

Research article

Open Access

## Sorting by reversals and block-interchanges with various weight assignments

Ying Chih Lin<sup>\*1</sup>, Chun-Yuan Lin<sup>2</sup> and Chunhung Richard Lin<sup>1</sup>

Address: <sup>1</sup>Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan and <sup>2</sup>Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan 33302, Taiwan

Email: Ying Chih Lin<sup>\*</sup> - [yclin@cse.nsysu.edu.tw](mailto:yclin@cse.nsysu.edu.tw); Chun-Yuan Lin - [cyulin@mail.cgu.edu.tw](mailto:cyulin@mail.cgu.edu.tw); Chunhung Richard Lin - [lin@cse.nsysu.edu.tw](mailto:lin@cse.nsysu.edu.tw)

<sup>\*</sup> Corresponding author

Published: 4 December 2009

Received: 3 August 2009

BMC Bioinformatics 2009, 10:398 doi:10.1186/1471-2105-10-398

Accepted: 4 December 2009

This article is available from: <http://www.biomedcentral.com/1471-2105/10/398>

© 2009 Lin et al; licensee BioMed Central Ltd.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/2.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

### Abstract

**Background:** A classical problem in studying genome rearrangements is understanding the series of rearrangement events involved in transforming one genome into another in accordance with the parsimonious principle when two genomes with the same set of genes differ in gene order. The most studied event is the reversal, but an increasing number of reports have considered reversals along with other genome rearrangement events. Some recent studies have investigated the use of reversals and block-interchanges simultaneously with a weight proportion of 1:2. However, there has been less progress towards exploring additional combinations of weights.

**Results:** In this paper, we present several approaches to examine genome rearrangement problems by considering reversals and block-interchanges together using various weight assignments. An exact algorithm for the weight proportion of 1:2 is developed, and then, its idea is extended to design approximation algorithms for other weight assignments. The results of our simulations suggest that the performance of our approximation algorithm is superior to its theoretical expectation.

**Conclusion:** If the weight of reversals is no more than that of block-interchanges, our algorithm provides an acceptable solution for the transformation of two permutations. Nevertheless whether there are more tractable results for studying the two events remains open.

### Background

In comparative genomics, the study of genome rearrangements has been one of the most promising methods for tracing the evolutionary history using gene order comparisons between organisms. The mathematical model simply treats a chromosome in the genome as a permutation of integers, where each integer represents a gene. Specifically, these integers are associated with signs, + or -, to indicate the corresponding orientation (strandedness) of the gene. A basic task in genome rearrangement studies is to economically transform one permutation into another

using restricted types of global mutations. Compared with local (point) mutations, global mutations are rare, but can provide valuable clues about the evolutionary history of organisms.

The most widely studied type of global mutations is the *reversal* (also called *inversion*) which inverts a segment in the permutation and changes the sign of each integer in that segment. If we only consider reversals, the so-called problem of *sorting by reversals* (SBR) is to find the shortest series composed of reversals that transforms the given per-

mutation into another, where the minimum number of reversals is often regarded as the (reversal) *distance* between two permutations. SBR is a well-studied subject in computational biology, and its first polynomial-time algorithm was proposed by Hannenhalli and Pevzner in 1995 [1]. Other groups have subsequently simplified and improved this algorithm [2-5]. To date, the best running time of an algorithm for SBR is  $O(n^{3/2})$  in theoretical analysis, as presented by Han [6]. It remains unclear whether SBR can be solved in  $O(n \log n)$  time, but a plausible answer was recently given by Swenson *et al.* [7], providing two new algorithms; the first runs in randomized  $O(n \log n)$  time, whereas the other is a deterministic algorithm with running time  $O(n \log n + kn)$ , where  $k$  is a data-dependent parameter and both its average and standard deviation are small constants derived from extensive experiments [7]. Moreover, a linear-time cost is sufficient to compute the reversal distance [8].

In addition to reversals, *transpositions* and *block-interchanges* are also global mutations that act on a permutation. The former exchanges two adjacent segments, and the latter is a generalization of a transposition in which exchanged segments do not have to be adjacent. The problem of transpositions is called *sorting by transpositions* (SBT), in which the minimum number of transpositions required to complete the transformation is sought. Currently, we know nothing about its complexity, but several approximation algorithms have been proposed [9-11]. However, the problem of *sorting by block-interchanges* (SBBI) using block-interchanges only is tractable and was first studied by Christie [12] using the graph approach and then by Lin *et al.* [13] using the algebraic formalism. Recently, Feng and Zhu [14] introduced a new data structure to improve the approximation and exact algorithms for SBT and SBBI, respectively, to achieve the time complexity  $O(n \log n)$ .

Considering reversals and transpositions together leads to the problem of *sorting by reversals and transpositions* (SBR+T), i.e., it allows one to perform reversals and transpositions alternatively during the transforming process. Because of the two operations used, we assign weights  $w_r$  to reversals and  $w_t$  to transpositions, and thus seek a transforming series with a minimum sum of weights. For  $w_r : w_t = 1 : 1$ , Lin and Xue [15] and Walter *et al.* [16] presented approximation algorithms with a factor of 2. By incorporating *inverted transposition*, which inverts one of two swapped segments of a transposition and usually has equal weight  $w_{it}$  to  $w_v$  in the transformation, 2-approximation algorithms have been reported by two groups [15,17]. Furthermore, Eriksen [18] developed a  $(1 + \varepsilon)$ -approximation algorithm for the weighted assignment of  $w_r : w_t(w_{it}) = 1 : 2$ . Bader and Ohlebusch [19] recently devised a 1.5-approximation algorithm with time  $O(n^2)$

for any weight proportion of  $w_r : w_t(w_{it})$  between 1 : 1 and 1 : 2. Nevertheless, it remains unknown whether tractable results can be derived for SBR+T.

In contrast, studying the block-interchanges (with each weight  $w_{bi}$ ) along with reversals seems easier, i.e., the problem of *sorting by reversals and block-interchanges* (SBR+BI). For  $w_r : w_{bi} = 1 : 2$ , three groups of researchers began from different perspectives but all achieved tractable results for SBR+BI [20-22]. Yancopoulos *et al.* [20] introduced a universal *double-cut-and-join* operation that accounts for reversals, *translocations*, *fissions*, *fusions* and block-interchanges by assigning a weight of 2 to block-interchanges and 1 to others. With a slight modification to their algorithm, one can optimally solve SBR+BI [21]. In addition, the approach of Lin *et al.* [21] based on the so-called *breakpoint graph* [1], whereas Mira and Meidanis [22] adopted the algebraic viewpoint by introducing the parameter *norm* to represent the weight of a rearrangement event. By adding a number of local mutations, Bader [23] tackled the problem of unequal gene content using a heuristic algorithm. Despite tractable results when studying SBR+BI under  $w_r : w_{bi} = 1 : 2$ , to our knowledge, this is the only type of weight assignments that have been considered so far. In this paper, we study genome rearrangement problems by considering reversals and block-interchanges simultaneously using various weight assignments.

On the other hand, a traditional yet effective way to approach a complex problem is to devise an approximate solution that is "not too far from" the exact solution. Approximation algorithms are, indeed, a well-developed branch of the computer sciences [24]. A  *$\beta$ -approximation algorithm* ( $\beta > 1$ ) for a minimization problem runs in time polynomial to the input size and returns a feasible solution having a quality value that is, at most,  $\beta$  times the optimum. More interestingly, since the factor  $\beta$  is obtained from the worst-case analysis, an approximation algorithm with a higher factor does not imply poor average performance. To address genome rearrangement problems, two approximation algorithms are developed in this work, together with theoretical analyses and experiments to evaluate their performance.

## Methods

### Preliminaries

A *signed linear permutation*  $\vec{\pi} = (\vec{\pi}_1, \vec{\pi}_2, \dots, \vec{\pi}_n)$  is a permutation of  $\{1, 2, \dots, n\}$ , where each element is labeled by + or - to indicate the orientation of its corresponding gene. A *reversal*  $r(i, j)$  (with  $1 \leq i \leq j \leq n$ ) is an operation that inverts the order of elements in a segment of  $\vec{\pi}$  by transforming  $\vec{\pi}$  into

$r(i, j) \cdot \bar{\pi} = (\bar{\pi}_1, \dots, \bar{\pi}_{i-1}, -\bar{\pi}_j, \dots, -\bar{\pi}_i, \bar{\pi}_{j+1}, \dots, \bar{\pi}_n)$ . Another operation, *block-interchange*  $bi(i, j, k, l)$  (with  $1 \leq i \leq j < k \leq l \leq n$ ), exchanges two non-intersecting segments  $(\bar{\pi}_i, \bar{\pi}_{i+1}, \dots, \bar{\pi}_j)$  and  $(\bar{\pi}_k, \bar{\pi}_{k+1}, \dots, \bar{\pi}_l)$  by converting  $\bar{\pi}$  to  $bi(i, j, k, l) \cdot \bar{\pi} = (\bar{\pi}_1, \dots, \bar{\pi}_{i-1}, \bar{\pi}_k, \dots, \bar{\pi}_l, \bar{\pi}_{j+1}, \dots, \bar{\pi}_{k-1}, \bar{\pi}_i, \dots, \bar{\pi}_j, \bar{\pi}_{l+1}, \dots, \bar{\pi}_n)$ . For the two operations considered in our study, the weights of reversals and block-interchanges are denoted by  $w_r$  and  $w_{bi}$ , respectively.

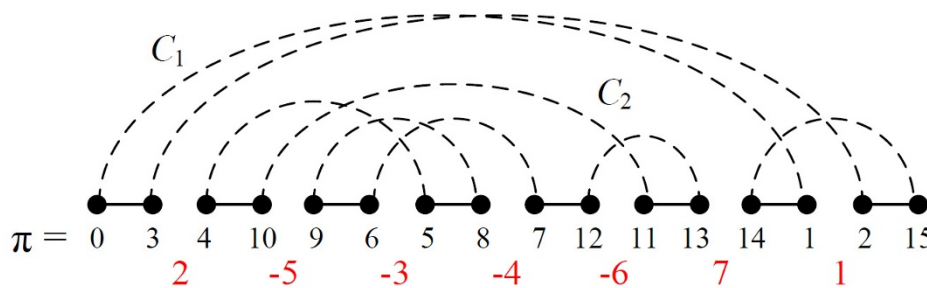
Given two permutations  $\bar{\pi}$  and  $\bar{\sigma}$ , the  $WGRP(w_r, w_{bi})$ , abbreviated from *Weighted Genome Rearrangement Problem with  $w_r$  and  $w_{bi}$* , is used to find a minimum weighted sequence of reversals and block-interchanges for transforming  $\bar{\pi}$  into  $\bar{\sigma}$ , and its sum of weights  $dist(\bar{\pi}, \bar{\sigma})$  is regarded as the *distance* between  $\bar{\pi}$  and  $\bar{\sigma}$ . In general, the problem is simplified as follows. First, the elements in  $\bar{\pi}$  and  $\bar{\sigma}$  are relabeled such that  $\bar{\sigma}$  becomes the identity permutation  $\bar{I} = (1, 2, \dots, n)$ , and therefore the transformation from  $\bar{\pi}$  to  $\bar{I}$  is similar to a sorting process. The distance  $dist(\bar{\pi}, \bar{I})$  is also simplified as  $dist(\bar{\pi})$ . Next, for  $w_r > 0$ , we replace  $w_{bi}$  with  $w_{bi}/w_r$  and fix  $w_r$  to 1.

When dealing with the signed permutation  $\bar{\pi}$  of size  $n$ , most studies extend and transform  $\bar{\pi}$  into an unsigned mapping  $\pi = (\pi_0, \pi_1, \dots, \pi_{2n+1})$  of  $\{0, 1, \dots, 2n + 1\}$  beforehand by replacing each positive element  $x$  of  $\bar{\pi}$  by  $2x - 1$  and  $2x$ , each negative element  $-x$  by  $2x$  and  $2x - 1$ , and adding two elements  $\pi_0 = 0$  and  $\pi_{2n+1} = 2n + 1$ . For example, if  $\bar{\pi} = (2, -5, -3, -4, -6, 7, 1)$ , then its unsigned mapping is  $\pi = (0, 3, 4, 10, 9, 6, 5, 8, 7, 12, 11, 13, 14, 1, 2, 15)$ . Each operation on  $\bar{\pi}$  also corresponds to a specific operation on  $\pi$  as follows: A reversal of the form  $r(2i + 1, 2j)$  is said to be *legal* for  $\pi$  since it mimics the reversal  $r(i + 1, j)$  on

$\bar{\pi}$  [1], and similarly a block-interchange  $bi(2i + 1, 2j, 2k + 1, 2l)$  is *legal* on  $\pi$  since it acts like the block-interchange  $bi(i + 1, j, k + 1, l)$  on  $\bar{\pi}$ . Considering the above  $\bar{\pi}$  as an example, the reversal  $r(5, 12)$  and block-interchange  $bi(1, 8, 11, 14)$  are legal, whereas  $r(3, 5)$  and  $bi(1, 9, 11, 14)$  are not. Furthermore, performing  $r(5, 12)$  (resp.  $bi(1, 8, 11, 14)$ ) on  $\pi$  is equivalent to performing  $r(3, 6)$  (resp.  $bi(1, 4, 6, 7)$ ) on  $\bar{\pi}$ . In other words, the  $WGRP(w_r, w_{bi})$  between  $\bar{\pi}$  and  $\bar{I}$  can be solved by computing a minimum weighted sequence of legal reversals and block-interchanges for converting  $\pi$  to  $I$ . We hereafter use  $\pi$  and  $I$  instead of  $\bar{\pi}$  and  $\bar{I}$ , and legal reversals and block-interchanges in our algorithms.

**Breakpoint graph**

Let  $\pi$  be the permutation mentioned previously. The so-called *breakpoint graph*  $BP(\pi)$  is a powerful analysis tool for studying genome rearrangement problems, and is defined as an edge-colored graph with  $2n + 2$  vertices as follows: For  $0 \leq i \leq n$ ,  $\pi_{2i}$  connects to  $\pi_{2i+1}$  by a *black edge* and  $2i$  is joined to  $2i + 1$  by a *gray edge* (Figure 1). In  $BP(\pi)$ , a gray edge  $(\pi_i, \pi_j)$  is said to be *oriented* if  $i + j$  is even, and otherwise it is *unoriented*. A cycle is said to be *alternating* if it contains alternating black and gray edges. Since the degree of each vertex is 2 (a black edge and a gray edge), the graph  $BP(\pi)$  can be uniquely decomposed into edge-disjoint and alternating cycles. In addition, a cycle is *oriented* as long as it has an oriented gray edge, otherwise, it is *unoriented*. The *length* of a cycle is the number of black (or equivalently, gray) edges it contains. We use  $l$ -cycle to denote an alternating cycle with length  $l$ , and  $c(\pi)$  to denote the number of cycles in  $BP(\pi)$ , e.g., in Figure 1,  $c(\pi) = 2$ : one is a 5-cycle and the other is a 3-cycle. Note that  $c(\pi) = n + 1$  if and only if  $\pi = I$ .



**Figure 1**  
**The breakpoint graph  $BP(\pi)$  of the permutation  $\pi$ , in which black edges are represented as solid lines and gray edges as dashed lines.** The gray edge (4, 5) is oriented whereas (2, 3) is unoriented. In addition, there are two components  $C_1$  and  $C_2$ , in which the former is a hurdle.

Each gray edge  $g = (\pi_i, \pi_j)$  is associated with the interval  $\langle i, j \rangle$ , and two gray edges *overlap* if their corresponding intervals overlap but neither of them properly contains the other. Moreover, two cycles *overlap* if their gray edges overlap, and a set of overlapping cycles forms a *component*. As with oriented cycles, a component is *oriented* if at least one of its cycles is oriented, and it is *unoriented* otherwise. Using the result of Bader *et al.* [8], the oriented and unoriented components can be efficiently determined in linear time.

A complex and interesting component of the Hannenhalli and Pevzner (HP) theory copes with the *hurdle*, which currently has several slightly different definitions [1,2,18,25,26]. Here we adopt a similar statement to the work of Eriksen [18] but with linear permutations. A *hurdle*  $H$  is an unoriented component such that there is an interval containing all vertices in  $H$  but no vertices in other unoriented components. Here we allow continuous intervals by setting 0 to be the successor of  $2n + 1$ . For the permutation  $\pi$  in Figure 1,  $C_1$  is a hurdle since  $\langle 12, 15 \rangle \cup \langle 0, 1 \rangle$  is an interval containing the unoriented component  $C_1$  only. Although  $\langle 2, 11 \rangle$  contains  $C_2$  only,  $C_2$  is not a hurdle since it is an oriented component. As a result, the number of hurdles of  $\pi$  in Figure 1 is one, i.e.,  $h(\pi) = 1$ .

The HP theory shows that the variations in  $c(\pi)$  and  $h(\pi)$  guide the transformation between two permutations. For an arbitrary operation  $\rho$  acting on  $\pi$ , let  $\Delta c_\rho = c(\rho \cdot \pi) - c(\pi)$  and  $\Delta h_\rho = h(\rho \cdot \pi) - h(\pi)$ . For convenience, we further abbreviate  $\Delta c_\rho$  (resp.  $\Delta h_\rho$ ) to  $\Delta c_r$  (resp.  $\Delta h_r$ ) if  $\rho$  is a reversal and to  $\Delta c_{bi}$  (resp.  $\Delta h_{bi}$ ) if  $\rho$  is a block-interchange. HP showed that  $\Delta c_r \leq 1$  and  $\Delta h_r \leq 2$  [1]. Christie presented that  $\Delta c_{bi} \leq 2$  but on unsigned permutations [12]. A similar

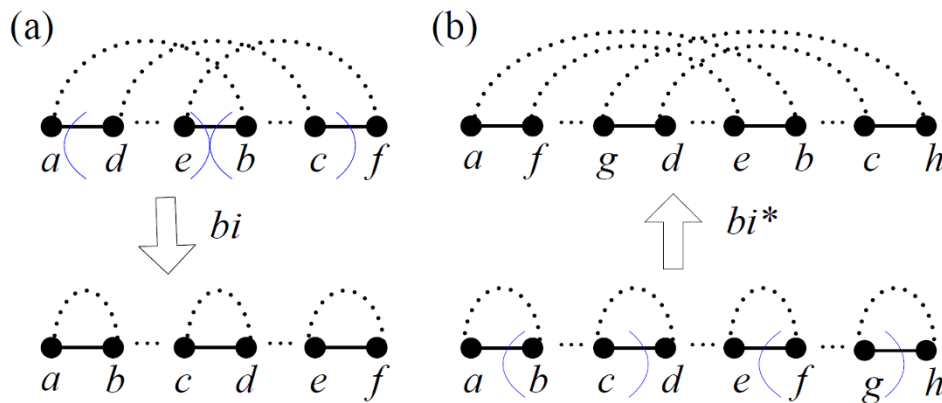
argument as Christie's work [12] can extend the upper bound of  $\Delta c_{bi}$  on signed permutations.

**Lemma 1** For every permutation  $\pi$  and block-interchange  $bi$ ,  $\Delta c_{bi} \leq 2$ .

**Proof:** A block-interchange exchanges two non-overlapping segments, whereas a segment can be specified by two black edges. Let  $V_{bi}$  be the set of vertices connected by the black edges for determining the block-interchange  $bi$ , and  $c(V_{bi})$  be the number of cycles containing the vertices in  $V_{bi}$ . For example in Figure 2a,  $V_{bi} = \{a, d, e, b, c, f\}$  and  $c(V_{bi}) = 1$ . According to the number of black edges containing vertices in  $V_{bi}$ , we have the following two cases:

**CASE1: Three black edges.** Applying  $bi$  to  $\pi$  affects only the cycles whose vertices are in  $V_{bi}$ . Due to the three black edges in this case, we have  $1 \leq c(V_{bi}) \leq 3$  and the same is true after applying  $bi$ , implying that  $\Delta c_{bi} \leq 2$  (Figure 2a).

**CASE2: Four black edges.** A similar statement as **CASE1** shows that  $\Delta c_{bi} \leq 3$  as a result of  $1 \leq c(V_{bi}) \leq 4$ . The only possibility in which  $\Delta c_{bi} = 3$  comes from the result of breaking the cycle in  $\pi$  into four cycles in  $bi \cdot \pi$ , but it cannot happen with the subsequent argument. As shown in Figure 2b, the block-interchange  $bi^*$  with  $c(V_{bi^*}) = 4$  results in  $c(V_{bi^*}) = 2$  after performing  $bi^*$ , and hence,  $\Delta c_{bi^*} = 1 - 4 = -3$ . However, if there is a  $bi$  such that  $\Delta c_{bi} = 3$ , then the vertices of  $V_{bi}$  will be in four cycles of  $BP(bi \cdot \pi)$ . Then the  $bi^*$  exchanging the two swapped segments of  $bi$  has  $\Delta c_{bi^*} = -3$  when it acts on  $bi \cdot \pi$ , a contradiction. Consequently,  $\Delta c_{bi} \leq 2$ .  $\square$



**Figure 2**  
**The block-interchange  $bi$  defined by (a) three black edges increases the number of cycles by two, whereas (b) four black edges decreases the number of cycles by two.** The pair of blue parentheses specifies one of two exchanged segments of  $bi$ , and the small dotted lines denote alternating paths.

WGRP( $w_r = 1, w_{bi} = 2$ )

For a sorting series  $S = \rho_1, \rho_2, \dots, \rho_l$  transforming  $\pi$  into  $I$ , where  $\rho_i$  represents either a reversal or a block-interchange, let the number of reversals be  $d_r(S)$  and the number of block-interchanges be  $d_{bi}(S)$ . Thus, the weighted sum of  $S$  is  $d(S) = w_r \cdot d_r(S) + w_{bi} \cdot d_{bi}(S)$ . The distance  $dist(\pi)$  is then the minimum  $d(S)$  among all sorting series  $S$  of converting  $\pi$  to  $I$ . First, we set  $w_{bi} = 2$  and consider WGRP (1, 2). Lemma 2 gives a lower bound of  $dist(\pi)$  in a more general case when  $2 \leq w_{bi}$ .

**Lemma 2**  $dist(\pi) \geq n + 1 - c(\pi)$  for WGRP(1,  $w_{bi}$ ) with  $2 \leq w_{bi}$ .

**Proof:** Since  $\Delta c_r \leq 1$  and  $\Delta c_{bi} \leq 2$ , an operation increasing the number of cycles by one costs at least  $\min\{\frac{w_r}{1}, \frac{w_{bi}}{2}\}$ , which equals 1 in the case of  $w_r = 1$  and  $2 \leq w_{bi}$ . However, in the best situation, there are at least  $n + 1 - c(\pi)$  cycles to be increased because of  $n + 1$  cycles in  $BP(I)$ . As a result, the cost of any transformation from  $\pi$  to  $I$  is at least  $n + 1 - c(\pi)$  for WGRP(1,  $w_{bi}$ ) with  $2 \leq w_{bi}$ .  $\square$

To deal with WGRP(1, 2), Lemma 2 shows that if the rearrangement sequences for sorting  $\pi$  are composed of reversals with  $\Delta c_r = 1$  and block-interchanges with  $\Delta c_{bi} = 2$ , the cost of such a sequence is equal to the lower bound of  $dist(\pi)$ , and hence is optimal. The strategy for selecting best reversals and block-interchanges is the core of the algorithm proposed by Lin *et al.* [21]. Their algorithm distinguished between oriented and unoriented components, and then sorted them separately, i.e., used the algorithm of Kaplan *et al.* [2] to sort all oriented components and the algorithm of Lin *et al.* [13] to deal with the unoriented components. Here we also utilize a known

algorithm for SBR, called ASBR, to tackle oriented components but we modify the method for sorting unoriented components using the following theorem.

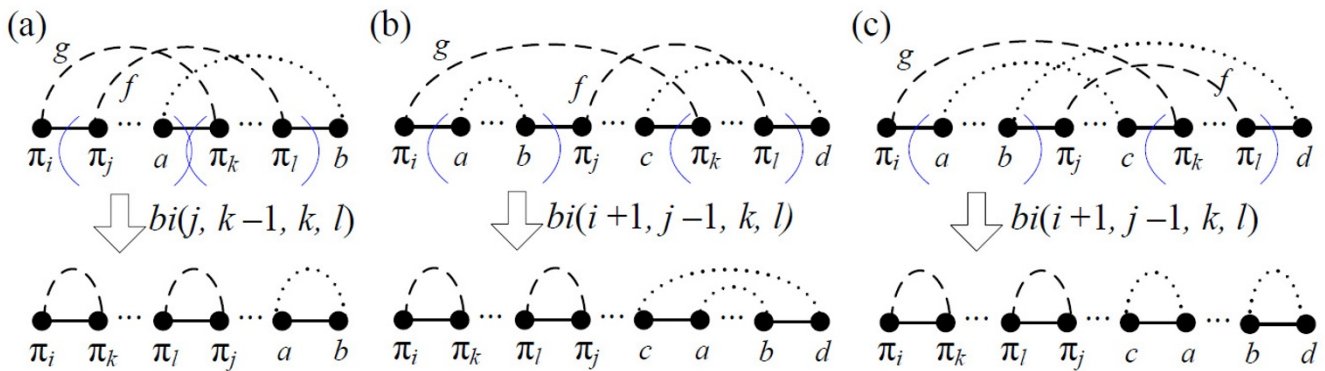
**Theorem 1** Let  $g = (\pi_i, \pi_k)$  and  $f = (\pi_j, \pi_l)$  be unoriented gray edges of a component. If  $g$  and  $f$  overlap, then there is a block-interchange with  $\Delta c_{bi} = 2$  in this component.

**Proof:** WLOG, we assume that  $i$  and  $l$  are even and  $j$  and  $k$  are odd with  $i < j < k < l$  (other cases of  $i, j, k$  and  $l$  can be illustrated similarly). According to the number of cycles containing  $g$  and  $f$ , there are two main cases:

CASE1:  $g$  and  $f$  are in the same cycle. We further consider two subcases according to whether  $\pi_i$  and  $\pi_j$  are connected by a black edge:

(1)  $j = i + 1$ , i.e., there is a black edge linking  $\pi_i$  and  $\pi_j$  (Figure 3a). Using the assumption of  $k < l$ , and that  $k$  is odd and  $l$  is even, there is no black edge between  $\pi_k$  and  $\pi_l$ . Therefore, we use the three black edges,  $(\pi_i, \pi_j)$ ,  $(a, \pi_k)$ , and  $(\pi_l, b)$  to determine the block-interchange  $bi(j, k - 1, k, l)$ . After performing it, the number of cycles is increased by two (Figure 3a), i.e.,  $\Delta c_{bi} = 2$ .

(2)  $j > i + 1$ . Let  $V_{bi} = \{\pi_i, a, b, \pi_j, c, \pi_k, \pi_l, d\}$  (Figure 3b). There are no alternating paths from vertex  $a$  to  $c$  without passing a vertex in  $V_{bi} \setminus \{a, c\}$  since  $g$  and  $f$  are in the same cycle. Consequently, one of the two cases of alternating paths linking vertices  $a, b, c$ , and  $d$  is demonstrated in Figure 3b. In this case, let the block-interchange be  $bi(i + 1, j - 1, k, l)$  and thus, in  $BP(bi(i + 1, j - 1, k, l) \cdot \pi)$  the four vertices,  $a, b, c$ , and  $d$ , belong to one cycle. (The other case can be similarly demonstrated.) We have  $c(bi(i + 1, j - 1, k, l) \cdot \pi) = c(\pi) + 2$ , which implies that  $\Delta c_{bi} = 2$ .



**Figure 3**  
Two unoriented gray edges  $g = (\pi_i, \pi_k)$  and  $f = (\pi_j, \pi_l)$  overlapping in a component are in the same cycle with (a)  $j = i + 1$  and (b)  $j > i + 1$ , whereas (c)  $g$  and  $f$  are in different cycles.

CASE2:  $g$  and  $f$  are in two different cycles (Figure 3c). Recall that the order and positions of  $i, j, k$ , and  $l$  are fixed via the assumption. On the condition that  $g$  and  $f$  are parts of different cycles,  $\pi_i$  and  $\pi_j$  are never joined by a black edge. In addition, the vertex  $a$  connects to  $b$  (or  $d$ ) by an alternating path that will result in the subcase (2) of CASE1. As a consequence, Figure 3c is the unique possibility in this case, and performing the block-interchange  $bi(i + 1, j - 1, k, l)$  leads to  $\Delta c_{bi} = 4 - 2 = 2$ .  $\square$

All gray edges are unoriented in unoriented components by definition, and furthermore, HP theory presents that for every gray edge  $g$  not in a 1-cycle, there is another gray edge  $f$  that overlaps with  $g$  [1]. In other words, it is always feasible to find two unoriented gray edges overlapping in unoriented components. By repeatedly applying the block-interchanges constructed in Theorem 1, all unoriented components are eventually sorted. We summarize the procedures as AWGRP(1,2) as follows:

**Algorithm for WGRP( $w_r = 1, w_{bi} = 2$ ) (AWGRP(1,2))**

**Input:** A signed permutation  $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2, \dots, \bar{\pi}_n)$ .

**Output:** A sorting series composed of reversals and block-interchanges for optimally transforming  $\bar{\pi}$  into  $\bar{I}$ .

- 1: Transform  $\bar{\pi}$  into its unsigned mapping  $\pi$  and construct  $BP(\pi)$ ;
- 2: Use the algorithm developed by Bader *et al.* [8] to distinguish between oriented and unoriented components;
- 3: Perform the algorithm of Han [6] to sort all oriented components;
- 4: Repeatedly apply the block-interchanges constructed by Theorem 1 to sort all unoriented components;
- 5: Mimic the sorting series of  $\pi$  to  $I$  to the transformation between  $\bar{\pi}$  and  $\bar{I}$ ;

In AWGRP(1,2), Step1 and Step2 cost linear time, while Step5 can be implemented in  $O(n \log n)$  time [14,27]. Recently, Feng and Zhu [14] developed a new data structure, called the *permutation tree*, to improve certain algorithms for SBT and SBBI, to achieve the time complexity  $O(n \log n)$ . This group used the permutation tree to implement two core procedures, *Query* and *Transposition*, which were developed by Hartman and Shamir [10] on the breakpoint graph. The former is used to find a pair of black edges *intersecting* the given pair of black edges, and the latter is used to adjust the data structures after apply-

ing transpositions. Although the term "intersecting" is defined on black edges [10], it is indeed the same concept as "overlap" here, and thus, can be used to find two overlapping unoriented gray edges to piece together block-interchanges. Moreover, since a block-interchange can be mimicked by two transpositions, a slight modification of the *Transposition* procedure [10] can be applied to retain the structures after performing block-interchanges. In short, the method of Feng and Zhu [14] to enhance the algorithm of Hartman and Shamir [10] can also be extended to cope with performing block-interchanges on unoriented components in Step4, for which we do not give a detailed description here. Accordingly, Step4 costs  $O(n \log n)$  time. The running time of Step3 is  $O(n^{3/2})$  in a theoretical analysis [6], which is currently the best, or  $O(n \log n)$  in most cases [7], depending on which algorithm is used to address SBR. As a result, theoretically, the total time complexity of AWGRP(1,2) is  $O(n^{3/2})$ .

**WGRP( $w_r = 1, 2 < w_{bi} < 3$ )**

In this subsection, we adjust the weight of block-interchanges to  $2 < w_{bi} < 3$  and investigate WGRP(1,  $2 < w_{bi} < 3$ ). A lower bound of  $n + 1 c(\pi)$  for  $dist(\pi)$  is given in Lemma 2, and on the other hand, taking the parameters  $\Delta h_r$  and  $\Delta h_{bi}$  into account can establish another lower bound. Let  $\Delta(c-h)_r = \Delta c_r - \Delta h_r$  and  $\Delta(c-h)_{bi} = \Delta c_{bi} - \Delta h_{bi}$ . We know that  $\Delta h_r \leq 2$  and  $\Delta(c-h)_r \leq 1$  from the literature [1], and subsequent work is required to obtain a lower bound of  $\Delta h_{bi}$  for bounding  $\Delta(c-h)_{bi}$ .

Let  $bi$  be a block-interchange and  $V_{bi}$  be the set of vertices connected to the black edges of  $bi$ . If a hurdle  $H$  has no vertices of  $V_{bi}$  in its interval  $I_H$ , then after performing  $bi$ ,  $I_H$  still contains all vertices of  $H$  but no vertices in other unoriented components, i.e.,  $H$  will be unchanged in  $BP(bi \cdot \pi)$ . This provides that  $\Delta h_{bi} \geq -h(V_{bi})$ , where  $h(V_{bi})$  is the number of hurdles including vertices of  $V_{bi}$ , since there are  $h(V_{bi})$  hurdles whose intervals contain the elements in  $V_{bi}$  and performing  $bi$  removes  $h(V_{bi})$  hurdles at most. By using the bound for  $\Delta h_{bi}$ , Lemma 3 immediately derives an upper bound for  $\Delta(c-h)_{bi}$ .

**Lemma 3** For every permutation and block-interchange  $bi$ ,  $\Delta(c-h)_{bi} \leq 3$ .

**Proof:** Let  $c_a(V_{bi})$  be the number of cycles containing vertices of  $V_{bi}$  after performing  $bi$ . Clearly,  $c(V_{bi}), c_a(V_{bi}) \in \{1, 2, 3, 4\}$  and recall that  $c_a(V_{bi}) - c(V_{bi}) = c(bi \cdot \pi) - c(\pi) \leq 2$ . We prove this lemma by first considering the achievable situations of  $c(V_{bi}) = 4$  and  $c_a(V_{bi}) = 4$ . Lemma 1 demonstrates that the only possibility for  $c_a(V_{bi}) = 4$  is  $\Delta c_{bi} = 4 - 2 = 2$ , in which the two cycles including vertices of  $V_{bi}$  belong to a component. Consequently,  $\Delta h_{bi} \geq -h(V_{bi}) \geq -1$ , and then  $\Delta(c-h)_{bi} \leq 2 - (-1) = 3$ . Using a similar argument, another case of  $c(V_{bi}) = 4$  has  $\Delta c_{bi} = 2 - 4 = -2$  and  $h(V_{bi}) \leq$



$c(V_{bi})$ , indicating that  $\Delta(c - h)_{bi} \leq -2 - (-4) = 2$ . Both cases satisfy this lemma.

Next, consider that  $c(V_{bi}), c_a(V_{bi}) \in \{1, 2, 3\}$  is sufficient to show the remaining instances. In these cases, we have  $\Delta h_{bi} \geq -h(V_{bi}) \geq -c(V_{bi})$ , and thus  $\Delta(c - h)_{bi} \leq (c_a(V_{bi}) - c(V_{bi})) - (-c(V_{bi})) = c_a(V_{bi}) \leq 3$ . This completes the proof.  $\square$

Next, from Lemma 3, we compute another lower bound for  $dist(\pi)$ . HP proved that one must decrease  $dist_r(\pi) = n + 1 - (c(\pi) - h(\pi) - f(\pi))$  to 0 to complete the sorting process if only reversals are allowed, where  $f(\pi)$  is the characteristic function for the existence of a fortress, i.e.,  $f(\pi)$  is 1 if  $\pi$  is a fortress and 0 otherwise. In addition, by using a similar argument as Lemma 2, since  $\Delta(c - h)_r \leq 1$  and  $\Delta(c - h)_{bi} \leq 3$ , an operation of increasing  $c(\pi) - h(\pi)$  by one costs at least  $\min\{\frac{w_r}{1}, \frac{w_{bi}}{3}\}$ , which equals  $\frac{w_{bi}}{3}$  when  $2 < w_{bi} < 3$ . There are, however, at least  $n + 1 - c(\pi) + h(\pi)$  to be increased, leading to a lower bound for  $dist(\pi)$  in the following lemma.

**Lemma 4**  $dist(\pi) \geq \frac{w_{bi}}{3} (n + 1 - c(\pi) + h(\pi))$  for  $WGRP(1, 2 < w_{bi} < 3)$ .

After obtaining two lower bounds of  $dist(\pi)$ , we can evaluate the approximation ratios of two proposed algorithms, **AWGRP(1,2)** and **ASBR**, as they are employed to solve  $WGRP(1, 2 < w_{bi} < 3)$ , where **ASBR** is an algorithm used to optimally solve SBR.

**Theorem 2** **ASBR** is an approximation algorithm for  $WGRP(1, 2 < w_{bi} < 3)$  with a ratio close to  $\frac{3}{w_{bi}}$ .

**Proof:** The sorting series given by **ASBR** comprises  $dist_r(\pi)$  reversals and therefore, to be an approximation algorithm for  $WGRP(1, 2 < w_{bi} < 3)$ , **ASBR** has the factor close to  $\square$

$$\frac{n+1-c(\pi)+h(\pi)+f(\pi)}{\frac{w_{bi}}{3}(n+1-c(\pi)+h(\pi))} \approx \frac{3}{w_{bi}}$$

In Theorem 2, we bypass the effect of  $f(\pi)$  for two reasons: First, the probability that a random signed permutation of size  $n$  contains a fortress is  $\Theta(n^{-15})$ , which is extremely rare [26]. Second, HP illustrated the concept of fortress with a permutation  $\pi$  having  $dist_r(\pi) = 23 + 1 - 12 + 3 + 1 = 16$  [1], which is, in fact, the minimal  $dist_r(\pi)$  for a permutation being a fortress. In other words, for  $f(\pi) = 1$ , the

ratio is at most  $\frac{3}{w_{bi}} + \frac{1}{5w_{bi}}$  when  $2 < w_{bi} < 3$ , which is nearly  $\frac{3}{w_{bi}}$ .

**Theorem 3** **AWGRP(1,2)** is a  $\frac{w_{bi}}{2}$ -approximation algorithm for  $WGRP(1, 2 < w_{bi} < 3)$ .

**Proof:** For sorting a permutation  $\pi$  with only oriented components, HP presented that  $\phi(\pi) = b(\pi) - c(\pi)$  reversals are sufficient, where  $b(\pi)$  is the number of black edges in  $\pi$ . More specifically, for sorting an oriented component  $C$ , we need  $\phi(C) = b(C) - c(C)$  reversals, in which  $b(C)$  (resp.  $c(C)$ ) is the number of black edges (resp. cycles) in  $C$ . Similarly, if sorting a set  $\mathcal{OC}_\pi$  of oriented components, an **ASBR** will produce  $\phi(\mathcal{OC}_\pi) = \sum_{C \in \mathcal{OC}_\pi} \phi(C)$  reversals, which is also the same in **AWGRP(1,2)**. When dealing with a set  $\mathcal{UC}_\pi$  of unoriented components, **AWGRP(1,2)** constructs  $\frac{\phi(\mathcal{UC}_\pi)}{2}$  block-interchanges since each decreases  $\phi(\mathcal{UC}_\pi)$  by two.

To convert  $\pi$  to  $I$ , **AWGRP(1,2)** outputs a sorting series with weight sum  $\phi(\mathcal{OC}_\pi) + w_{bi} \frac{\phi(\mathcal{UC}_\pi)}{2}$ , and a lower bound of  $dist(\pi)$  is  $\phi(\pi) = n + 1 - c(\pi)$  by Lemma 2. As a result, **AWGRP(1,2)** is an approximation algorithm for solving  $WGRP(1, 2 < w_{bi} < 3)$  with the factor given by  $\square$

$$\frac{\phi(\mathcal{OC}_\pi) + w_{bi} \frac{\phi(\mathcal{UC}_\pi)}{2}}{\phi(\pi)} \leq \frac{w_{bi} \frac{\phi(\mathcal{OC}_\pi)}{2} + w_{bi} \frac{\phi(\mathcal{UC}_\pi)}{2}}{\phi(\pi)} = \frac{w_{bi} \frac{\phi(\pi)}{2}}{\phi(\pi)} = \frac{w_{bi}}{2}$$

Theorems 2 and 3 give the approximation ratios of **ASBR** and **AWGRP(1,2)**, respectively, for approaching  $WGRP(1, 2 < w_{bi} < 3)$ , where their ratios are both at most 1.5. By always selecting the better result of **AWGRP(1,2)** and **ASBR**, we receive a smaller ratio of  $\min\{\frac{w_{bi}}{2}, \frac{3}{w_{bi}}\}$ , whose maximum is  $\sqrt{6}/2 \approx 1.225$  when the two terms coincide.

**WGRP( $w_r = 1, 1 \leq w_{bi} < 2$ )**

In the sequel, we readjust the weight of block-interchanges to  $1 \leq w_{bi} < 2$  and examine  $WGRP(1, 1 \leq w_{bi} < 2)$ .

Two lower bounds mentioned above,  $\frac{w_{bi}}{3} (n + 1 - c(\pi) + h(\pi))$  and  $\phi(\pi)$ , are not proper here since the former is too small and the latter is no longer correct. A concise way to

obtain a feasible lower bound is to take all oriented components in  $\pi$  as unoriented ones. Owing to the increase of at most two cycles by a block-interchange, a lower bound of  $dist(\pi)$  for  $WGRP(1, 0 < w_{bi} < 2)$  is  $w_{bi} \frac{\phi(\pi)}{2}$ .

With the bound, then we have the following theorem.

**Theorem 4**  $AWGRP(1,2)$  is a  $\frac{2}{w_{bi}}$ -approximation algorithm for  $WGRP(1, 0 < w_{bi} < 2)$ .

**Proof:** Recall that  $AWGRP(1,2)$  produces a sorting series with  $\varphi(\mathcal{OC}_\pi)$  reversals and  $\frac{\phi(\mathcal{UC}_\pi)}{2}$  block-interchanges. Consequently, to be an approximation algorithm for  $WGRP(1, 0 < w_{bi} < 2)$ ,  $AWGRP(1,2)$  has the factor of  $\square$

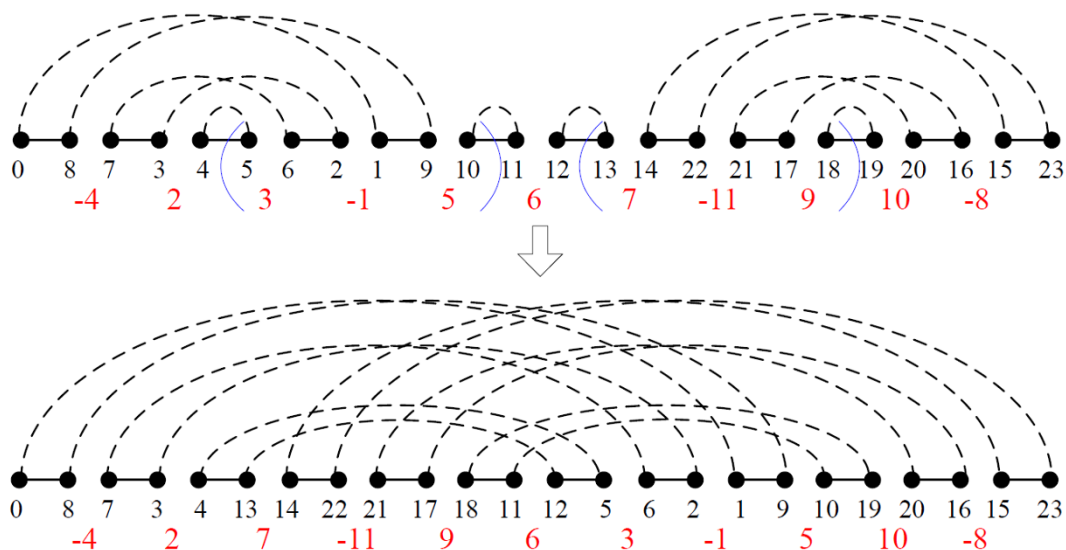
$$\frac{\phi(\mathcal{OC}_\pi) + w_{bi} \frac{\phi(\mathcal{UC}_\pi)}{2}}{w_{bi} \frac{\phi(\pi)}{2}} \leq \frac{\phi(\mathcal{OC}_\pi) + \phi(\mathcal{UC}_\pi)}{w_{bi} \frac{\phi(\pi)}{2}} = \frac{\phi(\pi)}{w_{bi} \frac{\phi(\pi)}{2}} = \frac{2}{w_{bi}}$$

Since reversals are main mutations from the evolutionary viewpoint, its weight is often no more than weights of other mutations. Therefore, we focus on improving the algorithm to efficiently cope with  $WGRP(1, 1 \leq w_{bi} < 2)$ .

We first observed the variation of the approximation ratio in Theorem 4. When  $w_{bi}$  is close to 1, the factor  $\frac{2}{w_{bi}}$  approaches 2, which is insufficient to be used in practice.

There are two ways to approach this inefficiency. The first is to make the lower bound higher by considering the fact that block-interchanges do not remove oriented components, and thus, an oriented component has at least one reversal to sort it. However, this does not indicate that  $\frac{\phi(\pi)}{2} + k$  is a new lower bound for  $k$  oriented components contained in  $\pi$ , since an operation may merge most of the oriented components into a single one. Figure 4 is an example of this, and this type of operations may result in the overestimate of  $\frac{\phi(\pi)}{2} + k$  becoming a lower bound. Therefore, we slightly enhance the lower bound by considering that if there is a permutation  $\pi$  whose  $BP(\pi)$  contains an oriented component, then  $dist(\pi) \geq w_{bi} \frac{\phi(\pi)-1}{2} + w_r$ , where the result of  $\varphi(\pi) - 1$  is caused by an optimal reversal.

Next, we improve the algorithm by adding a new component. When  $1 \leq w_{bi} < 2$ , the block-interchange is superior to the reversal since the former decreases  $\varphi(\pi)$  by at most two whereas the latter decreases it by at most one. Therefore, a straightforward idea is to use optimal block-interchanges whenever possible. Theorem 1 says that if two gray edges are unoriented and overlapping, then the corresponding block-interchange has  $\Delta c_{bi} = 2$ , which is true regardless of oriented or unoriented components. Nevertheless, there may be no gray edges to satisfy the conditions of Theorem 1 in oriented components. Whenever there are no gray edges to form a block-interchange, we



**Figure 4**  
The illustrated block-interchange merges four oriented components into one at a time.



adapt a heuristic method to choose the oriented gray edge  $oge$  with maximum  $P(oge) = N(ooge) - N(ouge)$ , where  $N(ooge)$  and  $N(ouge)$  are the number of oriented and unoriented gray edges overlapping with  $oge$ , respectively.

Let  $oge = (\pi_i, \pi_j)$  be an oriented gray edge, and  $r_{oge}$  be a reversal defined by two black edges linking  $\pi_i$  and  $\pi_j$ . Then, we immediately know that  $i + j$  is even, and hence, both  $i$  and  $j$  are either even or odd. The reversal  $r_{oge}$ , irrespective of "even" or "odd" case, results in breaking a cycle into two smaller ones, i.e.,  $\Delta c_{r_{oge}} = 1$ , as demonstrated in Figure 5. Notice that an  $oge$  can correspond to a reversal having  $\Delta c_r = 1$ , and it is false conversely, i.e., not all optimal reversals can map to oriented gray edges; take  $\bar{\pi} = (-1, -2, -3)$  and  $r(2, 2)$  as an example. Besides, a reversal  $r_{oge}$  complements the gray edges overlapping with  $oge$ . In other words, after applying  $r_{oge}$ , oriented gray edges overlapping with  $oge$  become unoriented and vice versa. The heuristic used to compute  $P(oge)$  and select the maximum results from which we want to leave as many unoriented gray edges as possible after performing a reversal. Then, the algorithm is summarized as follows:

**Approximation Algorithm for WGRP( $w_r=1, 1 \leq w_{bi} < 2$ ) (AAWGRP(1,1))**

**Input:** A signed permutation  $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2, \dots, \bar{\pi}_n)$ .

**Output:** A sorting series composed of reversals and block-interchanges for transforming  $\bar{\pi}$  into  $\bar{I}$ .

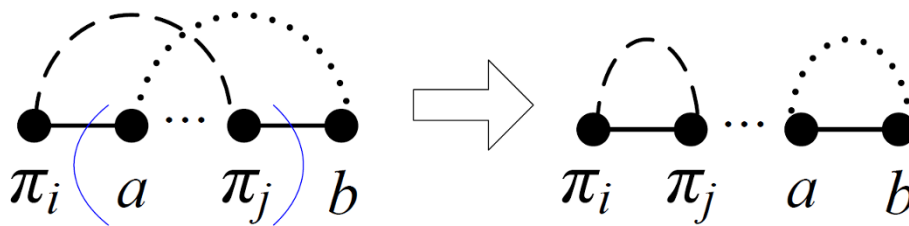
- 1: Transform  $\bar{\pi}$  into its unsigned mapping  $\pi$  and construct  $BP(\pi)$ ;
- 2: **While**  $\pi$  is not sorted

- 3: Repeatedly apply block-interchanges if Theorem 1 holds;
- 4: Compute  $P(oge)$  for each oriented gray edge  $oge$ ;
- 5: Select the maximum  $P(oge)$  and perform the corresponding reversal;
- 6: **End while**;
- 7: Mimic the sorting series of  $\pi$  to  $I$  to the transformation between  $\bar{\pi}$  and  $\bar{I}$ ;

**Lemma 5** After  $O(\varphi(\pi))$  steps, the algorithm AAWGRP(1,1) stops and returns a sorting series for converting  $\bar{\pi}$  to  $\bar{I}$ .

**Proof:** Let  $\pi$  be the unsigned mapping of  $\bar{\pi}$ . The block-interchanges used in Step 3 and reversals in Step 5 have  $\Delta c_{bi} = 2$  and  $\Delta c_r = 1$ , respectively. In other words,  $\varphi(\pi) = n + 1 - c(\pi)$  is strictly decreased after each applied operation. Due to this fact, AAWGRP(1,1) terminates after performing at most  $\varphi(\pi)$  operations.  $\square$

Now, let us examine the time complexity of AAWGRP(1,1). Step1 and Step7 are mentioned in AWGRP(1,2), and the two steps require  $O(n)$  and  $O(n \log n)$  time, respectively. To find two unoriented overlapping gray edges, a linear cost to scan  $\pi$  is sufficient. Applying a block-interchange also spends linear time, indicating that the running time to execute Step3 once is  $O(n)$ . The computation of  $P(oge)$  for an oriented gray edge  $oge$  can be done simply by visiting the vertices that lay on the interval of  $oge$  one by one, and then counting the number of oriented and unoriented gray edges overlapping with  $oge$ , which costs  $O(n)$  time at most. Furthermore, at most  $n$  computations for  $P(oge)$  implies that Step4 can be done within  $O(n^2)$  time. In Step5, an  $O(n)$ -time cost is needed to select the maximum  $P(oge)$  and next perform a corre-



**Figure 5**  
The reversal specified by a pair of blue parentheses comes from an oriented gray edge  $(\pi_i, \pi_j)$ , in which  $i$  and  $j$  are even.

sponding reversal. Therefore, to apply a reversal, the time complexity is  $O(n^2)$ . Finally, **AAWGRP(1,1)** terminates after constructing at most  $\phi(\pi)$  operations, and consequently, it takes at most  $O(n^3)$  time in the worst case.

Comparing **AAWGRP(1,1)** with **AWGRP(1,2)**, the former is preferable to the latter when analyzing oriented components provided that  $1 \leq w_{bi} < 2$ . **AAWGRP(1,1)** seems feasible for producing a sorting scenario with a smaller sum of weights, but its performance in worst cases is the same as that of **AWGRP(1,2)** for solving  $WGRP(1, 1 \leq w_{bi} < 2)$ . This is a consequence of certain specific permutations in which their weight sums conducted by both **AAWGRP(1,1)** and **AWGRP(1,2)** are far from the corresponding lower bounds. For example, if  $\pi$  has  $k$  oriented components, each with a 2-cycle only, in its  $BP(\pi)$ , then both **AAWGRP(1,1)** and **AWGRP(1,2)** output  $k$  reversals; however, the lower bound is just  $w_{bi} \frac{\phi(\pi)-1}{2} + w_r = \frac{k+1}{2}$  when  $w_r = w_{bi} = 1$ . Due to the existence of these challenging cases, the approximation ratio of **AAWGRP(1,1)** is identical to that of **AWGRP(1,2)** when they are used to analyze  $WGRP(1, 1 \leq w_{bi} < 2)$ .

$WGRP(w_r = 1, 3 \leq w_{bi})$

$WGRP(1, 3 \leq w_{bi})$  can be easily solved by considering the fact that an arbitrary block-interchange can be mimicked by three specific reversals. For example, performing the block-interchange  $bi(2, 4, 6, 7)$  on  $\vec{\pi} = (2, -5, -3, -4, -6, 7, 1)$  is the same as doing three reversals of  $r(2, 5)$ ,  $r(3, 7)$  and  $r(2, 4)$  in turn on  $\vec{\pi}$ . In other words, as long as a rearrangement sequence consists of a block-interchange, it can be replaced by three corresponding reversals without increasing the weighted sum. As a result, an **ASBR** is sufficient to optimally solve  $WGRP(1, 3 \leq w_{bi})$ , and its best running-time to date is  $O(n^{3/2})$  [6].

## Results and Discussion

### Simulation

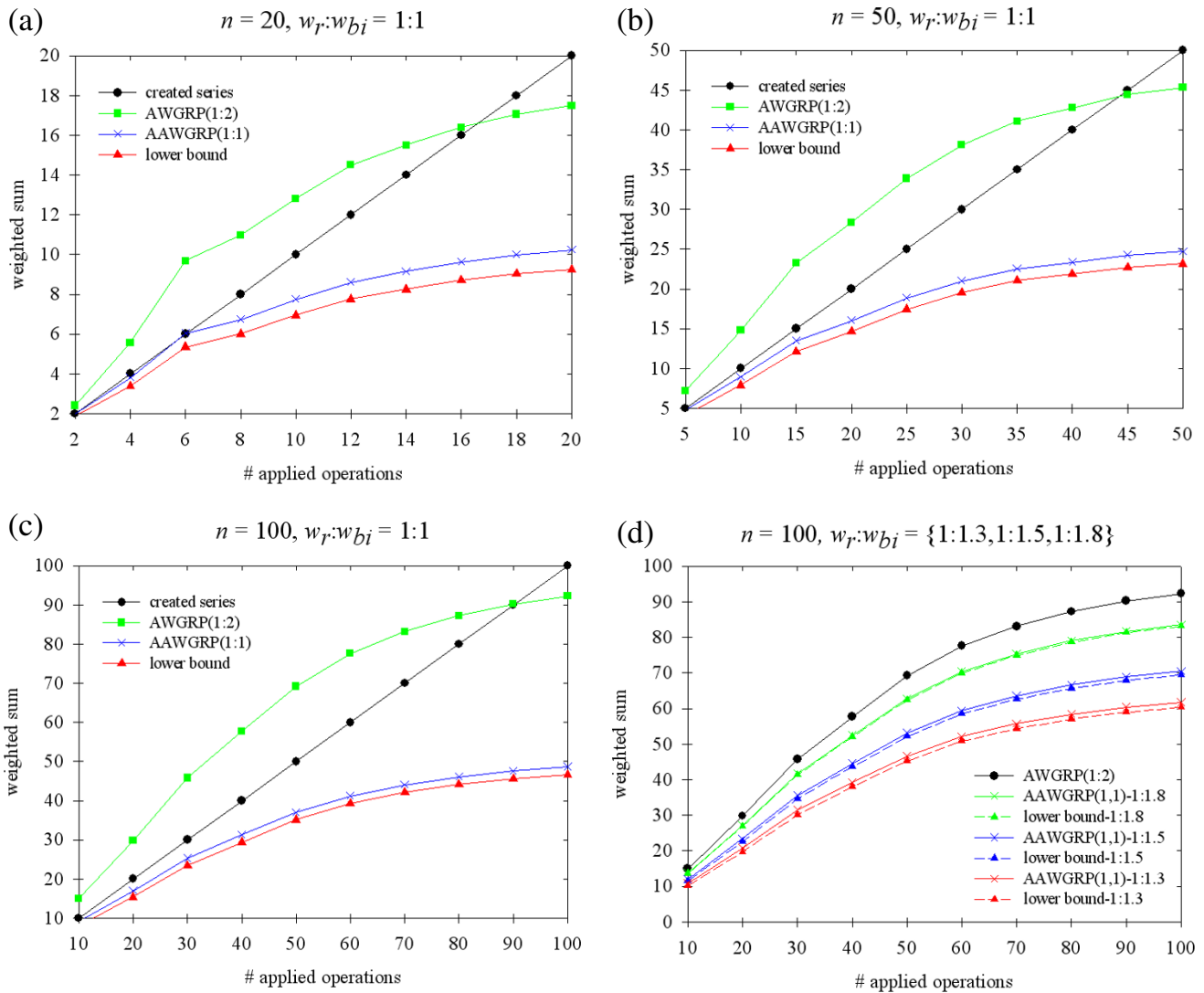
Despite the appearance of difficult cases with **AAWGRP(1,1)**, it works well in the general situation, even very close to the lower bounds when  $w_{bi}$  is near 2. To assess its performance, we conducted several experiments with the sample data generated by applying  $\alpha n$  operations on  $\vec{I} = (1, 2, \dots, n)$ , where  $n \in \{20, 50, 100\}$  and  $\alpha \in \{0.1, 0.2, \dots, 0.9, 1\}$ . The rearrangement operations of either reversals or block-interchanges were selected randomly with equal probability, and each operation was specified at

random by selecting two (for reversals) or four (for block-interchanges) integers ranging from 1 to  $n$ . Moreover, we examined  $10n$  test cases and kept track of the mean for each pair of  $\alpha$  and  $n$ .

At the beginning, we considered  $WGRP(1, 1)$ . Then for the simulated data, we computed the corresponding lower bounds as well as the average weight sums of sorting sequences created by **AAWGRP(1,1)**. For comparison, the results of **AWGRP(1,2)** were also marked (Figure 6). The weight sums of four sources, created series, **AWGRP(1,2)**, **AAWGRP(1,1)** and lower bounds, increased with the number of applied operations, but at different rates. Furthermore, in the first three diagrams of Figure 6, regardless of the size  $n$  or the number of applied operations on permutations, the two curves corresponding to **AAWGRP(1,1)** and the lower bound exhibited the same relative behavior, with only a small gap between them (about 80% of the gaps between the curves were within 2 in the experiment of Figure 6c). This result indicates that **AAWGRP(1,1)** consistently produces a closer estimate of the exact  $dist(\pi)$  for  $WGRP(1, 1)$ .

Subsequently, in Figure 6d, we fixed  $n = 100$  and adjusted  $w_{bi} = 1.3, 1.5$ , and  $1.8$  individually to investigate  $WGRP(1, 1.3)$ ,  $WGRP(1, 1.5)$ , and  $WGRP(1, 1.8)$ , respectively. Note that although three problems were included, we only plotted a curve to represent **AWGRP(1,2)**. In addition to simplifying the chart, there was hardly any difference among the reconstructed sequences of **AWGRP(1,2)** for the three problems. In other words, the vast majority of operations in the sorting sequences of **AWGRP(1,2)** were reversals, and hence, their weight sums for the three problems were virtually identical. This phenomenon is expected based on two facts: First, the probability that a component will be unoriented is the same as that of a hurdle, which is  $\Theta(n^{-2})$  on a random permutation of size  $n$  [26]. Second, the strategy of **AWGRP(1,2)** to remove oriented components is to use an **ASBR** to generate reversals. As a result, the components of the generated permutations are generally oriented, and the sorting sequences of **AWGRP(1,2)** consist mostly of reversals.

Notwithstanding **AWGRP(1,2)** was shown to be a factor 2 approximation algorithm for  $WGRP(1, 1)$  by Theorem 4, it is indeed infeasible in our experiments. The performance of **AWGRP(1,2)** is gradually improved as  $w_{bi}$  moves towards 2 (Figure 6d). In contrast, **AAWGRP(1,1)** improves dramatically when  $1 \leq w_{bi} < 2$ . Figure 6d suggests that the performance of **AAWGRP(1,1)** is superior to that of **AWGRP(1,2)** in such cases. Even in our simulation of  $w_{bi} = 1.8$ , two curves of **AAWGRP(1,1)** and the lower bound were almost the same (most of their differences were less than 1).



**Figure 6**  
**The diagrams (a), (b) and (c) consist of four curves each whereas (d) has four sets of curves, corresponding to the values of simulations and theoretical estimations. Specifically in (d), the expression of "AAWGRP(1,1)-1:1.3" means that AAWGRP(1,1) was used to solve WGRP(1, 1.3), and "lower bound-1:1.3" means the lower bound for WGRP(1, 1.3).**

**Contribution**

A large body of work has been devoted to genome rearrangement problems to study the evolutionary changes in the macrostructure of individual chromosomes according to the parsimonious principle. Here, we investigated the *Weighted Genome Rearrangement Problem* by considering reversals and block-interchanges simultaneously with various weight assignments, i.e.,  $WGRP(w_r, w_{b_i})$ . Our objective was to find a rearrangement series composed of reversals and block-interchanges for converting  $\bar{\pi}$  to  $\bar{I}$ , as well as the most parsimonious series, that is, the mini-

imum weight sum. We began studying the algorithm  $WGRP(w_r, w_{b_i})$  by setting  $w_r = 1$  and  $w_{b_i} = 2$ , and then developed  $AWGRP(1,2)$  to optimally solve it. The idea used in  $AWGRP(1,2)$  is similar to that of Lin *et al.* [21] but differs when coping with unoriented components. We also provided a rigorous proof to show the correctness of  $AWGRP(1,2)$ .

Furthermore, we adjusted the weight of block-interchanges so that  $2 < w_{b_i} < 3$  to study  $WGRP(1, 2 < w_{b_i} < 3)$ . Two algorithms ASBR and  $AWGRP(1,2)$  were employed as approximation algorithms, whose ratios were given by

**Table 1: Summary of our current and previous results for solving WGRP( $w_r, w_{bi}$ ).**

$w_r$	$w_{bi}$	Results
1	$1 \leq w_{bi} < 2$	$2/w_{bi}$ -app. with $O(n^3)$ time $O(n^{3/2})$ -time algorithm
	$2 < w_{bi} < 3$	1.225-app. with $O(n^{3/2})$ time
	$3 \leq w_{bi}$	$O(n^{3/2})$ -time algorithm [6]

Theorems 2 and 3, respectively. The approximation ratio of ASBR is  $\frac{3}{w_{bi}}$ , and hence it decreases if  $w_{bi}$  is close to 3;

however, the ratio of AWGRP(1,2)  $\frac{w_{bi}}{2}$ , which decreases when  $w_{bi}$  is near 2. Even if both factors are at most 1.5 for  $2 < w_{bi} < 3$ , their behaviors are completely opposite. Consequently, we obtained a better result by always selecting the best output of the two algorithms to acquire a smaller approximation ratio around 1.225.

Later, the weight of block-interchanges is again varied to fit WGRP( $1, 1 \leq w_{bi} < 2$ ). To address this problem, we first showed that AWGRP(1,2) is a  $\frac{2}{w_{bi}}$ -approximation algorithm. Nevertheless, the factor becomes larger as  $w_{bi}$  moves towards 1. From our experimental results on WGRP(1, 1), most of the weighted sums of sorting sequences provided by AWGRP(1,2) were more aggravated than the weighted sums of created sequences. Therefore, we improved it with AAWGRP(1,1) by adding a new component for selecting operations. Our idea was to choose as many best block-interchanges as possible, and determine plausible candidates for the best reversals once no best block-interchanges were available. As a heuristic, AAWGRP(1,1) does not have a smaller approximation ratio than AWGRP(1,2).

Consequently, we conducted several experiments to evaluate its performance and illustrated the results in Figure 6. Our result indicated that, although the theoretical approximation ratio of AAWGRP(1,1) trends towards 2 if  $w_{bi}$  is close to 1, its average performance is significantly improved. Table 1 further summarizes our current and previous results for solving WGRP( $w_r, w_{bi}$ ).

## Conclusion

In this work, we present several approaches to examine genome rearrangement problems by considering reversals and block-interchanges together under various weight assignments. Provided that the weight of reversals is no more than that of block-interchanges, our algorithm

reports an acceptable solution with theoretical guarantees and experimental evidences. Our results are promising, and these approaches should be used as an initial step for considering the two operations simultaneously. Future research must focus on improving both the approximation ratios and running times of these algorithms.

## Authors' contributions

YCL conceived the research, implemented the program and wrote the manuscript. CYL provided comments and discussion, and also assisted in revising the paper. CRL helped to draft and revise the manuscript. All authors read and approved the final manuscript.

## Acknowledgements

We would like to thank the anonymous referees for many constructive comments during the revision. Part of this work was supported by the National Science Council (NSC) under grant NSC97-2221-E-182-033-MY3 and NSC96-2628-E-110-010-MY3.

## References

- Hannenhalli S, Pevzner PA: **Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals.** *J ACM* 1999, **46**:1-27.
- Kaplan H, Shamir R, Tarjan RE: **A Faster and simpler algorithm for sorting signed permutations by reversals.** *SIAM J Comput* 1999, **29**:880-892.
- Bergeron A: **A very elementary presentation of the Hannenhalli-Pevzner theory.** *Dis Math* 2005, **146**:134-145.
- Bergeron A, Mixtacki J, Stoye J: **Reversal distance without hurdles and fortresses.** In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching: 5-7 July 2004; Istanbul, Turkey Volume 3109*. Edited by: Sahinalp SC, Muthukrishnan S, Dogrusöz U. Lecture Notes in Computer Science, Springer-Verlag; 2004:388-399.
- Tannier E, Bergeron A, Sagot MF: **Advances on sorting by reversals.** *Dis Math* 2007, **155**:881-888.
- Han Y: **Improving the Efficiency of Sorting by Reversals.** In *Proceedings of the 2006 International Conference on Bioinformatics and Computational Biology: June 26-29 2006; Las Vegas, Nevada, USA* Edited by: Arabnia HR, Valafar H. CSREA Press; 2006:406-409.
- Swenson KM, Rajan V, Lin Y, Moret BME: **Sorting signed permutations by inversions in  $O(n \log n)$  time.** In *Proceedings of the 13th Annual International Conference on Research in Computational Molecular Biology: 18-21 May 2009; Tucson, Arizona Volume 5541*. Edited by: Batzoglou S. Lecture Notes in Computer Science, Springer-Verlag; 2009:386-399.
- Bader DA, Moret BME, Yan M: **A linear-time algorithm for computing inversion distance between signed permutations with an experimental study.** *J Comput Biol* 2001, **8**:483-491.
- Bafna V, Pevzner PA: **Sorting by transpositions.** *SIAM J Dis Math* 1998, **11**:221-240.
- Hartman T, Shamir R: **A simpler and faster 1.5-approximation algorithm for sorting by transpositions.** *Inf Comput* 2006, **204**:275-290.
- Elias I, Hartman T: **A 1.375-approximation algorithm for sorting by transpositions.** *IEEE/ACM Trans Comput Biol and Bioinformatics* 2006, **3**:369-379.
- Christie DA: **Sorting by block-interchanges.** *Inform Process Lett* 1996, **60**:165-169.
- Lin YC, Lu CL, Chang HY, Tang CY: **An efficient algorithm for sorting by block-interchanges and its application to the evolution of Vibrio species.** *J Comput Biol* 2005, **12**:102-112.
- Feng J, Zhu D: **Faster algorithms for sorting by transpositions and sorting by block-interchanges.** *ACM T Algorithm* 2007, **3**:1-14.
- Lin GH, Xue G: **Signed genome rearrangement by reversals and transpositions: models and approximations.** *Theoret Comput Sci* 2001, **259**:513-531.

16. Walter MEMT, Dias Z, Meidanis J: **Reversal and transposition distance of linear chromosomes.** In *Proceedings of String Processing and Information Retrieval: 9-11 September 1998; Santa Cruz, Bolivia* Edited by: Bolivia SCS. IEEE Computer Society; 1998:96-102.
17. Gu QP, Peng S, Sudborough H: **A 2-approximation algorithms for genome rearrangements by reversals and transpositions.** *Theoret Comput Sci* 1999, **210**:327-339.
18. Eriksen N: **(1+ε)-approximation of sorting by reversals and transpositions.** *Theoret Comput Sci* 2002, **289**:517-529.
19. Bader M, Ohlebusch E: **Sorting by Weighted Reversals, Transpositions, and Inverted Transpositions.** *J Comput Biol* 2007, **14**:615-636.
20. Yancopoulos S, Attie O, Friedberg R: **Efficient sorting of genomic permutations by translocation, inversion & block interchange.** *Bioinformatics* 2005, **21**:3340-3346.
21. Lin YC, Lu CL, Liu YH, Tang CY: **SPRING: a tool for the analysis of genome rearrangement using reversals and block-interchanges.** *Nucleic Acids Res* 2006, **34**:W696-W699.
22. Mira C, Meidanis J: **Sorting by Block-Interchanges and Signed Reversals.** In *4th International Conference on Information Technology: 2-4 April 2007; Las Vegas, Nevada, USA* Edited by: Latifi S. IEEE Computer Society; 2007:670-676.
23. Bader M: **Sorting by reversals, block interchanges, tandem duplications, and deletions.** *BMC Bio* 2009, **10**:S9.
24. Vazirani VV: *Approximation algorithms* New York: Springer-Verlag; 2001.
25. El-Mabrouk N, Sankoff D: **On the Reconstruction of Ancient Doubled Circular Genomes Using Minimum Reversal.** *Genome Informatics* 1999, **10**:83-93.
26. Swenson KM, Lin Y, Rajan V, Moret BME: **Hurdles hardly have to be heeded.** In *Proceedings of the 6th RECOMB Comparative Genomics Satellite Workshop: 13-15 October 2008; Paris, France Volume 5267.* Edited by: Nelson CE, Vialette S. Lecture Notes in Computer Science, Springer-Verlag; 2008:241-251.
27. Gog S, Bader M: **Fast Algorithms for Transforming Back and Forth between a Signed Permutation and Its Equivalent Simple Permutation.** *J Comput Biol* 2008, **15**:1029-1041.

Publish with **BioMed Central** and every scientist can read your work free of charge

"BioMed Central will be the most significant development for disseminating the results of biomedical research in our lifetime."

Sir Paul Nurse, Cancer Research UK

Your research papers will be:

- available free of charge to the entire biomedical community
- peer reviewed and published immediately upon acceptance
- cited in PubMed and archived on PubMed Central
- yours — you keep the copyright

Submit your manuscript here:  
[http://www.biomedcentral.com/info/publishing\\_adv.asp](http://www.biomedcentral.com/info/publishing_adv.asp)

