BIOINFORMATICS ORIGINAL PAPER
Vol. 21 no. 16 2005, pages 3340–3346
doi:10.1093/bioinformatics/bti535

Genome analysis

Efficient sorting of genomic permutations by translocation, inversion and block interchange

Sophia Yancopoulos[1,*], Oliver Attie[2] and Richard Friedberg[3]

[1]The Feinstein Institute for Medical Research, North Shore-LIJ Health System, Manhasset, NY 11030, USA,
[2]Center for the Study of Gene Structure and Function, Hunter College, NY, NY 10021, USA and
[3]Department of Physics, Columbia University, NY, NY 10027, USA

Received on May 3, 2005; revised on June 2, 2005; accepted on June 8, 2005
Advance Access publication June 13, 2005

ABSTRACT

Motivation: Finding genomic distance based on gene order is a classic problem in genome rearrangements. Efficient exact algorithms for genomic distances based on inversions and/or translocations have been found but are complicated by special cases, rare in simulations and empirical data. We seek a universal operation underlying a more inclusive set of evolutionary operations and yielding a tractable genomic distance with simple mathematical form.

Results: We study a universal double-cut-and-join operation that accounts for inversions, translocations, fissions and fusions, but also produces circular intermediates which can be reabsorbed. The genomic distance, computable in linear time, is given by the number of breakpoints minus the number of cycles (b − c) in the comparison graph of the two genomes; the number of hurdles does not enter into it. Without changing the formula, we can replace generation and re-absorption of a circular intermediate by a generalized transposition, equivalent to a block interchange, with weight two. Our simple algorithm converts one multi-linear chromosome genome to another in the minimum distance.

Contact: syancopo@nshs.edu

1 INTRODUCTION

The study of pairwise genome rearrangement is rooted in the problem of computing an edit distance based on gene order and orientation (Sankoff, 1992) rather than on nucleotide sequence. In this study it has become important to construct fast algorithms determining the minimal path. The problem of transforming one genome to another can be reduced to that of sorting a signed permutation by certain mutational operations (Sturtevant and Novitski 1941; Waterston *et al.*, 1982; Palmer and Herbon, 1986). The minimal number of operations necessary to transform two genomes is referred to as a distance, and the number of reversals, for example, has been called their reversal or inversion distance (Caprara, 1997). The challenge has been to find a consistent and biologically meaningful set of operations for which the genomic distance problem is tractable.

A variety of biological operations have been proposed to affect gene order, but no consensus has been reached on which set, if any, would be definitive. The strategy, it appears, has been to discover first how to sort genomic permutations by a single operation at a time, and then with larger, more realistic combinations of operations. Hannenhalli and Pevzner (1995a, hereafter HPa) found a polynomial-time algorithm for sorting a single chromosome by inversions, and Hannenhalli (1996) used the same techniques to design a polynomial-time algorithm for optimal rearrangements by translocations, but to date, no polynomial-time algorithm has been found for optimal sorting solely by transpositions, despite more than 10 years of research by many researchers (Bafna and Pevzner, 1995; Walter *et al.*, 2000, 2003; Hartman, 2003).

As for combinations of operations, it has not been clear how to assess the relative contributions of operations (Blanchette *et al.*, 1996) although a variety of combinations have been considered (Sankoff, 1992; Dalevi *et al.*, 2002; Meidanis and Dias, 2002). Many of these are approached by heuristics or approximation algorithms. Meidanis and Dias (2001) found efficient algorithms for transpositions, fissions and fusions. Studies which include transpositions have indicated that they should cost about twice as much as other operations, but this has not been proven rigorously despite serious attempts to do so (Eriksen, 2002).

Perhaps the most realistic combination to date has been solved by Hannenhalli and Pevzner (1995b, hereafter HPb) for inversions, translocations, fissions and fusions. They showed that distance can be calculated close to linear time. The algorithm has subsequently been improved by a number of people including Bergeron (2001), Tesler (2002) and Ozery-Flato and Shamir (2003).

Transpositions (exchange of two contiguous segments) have been less favored as fundamental evolutionary operations. A justification may be that large-scale transpositions are much less frequently observed than inversions and translocations, but the apparent computational intractability of transpositions is undoubtedly as important a factor in neglecting them. There is clearly evidence for the occurrence of transpositions (Andersson and Eriksson, 2000; Dalevi *et al.*, 2002; Kent *et al.*, 2003) The rarity of computationally observed transpositions may be a case of not finding what you do not look for. Also, a surprisingly high percentage of DNA may originate from transposons or other mobile elements (Deininger and Roy-Engel, 2001).

A combination problem of great biological relevance should include transpositions in addition to inversions, translocations, fissions and fusions. In this paper, we show how this problem becomes far more tractable if the transposition operation is generalized to what Christie (1996) has called a block interchange (exchange of any two segments).

*To whom correspondence should be addressed.

2 OPERATIONS AND ALGORITHM

We assume that genomes A and B have the same gene content, organized into synteny blocks. A synteny block (SB) is a maximal sequence of genes on a chromosome of genome A, occurring unchanged in genome B. As is customary, we deal with the two ends (vertices) of each SB as independent entities. If we connect them in pairs arbitrarily, this defines a genome with circular chromosomes. By introducing extra single vertices Hannenhalli and Pevzner (HP) call 'caps' denoting the end of a chromosome, we can represent linear chromosomes. Following others, we introduce null chromosomes: two connected caps.

A breakpoint is the connection between two successive SB in either genome, between an SB and a cap (provided the same connection does not occur in the other genome), or between two caps in a null chromosome.

2.1 The double-cut-and-join operation

We call our elementary operation, 'double-cut-and-join' (DCJ), a local operation on four vertices, initially connected in pairs. It consists of cutting two connections (breakpoints) in the first genome, and rejoining the resulting four unconnected vertices in two new pairs, which can be done in either of two ways (Fig. 1). We weight this operation 1, regardless of which end of the SB we are dealing with, whether the two initial pairs are on the same chromosome or not, or which of the two ways we rejoin. We study here the transformation of 'initial' genome A to 'target' genome B by a minimal number of DCJ operations.

Our main tool for visualization (examples in Figs 7–9, of section 3) is the breakpoint graph (also called edge graph, comparison graph) introduced by Bafna and Pevzner (1993) and used since by HP and others (Bergeron, 2001; Tesler, 2002; Ozery-Flato and Shamir, 2003). Each end of an SB, and each cap at an end of a linear chromosome, is a vertex in the graph; two vertices adjacent in genome A, but not in B, are connected by a (straight) 'black line'; two vertices adjacent in B, but not in A, are connected by a 'gray line' (usually an arc), but the line representing the SB itself is customarily not drawn. We arrange for each cap to appear only once in each genome (Section 2.3, Phase 0).

In the course of transforming genome A to B, black lines are altered until the connections correspond to those of the gray lines specified by the target genome. Each DCJ cuts two black lines and rejoins the four cut ends into two new black lines.

At every stage, each vertex is connected to one black line and one gray line. Hence each vertex is order 2, and the graph consists of separate cycles alternating black and gray lines. Consequently, the number of breakpoints in genome A (black lines) equals the number of breakpoints (gray lines) in B, in each cycle. We denote a cycle with L-black and L-gray lines as an L-cycle.

We define b as the number of breakpoints in either genome, and c as the number of cycles as is customary. If a DCJ starts by cutting two black lines belonging to different cycles, either way of rejoining them will fuse the cycle into one. Therefore c will decrease by 1, and b remains unchanged. If the two black lines to be cut belong to the same cycle, one way of rejoining, which we call 'proper', breaks the cycle in two, and the other, called 'improper', does not (Fig. 2). With improper joining, both b and c remain unchanged.

With proper joining, there are three cases: First, if the two black lines to be cut are not successive (separated by a single gray line) in the cycle, proper joining keeps b unchanged and increases c by 1 (Fig. 2). Second, if the initial cycle has more than two black lines and the two to be cut are successive, then the DCJ with proper rejoining isolates a 1-cycle, consisting of a pair of vertices connected in both genomes. Following convention, we consider the breakpoint in each genome to be eliminated and b decreases by 1. We call the eliminated breakpoints 'healed'. The 1-cycle is deemed not to exist, so c remains unchanged (Fig. 3a). Third, if the initial cycle is a 2-cycle, then two breakpoints disappear along with their 1-cycles (Fig. 3b); b decreases by 2 and c by 1. In all three cases, $b - c$ decreases by 1.

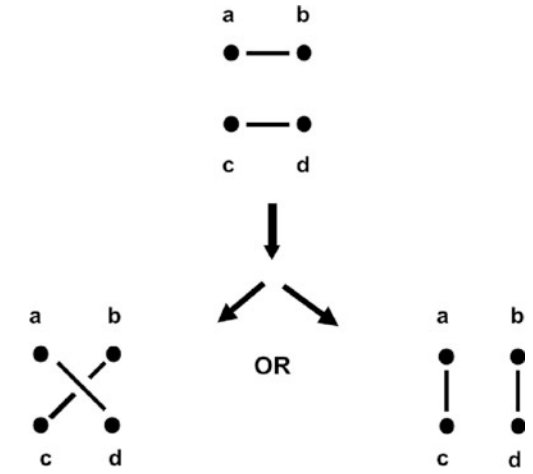

Fig. 1. The general DCJ operation. The SB to which the endpoints a, b, c and d belong are not shown. It is permitted for two of the four endpoints to represent opposite ends of the same SB.

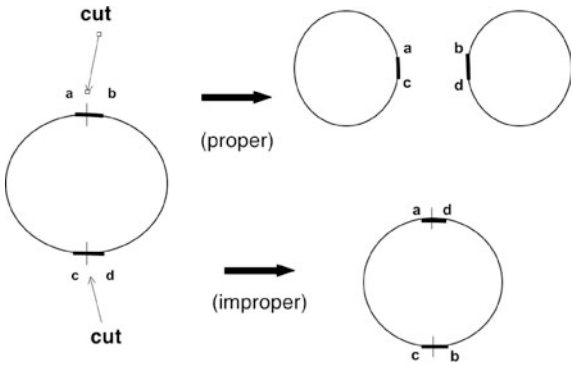

Fig. 2. A DCJ cuts the two black line connections belonging to the same HP cycle. The rest of the cycle, understood as an alternation of HP gray and black lines, is shown as a single line. 'Proper' rejoining cuts the cycle into two; 'improper' rejoining does not.

We see that a DCJ never decreases the number $b-c$ by more than 1. But since this number is zero at the end, when all breakpoints have been healed, the genomic distance, or minimum number of DCJ's to get from genome A to B, is at least as great as the initial value of $b - c$.

It is possible to choose each step so that the two black lines to be cut belong to the same cycle (since 1-cycles disappear when formed) and so that the rejoining is proper. Then $b - c$ will decrease at each step. Therefore, the optimal path will consist of DCJs in which a

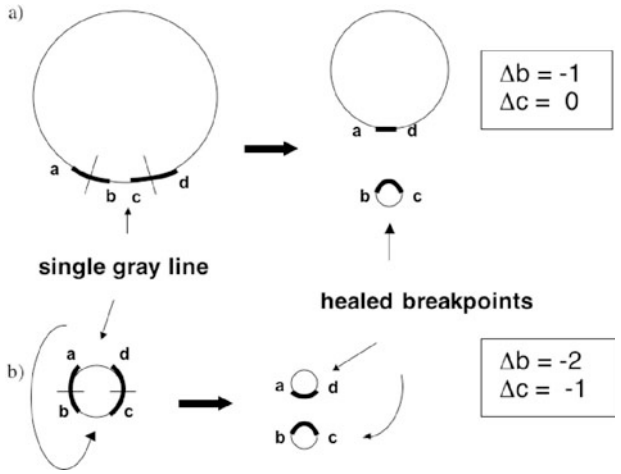

Fig. 3. (**a**) For (*ab*) and (*cd*) separated by a single gray line, adjacent in an HP cycle, proper rejoining after the double cut results in connecting b and c so that the breakpoint between them is 'healed' (eliminated) and the 1-cycle is dropped. (**b**) Both *bc* and *ad* are single gray lines so that the initial cycle has length $L = 2$. Proper rejoining heals two breakpoints and two 1-cycles are dropped.

single initial cycle is cut in two places and rejoined in the proper way. The number of steps required is $d = b - c$, where b is the initial number of breakpoints and c is the initial number of cycles.

2.2 Menu of operations on linear chromosomes

In particular contexts, we can identify the DCJ operation we have defined with more familiar operations having a recognized biological status. In this paper, we concern ourselves mainly with operations performed on linear chromosomes.

First, consider the case in which each genome has only one chromosome. In that case the cutting of a cycle in two places admits two ways of rejoining; one reverses a segment of the chromosome and the other snips out a circular fragment (Fig. 4).

In their work (HPa) on the pure reversal problem, HP speak of reversals as induced by an arc which may be 'oriented' or 'unoriented'. If the arc is oriented, it induces a reversal, equivalent in our terminology to a DCJ with proper rejoining (Fig. 4a). If unoriented, the reversal corresponds to a DCJ with improper rejoining (Fig. 4b). This is why 'hurdles', which consist entirely of unoriented arcs, add to the genomic distance, since they force one to introduce a reversal which does not decrease $b - c$. (A hurdle is a particular configuration of unoriented arcs defined in HPa). In our problem, we have a more general operation, so, upon encountering a hurdle, we can apply the same cuts (induced by an unoriented arc) as if we were doing a reversal, but with the other rejoining, which produces a circular intermediate (CI) (Fig. 4b). Being proper, this rejoining decreases $b - c$, yielding our $d = b - c$ instead of HP's $d = b - c + h + f$, with h the number of hurdles, and $f = 0$ or 1.

With more than one chromosome, a cycle may extend between them and the two cuts may be made on different chromosomes. In such a case the DCJ results in a translocation (Fig. 5). The two rejoinings differ by a reversal of one entire chromosome relative to the other, prior to the operation. Only one choice of relative direction corresponds to proper rejoining; one must examine the whole cycle to find it.

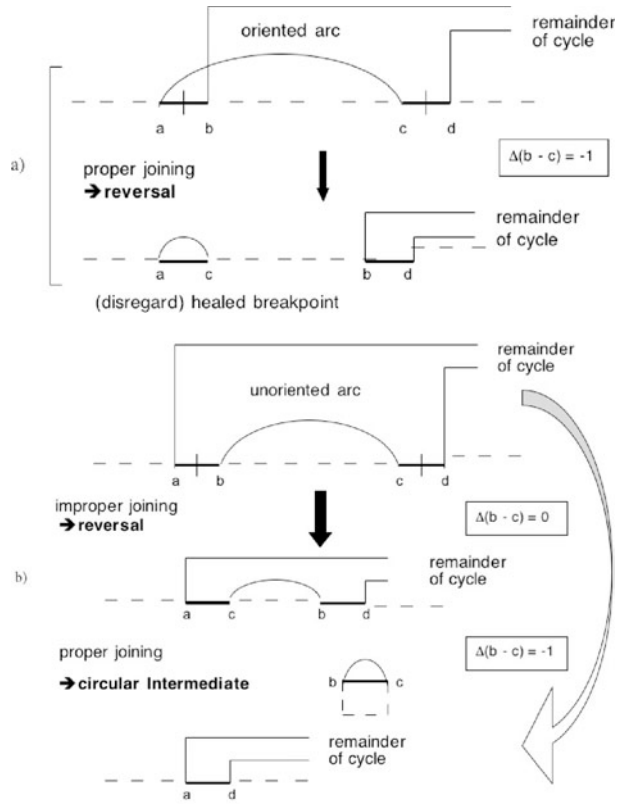

Fig. 4. (**a**) A DCJ applied to black lines (*ab*) and (*cd*) in a linear chromosome whose two left ends are joined by a gray line (called an 'oriented arc') results in a reversal (or inversion) with proper joining, decreasing $b - c$. The remainder of the chromosome is represented by dashed lines and of the cycle by thin lines. (**b**) The gray arc joining the left end of black line (*cd*) to the right end of (*ab*) is 'unoriented'. Reversal of segment bc arises from improper joining and does not increase $b - c$. Proper rejoining results in a CI with $b - c$ decreasing by 1. A block interchange results when the CI is reabsorbed after being cut somewhere other than at the 'healed' breakpoint. (Fig. 6)

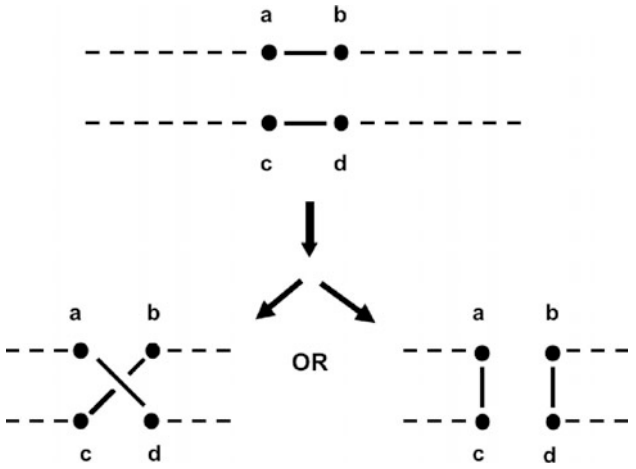

Fig. 5. A DCJ cutting two black lines on different chromosomes, but in the same cycle, results in a translocation with either rejoining. To find which is proper, one must see the whole cycle (not shown). Dashed lines represent the rest of the chromosome.

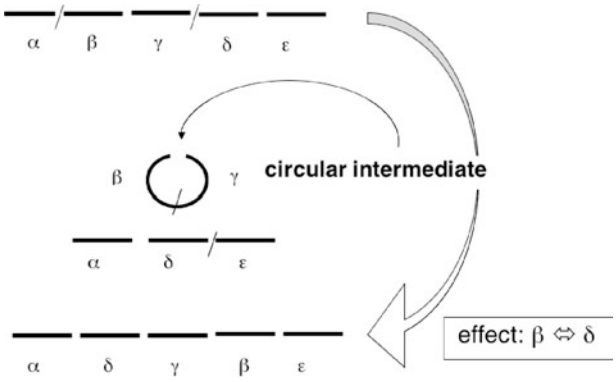

Fig. 6. A generalized transposition or block interchange is effected by creation and reabsorption of a CI, Greek letters represent segments of arbitrary length in a linear chromosome. Segment $\beta\gamma$ is cut and self-connected to make a CI. It is then cut between β and γ in the CI so that insertion occurs in the order $\gamma\beta$, resulting in exchanging β with δ. Cutting the CI in the same place where it was joined when created results in a transposition (Bafna and Pevzner, 1998; Eriksen, 2002).

By cutting off a cap from each of two linear chromosomes, one achieves a fusion, along with the production of a null chromosome consisting of the two end-blocks with a connecting line. Fission is accomplished by having a null chromosome in the initial state. Following HPb, we consider fusion and fission as special cases of translocation.

It is possible to split and fuse the CIs, and to absorb them into a linear chromosome. For initial and final genomes consisting only of linear chromosomes, it is possible to reabsorb a CI into a linear chromosome by a proper DCJ whenever it has been created, decreasing $b - c$. Without increasing the genomic distance we, therefore, can forbid fusion and fission of CIs, requiring their absorption as soon as created.

The resulting menu of operations on linear chromosomes is:

- Translocations (including fission and fusion): weight 1.
- Inversions: weight 1.
- Creation (weight 1) and immediate absorption (weight 1) of a CI.

We may further require that the CI be absorbed onto the linear chromosome that produced it, since its creation and absorption in separate chromosomes can be accomplished by two translocations. Also, creation and absorption with reversal can be achieved by two inversions. Replacing the third (dual DCJ) operation by block interchange (Fig. 6) we can rewrite the menu without CI's:

- Translocations (including fission and fusion): weight 1.
- Inversions: weight 1.
- Block interchanges: weight 2.

The block interchange operation is an exchange (without reversal) of two segments in the same chromosome, which is either contiguous or remote in the originating genome. Christie (1996) pointed out that block interchange can be viewed as a generalized transposition operation, in which exchanged segments need not be contiguous.

2.3 A simple algorithm to find an optimal path

Using our menu of operations, we devised an algorithm to find an optimal path between genomes with linear chromosomes not necessarily having the same number of chromosomes. It performs operations in a definite order: translocations (including fissions and fusions) first, then inversions, finally block interchanges. Decreasing $b - c$ by eliminating at least one breakpoint at each step, it never increases c.

2.3.1 Phase 0—end-capping (prior to any DCJ steps) We start by adding unlabeled caps to both ends of each chromosome in genome A and B in the edge graph. We connect each A-cap to the adjacent SB end by a black line, each B-cap to the adjacent SB end by a gray line, so that every SB end is the terminus of both a black and a gray line, but each cap is attached to only one line, gray or black.

We then trace a path from any A-cap along alternating black and gray lines until it terminates on another A-cap (an AA-path) or on a B-cap (an AB path). These are the same as the Pi–Pi and Pi–Gamma paths of HPb, except that no caps are added in genome B so their AB path terminates at an SB end. We repeat this for each A-cap that is not part of a path. When all A-caps have been so treated, we start a path with any B-cap that is not part of a path; it necessarily terminates on another B-cap. We repeat until all B-caps have been treated and all caps are part of an AA, AB or BB path.

We next identify the two caps at the end of each AB path with each other, closing the path into a cycle containing just one cap. HPb achieve this by adding a gray line connecting the A-cap at one end of the path to the naked 'tail' SB end at the other end.

Because they occupy different positions in the same genome, the caps at both ends of an AA path cannot be identified, and nor can those at the ends of a BB path. We do not pair up AA and BB paths as is sometimes done to reduce the number of null chromosomes to a minimum, diminishing both the number of breakpoints and the number of cycles so that $b - c$ remains the same. Instead, we connect the two caps of each AA path by a new gray line which introduces a null chromosome in genome B. Likewise we connect the two caps of each BB path by a new black line, introducing a null chromosome in genome A. Each AA or BB path is closed into a cycle containing just two caps in immediate succession, and all caps are incorporated into cycles.

We prefer our method because it treats both genomes the same and all AA and BB paths the same way, with a unique prescription and no arbitrary choices. In algorithms for sorting by reversals and translocations alone (such as HPb) chromosomes are first concatenated into one, and various additional operations and precautions are needed to manipulate the concatenation and avoid the later creation of further 'hurdles' or other obstacles, which would increase the eventual number of steps needed. These precautions sometimes require combining paths into a larger cycle. With our expanded menu of operations we need have no fear of causing more hurdles and do not need precautions, additional operations or to concatenate the chromosomes. This simplifies understanding and saves running time, requiring only $O[n]$ time .

We are now ready to undertake the DCJ steps. At each step, we refer to the genome defined by current black lines as the current genome, initially identical to genome A, and at the end to genome B. (See Section 3 for phase 0–3 examples.)

2.3.2 Phase 1—translocations (including fissions and fusions) In this phase we make no distinctions between null chromosomes and

others, nor between cycles containing caps and those that do not. Choosing any chromosome in the current genome, we follow it, starting from one of the caps, until we find an element connected by a gray line to an element in a different chromosome in the current genome. A chromosome having none is 'robust', i.e. all its material also belongs to a single chromosome in genome B. We cut the two successive black lines adjacent to the gray line, in the manner of Fig. 3. Naturally they belong to the same cycle. By proper rejoining, the cycle is cut in two, at least one of which is a 1-cycle that 'disappears' as the corresponding breakpoint is 'healed', resulting in a translocation (possibly a fission or fusion) and diminishing $b - c$ by 1.

We repeat the procedure until all chromosomes in the current genome are robust. Both genomes started phase 1 with the same number of chromosomes after padding with null chromosomes in phase 0, and the operations performed do not change this equality (fission uses up a null chromosome, fusion creates one). Chromosomes in the current genome, including caps, correspond one-to-one to those in genome B with only rearrangements within a chromosome still necessary. Fissions needed to eliminate the null chromosomes in genome A, and fusions needed to create the null chromosomes in genome B, have been carried out automatically. Null chromosomes in the current genome are now the same as in genome B. There have been no wasted steps, every step decreases $b - c$.

2.3.3 Phase 2—inversions From now on each chromosome is treated separately. We start from the beginning of the chromosome, and for each vertex we examine the gray line attached to it. If this gray line is 'oriented' (Fig. 4a) we cut the two black lines adjacent to it and rejoin in the proper way, achieving an inversion and decreasing $b - c$ by 1. If 'unoriented' (Fig. 4b) we go on to the next gray line. We continue thus until all gray lines are unoriented, so that all SBs in the chromosome have the same direction in the current genome as in genome B.

In both phases 1 and 2 it is unnecessary for us to provide against the creation of new hurdles by computing a 'score' (Bergeron, 2001), since hurdles will be handled in phase 3 without loss of efficiency. This considerably simplifies our procedure, shortening running time compared to the problem of sorting by translocations and inversions alone.

2.3.4 Phase 3—block interchanges Choosing a gray line arbitrarily, we cut the two black lines adjacent to it. Since all gray lines are now unoriented, proper rejoining will create a CI. Since genome B has no circular chromosome, there must be a gray line connecting this CI to the rest of the linear chromosome. We cut the two black lines adjacent to it, and perform a proper rejoining. Since one of these black lines was on the CI and the other on the linear chromosome, the operation reabsorbs the CI. The two DCJ's are equivalent to a block interchange (Fig. 6) decreasing $b - c$ by 2. We may regard the DCJ as a convenient way of determining which block interchange to make. We repeat until all breakpoints are healed.

Our procedure for phase 3 is the same as that of Lin *et al.* (2005) for sorting by block interchange despite differences in formulation. They set up the problem as one of sorting a permutation of SB's, equivalent to re-pairing SB ends when no SB's need to be reversed. Their 'pair exchange' is equivalent to our DCJ. Their 'split operation' is our DCJ creating a CI and their 'join operation' is our DCJ absorbing the CI.

In phases 1, 2 and 3 the time is $O[n\delta]$, where δ is the number of DCJ steps and n is the initial value of b. The reason it is not $O[n]$ is that the placement of SB's on the chromosomes must be updated after each step. If CIs were not required to be absorbed immediately and we were willing to perform the operations in any order, it would be unnecessary to keep track of which SB's are on which chromosomes, and the running time would be $O[n]$.

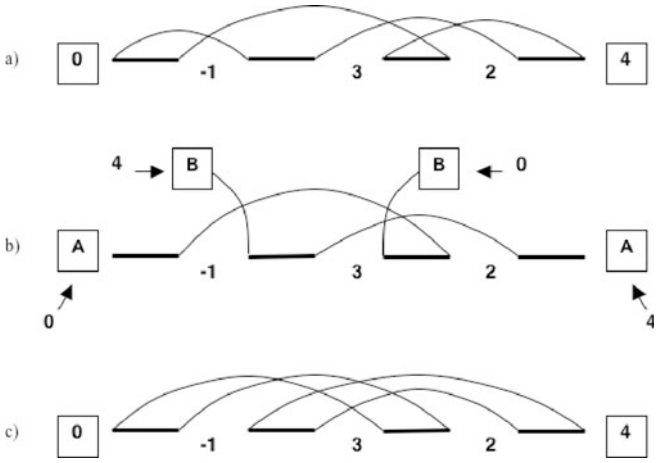

Fig. 7. Capping procedures for sorting chromosome $(-1, 3, 2)$ in genome A, to $(1, 2, 3)$ in genome B. (**a**) A naive method caps A as $(0, -1, 3, 2, 4)$ and B as $(0, 1, 2, 3, 4)$ yielding $b - c = 4 - 1 = 3$. (**b**) Our phase 0, closes the two AB paths by identifying the A cap at one end of a path with the B cap at the other end. (**c**) Performing the identifications indicated in (b) by arrows gives $b - c = 4 - 2 = 2$.

3 DISCUSSION

Closure, in phase 0, of each AB path into a cycle guarantees the number of cycles thus obtained is maximal, so that $b - c$ is minimal. Illustrated in Figure 7, the breakpoint graph for sorting $(-1, 3, 2)$ into $(1, 2, 3)$ undergoes initial capping.

As shown in Fig. 7b and c, there are two AB paths so that our capping procedure yields two cycles, and $b - c = 2$. Indeed, sorting $(-1, 3, 2)$ can be accomplished in two inversions, first to $(-3, 1, 2)$ and then $(-3, -2, -1)$, producing genome B backwards. This is quite acceptable, but if the capping is done as in Figure 7a, it effectively insists on getting genome B in the 'forwards' order, which costs an extra inversion. Such capping merges the two AB paths into one cycle so that $b - c = 4 - 1 = 3$. In our algorithm the problem of ensuring chromosomes are 'optimally flipped' is handled automatically.

Figure 8 contrasts our method of handling AA and BB paths with the more usual procedure that minimizes the number of null chromosomes by pairing AA with BB paths, where possible. Sorting $(-1, -3, 2)$ into $(1, 2, 3)$ via the usual pairing procedure (Fig. 8a) yields $b - c = 4 - 1 = 3$. Our method (Fig. 8b) produces an extra cycle, but at the cost of an extra breakpoint in the null chromosome in either genome, so that $b - c = 5 - 2 = 3$. In this case the two methods yield the same distance $b - c$; but in multi-chromosomal sorting there generally exist non-optimal ways of pairing the AA and BB paths, which would yield an unnecessarily high value of $b - c$ that reflects meaningless reversals and exchanges of whole chromosomes.

The advantages of our capping procedure do not mean we have found an easier way of dealing with the capping problem treated by HPb and later authors. They are possible because we have added block interchange to the menu.

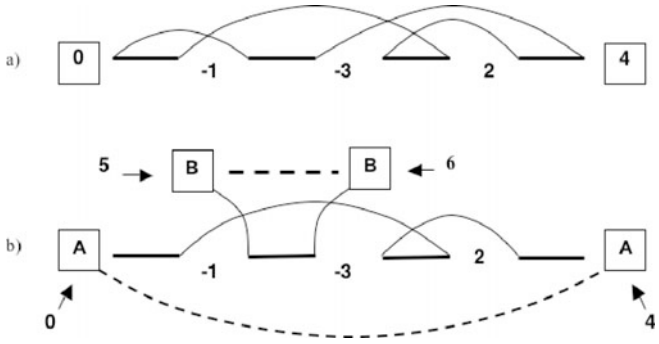

Fig. 8. A second illustration of our capping procedure. Genome A: $(-1, -3, 2)$, is to be sorted to genome B: (1, 2, 3). (**a**) The standard capping precedes the chromosome with (0) and ends with (4), giving $b-c = 4-1 = 3$. (**b**) Our phase 0 caps A with (0) and (4), and B with (5) and (6). There is an AA and a BB path. Each is closed by an extra (dotted) line which amounts to creating a null chromosome in the opposite genome providing an extra breakpoint so that $b - c = 5 - 2 = 3$. The null chromosome in A is (5,6) and in B, (0,4).

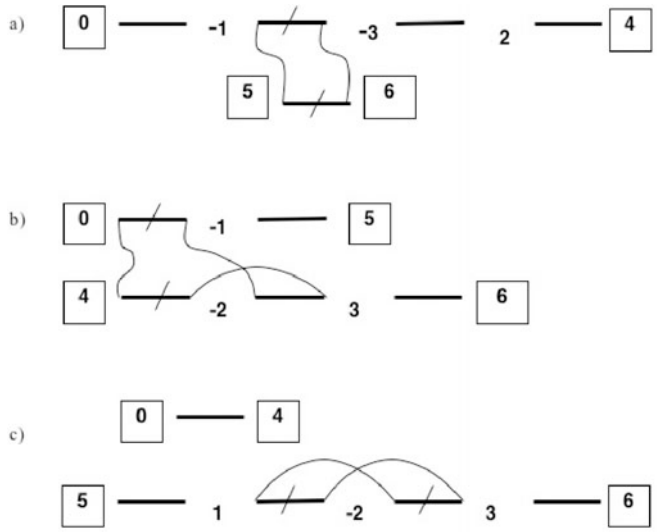

Fig. 9. How phases 1 and 2 sort Genome A of Figure 8. (**a**) Genome A with caps is $(0, -1, -3, 2, 4)$, (5,6). Our algorithm recognizes the null chromosome as a second chromosome and looks for a translocation. Proceeding along the first chromosome from cap 0, the first inter-chromosomal gray line connects the right end of -1 to the left cap of the null chromosome. The two black lines to be cut are marked with cross-slashes. Only those gray lines are shown that contribute to the cycle being cut. (**b**) The translocation indicated in (a) (actually a fission) results in two real chromosomes. The algorithm again starts from cap 0 and stops at the first inter-chromosomal gray line. This translocation will be a fusion. (**c**) Two steps of phase 1 result in a genome differing from genome B only by intra-chromosomal rearrangement. The algorithm proceeds to phase 2 and cuts at both ends of the first oriented gray line encountered, inducing a reversal that leads to: (5, 1, 2, 3, 6), (0, 4).

The operations of phase 1 and 2 are illustrated in Figure 9, using the sorting problem treated in Figure 8. Since the capping has produced a null chromosome in each genome, we now have a two-chromosome problem and our algorithm begins by looking for translocations (phase 1). Indeed, it first performs a fission (Fig. 9a), then a fusion

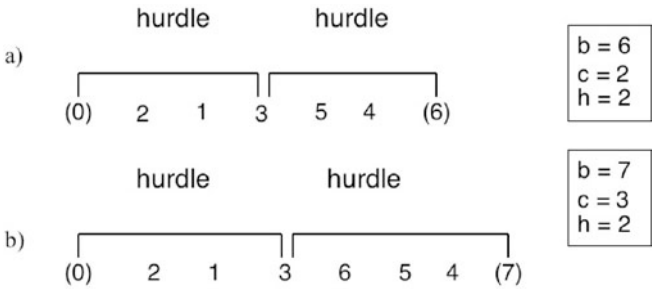

Fig. 10. (**a**) A positive permutation on five elements containing two hurdles. (0) and (6) are caps. (**b**) Longer permutation with two hurdles, with caps (0) and (7).

(Fig. 9b), and then goes to phase 2 for an inversion, which completes the sorting. Phase 3 is not necessary. (The whole sorting can be done, if desired, by three inversions, but we have designed the algorithm to perform translocations first.)

Phase 3 of the algorithm follows a trajectory that does not correspond to the HPa algorithm for eliminating hurdles. In Figure 10(a) we show a positive permutation with two hurdles. Having $b = 6, c = 2$ and $b-c = 4$, it can be sorted with two block interchanges ($21 \Leftrightarrow 12$ and $54 \Leftrightarrow 45$) which in this case are contiguous exchanges or transpositions. The HP algorithm for sorting by pure inversions starts by inverting three, merging the two hurdles to yield $b = 6$, $c = 1$ and no hurdles, after which five more inversions are required for a total of distance $6 = b-c+h$. An alternative path by inversion is to treat each hurdle separately mimicking our contiguous exchanges with three inversions each, also resulting in a distance of six.

The permutation of Figure 7b, however, requires a non-contiguous block interchange ($64 \Leftrightarrow 46$) to sort the second hurdle. For a distance of seven, this takes four inversions to mimic, whereas the optimal inversion path, which starts by inverting three, still requires only $6 = b - c + h$. Thus in general, our optimal sorting procedure (with length $b - c$) does not parallel an optimal procedure for pure inversions.

Bafna and Pevzner (1998) have shown that in sorting a positive permutation by transpositions (that is, the exchange of contiguous segments) one needs at least $(b - c_{\text{odd}})/2$ steps where c_{odd} is the number of odd cycles. But if the exchanged segments need not be contiguous, it is easily seen that only $(b - c)/2$ steps are required, since each step decreases $b - c$ by 2 as shown above in phase 3 of the algorithm. It is thus natural to weight each block interchange two when combined with inversions and translocations.

From the DCJ perspective, however, a block interchange has a weight of two simply because it is a shorthand for describing two successive DCJ's. The operations allowed by Eriksen (2002) for a single chromosome, inversions and contiguous exchanges (transpositions), are more powerful than those of HP (inversions alone for single chromosomes) but less powerful than ours (inversions and all block interchanges). Eriksen's shortest distance is consequently intermediary between HP's $b - c + f + h$ and our $b - c$.

The computation of genomic distance based on DCJs is highly efficient in that it just requires the computation of c, the number of cycles, which can be computed in linear time. There are currently no exact polynomial-time algorithms for sorting genomic permutations that combine transpositions with inversions and translocations.

4 LOOKING AHEAD

The general DCJ operation is a plausible elementary operation for genome evolution. Producing the familiar processes of inversion, translocation, fission and fusion it also, via a double step, creates and reabsorbs CIs, equivalent to a block interchange which exchanges two not necessarily contiguous segments along a single chromosome. Although we have focused here on linear chromosomes, our approach is applicable to genomes consisting of circular genomes, as well as combinations of the two, such as in *Borrelia burgdorferi*, consisting of ∼1500 genes, half of which are located on a long linear chromosome, with the rest interspersed among ∼21 plasmids—some circular and some linear. Qiu *et al*. (2004) showed extensive recombination takes place between the main chromosome and the plasmids and amongst the plasmids. Another intriguing idea is to apply our method to the formation and re-absorption of small replicating circular DNA sequences (amplisomes) thought to be responsible for rearranging tumor genomes as considered by Raphael and Pevzner (2004).

ACKNOWLEDGEMENTS

S.Y. thanks Nick Chiorazzi and Bettie Steinberg for their enthusiasm, G.D. Yancopoulos and Lynn Caporale for inspiration, and Bud Mishra for elucidating edit distance. OA thanks Weigang Qui for support on SCORE grant GM060654. We thank Betty Harris for help with figures, the Columbia Physics Department for use of the facilities, and the reviewers for their extensive and thoughtful comments.

Conflict of Interest: none declared.

REFERENCES

Andersson,S. and Eriksson,K. (2000) Dynamics of gene order structures and genomic architectures. In Sankoff,D. and Nadeau,J. (eds), *Comparative Genomics*, Kluwer Academic Publishers, pp. 267–280.

Bafna,V. and Pevzner,P.A. (1993) Genome rearrangements and sorting by reversals. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Press, pp. 148–157.

Bafna,V. and Pevzner,P. (1995) Sorting by transpositions. In *Proceedings of 6th Annual ACM-SIAM Symposium on Discrete Algorithms*. January 22–24, San Francisco, CA, pp. 614–623.

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

Bergeron,A. (2001) A very elementary presentation of the Hanenhalli–Pevzner theory. *CPM' 01 Proceedings, LNCS 2089*, Springer-Verlag, Berlin, pp. 106–117. [updated version appeared in (2005) *Discrete Appl. Math.*, **146**, 134–145.]

Blanchette,M. *et al*. (1996) Parametric genome rearrangement. *Gene*, **172**, 11–17.

Caprara,A. (1997) Sorting by reversals is difficult. In *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB '97)*, ACM, New York, pp. 75–83.

Christie,D.A. (1996) Sorting permutations by block interchanges. *Inform. Processing Lett.*, **60**,165–169.

Dalevi,D.A. *et al*. (2002) Measuring genome divergence in bacteria: a case study using chlamydian data. *J. Mol. Evol.*, **55**, 24–36.

Deininger,P.L. and Roy-Engel,A.M. (2001) Mobile elements in animal and plant genomes. In Craigie,R., Gellert,M. and Lambowitz,A.M., (eds), *Mobile DNA II*, ASM Press, Washington, DC, pp. 1071–1092.

Eriksen,N. (2002) $(1+\varepsilon)$-approximation of sorting by reversals and transpositions. *J. Theor. Comp. Sci.*, **289**, 517–529.

Hannenhalli,S. (1996) Polynomial-time algorithm for computing translocation distance between genomes. *Discrete appl. math.*, **71**, 137–151.

Hannenhalli,S. and Pevzner,P.A. (1995a) Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pp. 178–189 (full version appeared in *J. ACM*, **46**, 1–27, 1999).

Hannenhalli,S. and Pevzner,P.A. (1995b) Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, Milwaukee, WI, pp. 581–592.

Hartman,T. (2003) A simpler 1.5-approximation algorithm for sorting by transpositions. In *Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM '03)*, Springer, Berlin, pp. 156–169.

Kent,W.J. *et al*. (2003) Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc. Natl Acad. Sci. USA*, **100**, 11484–11489.

Lin,Y.C. *et al*. (2005) An efficient algorithm for sorting by block-interchanges and its application to the evolution of *Vibrio* species. *J. Comput. Biol.,* **12**, 102–112.

Meidanis,J. and Dias,Z. (2001) Genome rearrangements distance by fusion, fission, and transposition is easy. In *Proceedings of SPIRE'2001—String Processing and Information Retrieval*, Laguna de San Rafael, Chile, pp. 13–15.

Meidanis,J. and Dias,Z. (2002) The genome rearrangement distance by fusion, fission, and transposition with arbitrary weights. *Technical Report IC-02–01*, Institute of Computing, University of Campinas.

Ozery-Flato,M. and Shamir,R. (2003) Two notes on genome rearrangements. *J. Comput. Biol.*, **1**, 71–94.

Palmer,J.D. and Herbon,L.A. (1986) Tricircular mitochondrial genomes of *Brassica* and *Raphanus*: reversal of repeat configurations by inversion. *Nucleic Acids Res.*, **14**, 9755–9764.

Qiu,W. *et al*. (2004) Genetic exchange and plasmid transfers in *Borrelia burgdorferi sensu stricto* revealed by three-way genome comparisons and multilocus sequence typing. *Proc. Natl Acad. Sci. USA,* **101**, 14150–14155.

Raphael,B. and Pevzner,P. (2004) Reconstructing tumor amplisomes. *Bioinformatics*, **20**(Suppl 1), i265–i273. (Special ISMB/ECCB 2004 Issue).

Sankoff,D. (1992) Edit distance for genome comparison based on non-local operations. In Apostolico,A., Crochemore,M., Galil,Z. and Manber,U., (eds), *Combinatorial Pattern Matching. Third Annual Symposium*, *Lecture Notes in Computer Science,* Springer Verlag, **644**, 121–135.

Sturtevant,A.H. and Dobzhansky,T. (1936) Inversions in the third chromosome of wild races of *Drosophila pseudoobscura*, and their use in the study of the history of the species. *Proc. Natl Acad. Sci. USA*, **22**, 448–450.

Tesler,G. (2002) Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Sys. Sci.*, **65**, 587–609.

Walter,M.E., Dias,Z. and Meidanis,J. (2000) A new approach for approximating the transposition distance. In *Proceedings of SPIRE'2000—String Processing and Information Retrieval*, La Coruna, Spain, pp. 27–29.

Walter,M.E., Reginaldo,L., Curado,A.F. and Oliveira,A.G. (2003) Working on the problem of sorting by transpositions on genome rearrangements. In *Proceedins of the 14th Annual Symposium on Combinatorial Pattern Matching* (CPM '03), Springer, Berlin, pp. 372–383.

Waterston,G. *et al*. (1982) The chromosome inversion problem. *J. Theor. Biol.*, **99**, 1–7.