in a cycle of \( \hat{x} \hat{x}^{-1} \) with \( (\text{char}(x, \hat{x}), \text{char}(y, \hat{x})) \) \( \in \) \( \text{CEpair} \).

Let \( \sigma_i = (\hat{x} \hat{x}^T(5\text{cap}(y, \hat{x})), \hat{x} \hat{x}^T(5\text{cap}(x, \hat{x}))) \hat{x} \hat{x}^T(y), 5\text{cap}(y, \hat{x}))(x, y), 5\text{cap}(y, \hat{x}))(x, y). \)

Calculate new \( \hat{x} = \sigma_i \hat{x}, \) new \( \hat{x} \hat{x}^T = \sigma_i \hat{x} \hat{x}^T \) and new \( \hat{\sigma} \hat{x}^{-1} = \hat{\sigma} \hat{x}^{-1} \hat{x}_i^{-1} \).
end while

4. /* To find consecutive \( m - 1 \) fusions */

Let \( i = 0 \).
while there are two adjacent elements \( x \) and \( y \) in a cycle of \( \hat{x} \hat{x}^{-1} \) such that \( (\text{char}(x, \hat{x}), \text{char}(y, \hat{x})) \) \( \in \) \( \{ (T, C3) \) and \( (x, y) \hat{x} \) do

Let \( i = i + 1 \).
Find two adjacent elements \( x \) and \( y \) in a cycle of \( \hat{x} \hat{x}^{-1} \) such that \( (\text{char}(x, \hat{x}), \text{char}(y, \hat{x})) \) \( \in \) \( \{ (T, C3) \) and \( (x, y) \hat{x} \).

Let \( \tau_i = (\hat{x} \hat{x}^T(5\text{cap}(y, \hat{x})), \hat{x} \hat{x}^T(5\text{cap}(x, \hat{x}))(\hat{x} \hat{x}^T(y), 5\text{cap}(y, \hat{x}))(x, y), 5\text{cap}(y, \hat{x}))(x, y). \)

Calculate new \( \hat{x} = \tau_i \hat{x}, \) new \( \hat{x} \hat{x}^T = \tau_i \hat{x} \hat{x}^T \) and new \( \hat{\sigma} \hat{x}^{-1} = \hat{\sigma} \hat{x}^{-1} \tau_i^{-1} \).
end while

while \( i < m - 1 \) do

Let \( i = i + 1 \).
Find two adjacent elements $x$ and $y$ in a cycle of $\delta \hat{\pi}^{-1}$ such that $(\text{char}(x, \hat{\pi}), \text{char}(y, \hat{\pi})) = (T, C3)$ and $(x, y) \in \hat{\pi}$.

Find the strand of a different contig in $\hat{\pi}$ with at least a non-cap integer and its $3'$ cap, say $z$, different from $y$.

Let $\tau_i = (\hat{\pi}(z), \hat{\pi}(x))$, $(\hat{\pi}(z), \hat{\pi}(y))(y, z)(x, z)$.

Calculate new $\hat{\pi} = \tau_i \hat{\pi}$, new $\hat{\pi} = \tau_i \hat{\pi}$ and new $\hat{\pi}^{-1} = \hat{\pi}^{-1} \tau_i^{-1}$.

**end while**

Let $\text{assembly}(\pi)$ denote the assembled contig in current $\hat{\pi}$ whose caps are removed.

5: /* To find reversals */

Let $n_{r} = 0$.

while there are two adjacent elements $x$ and $y$ in a cycle of $\delta \hat{\pi}^{-1}$ such that $(x, \hat{\pi}(y)) \in \hat{\pi}$ do

Let $n_{r} = n_{r} + 1$.

Find two adjacent elements $x$ and $y$ in a cycle of $\delta \hat{\pi}^{-1}$ such that $(x, \hat{\pi}(y)) \in \hat{\pi}$.

Let $\gamma_{n_{r}} = (\hat{\pi}(y), \hat{\pi}(x))(x, y)$.

Calculate new $\hat{\pi} = \gamma_{n_{r}} \hat{\pi}$, new $\hat{\pi} = \gamma_{n_{r}} \hat{\pi}$ and new $\hat{\pi}^{-1} = \hat{\pi}^{-1} \gamma_{n_{r}}^{-1}$.

**end while**

6: /* To find block-interchanges */

Let $n_{b} = 0$.

while $\delta \hat{\pi}^{-1} \neq 1$ do

Let $n_{b} = n_{b} + 1$.

Choose any two adjacent elements $x$ and $y$ in a cycle of $\delta \hat{\pi}^{-1}$.

Find two adjacent integers $u$ and $v$ in a cycle of $\delta \hat{\pi}^{-1}(x, y)$ such that $(u, v) \notin (x, y) \hat{\pi}$.

Let $\beta_{b} = (\hat{\pi}(u), \hat{\pi}(v))(\hat{\pi}(y), \hat{\pi}(x))(u, v)(x, y)$.

Calculate new $\hat{\pi} = \beta_{b} \hat{\pi}$, new $\hat{\pi} = \beta_{b} \hat{\pi}$ and new $\hat{\pi}^{-1} = \hat{\pi}^{-1} \beta_{b}^{-1}$.

**end while**

7: Output $\text{assembly}(\pi)$ and $\Delta(\pi, \sigma) = n_{r} + 2n_{b}$.

Below, we consider an example to clarify Algorithm 1.

Let $\pi = [1, 4, [-5, 6], [3, 2]]$ and $\sigma = [1, 2, ..., 6]$ be the input linear, uni-chromosomal genomes of Algorithm 1. In our algorithm, these two genomes will be further represented by $\pi = (1, 4)(-4, -1)(-5, 6)(-6, 5)(3, 2)(-2, -3, -6)$ and $\sigma = (1, 2, ..., 6)(-6, -5, ..., -1)$. First of all, we add two null contigs into $\sigma$ and cap all the contigs in $\pi$ and $\sigma$ in a way such that

$\hat{\pi} = (7, 1, 4, 8)(-8, -4, -1, -7)(9, -5, 6, 10)(-10, -6, 5, -9)(11, 3, 2, 12)(-12, -2, -3, -11)$ and

$\hat{\sigma} = (7, 1, 2, ..., 6)(-8, -6, -5, ..., -1)(9, 10)(-10, -9)[11, 12](-12, -11)$.

Next, we compute $\hat{\pi}^{-1} = (7, 1, 4, 8)(-8, -4, -1, -7)(9, -5, 6, 10)(-10, -6, 5, -9)(11, 3, 2, 12)(-12, -2, -3, -11)$.

It can be found that $7$ and $10$ are in a cycle of current $\delta \hat{\pi}^{-1}$ with $\text{char}(10, \hat{\pi}) = (C3, C3) \in \text{Cpair}$. We perform a cap exchange on $\hat{\pi}$ by multiplying $(\hat{\pi}(5)(\hat{\pi}(8, \hat{\pi}))) \hat{\pi}(5)(\hat{\pi}(8, \hat{\pi}))((\hat{\pi}(8, \hat{\pi}))(10, 8) = (-8, -10)(-4, -6, 9, 7, 10, 8) \hat{\pi}(8, \hat{\pi}))(10, 8) = (-8, -10)(-4, -6, 9, 7, 10, 8)$ with $\hat{\pi}$, resulting in new $\hat{\pi}$. In addition, we have new $\hat{\pi}^{-1} = (7, 1, 2, ..., 6)(-8, -6, -5, ..., -1)(9, 10)(-10, -9)[11, 12](-12, -11)$.

It can be observed that $-5$ and $10$ are in the same cycle of $\delta \hat{\pi}^{-1}$ with satisfying that $\text{char}(-5, \hat{\pi}) = T$, $\text{char}(10, \hat{\pi}) = C3$ and $(-5, 10) \not\in \hat{\pi}$ (since $-5$ and $10$ are in different contigs in current $\hat{\pi}$). Therefore, we perform a fusion on $\hat{\pi}$, by multiplying $\hat{\pi}(5)(\hat{\pi}(8, \hat{\pi}))) \hat{\pi}(5)(\hat{\pi}(8, \hat{\pi}))((\hat{\pi}(8, \hat{\pi}))(10, 8) = (-8, -10)(-4, -6, 9, 7, 10, 8)$.

Moreover, we have two null contigs into $\sigma$ and new $\hat{\pi} = (2, 4)(-1, -3, 5, -5, -4, 6, 3, 8)(2, -6)$. Notice that $-4$ and $6$ are adjacent in a cycle of current $\hat{\pi}$ and they are in different strands in current $\hat{\pi}$ since $(-4, \hat{\pi}(6))\hat{\pi}$. Thus, we can find a reversal, which is $(\hat{\pi}(6), \hat{\pi}(4))(-4, 6) = (5, -5)(-4, 6), \hat{\pi}$ can serve as a block-interchange to further transform $\hat{\pi}$ into $(7, 1, 2, 3, 4, 5, 6, 8)(-8, -6, -5, -4, -3, -2, -1, -7)(9, 10)(-10, -9)(11, 12)(-12, -11)$. After that, we have new $\hat{\pi}^{-1} = (2, 4)(-1, -3)(3, 8)(-2, -6), \hat{\pi}$ can serve as a block-interchange to further transform $\hat{\pi}$ into $(7, 1, 2, 3, 4, 5, 6, 8)(-8, -6, -5, -4, -3, -2, -1, -7)(9, 10)(-10, -9)(11, 12)(-12, -11)$, which is equal to $\hat{\sigma}$.

As a result, we obtain an ordering $(1, 14, [-5, 6], [3, 2])$ of $\pi$ whose induced permutation $[1, 4] \odot [-5, 6]$ $[3, 2]$ of $\sigma$ using a reversal and a block-interchange (i.e., $\Delta(\pi, \sigma) = 3$).

Actually, after running the step 3 of Algorithm 1, it can be verified according to the capping of $\pi$ and $\hat{\pi}$ and Lemma 3 that for any two adjacent elements $x$ and $y$ in a cycle of $\delta \hat{\pi}^{-1}$ with $\text{char}(x, \hat{\pi}), \text{char}(y, \hat{\pi}) = (T, C3)$, if $(x, y) \notin \hat{\pi}$, then $(\hat{\pi}(y), \hat{\pi}(x)) \not\in \hat{\pi}$. Moreover, the operation $\tau_i = (\hat{\pi}(z), \hat{\pi}(x))(\hat{\pi}(z), \hat{\pi}(y))(y, z)(x, z)$ used in the step 3 of Algorithm 1 acts on $\hat{\pi}$ still as a fusion of $\pi$, as explained as follows. Notice that $(x, y) \hat{\pi}$, meaning
that $x$ and $y$ are in the same cycle of $\hat{\sigma}$ and hence $5\text{cap}(x, \hat{\sigma}) = 5\text{cap}(y, \hat{\sigma})$. It can be verified that $(5\text{cap}(y, \hat{\sigma}), 5\text{cap}(z, \hat{\sigma})) = 1$, $5\text{cap}(x, \hat{\sigma}) = 1$. Since $(x, y)\hat{\sigma},$ we have $(5\text{cap}(y, \hat{\sigma}), 5\text{cap}(z, \hat{\sigma}))$ and hence $(\text{cap}(x, \hat{\sigma}), \text{cap}(z, \hat{\sigma})) \equiv (\text{cap}(y, \hat{\sigma}), \text{cap}(z, \hat{\sigma}))$, $5\text{cap}(\hat{\sigma}\hat{\Gamma}(x), \hat{\sigma}) = 1$. It is not hard to see that $(\text{cap}(x, \hat{\sigma}), \text{cap}(z, \hat{\sigma})) = (\text{cap}(y, \hat{\sigma}), \text{cap}(z, \hat{\sigma})) = (\text{cap}(x, \hat{\sigma}), \text{cap}(y, \hat{\sigma})) = (\text{cap}(x, \hat{\sigma}), \text{cap}(y, \hat{\sigma}))$. Thus, $t_1$ can be rewritten as $\tau_1 = \alpha_2\alpha_1$, where $\alpha_1 = (5\text{cap}(\hat{\sigma}\hat{\Gamma}(z), \hat{\sigma}), 5\text{cap}(\hat{\sigma}\hat{\Gamma}(x), z), 5\text{cap}(\hat{\sigma}\hat{\Gamma}(z), y), 5\text{cap}(\hat{\sigma}\hat{\Gamma}(y), z), 5\text{cap}(\hat{\sigma}\hat{\Gamma}(x), y, z))$. It can be verified that $\alpha_2 = (5\text{cap}(\hat{\sigma}\hat{\Gamma}(x), y, z))$. By Lemma 5, as well as the previous discussion, it can be realized that $a_1$ acts on $\hat{\sigma}$ as a fusion of $\sigma$ and $a_2$ continues to act on $\alpha_1\hat{\sigma}$ as a cap exchange. As a result, the rearrangement effect of acting $t_1$ on $\hat{\sigma}$ is still equivalent to a fusion acting on $\sigma$. The above discussion indicates that a fusion to $\sigma$ can be mimicked by a translocation $r$, which acts on $\hat{\sigma}$ as a fusion of $\sigma$, followed by zero or more translocations acting on $\tau\hat{\sigma}$ as cap exchanges.

In the following, we prove the correctness of Algorithm 1. Initially, it is not hard to see that all the $5'$ caps are fixed in $\hat{\sigma}\hat{\Gamma}^{-1}$ and $\text{char}(x, \hat{\sigma}) \neq \text{N3}$ for all $x \in \hat{E}$. For any element $x \in \hat{E}$ with $\text{char}(x, \hat{\sigma}) = T$, where $1 \leq i \leq m$, if $\hat{\sigma}^{-1}(x) \neq \hat{\sigma}^{-1}[1]$ and $\hat{\sigma}^{-1}(x) \neq \hat{\sigma}^{-1}[1]$, that is, the $5'$ cap of $\hat{\sigma}$ is not equal to that of $\sigma_1$, then the character of $\hat{\sigma}\hat{\Gamma}^{-1}(x)$ in $\hat{\sigma}$ must be C3. If any cycle in $\hat{\sigma}\hat{\Gamma}^{-1}$ contains any two elements $x$ and $y$ with the same character (either T or C3) in $\hat{\sigma}$, then we can extract two 2-cycles $c_1 = (x, y)$ and $c_1' = (\hat{\sigma}\hat{\Gamma}(y), \hat{\sigma}\hat{\Gamma}(x))$ from two mate cycles in $\hat{\sigma}\hat{\Gamma}^{-1}$ and multiply $c_2c_1'c_2c_1$ with $\hat{\sigma}$ to exchange the caps of the contigs containing $x$ and $y$, respectively, in $\hat{\sigma}$, where $c_2 = (5\text{cap}(x, \hat{\sigma}), 5\text{cap}(y, \hat{\sigma}))$ and $c_2' = (\hat{\sigma}\hat{\Gamma}(5\text{cap}(x, \hat{\sigma})), \hat{\sigma}\hat{\Gamma}(5\text{cap}(y, \hat{\sigma})))$. This is the job to be performed in the step 3 in Algorithm 1. Moreover, after finishing the cap exchanges in the step 3, each cycle in the remaining $\hat{\sigma}\hat{\Gamma}^{-1}$ has at most one element with T character and at most one element with C3 character. In other words, after running the step 3, there are at least $2(m-1)$ cycles in the resulting $\hat{\sigma}\hat{\Gamma}^{-1}$ such that each such a cycle contains exactly one element, say $x$, with $(x, \hat{\sigma}) = T$ and exactly one element, say $y$, with $\text{char}(y, \hat{\sigma}) = C3$, and $\hat{\sigma}\hat{\Gamma}^{-1}(x) = y$. In this case, we can further derive $2(m-1)$ 2-cycles from these cycles in $\hat{\sigma}\hat{\Gamma}^{-1}$ with each 2-cycle having a character pair of $(T, C3)$. Intriguingly, we shall show below that these $2(m-1)$ 2-cycles with character pair $(T, C3)$, denoted by $f_1, f_1', \ldots, f_{m-1}, f_{m-1}'$, can be used to obtain an optimal ordering of $\pi$ such that the weighted reversal and block-interchange distance between the permutation induced by this ordering of $\pi$ and $\sigma$ is minimum.
char \( (x, \hat{\pi}) = T \) and char \( (y, \hat{\pi}) = C3 \). Then
\[
||\psi(\hat{\sigma}\hat{\pi}^{-1})|| - ||\psi(\hat{\sigma}\hat{\pi}^{-1}\tau^{-1})|| \in \{-2, 0, 2\}.
\]
Proof. For simplicity, it is assumed that we cannot find any cap exchange from \( \hat{\sigma}\hat{\pi}^{-1} \) to perform on \( \hat{\pi} \). We then consider the following two cases.

Case 1: Suppose that \((x, y) \hat{\sigma}\hat{\pi}^{-1}\), that is, both \(x \) and \(y \) lie in the same cycle, say \(\alpha \) in \( \hat{\sigma}\hat{\pi}^{-1} \). Without loss of generality, let \(\alpha = (a_1, a_2, \ldots, a_i \equiv x, \ldots, a_j \equiv y)\). Then \(\alpha \) can be expressed as \(\alpha = \alpha_1\alpha_2(\hat{\pi}\hat{\Gamma}(x), \hat{\pi}\hat{\Gamma}(y))\), where \(\alpha_1 = \hat{\pi}\hat{\Gamma}(a_{i-1}), \ldots, \hat{\pi}\hat{\Gamma}(a_1), \hat{\pi}\hat{\Gamma}(a_2), \ldots, \hat{\pi}\hat{\Gamma}(a_j)\) and \(\alpha_2 = \hat{\pi}\hat{\Gamma}(a_{j-1}), \ldots, \hat{\pi}\hat{\Gamma}(a_i), \hat{\pi}\hat{\Gamma}(a_{j-2}), \ldots, \hat{\pi}\hat{\Gamma}(a_2)\). Clearly, after applying \(\tau \) to \(\hat{\pi}\), the cycle \(\alpha \) becomes two disjoint cycles \(\alpha_1 \) and \(\alpha_2 \) in \(\hat{\sigma}\hat{\pi}^{-1}\). 

Let \(\hat{\sigma}\) and \(\hat{\pi} \) denote the mate cycle of \(\alpha \) in \(\hat{\sigma}\hat{\pi}^{-1}\). Then it can be expressed as \(\alpha' = \alpha'_1\alpha'_2(\hat{\pi}\hat{\Gamma}(x), \hat{\pi}\hat{\Gamma}(y))\), where \(\alpha'_1 = (\hat{\pi}\hat{\Gamma}(a_{i-1}), \ldots, \hat{\pi}\hat{\Gamma}(a_1), \hat{\pi}\hat{\Gamma}(a_2))\) and \(\alpha'_2 = (\hat{\pi}\hat{\Gamma}(a_{j-1}), \ldots, \hat{\pi}\hat{\Gamma}(a_i), \hat{\pi}\hat{\Gamma}(a_{j-2}), \ldots, \hat{\pi}\hat{\Gamma}(a_2))\). According to Corollary 1, it can be realized that \(\hat{\sigma}\hat{\pi}^{-1}\) and \(\hat{\sigma}\hat{\pi}^{-1}\) are a fusion of \(\pi \) decreases the norm \(||\psi(\hat{\sigma}\hat{\pi}^{-1})||\) at most by two. Hence, we call \(\tau \) as a good fusion of \(\pi \) if

\[
||\psi(\hat{\sigma}\hat{\pi}^{-1})|| - ||\psi(\hat{\sigma}\hat{\pi}^{-1}\tau^{-1})|| = 2.
\]

Notice that if \(\hat{\pi} = \hat{\sigma} \), then \(||\psi(\hat{\sigma}\hat{\pi}^{-1})|| = 0\). According to Lemmas 5 and 6, any translocation \(\tau \) that acts on \(\hat{\sigma}\) as a fusion of \(\pi \) decreases the norm \(||\psi(\hat{\sigma}\hat{\pi}^{-1})||\) at most by two. Hence, we call \(\tau \) as a good fusion of \(\pi \) if \(\hat{\sigma}\). If \(\hat{\sigma}\hat{\pi}^{-1} \) and \(\hat{\sigma}\hat{\pi}^{-1}\) are a fusion of \(\pi \) decreases the norm \(||\psi(\hat{\sigma}\hat{\pi}^{-1})||\) at most by two. Hence, we call \(\tau \) as a good fusion of \(\pi \) if \(\hat{\sigma}\).
τ, to act on π, then the C3 elements in both f and f′ must locate at a contig whose T elements are in some fk and fk′, respectively, where 1 ≤ k ≤ λ. This implies that the good fusion τ cannot act on π together with τ1, τ2, . . . , τm−1 at the same time, since they will assemble a circular contig that is not allowed. Therefore, we must show that τ1, τ2, . . . , τm−1 are the fusions obtained by the step 4 of Algorithm 1. Clearly, for 1 ≤ k ≤ λ, |d(ω−1 1−k)| = |d(ω−1 1−k)| + 2 since τk is a good fusion to ω−1 1−k. Moreover, for

x = x, where 1 ≤ j ≤ m − 1, char (xj, ω−1 1−j) = T and char (yj, ω−1 1−j) = C3. According to Corollary 1, we have (x, y) ̸∈ ω−1 1−1, that is, x and y are in different cycles of ω−1 1−1. Moreover, char (x, τω−1 1−1) = O and char (y, τω−1 1−1) = N3 after applying τ to ω−1 1−1. Below, we consider two cases.

Case 1: Suppose that there is a 2-cycle f = (xj, yj) such that xj = x, where 1 ≤ j ≤ m − 1, char (xj, ω−1 1−j) = T and char (yj, ω−1 1−j) = C3. For simplifying our discussion, we assume that f is disjoint from the other cycles in ψ(ω−1 1−1) and y is in the cycle α = (a1, a2, . . . , an) of ψ(ω−1 1−1). Then in ψ(ω−1 1−1)τ−1, the cycles fj and α are joined into a cycle β = (a1, a2, . . . , an−1, y, y′, x), which can be expressed as γ(y, y′), where γ = (a1, a2, . . . , an−1, y, x), char (γ, τω−1 1−1) = N3 and char (y, τω−1 1−1) = C3. According to Lemma 3, there is a cycle β′ = (τω−1 1−1γ), ψ−β′ that is the mate cycle of β in ψ(ω−1 1−1)τ−1. In other words, we can extract c1 = (y, y′) from β and c′1 = (τω−1 1−1γ), τω−1 1−1γ) from β′, and then apply τ′ 1 = c′1c2c1 to τω−1 1−1 as a cap exchange, where c2 = (5cap(y, τω−1 1−1), 5cap(y, τω−1 1−1)), τω−1 1−1γ)τω−1 1−1γ(5cap(y, τω−1 1−1)), τω−1 1−1γ(5cap(y, τω−1 1−1)), since the character pair (C3, N3) of (y, y′) belongs to CEpair. After that, y′, as well as τω−1 1−1γ), will be fixed in the resulting ψ(ω−1 1−1) and char (y, ω−1 1−1) will become C3. As a result, n(ψ(ω−1 1−1)) = n(ψ(ω−1 1−1)) and hence ||ψ(ω−1 1−1)|| − ||ψ(ω−1 1−1)|| = 0. According to the above discussion, if j ≤ λ, that is, fj can be used to derive a good fusion to ω−1 1−1, then this good fusion will be destroyed when we perform τ′ 1τ to ω−1 1−1. On the other hand, if char (ak−1, ω−1 1−k−1) = T and (ak, ak−1) = fk with k ≤ λ, that is, fk can also be used to derive a good fusion to ω−1 1−k−1, then this good fusion will be destroyed after applying τ′ 1τ to ω−1 1−k−1, since fk will become a 2-cycle with character pair of (T, N3) in the resulting ω−1 1−k−1. Based on the above discussion, the number of good fusions newly created by τj and τ′ j minus that of good fusions currently destroyed by τj and τ′ j must be less than or equal to zero.

Case 2: Suppose that there is no fj = (xj, yj) such that xj = x, where 1 ≤ j ≤ m − 1, char (xj, ω−1 1−j) = T and char (yj, ω−1 1−j) = C3. Let α1 denote the cycle containing x and α2 denote the cycle containing y in ω−1 1−1. Also let α′ 1 and α′ 2 be the mate cycles of α1 and α2, respectively, in ω−1 1−1. Note that after applying τ to ω−1 1−1, the cycles α1 and α2 will be merged into a single cycle, say α′ in ω−1 1−1τ−1 and α′ 1 and α′ 2 will be merged into a single cycle, say α′. Moreover, the characters of x and y
in $\tau \omega_{k-1}$ will become O and N3, respectively. As discussed in the proof of Lemma 6, if both $\alpha_1$ and $\alpha_2$, as well as both $\alpha_1'$ and $\alpha_2'$, do not contain T and C3 elements simultaneously, then $||\psi(\tilde{\sigma}\omega_{k-1})|| - ||\psi(\tilde{\sigma}\omega_{k-1})|| = -2$. In this case, it can be verified that no existing good fusion is destroyed by $\tau$ and no new good fusion is created by $\tau$. In other words, the number of the increased good fusions minus that of the destroyed good fusions is zero. If at least one of $\alpha_1$ and $\alpha_2$, as well as at least one of $\alpha_1'$ and $\alpha_2'$, has both T and C3 elements at the same time, then $||\psi(\tilde{\sigma}\omega_{k-1})|| - ||\psi(\tilde{\sigma}\omega_{k-1})|| = 0$ according to the discussion in the proof of Lemma 6. Now suppose that $\alpha_1$ has no C3 element. Then T and C3 elements in $\alpha_2$ can form a 2-cycle that equals to some $f_k = (x_k, y_k)$, where $1 \leq k \leq m - 1$ and $y_k = y$. After applying $\tau$ to $\omega_{k-1}$, the T element $x$ from $\alpha_1$ becomes an O element in $\alpha$ and the C3 element $y$ from $\alpha_2$ becomes an N3 element in $\alpha$. We can continue to extract $(x_k, y)$, which is now a 2-cycle of ($T$, N3), from $\alpha$ and act $\tau_i = c_1' c_2 c_1$ on $\tau \omega_{k-1}$ as a cap exchange, where $c_2 = (5 \text{cap}(x_k, \tau \omega_{k-1}), c_2 = (5 \text{cap}(x, \tau \omega_{k-1}), 5 \text{cap}(y, \tau \omega_{k-1})))$, $c_1' = (\tilde{\omega}_{k-1} \tilde{\Gamma}(y), \tilde{\omega}_{k-1} \tilde{\Gamma}(c_2))$ and $c_2' = (\tilde{\omega}_{k-1} \tilde{\Gamma}(5 \text{cap}(y, \tau \omega_{k-1})), \tilde{\omega}_{k-1} \tilde{\Gamma}(5 \text{cap}(x_k, \tau \omega_{k-1})))$. Clearly, no new good fusion is created in this case and one existing good fusion derived by $f_k$ will be destroyed if $k \leq \lambda$. Therefore, the number of the increased good fusions minus that of the destroyed good fusions is less than or equal to zero. Suppose that $\alpha_1$ contains both T and C3 elements, where we denote the C3 element by $z$ for convenience. Then $x$ and $z$ can form a 2-cycle of (T, C3) pair, which can derive a good fusion $\tau = c_1' c_2 c_1$ to $\omega_{k-1}$ if $(x, z) \not\in \omega_{k-1}$, where $c_2 = (5 \text{cap}(x, \omega_{k-1}), c_2 = (5 \text{cap}(x, \omega_{k-1}), 5 \text{cap}(z, \omega_{k-1})))$, $c_1' = (\omega_{k-1} \tilde{\Gamma}(y), \omega_{k-1} \tilde{\Gamma}(c_2))$ and $c_2' = (\omega_{k-1} \tilde{\Gamma}(5 \text{cap}(z, \omega_{k-1})), \omega_{k-1} \tilde{\Gamma}(5 \text{cap}(x, \omega_{k-1})))$. If $(x, z) \not\in \omega_{k-1}$, then, as mentioned previously, $\tau$ cannot work together with $\lambda$ other good fusions derived by $f_1, f_2, \ldots, f_k$, at the same time, since they will assemble a circular contig that is not allowed. For the case in which $\alpha_2$ contains no T element, it is not hard to see that no new good fusion will be created and no existing good fusion will be destroyed when performing $\tau_i$ and its following cap exchange on $\omega_{i-1}$, resulting in that the number of the created good fusions minus that of the destroyed good fusions is zero. We now assume that $\alpha_2$ contains a T element, say $w$, and a C3 element $y$. Then $w$ and $y$ are adjacent in $\alpha_2$ and $(w, y)$ equals to some $f_k$, where $1 \leq k \leq m - 1$. After applying $\tau_i$ to $\omega_{k-1}$, $\alpha$ has a C3 element $z$, a T element $w$ and an N3 element $y$. Then a 2-cycle $(w, y)$ can be extracted from $\alpha$ such that $\alpha' = c_1' c_2 c_1$ can further perform on $\tau \omega_{k-1}$ as a cap exchange, where $c_1 = (w, y)$, $\alpha' = (5 \text{cap}(w, \omega_{k-1})), c_1' = (\omega_{k-1} \tilde{\Gamma}(5 \text{cap}(w, \omega_{k-1}))), \tilde{\omega}_{k-1} \tilde{\Gamma}(5 \text{cap}(w, \omega_{k-1})))$, $\tilde{\omega}_{k-1} \tilde{\Gamma}(5 \text{cap}(w, \omega_{k-1})))$, $\tilde{\omega}_{k-1} \tilde{\Gamma}(5 \text{cap}(w, \omega_{k-1})))$. Hence, if $k \leq \lambda$, then the good fusion derived by $f_k$ will be destroyed by $\tau$ and $\tau_i$. However, the remaining $\alpha$ still contains a C3 element $z$ and a T element $w$, which can derive a good fusion, say $\tau'$. Hence, the number of the increased good fusions minus that of the destroyed good fusions is zero. On the other hand, if $k > \lambda$, then no existing good fusion is destroyed by $\tau$ and $\tau_i$. In this case, the number of the increased good fusions minus that of the destroyed good fusions is equal to one. However, it can be verified that $\tau'$ cannot work with $\lambda$ other good fusions derived by $f_1, f_2, \ldots, f_k$, because they will produce a circular contig that is not allowed. In other words, no more contigs can be assembled after performing $\tau_i$ and $\tau_i'$ on $\omega_{k-1}$.

According to the above discussion, we can conclude that $||\psi(\tilde{\sigma}\omega_{k-1})|| - ||\psi(\tilde{\sigma}\omega_{m-1})|| \leq 2\lambda$. □

Based on Lemma 7, as well as the discussion in its proof, the $m - 1$ fusions derived by Algorithm 1 correspond to an optimal ordering of $\pi$ with an induced permutation assembly (a) such that the weighted rearrangement distance $\Delta(\pi, \sigma)$ between assembly (a) and $\sigma$ is minimized. The obtained rearrangement distance $\Delta(\pi, \sigma)$ is calculated based on the algorithm in our previous study [17], and is equal to $\tilde{\Delta}(\pi, \sigma)$, where $\tilde{\sigma}$ is the genome obtained by performing the cap exchanges and $m - 1$ fusions on the initial capping of $\pi$, as done in the steps 3 and 4 in Algorithm 1, respectively. The total time complexity of Algorithm 1 is $O(\delta n)$, where $\delta$ is the number of reversals and block-interchanges used to transform assembly (a) into $\sigma$. The reason is as follows. Since $m \leq n$, the cost of the step 1 for capping the input genomes is $O(n)$. The computation of $\tilde{\Delta}(\pi, \sigma)$ in the step 2 still can be done in $O(n)$ time. Recall that after running the step 3, each cycle in $\tilde{\Delta}(\pi, \sigma)$ has at most a T element and at most a C3 element. Totally, there are $2m\Gamma(\pi)$ elements and $2m$ C3 elements in the cycles of $\tilde{\Delta}(\pi, \sigma)$. Moreover, deriving two 2-cycles to serve as a cap exchange from two long mate cycles in $\tilde{\Delta}(\pi, \sigma)$ will divide these two long cycles into four smaller cycles. Hence, there are $O(n)$ cap exchanges to be performed in the step 3, which totally cost $O(n)$ time since each cap exchange needs only constant time. The step 4 assembles $m$ contigs by utilizing $2m - 1$ 2-cycle $f_1, f_2, \ldots, f_{m-1}, 1_{m-1}$, which can be derived in advance from $\tilde{\Delta}(\pi, \sigma)$ in $O(n)$ time. Since each fusion requires only constant time, the cost of the step 4 is $O(m + n)$, which is equal to $O(n)$. As to the steps 5 and 6, they can be done in $O(\delta n)$ time in total,
since there are totally δ iterations to find the reversals and block-interchanges and the time complexity of each iteration is dominated by the cost of finding a reversal or block-interchange that is \( O(n) \) time. Notice that although Algorithm 1 we described above is dedicated to linear, uni-chromosomal genomes, it can still be applied to circular, uni-chromosomal genomes, or to multi-chromosomal genomes with linear or circular chromosomes in a way of chromosome by chromosome.

**Theorem 1** Given a partially assembled genome \( \pi \) and a completely assembled genome \( \sigma \), the one-sided block ordering problem can be solved in \( O(\delta n) \) time and the weighted rearrangement distance between the permutation assembly \( \pi \) induced by the optimal ordering of \( \pi \) and \( \sigma \) is \( \frac{|\hat{\sigma}_\pi - 1|}{2} \) that can be computed in \( O(n) \) time, where \( \hat{\pi} \) is the capping genome of \( \pi \) with the cap exchanges and \( m - 1 \) fusions being done, \( \hat{\sigma} \) is the capping genome of \( \sigma \), \( n \) is the number of genes or markers, and \( \delta \) is the number of reversals and block-interchanges used to transform assembly \( \pi \) into \( \sigma \).

As mentioned in the introduction, any algorithm to solve the one-sided block ordering problem can be used to assemble (i.e., order and orient) the contigs in a draft genome based on a reference genome, if we denote this draft genome as \( \pi \) and use the reference genome as \( \sigma \). For this application, our Algorithm 1 can finish its job just in \( O(n) \) time, because it does not need to do the steps 5 and 6 in this situation.

**Experimental results**

We have implemented Algorithm 1 as mentioned in the previous section into a program and also compared its accuracy performance to SIS on assembling the contigs of partially assembled genomes using some simulated datasets of linear, uni-chromosomal genomes. For this purpose, we compared the permutation induced by an assembly algorithm for a partially assembled genome

![Figure 1 Comparison of accuracy performance between our program and SIS on simulated datasets with different ratios of the involved reversals and transpositions](http://www.biomedcentral.com/1471-2105/14/S5/S9)
with its actual permutation by counting the number of breakpoints between them, where each breakpoint corresponds to an error of incorrectly joining two contigs (i.e., a mis-join error) caused by the assembly algorithm. This breakpoint number is then normalized by the number of contigs minus ρ to represent a fraction of incorrect contig joins, where ρ = 1 if the chromosome is linear; otherwise, ρ = 0. Each of partially assembled genomes with single linear chromosome in our simulated datasets was prepared and tested as follows. First, we generated the reference genome σ = (1, 2, ..., n) with a linear chromosome of n genes, where n varies from 50 to 1000 with in the step of 50, and performed δ random rearrangement events (reversals and/or transpositions) on s to obtain a permutation of a linear, uni-chromosomal genome π′, where δ varies from zero to 100 in the step of 1. Among the δ rearrangement events in our simulations, we used four different occurrence ratios to randomly generate reversals and transpositions: (1) 1:0, (2) 2:1, (3) 1:1 and (4) 0:1. Next, the genome π′ is randomly fragmented into m contigs of various sizes to simulate the partially assembled genome π, where m varies from 50 to 500 with step 50. Finally, for each choice of n, m, δ and reversal/transposition ratio, we repeated the experiments 10 times and compared our program with SIS using their averaged normalized mis-join errors. As shown in Figure 1, the averaged normalized contig mis-join errors of our program are lower than those of SIS for all simulated datasets when the number of the involved reversals and transpositions is increased. In particular, if there are more transpositions involved in the rearrangement events, then the gap of accuracy performance between our program and SIS is increasing. The main reason may be due to the fact that our program can deal with both reversals and block-interchanges (including transpositions as a special case), while SIS considers only reversals without taking into account transpositions.

Conclusions
In this study, we introduced and studied the one-sided block/contig problem with optimizing the weighted reversal and block-interchange distance, which particularly has a useful application in genome resequencing. We finally designed an efficient algorithm to solve this problem in O(δn) time, where n is the number of genes or markers and δ is the number of used reversals and block-interchanges. In addition, we showed that the assembly of the partially assembled genome can be done in O(n) time and its weighted rearrangement distance from the completely assembled genome can be calculated in advance in O(n) time. Finally, our simulation results showed that the accuracy performance of our program is better than that of the currently existing tool SIS when the number of the involved reversals and transpositions is increased. Moreover, the gap of this accuracy performance between our program and SIS is increasing, if there are more transpositions involved in the rearrangement events.

Authors’ contributions
Corresponding author CLL conceived of the study, designed and analyzed the algorithm, and drafted the manuscript. The other authors CLL and KTC participated in the development of the program, as well as in the simulated experiments and their result discussion. The authors wish it to be known that the first two authors CLL and KTC contributed equally to this work and should be considered co-first authors. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work was supported in part by National Science Council of Republic of China under grant NSC100-2221-E-007-129-MY3.

Declarations
The publication costs for this article were funded by National Science Council of Republic of China under grant NSC100-2221-E-007-129-MY3.

References
1. Shendure J, Ji HL: Next-generation DNA sequencing. Nature Biotechnology 2008, 26:1135-1145.
2. Mardis ER: The impact of next-generation sequencing technology on genetics. Trends in Genetics 2008, 24:133-141.
3. Metzker ML: Sequencing technologies - the next generation. Nature Reviews Genetics 2010, 11:33-44.
4. Fertin G, Labarre A, Rusu I, Tannier E, Vialette S: Combinatorics of Genome Rearrangements. Cambridge, Massachusetts: The MIT Press; 2009.
5. Gual E, Blanchette M: Ordering partially assembled genomes using gene arrangements. Lecture Notes in Computer Science 2006, 4205:113-128.
6. Bourque G, Pevzner PA: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Research 2002, 12:26-36.
7. Hannenhalli S, Pevzner PA: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM 1999, 46:1-27.
8. Bentley DR: Whole-genome re-sequencing. Current Opinion in Genetics and Development 2006, 16:545-552.
9. Koboldt DC, Ding L, Mardis ER, Wilson RK: Challenges of sequencing human genomes. Briefings in Bioinformatics 2010, 11:484-498.
10. van Huij JAF, Zomer AL, Kupers OP, Kok J: Projector 2: contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies. Nucleic Acids Research 2005, 33:W560-W566.
11. Richter DC, Schuster SC, Huson DH: OSLay: optimal synteny layout of unfinished assemblies. Bioinformatics 2003, 19:1573-1579.
12. Assefa S, Keane TM, Otto TD, Newbold C, Berriman M: ABACAS: algorithm-based automatic contigulation of assembled sequences. Bioinformatics 2009, 25:1968-1969.
13. Rissman AJ, Mau B, Biehl BS, Darling AE, Glasner JD, Perna NT: Reordering contigs of draft genomes using the Mauve Aligner. Bioinformatics 2009, 25:2071-2073.
14. Muñoz A, Zheng CF, Zhu QA, Albert VA, Rounsley S, Sankoff D: Scaffold filling, contig fusion and comparative gene order inference. BMC Bioinformatics 2010, 11:304.
15. Husemann P, Stoye J. r2cat: synteny plots and comparative assembly. Bioinformatics 2010, 26:570-571.

16. Dias Z, Dias U, Setubal JC. STS: a program to generate draft genome sequence scaffolds for prokaryotes. BMC Bioinformatics 2012, 13:96.

17. Huang YL, Lu CL. Sorting by reversals, generalized transpositions, and translocations using permutation groups. Journal of Computational Biology 2010, 17:685-705.

18. Blanchette M, Kunisawa T, Sankoff D. Parametric genome rearrangement. Gene 1996, 172:GC11-GC17.

doi:10.1186/1471-2105-14-S5-S9

Cite this article as: Li et al. Assembling contigs in draft genomes using reversals and block-interchanges. BMC Bioinformatics 2013 14(Suppl 5):S9.