JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 10, Numbers 3–4, 2003

Pp. 575–597

An Algorithm to Enumerate Sorting Reversals for Signed Permutations

ADAM C. SIEPEL

ABSTRACT

The rearrangement distance between single-chromosome genomes can be estimated as the minimum number of inversions required to transform the gene ordering observed in one into that observed in the other. This measure, known as "inversion distance," can be computed as the reversal distance between signed permutations. During the past decade, much progress has been made both on the problem of computing reversal distance and on the related problem of finding a minimum-length sequence of reversals, which is known as "sorting by reversals." For most problem instances, however, many minimum-length sequences of reversals exist, and in the absence of auxiliary information, no one is of greater value than the others. The problem of finding *all* minimum-length sequences of reversals is thus a natural generalization of sorting by reversals, yet it has received little attention. This problem reduces easily to the problem of finding all "sorting reversals" of one permutation with respect to another—that is, all reversals ρ such that, if ρ is applied to one permutation, then the reversal distance of that permutation from the other is decreased. In this paper, an efficient algorithm is derived to solve the problem of finding all sorting reversals, and experimental results are presented indicating that, while the new algorithm does not represent a significant improvement in asymptotic terms (it takes $O(n^3)$ time, for permutations of size n; the problem can now be solved by brute force in $\Theta(n^3)$ time), it performs dramatically better in practice than the best known alternative. An implementation of the algorithm is available at *www.cse.ucsc.edu/~acs*.

Key words: genome rearrangements, sorting by reversals.

1. INTRODUCTION

CHROMOSOMAL REARRANGEMENTS ARE BELIEVED TO OCCUR primarily by a few simple mechanisms, including inversion, transposition, and reciprocal translocation (Sankoff and El-Mabrouk, 2002). Of these, inversion is considered the most important for many genomes (McLysaght *et al.*, 2000; Blanchette *et al.*, 1996). A model based on inversions is therefore a natural starting place for a mathematical treatment of genome rearrangements, and the minimum number of inversions required to transform one genome into another is a natural measure of evolutionary distance. If each of two genomes has exactly one copy of each

Department of Computer Science, University of New Mexico, Albuquerque, NM 87131 and National Center for Genome Resources, Santa Fe, NM 87505.

of n genes, then the genomes can be represented by permutations of size n, and their inversion distance is equal to the minimum number of "reversals" required to transform one permutation into the other, known as the *reversal distance* between the permutations. Here, a reversal is an operation by which contiguous elements of a permutation are changed in order: for example, $(1, 2, 3, 4) \rightarrow (3, 2, 1, 4)$. There has been considerable interest during the past decade in the reversal distance problem and in the related but distinct problem of finding an actual sequence of reversals that will "sort" one permutation with respect to another. Both of these problems have been shown to be NP-hard with ordinary permutations (Caprara, 1997), but in the case of *signed* permutations, where each permutation element is assigned a "+" or "−" sign, they have polynomial-time solutions (Hannenhalli and Pevzner, 1995) (with signed permutations, a reversal changes the sign of affected elements, as well as their order). The genome rearrangement problem can be modeled with signed permutations if the direction of transcription is known of each gene in each genome.

The first major step in solving the reversal-distance and sorting-by-reversals problems was apparently the recognition, by Bafna and Pevzner (1993), that the reversal distance between signed permutations was closely related to the number of cycles in a particular diagram—the "breakpoint graph," or (more colorfully) "Diagram of Reality and Desire" (Setubal and Meidanis, 1997). The breakthrough came when Hannenhalli and Pevzner (1995) characterized certain peculiar structures in the breakpoint graph—which they called "hurdles" and "fortresses"—that caused the relationship between cycles and distance not to be exact. Hannenhalli and Pevzner proved that reversal distance can be exactly expressed as a function of the numbers of cycles, hurdles, and fortresses and derived a $O(n^4)$-time algorithm to sort by reversals (where n is the permutation size). Berman and Hannenhalli (1996) soon improved the bound for the sorting problem to $O(n^2\alpha(n))$ (where α is the inverse of Ackermann's function), and it was then further improved by Kaplan, Shamir, and Tarjan (1999) to $O(n^2)$. Recently, Bader, Moret, and Yan (2001) have shown how to compute reversal distance (without actually sorting) in $O(n)$ time, and Bergeron (2001) and Bergeron and Strasbourg (2001) have described an alternative sorting algorithm that takes $O(n^2)$ time but sidesteps much of the complexity of earlier algorithms.

All sorting-by-reversals algorithms published so far find a single minimum-length sequence of sorting reversals. While they generally can be adapted to find multiple sequences of sorting reversals, none will find *all* sequences. For certain search problems in the space of genome rearrangements, it can be very useful to obtain all minimum-length sequences of sorting reversals, as has been shown in the case of the reversal median problem (Siepel, 2001). Knowing all minimum-length sequences of sorting reversals also might improve the usefulness in real scientific applications of reversal sorting algorithms. One might attempt, for example, to assess the biological merits of various parsimonious rearrangement scenarios. Indeed, from a biological perspective, a single minimum-length sequence of sorting reversals is of limited value, even aside from the limitations of an inversions-only model of rearrangement. Many such sequences exist (as will be shown below), and in the absence of additional data or a richer model, no one is more plausible than the others.

The problem of finding all minimum-length sequences of sorting reversals between a permutation π and a permutation ϕ reduces easily to the problem of finding all individual sorting reversals of an intermediate permutation π' with respect to ϕ. It is this "inner" or "branching" problem—which I will call the "all sorting reversals problem" (*ASR*)—that this paper addresses. The paper begins with a straighforward classification scheme for all possible reversals. Next, a simplified version of the problem is introduced, called the "Fortress-Free Model" (*FFM*), and it is shown, under the *FFM*, what criteria the reversals of each class must meet in order to be sorting reversals. Next, fortresses are reintroduced, the results of the previous section are adapted for the general case, and an algorithm is presented that solves *ASR*. Finally, experimental results are shown that demonstrate the efficiency of the algorithm and affirm its correctness.

2. NOTATION AND DEFINITIONS

Let π and ϕ be signed permutations of size n, such that $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and $\phi = (\phi_1, \phi_2, \ldots, \phi_n)$. Let the unsigned permutation $\pi' = (\pi'_0, \pi'_1, \ldots, \pi'_{2n+1})$ be defined such that $\pi'_0 = 0$, $\pi'_{2n+1} = 2n + 1$, and for all i $(1 \leq i \leq n)$, $\pi'_{2i} = 2\pi_i$, $\pi'_{2i-1} = 2\pi_i - 1$ (if $\pi_i > 0$) or $\pi'_{2i} = 2|\pi_i| - 1$, $\pi'_{2i-1} = 2|\pi_i|$ (if $\pi_i < 0$); let the unsigned permutation $\phi' = (\phi'_0, \phi'_1, \ldots, \phi'_{2n+1})$ be defined exactly the same way with respect to ϕ. Two elements π_i and π_{i+1} are said to be *adjacent* in π, and the corresponding elements π'_{2i} and π'_{2i+1}

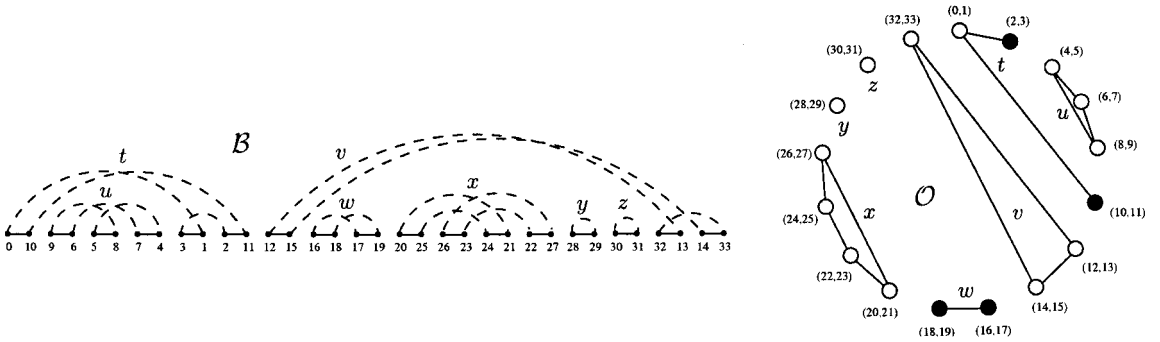

FIG. 1. Breakpoint graph $\mathcal{B}$ and overlap graph $\mathcal{O}$ for the permutation $\pi = \{-5, -3, -4, -2, +1, +6, +8, -9, +10, +13, +12, +11, +14, +15, +16, +7\}$ with respect to the identity permutation of size $n = 16$. Connected components t and w are oriented, y and z are trivial, and u, v, and x are unoriented. Unoriented components u and x are hurdles, but unoriented component v is a protected nonhurdle because it separates u and x. Oriented gray edges are represented in $\mathcal{O}$ by solid circles.

are said to be adjacent in π'; similarly for ϕ and ϕ'. The *breakpoint graph* $\mathcal{B}$ of π with respect to ϕ is constructed by arranging in a line a sequence of $2n + 2$ vertices, corresponding to the elements of π' (the vertices must obey the order of π'; see Fig. 1), and connecting every two of these vertices that reflect an adjacency in π' with a *black* edge (a "reality" edge) and every two that reflect an adjacency in ϕ' with a *gray* edge (a "desire" edge, often depicted as a dashed line). The black edges necessarily span alternating intervals between adjacent vertices; the gray edges are by convention drawn as arcs extending above the baseline of the diagram. I will refer to the process of visiting the vertices of $\mathcal{B}$ in the order in which they appear as *traversing* the breakpoint graph. A traversal may begin at any vertex but must include all vertices (it is circular); it can proceed in either direction.

A *cycle* in $\mathcal{B}$ is a sequence of connected vertices $(v_0, v_1, v_2, \ldots, v_{2m}, v_{2m+1}, v_{2m+2})$, such that $m \geq 0$, $v_{2m+2} = v_0$, and for all i $(0 \leq i \leq m)$, v_{2i} and v_{2i+1} are connected with a black edge, and v_{2i+1} and v_{2i+2} are connected with a gray edge. A traversal of the cycle is a visitation of vertices and edges in the order of such a sequence. As with a traversal of $\mathcal{B}$, a traversal of a cycle can begin at any place and proceed in either direction. A given traversal of a cycle induces at every black edge a positive or negative ordering of the vertices of $\mathcal{B}$. For example, in Fig. 1, a traversal of the cycle containing vertices 0, 10, 3, 1, 2, and 11, beginning along the black edge from 0 to 10, induces a positive ordering of vertices at edge (0,10) and a negative ordering at edge (11,2). The black and gray edges that connect the vertices of a cycle are said to belong to the cycle.

Let the *overlap graph* $\mathcal{O} = (V, E)$ for $\mathcal{B}$ be defined such that there exists a distinct $v_e \in V$ for every gray edge e in $\mathcal{B}$ and two vertices v_e and $v_{e'}$ are connected by an edge $(\{v_e, v_{e'}\} \in E)$ iff gray edges e and e' *overlap* in $\mathcal{B}$ (see Fig. 1). Two edges overlap if they span overlapping intervals in the sequence of vertices (if the diagram is drawn according to convention, overlapping edges must intersect). A *connected component* in $\mathcal{O}$ has the usual meaning and is sometimes called simply a "component."

A gray edge is said to be *oriented* if it spans an odd number of vertices in $\mathcal{B}$, and *unoriented* otherwise. A cycle in $\mathcal{B}$ and a connected component in $\mathcal{O}$ are each said to be *oriented* if they have at least one oriented gray edge. Cycles and components are called *unoriented* if they are not oriented, except when they are *trivial.* A trivial cycle consists of a single gray edge and a single black edge and corresponds to an adjacency shared in permutations π and ϕ. A component is trivial if it consists of a single, isolated vertex in $\mathcal{O}$. Such a component always corresponds to a single trivial cycle.[1] Note that the gray edges of a cycle always belong to the same connected component, so we can say that the cycle belongs to that component. In some cases, it is useful to group together oriented and trival components; I will refer to a component that is either oriented or trivial as a *benign* component.

[1]There has been some variation in the literature in the treatment of trivial cycles and components. In this paper, they will be distinguished explicitly from unoriented cycles and components.

Every unoriented component can be classified as either a *hurdle* or a *protected nonhurdle*. A hurdle is an unoriented component that does not *separate* other unoriented components, and a protected nonhurdle is one that does. A component u is said to separate two other components v and w if, in a traversal of $\mathcal{B}$, it is impossible to pass (in either circular direction) from a vertex belonging to v to a vertex belonging to w without encountering a vertex belonging to u. Note that, while separation is primarily used with respect to unoriented components, the definition (as stated here) applies as well to benign components. A hurdle is called a *superhurdle* if, were it eliminated, a protected nonhurdle would emerge as a hurdle; otherwise, it is called a *simple hurdle*.

By Hannenhalli and Pevzner's duality theorem, the distance $d(\pi, \phi)$ between π and ϕ is given by $d(\pi, \phi) = n + 1 - c + h + f$, where c is the number of cycles and h is the number of hurdles in $\mathcal{B}$. The parameter f is equal to one if there is a *fortress* in $\mathcal{B}$ and zero otherwise. A fortress exists iff there is an odd number of hurdles and all are superhurdles (Setubal and Meidanis, 1997). The function $d(\pi, \phi)$ is sometimes represented by the variable d when π and ϕ are clear from context.

A reversal $\rho_{i,j}$ $(1 \leq i < j \leq n)$ applied to $\pi = (\pi_1, \ldots, \pi_n)$ transforms π into $\rho_{i,j}(\pi) = \pi_1, \ldots, -\pi_j, \ldots, -\pi_i, \ldots, \pi_n)$. A reversal ρ (the subscripts will be omitted when unimportant) is a *sorting reversal* iff $d(\rho(\pi), \phi) = d(\pi, \phi) - 1$. I will use the term Δd to indicate the quantity $d(\rho(\pi), \phi) - d(\pi, \phi)$ and, similarly, the terms Δc, Δh, and Δf to indicate the changes caused by a given reversal to the numbers of cycles, hurdles, and fortresses, respectively ($\Delta d = -\Delta c + \Delta h + \Delta f$). Note that these quantities have meaning only with respect to a given reversal ρ and pair of permutations, (π, ϕ), one of which is designated as the "reference" or "target" (ϕ); in this paper, ρ, π, and ϕ will generally be clear from context. A given reversal can only increase by one, decrease by one, or leave unchanged the reversal distance of π from ϕ; thus, $\Delta d \in \{-1, 0, 1\}$. The end-points i and j of a reversal $\rho_{i,j}$ correspond to the ith and $(j+1)$st black edges of $\mathcal{B}$. The reversal ρ is said to *act on* these edges.

Setubal and Meidanis (1997) have presented an alternative distinction to the one between oriented and unoriented gray edges, based on black edges. They define two black edges of the same cycle as *convergent* if in a traversal of the cycle, these edges induce the same ordering of the vertices of $\mathcal{B}$; otherwise, the edges are *divergent*. It can be shown easily that an oriented gray edge always connects nodes adjoining divergent black edges and an unoriented gray edge always connects nodes adjoining convergent black edges (if the unoriented gray edge is part of a trivial cycle, then it connects nodes that adjoin the same black edge; thus, the rule holds, as long as any black edge is considered to be convergent with itself). Setubal and Meidanis have shown that a reversal splits a cycle iff it acts on divergent black edges of the cycle. This result is stronger than the corresponding one for oriented gray edges, which says that a reversal splits a cycle *if* (not iff) it acts on black edges corresponding to an oriented gray edge. They have also shown that a reversal does not change the number of cycles iff it acts on convergent edges of the same cycle, and a reversal combines two cycles iff it acts on edges belonging to different cycles. These results are especially useful when enumerating all sorting reversals, as will be seen below.

When a gray edge has one vertex within and one vertex outside the range of a reversal, the edge is said to be *affected by* the reversal.[2] A component is affected by a reversal iff it has at least one gray edge that is affected by the reversal. It can be shown that a reversal causes an edge to change orientation (that is, to change from oriented to unoriented or vice versa) iff it affects the edge (two black edges adjoining a gray edge change from convergent to divergent, or vice versa, iff the gray edge is affected) (Siepel, 2001). As a result, a reversal will cause an unoriented component to become oriented iff it affects the component. This observation provides a simple explanation for the phenomena Hannenhalli and Pevzner (1995) have called "hurdle cutting" and "hurdles merging."

I will denote as follows all connected components, the cycles within them, and the black edges within the cycles. Let $M = \{m_i\}$ be the set of components in $\mathcal{O}$, let $C_i = \{c_{i,j}\}$ be the set of cycles that belong to m_i, and let $B_{i,j} = \{b_{i,j,k}\}$ be the set of black edges that belong to $c_{i,j}$. This notation allows several of the definitions above to be summarized concisely, as follows. Any two black edges belonging to the same cycle, $b_{i,j,k}$ and $b_{i,j,l}$, must be convergent or divergent. If there exist divergent black edges $b_{i,j,k}$

[2]More precisely, a reversal induces a bipartitioning of the vertices of the breakpoint graph, and each partition can be considered to represent "a range" of the reversal (the ranges are complementary). An edge is said to be affected by the reversal iff it connects a vertex of one range to a vertex of the complementary range (Siepel, 2001).

and $b_{i,j,l}$, then cycle $c_{i,j}$ is oriented; otherwise, $c_{i,j}$ is unoriented, unless it has only a single black edge ($|B_{i,j}| = 1$), in which case $c_{i,j}$ is trivial. If there exists $c_{i,j} \in C_i$ such that $c_{i,j}$ is oriented, then m_i is oriented; otherwise, m_i is unoriented, unless $|C_i| = 1$ and the single element of C_i is trivial, in which case m_i also is trivial. If m_i is unoriented, then m_i is either a hurdle or a protected nonhurdle, depending on whether it separates other unoriented components.

3. SORTING REVERSALS IN THE ABSENCE OF FORTRESSES

These definitions lead directly to an exhaustive classification scheme for reversals.

Lemma 1. *Suppose ρ is a reversal acting on two black edges, $b_{i,j,k}$ and $b_{i',j',k'}$, which belong respectively to cycles $c_{i,j}$ and $c_{i'j'}$ and to connected components m_i and $m_{i'}$. Then one and only one of the following is true:*

1. *($i = i'$ and $j = j'$) $b_{i,j,k}$ and $b_{i',j',k'}$ belong to the same cycle, $c_{i,j}$, and either:*
   (a) $c_{i,j}$ is oriented and m_i is oriented; or
   (b) $c_{i,j}$ is unoriented, $b_{i,j,k}$ and $b_{i',j',k'}$ are convergent, and m_i is either oriented or unoriented;
2. *($i = i'$ and $j \neq j'$) $b_{i,j,k}$ and $b_{i',j',k'}$ belong to different cycles of the same component, m_i, which is either oriented or unoriented;*
3. *($i \neq i'$) $b_{i,j,k}$ and $b_{i',j',k'}$ belong to different components, m_i and $m_{i'}$.*

Proof. Either $i = i'$ or $i \neq i'$ (the edges are part of the same or different components); and if $i = i'$, then either $j = j'$ or $j \neq j'$ (the edges are part of the same or different cycles). Each of $c_{i,j}$, $c_{i',j'}$, m_i, and $m_{i'}$ is by definition oriented, unoriented, or trivial. A component that has at least two edges cannot be trivial, however, and can contain no trivial cycle, so Cases 1 and 2 need only consider oriented and unoriented cycles and components. Moreover, a component containing an oriented cycle is by definition oriented, so Case 1a need not allow m_i to be unoriented. Similarly, Case 1b need not consider the possibility of divergent black edges, because every two edges of an unoriented cycle must be convergent. ■

The "Fortress-Free Model" (*FFM*) of *ASR* will be defined in terms of a surrogate measure of distance, $d'(\pi, \phi)$, that ignores fortresses. Specifically, let $d'(\pi, \phi) = n+1-c+h$ and $\Delta d' = d'(\rho(\pi), \phi) - d'(\pi, \phi)$. Let the *FFM* be a version of *ASR* in which a *sorting reversal* is redefined to be a reversal that causes $\Delta d' = -1$.

The *FFM* allows for a simple but powerful rule I will call "conversation of distance," which will be used heavily throughout this section of the paper.

Lemma 2 (Conservation of Distance). *Under the FFM, a reversal is a sorting reversal iff one of the following is true:*

1. *$\Delta c = -1$ and $\Delta h = -2$,*
2. *$\Delta c = 0$ and $\Delta h = -1$,*
3. *$\Delta c = 1$ and $\Delta h = 0$.*

Proof. By definition, the reversal is a sorting reversal iff $\Delta d' = -1$. Because $d' = n + 1 - c + h$, $\Delta d' = -1$ iff $\Delta h - \Delta c = -1$. Clearly, $\Delta h - \Delta c = -1$ if any of the cases of the lemma apply. Furthermore, we know that $\Delta c \in \{-1, 0, 1\}$, because a reversal can only merge cycles, be neutral with respect to cycle number, or split a cycle. Therefore, $\Delta h - \Delta c = -1$ only if one of the cases of the lemma applies. ■

I will now address Cases 1a, 1b, 2, and 3 of Lemma 1 in turn, under the assumptions of the *FFM*.

Lemma 3 (Case 1a). *Under the FFM, a reversal ρ that acts on two black edges belonging to the same oriented cycle is a sorting reversal iff the edges are divergent and ρ does not change the number of hurdles.*

Proof. First consider the claim that ρ is a sorting reversal if the edges are divergent and the number of hurdles is left unchanged. Suppose ρ acts on two divergent black edges of the same cycle and ρ does not change the number of hurdles. Then $\Delta c = 1$ (ρ must split the cycle) and $\Delta h = 0$; therefore, ρ is a sorting reversal by conservation of distance. Now consider the converse claim, that ρ is a sorting reversal only if the edges are divergent and the number of hurdles does not change. Suppose ρ acts on black edges of the same cycle and is a sorting reversal. Either the black edges are divergent and ρ splits the cycle, or the black edges are convergent and ρ is neutral with respect to cycle number. If ρ splits the cycle ($\Delta c = 1$), then $\Delta h = 0$, by conservation of distance. Similarly, if ρ is neutral with respect to cycle number ($\Delta c = 0$), then $\Delta h = -1$. It must be true, however, that $\Delta h \geq 0$, because ρ acts on black edges of an oriented cycle, and therefore of an oriented component (thus, ρ can affect no unoriented component[3] and can eliminate no hurdle). Consequently, ρ is a sorting reversal only if the edges are divergent and the number of hurdles is left unchanged. ■

Lemma 4 (Case 1b). *Under the FFM, a reversal ρ that acts on two black edges belonging to the same unoriented cycle $c_{i,j}$ is a sorting reversal iff the component m_i to which $c_{i,j}$ belongs is a simple hurdle.*

Proof. First consider the claim that ρ is a sorting reversal if m_i is a simple hurdle. Suppose that ρ acts on two black edges belonging to the same unoriented cycle $c_{i,j}$, which belongs to a simple hurdle m_i. Because $c_{i,j}$ is unoriented, all of its black edges are convergent; therefore, ρ must cause $\Delta c = 0$. Because m_i is a simple hurdle, ρ must orient the hurdle; furthermore, ρ can eliminate or introduce no other hurdle, because it can neither affect another existing component, nor introduce a new one. Thus, ρ must also cause $\Delta h = -1$, and must be a sorting reversal by conservation of distance. Now consider the converse claim, that ρ is a sorting reversal only if m_i is a simple hurdle. Suppose that ρ is a sorting reversal that acts on two black edges of the same unoriented cycle, $c_{i,j}$. The component m_i to which $c_{i,j}$ belongs is either oriented, a hurdle, or a protected nonhurdle (it cannot be trivial because it has at least two black edges). If m_i is oriented or a protected nonhurdle, then $\Delta h \geq 0$, because ρ can affect no hurdle[4]; these cases are impossible, because they would not allow ρ to be a sorting reversal. Therefore, m_i must be a hurdle. If m_i is a superhurdle, however, then when ρ orients it, another hurdle will emerge, causing $\Delta h = 0$ and preventing ρ from being a sorting reversal. Therefore, m_i must be a simple hurdle. ■

A reversal that affects a hurdle in the way described in Lemma 4—by acting on two black edges of the same cycle—is said to *cut* the hurdle.

Lemma 5 (Case 2). *Under the FFM, a reversal ρ cannot be a sorting reversal if ρ acts on two black edges belonging to different cycles of the same component.*

Proof. Suppose to the contrary that ρ acts on two black edges belonging to different cycles of the same component and ρ is a sorting reversal. Because ρ acts on different cycles, $\Delta c = -1$; therefore, by conservation of distance, $\Delta h = -2$. It is impossible, however, for ρ to eliminate more than one hurdle, because it affects only a single component. ■

Before addressing Case 3 of Lemma 1, I must introduce two new ideas (see Figures 2 and 3).

---

[3] A reversal that acts on two edges belonging to the same component can affect no other component. The proof of this fact goes essentially as follows. Assume to the contrary that a reversal ρ acts on two edges belonging to a component u and affects another component v. Then u and v must each have at least one vertex in each of the ranges of ρ. It follows (omitting some details) that an edge of u and an edge of v must overlap, and hence that u and v are not separate components—a contradiction.

[4] In general, a reversal can decrease the number of hurdles by no more than the number of hurdles it directly affects. The reason is that a hurdle can cease to be a hurdle only if it is affected (and hence oriented) or caused to separate other unoriented components, but it cannot be caused to separate other unoriented components without being affected; therefore, a hurdle can be eliminated only if it is affected. This fact can be shown by contradiction, using an argument based on the ranges of ρ.

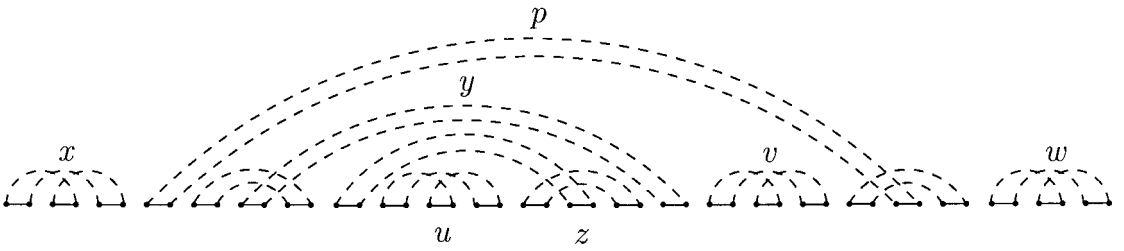

FIG. 2. Breakpoint graph in which hurdles u and v form a double superhurdle with respect to protected nonhurdle p. Note that all unoriented components besides u, v, and p either separate u and v (y and z) or are separated from both u and v by p (x and w). This construct is called a "double superhurdle," by analogy to ordinary superhurdles, because a reversal that simultaneously destroys u and v will cause p to emerge as a hurdle. In this example, x and w also form a double superhurdle with respect to p.

Definition 1. *Two hurdles u and v form a* double superhurdle *iff two other unoriented components w and p coexist with u and v such that:*

1. *p separates u and v from w but does not separate u and v from each other,*
2. *every other unoriented component either separates u and v or is separated by p from both u and v, and*
3. *p does not separate from each other any two of the components that it separates from u and v.*

Definition 2. *A hurdle that separates a benign component from all other unoriented components is said to be the* separating hurdle *of the benign component.*

It follows from Definition 2 that a benign component may have at most one separating hurdle, but a hurdle may be the separator of multiple benign components. All separating hurdles and the benign components that they separate can be found easily in a traversal of the breakpoint graph (Siepel, 2001).

Lemma 6 (Case 3). *Under the FFM, a reversal ρ that acts on black edges belonging to different components u and v is a sorting reversal iff all of the following are true:*

1. *each of u and v is a hurdle or a benign component that has a separating hurdle,*
2. *u and v are not benign components sharing the same separating hurdle, and*
3. *u and v or their separating hurdles do not form a double superhurdle.*

Proof. A reversal ρ that acts on black edges belonging to different components must act on black edges belonging to different cycles and therefore must cause $\Delta c = -1$; hence, such a reversal is a sorting reversal iff it causes $\Delta h = -2$ (conservation of distance). Therefore, to prove the claim in the "if" direction, it is sufficient to show that ρ causes $\Delta h = -2$ if it meets the criteria of the lemma. Suppose ρ acts on black edges belonging to different component u and v, each of u and v is a hurdle or a benign component that has a separating hurdle, and u and v are not benign components sharing the same separating hurdle (see Fig. 3). Then ρ must affect two hurdles and hence must eliminate two hurdles. Therefore, as long

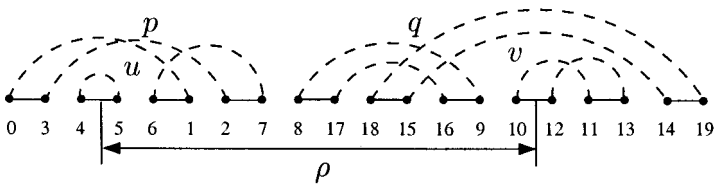

FIG. 3. A reversal ρ that acts on black edges belonging to two benign components (u and v) can still cause the destruction of two hurdles (p and q), if the hurdles are distinct separating hurdles of the benign components. The effect is similar to what would be observed if the hurdles were merged.

as ρ does not introduce any new hurdles, $\Delta h = -2$. Assume temporarily that ρ introduces one or more new hurdles iff u and v or their separating hurdles form a double superhurdle. Because this circumstance is prohibited by the third criterion of the lemma, this part of the proof is complete, conditional on the assumption about double superhurdles (to be addressed below).

For the claim in the "only if" direction, it is sufficient to show that $\Delta h = -2$ only if the criteria of the lemma hold. Suppose $\Delta h = -2$. If $\Delta h = -2$, ρ must orient at least two hurdles, and if ρ orients at least two hurdles, it must affect at least two hurdles. It is impossible for a single reversal to affect more than two hurdles,[5] so ρ must affect (and orient) exactly two hurdles. There are three possible ways that ρ can affect two hurdles: either (1) u and v are both hurdles; or (2) u and v are both nonhurdles and are separated by two hurdles; or (3) one of u and v is a hurdle—call it u, without loss of generality—and the other, v, is a nonhurdle that is separated from u by a second hurdle. Case (2) can occur only if the two components u and v are benign and have distinct separating hurdles. The reason is that u and v cannot be unoriented (otherwise a hurdle would separate unoriented components, which is prohibited by definition), so they must be benign; and if two benign components are separated by two hurdles, those hurdles by definition are their separating hurdles. Similarly, case (3) can only occur if u is benign and has a separating hurdle and that separating hurdle is distinct from v. Thus, the first two criteria of the lemma must hold. To address the third criterion, notice that if $\Delta h = -2$, then ρ must not only eliminate two hurdles, but avoid introducing new ones. If we invoke again the assumption about double superhurdles, then u and v or their separating hurdles cannot form a double superhurdle; thus, this part of the proof is complete, again conditional on the assumption.

Let me now prove that ρ introduces one or more new hurdles iff u and v or their separating hurdles form a double superhurdle. First, consider the claim in the "if" direction. Let u' be u (if u is a hurdle) or the separating hurdle of u (if u is benign), let v' be v or the separating hurdle of v, and suppose u' and v' form a double superhurdle. Let U be the set consisting of u', v', and all unoriented components that separate them. By Definition 1, there exists an unoriented component p that separates all members of U from all members of a nonempty set V, consisting of all unoriented components besides p that are not in U. Component ρ will affect and orient every member of U and no member of V; consequently, ρ will cause p to emerge as a hurdle. Now consider the converse claim, that ρ introduces one or more new hurdles only if u and v or their separating hurdles form a double superhurdle. Suppose ρ causes a new hurdle p to emerge. Prior to ρ, p must have been a protected nonhurdle separating each member of a nonempty set U of unoriented components from each member of a nonempty set V of unoriented components. Furthermore, ρ must affect all members of U or all members of V but not members of both (if ρ affected members of both U and V, it would orient p). Without loss of generality, assume that ρ affects all members of U. Then $u', v' \in U$, and u' and v' are separated from all members of V. Furthermore, u and v must be separated from one another by all other members of U (otherwise, either u and v could not be hurdles, or a reversal that destroyed them could not also destroy the other members of U) and p must not separate any two members of V (otherwise, it could not emerge as a hurdle). Thus, u' and v' meet the definition of a double superhurdle. ■

Notice that the proof of this fact about double superhurdles does not depend on the assumptions of the *FFM*: in general, a reversal ρ acting on edges belonging to different components u and v introduces one or more new hurdles iff the components or their separating hurdles form a double superhurdle.

4. ACCOMMODATING FORTRESSES

In this section, I adapt what has been established under the *FFM* to allow for the possibility of a fortress. The *FFM* differs from the general case exactly when $\Delta f \neq 0$. As a result, two possibilities must

[5]This fact can be shown easily by contradiction. Suppose ρ affects two hurdles, u and v. Then, each of u and v must have vertices in each range of ρ. Now suppose ρ also affects a third hurdle w. If, as required, w has vertices in each range of ρ, then w must either separate u and v or have a gray edge that overlaps a gray edge of u or v. Either case represents a contradiction.

be considered. The first occurs when a fortress does not exist before a candidate reversal ρ; here it is possible that ρ is a sorting reversal under the *FFM*, but introduces a fortress, and thus is not a sorting reversal in the general model. The second possibility occurs when there exists a fortress; here it is possible that ρ does not sort under the *FFM*, but eliminates the fortress, and thus is a sorting reversal in the general model. The first possibility is addressed by the following lemma.

Lemma 7. *A reversal ρ that meets the criteria for a sorting reversal under the FFM will introduce a fortress iff one of the following is true:*

1. *ρ acts on divergent black edges of the same oriented cycle, and ρ introduces at least one unoriented component such that the number of hurdles becomes odd and all of them are superhurdles;*
2. *ρ cuts the only simple hurdle and there are an odd number of superhurdles;*
3. *ρ acts on two black edges belonging to different components such that two hurdles are destroyed, and the set of hurdles is altered so that there remain an odd number and all are superhurdles.*

Proof. First consider the claim that ρ will introduce a fortress if one of the cases of the lemma applies. A fortress is introduced if the set of hurdles is changed such that it becomes odd in size and consists only of superhurdles. Each of the three cases of the lemma explicitly accomplishes such a change, so this part of the proof is complete. Now consider the claim that ρ will introduce a fortress only if one of the cases of the lemma applies. Suppose that ρ is a sorting reversal under the *FFM* and ρ introduces a fortress. Any sorting reversal must belong to one of three classes: (a) those that split cycles (Lemma 3); (b) those that cut simple hurdles (Lemma 4); and (c) those that destroy pairs of hurdles (Lemma 6). If ρ belongs to class (a), it can change the set of hurdles only by introducing new unoriented components, because it only affects a single oriented component. Thus, case 1 must apply. If ρ belongs to class (b), it can affect only a single simple hurdle. As a result, it can introduce a fortress only if there already exist an odd number of superhurdles, and the reversal removes the sole simple hurdle; thus, case 2 must apply. If ρ belongs to class (c), then case 3 must apply. ■

The second possibility above, that of a sorting reversal that eliminates a fortress, is more difficult and requires the introduction of several new concepts.

Definition 3. *Two unoriented components u and v are* adjacent *in a breakpoint graph $\mathcal{B}$ iff at least one pair of vertices from u and v have between them, in a traversal of the breakpoint graph, no vertex belonging to another unoriented component.*

Definition 4. *Let U be the set of unoriented components in a breakpoint graph $\mathcal{B}$. The* hurdle graph *for $\mathcal{B}$ is a graph $\mathcal{H} = (V, E)$ such that $V = U$ and $E = \{\{v_i, v_j\} \mid v_i, v_j \in V$ and v_i, v_j are adjacent in $\mathcal{B}\}$.*

The hurdle graph (Fig. 4) can easily be constructed in a single traversal of the breakpoint graph. It has many useful properties. For example, hurdles, protected nonhurdles, and superhurdles can all be identified from local properties of vertices in the hurdle graph, as described below (proofs omitted).

1. Suppose u, v, and w are three unoriented components and u', v', and w' are the corresponding vertices in the hurdle graph. Then, w separates u and v iff there exists no path in the hurdle graph between u' and v' that does not pass through w'. For convenience, I will say that node w' separates nodes u' and v'.
2. A vertex in the hurdle graph does not separate other vertices iff it belongs to a cycle and has degree 2, or it has degree less than 2. Thus, such a vertex corresponds to a hurdle.
3. A vertex in the hurdle graph separates other vertices iff it belongs to a cycle and has degree greater than 2, or it does not belong to a cycle and has degree 2. Thus, such a vertex corresponds to a protected nonhurdle.
4. A vertex corresponds to a superhurdle iff it has degree 1 and its neighbor either has degree 3 and belongs to a cycle, or has degree 2 and does not belong to a cycle.

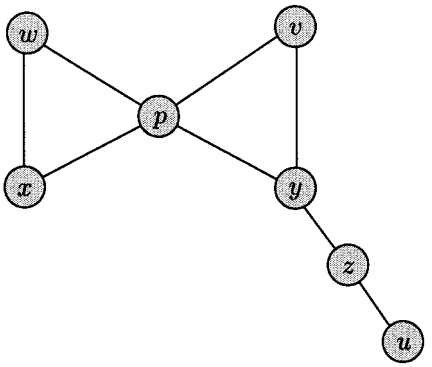

FIG. 4. The hurdle graph for the breakpoint graph of Fig. 2. Every unoriented component is represented by a vertex, and adjacencies between unoriented components are represented by edges. Here, unoriented components y, z, and u form a superhurdle chain for superhurdle u, with y as the anchor.

It is certain higher-level structures in the hurdle graph, however, that are most important for the problem of identifying sorting reversals in the case of a fortress.

Definition 5. *A* hurdle chain *is a chain of vertices in the hurdle graph consisting of a hurdle and zero or more other vertices, such that every vertex v in the chain is a hurdle, has degree 2, and does not belong to a cycle, or is the last vertex in the chain and either belongs to a cycle or has degree greater than 2.*

Definition 6. *If a hurdle chain has one end that is not a hurdle, the vertex at that end is the* anchor *of the chain.*

A hurdle chain that has hurdles at both ends is said to be "unanchored" (such a chain must have exactly two hurdles). If there exists an unanchored chain, it must encompass the entire hurdle graph (that the hurdle graph must be connected is implicit in its definition); therefore, if there exists at least one anchored chain, all chains must be anchored. A hurdle chain that contains a superhurdle is called a "superhurdle chain." Such a chain cannot have a simple hurdle; it must either be anchored and have a single superhurdle or be unanchored and have two superhurdles.

Double superhurdles can also be identified readily from the hurdle graph, using the ideas of hurdle chains and anchors. Specifically, two hurdles u and v form a double superhurdle iff u and v belong to hurdle chains anchored by vertices w and x, such that w and x belong to a 3-vertex cycle, the third vertex of that cycle has degree 3, and each of w and x has degree of at most 3 (Siepel, 2001).

Two final definitions are necessary.

Definition 7. *A superhurdle is a* single protector *if it belongs to an anchored hurdle chain of length 2.*

Definition 8. *The neighbor of a single protector is a* pseudohurdle.

The reason for the name "pseudohurdle" will become apparent below. Essentially, when there is a fortress, such a component can be treated much like a hurdle.

Lemma 8. *A reversal ρ that cuts a pseudohurdle or a single protector leaves the total number of hurdles unchanged, but causes a superhurdle to be replaced by a simple hurdle.*

Proof. Suppose ρ cuts a single protector u having pseudohurdle v. Then the only effect of ρ on the hurdle graph is to shorten an anchored hurdle chain from length 2 to length 1, because ρ orients u, but can affect no other unoriented component, and because u is adjacent to no other unoriented component besides v. By definition, u is a superhurdle and belongs to a chain of length 2; therefore, v must initially have

degree 3 and belong to a cycle, and after u is eliminated, v must have degree two and belong to a cycle. Thus, the number of hurdles stays the same, but a superhurdle is replaced by a simple hurdle. Suppose instead that ρ cuts a pseudohurdle v having single protector u. Once again, the effect of ρ is to shorten an anchored hurdle chain from length 2 to length 1. Here, ρ orients v, and the chain is shortened from its anchor rather than from its terminus. The reason that the chain is simply shortened is that any unoriented component that was adjacent to v now becomes adjacent to u. Thus, after the reversal, u must have degree two and belong to a cycle, and once again, the number of hurdles stays the same, but a superhurdle is replaced by a simple hurdle. ■

Now I am prepared to address the second possibility described at the beginning of this section—that of a sorting reversal that eliminates a fortress. In the discussion below, I return to the original, general-purpose definition of a *sorting reversal*.

Lemma 9. *A reversal ρ will eliminate a fortress and be a sorting reversal iff there exists a fortress and one of the following is true:*

1. *ρ acts on divergent edges of the same cycle and introduces at least one new unoriented component such that the number of hurdles increases by exactly one.*
2. *ρ cuts a single protector or a pseudohurdle.*
3. *ρ affects two components u and v such that one of u and v is a superhurdle or a benign component that has a separating superhurdle. Let u be this component, let the hurdle u' be either u or the separating hurdle of u, and let k be the hurdle chain of u'. One of the following must be true of v:*
   (a) *v is the anchor of chain k.*
   (b) *v is a protected nonhurdle not belonging to chain k.*
   (c) *v is a benign component that has no separating hurdle, and no one component in chain k separates v from another component in chain k.*
   (d) *v is a superhurdle or a benign component with a separating superhurdle, and v or its separating superhurdle forms a double superhurdle with u'.*

The proof of Lemma 9 is rather complicated and depends on establishing several intermediate results. It is left for the appendix.

Lemmas 7 and 9 allow Lemmas 3, 4, 5, and 6 to be generalized to accommodate fortresses.

Theorem 1 (Generalization of Lemma 3). *A reversal ρ that acts on two black edges belonging to the same oriented cycle is a sorting reversal iff the edges are divergent and one of the following is true:*

1. *ρ does not introduce an unoriented component.*
2. *There exists no fortress ($f = 0$) and ρ introduces at least one unoriented component, but does not change the number of hurdles, and does not cause there to be an odd number of hurdles all of which are superhurdles.*
3. *There exists a fortress ($f = 1$), and ρ introduces at least one unoriented component such that the number of hurdles remains the same or increases by exactly one.*

Proof. The reversal is a sorting reversal iff $-\Delta c + \Delta h + \Delta f = -1$. For a reversal that acts on two black edges of the same oriented cycle, $\Delta h + \Delta f \geq 0$, because no unoriented component is affected. Thus, a sorting reversal occurs iff $\Delta c = 1$ and $\Delta h + \Delta f = 0$ (because $\Delta c \in \{-1, 0, 1\}$). As before, $\Delta c = 1$ iff the edges are divergent. Therefore, it is sufficient to show that $\Delta h + \Delta f = 0$ iff one of the three cases of the lemma applies. First, consider the claim that $\Delta h + \Delta f = 0$ if one of the cases of the lemma applies. If the first case applies, ρ cannot alter the set of unoriented components, because it affects only a single oriented component and introduces no new unoriented components; thus, $\Delta h = 0$ and $\Delta f = 0$. The second case requires directly that $\Delta h = 0$ and indirectly (by Lemma 7) that $\Delta f = 0$; thus, it ensures $\Delta h + \Delta f = 0$. The third case states that $\Delta h \in \{0, 1\}$. Furthermore, by Lemma 9, if $\Delta h = 0$, then $\Delta f = 0$, and if $\Delta h = 1$, then $\Delta f = -1$. Thus, in the third case also, $\Delta h + \Delta f = 0$. Now consider the converse claim, that $\Delta h + \Delta f = 0$ only if one of the cases applies. Suppose $\Delta h + \Delta f = 0$. Either (1) ρ does not

introduce at least one unoriented component, (2) ρ introduces at least one unoriented component and there is not a fortress, or (3) ρ introduces at least one unoriented component and there is a fortress. Each of these three possibilities implies the corresponding case of the lemma, as follows. Possibility (1) is equal to the first case. Under possibility (2), either $\Delta f = 0$ and $\Delta h = 0$, or $\Delta f = 1$ and $\Delta h = -1$ (because $f = 0$); but we know $\Delta h \geq 0$, because ρ affects only a single oriented component, so $\Delta f = 0$ and $\Delta h = 0$. Thus, ρ cannot increase the number of hurdles and, by Lemma 7, cannot cause there to be an odd number of hurdles all of which are superhurdles. Under possibility (3), either $\Delta f = 0$ and $\Delta h = 0$, or $\Delta f = -1$ and $\Delta h = 1$ (because $f = 1$); thus, the number of hurdles either remains the same or increases by exactly one. ■

Theorem 2 (Generalization of Lemma 4). *A reversal ρ that acts on two black edges belonging to the same unoriented cycle $c_{i,j}$ is a sorting reversal iff one of the following is true of the component m_i to which $c_{i,j}$ belongs:*

1. *There exists no fortress ($f = 0$), m_i is a simple hurdle, and either m_i is not the only simple hurdle or the number of superhurdles is even.*
2. *There exists a fortress ($f = 1$) and m_i is a single protector or a pseudohurdle.*

Proof. If ρ acts on two black edges of an unoriented cycle, then $\Delta c = 0$; thus, ρ is a sorting reversal iff $\Delta h + \Delta f = -1$. The claim in the "if" direction can be proved by showing that $\Delta h + \Delta f = -1$ if one of the cases of the lemma applies. Suppose there exists no fortress, ρ acts on two black edges belonging to the same cycle $c_{i,j}$ of a simple hurdle m_i, and either m_i is not the only simple hurdle or the number of superhurdles is even. Then, as in Lemma 4, ρ eliminates m_i and affects no other hurdle, causing $\Delta h = -1$. Furthermore, ρ cannot introduce a fortress (Lemma 7), so $\Delta f = 0$. Suppose instead that there exists a fortress and m_i is a single protector or a pseudohurdle. Then by Lemma 8, $\Delta h = 0$, and by Lemma 9, $\Delta f = -1$. Therefore, both of the cases in the lemma imply that $\Delta h + \Delta f = -1$. The claim in the "only if" direction can be proved by showing that $\Delta h + \Delta f = -1$ only if one of the cases of the lemma applies. Suppose $\Delta h + \Delta f = -1$. Either $\Delta f = -1$ and $\Delta h = 0$ or $\Delta f = 0$ and $\Delta h = -1$; it cannot be true that $\Delta f = 1$ and $\Delta h = -2$, because ρ affects only a single component. If $\Delta f = -1$ and $\Delta h = 0$, then a fortress must exist initially and be eliminated; thus, $c_{i,j}$ must belong to a single protector or pseudohurdle (Lemma 9). If $\Delta f = 0$, then Lemma 4 applies, and m_i must be a simple hurdle. Furthermore, by Lemma 7, it must not be true that there are an odd number of superhurdles and m_i is the only simple hurdle. Therefore, one of the cases of the lemma must apply. ■

Theorem 3 (Generalization of Lemma 5). *A reversal ρ cannot be a sorting reversal if ρ acts on two black edges belonging to different cycles of the same component.*

Proof. Suppose to the contrary that ρ acts on two black edges belonging to different cycles of the same component and ρ is a sorting reversal. Because ρ acts edges of different cycles, $\Delta c = -1$; therefore, it must be true that $\Delta h + \Delta f = -2$. Because $\Delta f \in \{-1, 0, 1\}$ and $\Delta h \geq -1$ (the latter because ρ affects a single component), it must be true that $\Delta f = -1$ and $\Delta h = -1$. Thus, a fortress must exist initially and be destroyed by ρ. But a reversal that affects a single unoriented component eliminates a fortress iff it cuts a single protector or a pseudohurdle (Lemma 9)—a contradiction, because ρ acts on edges of different cycles. ■

Theorem 4 (Generalization of Lemma 6). *A reversal ρ that acts on black edges belonging to different components u and v is a sorting reversal iff either:*

1. *There exists no fortress ($f = 0$) and all of the following are true:*
   (a) *Each of u and v is a hurdle or a benign component that has a separating hurdle.*
   (b) *u and v are not benign components sharing the same separating hurdle.*
   (c) *u and v or their separating hurdles do not form a double superhurdle.*
   (d) *The elimination of the hurdles associated with u and v will not leave an odd number of hurdles all of which are superhurdles.*

2. *There exists a fortress ($f = 1$) and either:*
   (a) *Each of u and v is a superhurdle or a benign component that has a separating superhurdle, u and v are not benign components sharing the same separating hurdle, and u and v or their separating hurdles do not form a double superhurdle; or*
   (b) *Case 3 of Lemma 9 applies.*

Proof. First, consider the claim that ρ is a sorting reversal if either case of the lemma applies. If case 1 applies, then $\Delta f = 0$ by criterion 1d (Lemma 7); as a result, Lemma 6 applies, and by virtue of criteria 1a, 1b, and 1c, ρ is a sorting reversal. If case 2 applies, then either case 2a applies or case 2b applies. If case 2a applies, then either $\Delta f = 0$ or $\Delta f = -1$ (because $f = 1$). If $\Delta f = 0$, then Lemma 6 applies, and ρ is a sorting reversal. Otherwise, if $\Delta f = -1$, then it must be true that $\Delta h \geq -1$, because $\Delta c = -1$ and $\Delta d \in \{-1, 0, 1\}$; but this is impossible, because ρ will eliminate two hurdles and introduce no new ones, as described in the proof of Lemma 6. If, instead, case 2b applies, then ρ is a sorting reversal by Lemma 9. Thus, ρ is a sorting reversal if either case of the lemma applies. Now consider the converse claim, that ρ is a sorting reversal only if one of the cases of the lemma applies. For all reversals, $\Delta f \in \{-1, 0, 1\}$. In addition, if $\Delta f = 0$, then $f \in \{0, 1\}$; if $\Delta f = -1$, then $f = 1$; and if $\Delta f = 1$, then $f = 0$. Suppose $\Delta f = 0$. Then Lemma 6 applies, and ρ is a sorting reversal only if $f = 0$ and criteria 1a, 1b, and 1c are met, or if $f = 1$ and case 2a applies. In addition, if $f = 0$, then criterion 1d must be met, as required by Lemma 7. Suppose instead that $\Delta f = -1$. Then ρ is a sorting reversal only if case 3 of Lemma 9 applies, and equivalently, case 2b of the present lemma. Finally, suppose $\Delta f = 1$. Then, because $\Delta c = -1$, ρ is a sorting reversal only if $\Delta h = -3$, which is impossible, because $\Delta h \geq -2$ in all cases (see the proof of Lemma 6). Therefore, ρ is a sorting reversal only if one of the cases of the lemma applies. ■

5. AN ALGORITHM TO ENUMERATE ALL SORTING REVERSALS

Theorems 1, 2, 3, and 4 lead directly to an algorithm to address *ASR* (Fig. 5). For clarity of presentation and economy of space, I have described as separate subroutines the steps of the algorithm that split cycles (`get_revs_split_cycles`, see Fig. 6; Theorem 1), cut hurdles or pseudohurdles (`get_revs_cut_hurdles`, see Fig. 7; Theorem 2), and merge components (`get_revs_merge_nofort`, see Fig. 8, and `get_revs_merge_fort`, see Fig. 9; Theorem 4). Theorem 3 is addressed implicitly, by the absence of a corresponding step. The correctness of algorithm `find_all_sorting_reversals` follows from Lemma 1 and Theorems 1, 2, 3, and 4.

One step in the routine to enumerate sorting reversals that split cycles (`get_revs_split_cycles`) is particularly important: the step that detects whether a reversal that splits a cycle introduces new unoriented components. This step is relatively expensive, and turns out to be a bottleneck for the algorithm; it must be performed for every candidate reversal of the class that splits cycles, and candidates of this class are by far the most common (candidates of the other classes are possible only when hurdles are present, and hurdles are rare). I refer to the routine that executes this step as `detect`.

I implemented two versions of `detect` and compared their performance. The first version simply reruns the `connected_components` algorithm of Bader *et al.* (2001) and tests whether more unoriented components are present after a reversal than before. This approach takes time linear in the number of vertices of the affected component (the `connected_components` routine can be constrained to run only on the affected component). The second version of `detect` uses a novel algorithm that extends ideas presented by Bergeron (2001) and Bergeron and Strasbourg (2001) for modeling changes to the overlap graph induced by a reversal. The key to the algorithm is the following lemma.

Lemma 10 (Siepel, 2001). *The effect on the overlap graph of any reversal is to complement the subgraph of all vertices corresponding to affected gray edges.*

Lemma 10 is a generalization of Lemma 11 (below), which was central to Bergeron and Strasbourg's approach to sorting by reversals. Lemma 11 was not adequate for my purposes because it applies only to reversals induced by oriented gray edges in the breakpoint graph. As has been shown above, many other classes of sorting reversals exist.

Input: Two signed permutations of size n, π and ϕ.
Output: A list L of all sorting reversals of π with respect to ϕ.
begin
Construct the breakpoint graph $\mathcal{B}$ of π with respect to ϕ;
Identify all black edges, cycles, and connected components in $\mathcal{B}$. Let $e_{i,j,k}$ represent the kth black edge belonging to the jth cycle of the ith component;
Define a value $o_{i,j,k}$ corresponding to each $e_{i,j,k}$ such that $o_{i,j,k} \in \{-1, +1\}$ and $o_{i,j,k_1} \cdot o_{i,j,k_2} = -1$ iff e_{i,j,k_1} and e_{i,j,k_2} are divergent;
Label each component as oriented, trivial, or unoriented;
Build hurdle graph $\mathcal{H}$; use $\mathcal{H}$ to label each unoriented component as a simple hurdle, a single protector, a pseudohurdle, a (non-single-protector) superhurdle, or a (non-pseudohurdle) protected nonhurdle. Also, identify any double superhurdles, and label each member of a double superhurdle with its partner;
Let c be the number of cycles in $\mathcal{B}$, h be the number of hurdles, and s be the number of superhurdles; let $f = 1$ if $h = s$ and s is odd; otherwise let $f = 0$.
Initialize list L;
`append_all(`L`, get_revs_split_cycles());`
`append_all(`L`, get_revs_cut_hurdles());`
if $f = 0$ **then** `append_all(`L`, get_revs_merge_nofort());`
else `append_all(`L`, get_revs_merge_fort());`
return L;
end

FIG. 5. Algorithm `find_all_sorting_reversals`. The subroutines `get_revs_split_cycles`, `get_revs_cut_hurdles`, `get_revs_merge_nofort`, and `get_revs_merge_fort` are assumed to return lists of all sorting reversals that, respectively, split cycles, cut hurdles (or pseudohurdles), merge components when there is no fortress, and merge components when there is a fortress. They are defined below in Figs. 6, 7, 8, and 9.

Lemma 11 (Kaplan *et al.*, 1999; Bergeron, 2001). *If one performs the reversal corresponding to an oriented vertex v, the effect on the overlap graph will be to complement the subgraph of v and its adjacent vertices.*

A complete description of the new `detect` algorithm is outside the scope of this paper. Essentially, this algorithm obtains the new overlap graph that would result from a candidate reversal using Lemma 10 and the "bitwise" techniques introduced by Bergeron and Strasbourg, by which the overlap graph is represented as a "bit matrix" and changes to it are modeled using bitwise *and*, *not*, and *exclusive or* operations. The algorithm searches the new graph for unoriented components, using certain tricks to avoid an exhaustive search. I will refer to this algorithm as the "bitwise" version of detect. It takes $O(k^2)$ time, where k is the number of vertices in the affected component.

Algorithm `find_all_sorting_reversals` takes $O(n^3)$ or $O(n^4)$ time, depending on whether the connected-components or bitwise version of `detect` is used. While all sorting reversals can be enumerated by brute force in $\Theta(n^3)$ time, in practice the new algorithm offers a considerable improvement in performance, as will be shown in the next section.

6. EXPERIMENTAL METHODS AND RESULTS

I implemented Algorithm 5 in C and tested it for correctness and speed. My implementation, program `find-all-sr`, allows either `detect` algorithm to be selected at compile time. The program comprises about 1,600 lines of code (source code is available at *www.cse.ucsc.edu/~acs*).

Input: All $e_{i,j,k}, o_{i,j,k}$, and the values of h, s, and f from `find_all_sorting_reversals`; assume the existence of a function `detect` that returns a list of the new unoriented components introduced by a given reversal, or the empty set ($\emptyset$) if none is introduced.
Output: A list M of all sorting reversals that split cycles.
begin
  Initialize list M;
  /* *enumerate all divergent edges of the same cycle* */
  foreach e_{i,j,k_1} *and* e_{i,j,k_2} *such that* $o_{i,j,k_1} \cdot o_{i,j,k_2} = -1$ **do**
    $P \leftarrow$ `detect`$(\rho(e_{i,j,k_1}, e_{i,j,k_2}))$;
    if $P = \emptyset$ **then**
      /* *no new unoriented components (Theorem 1, Case 1)* */
      `append`$(M, \rho(e_{i,j,k_1}, e_{i,j,k_2}))$;
    end
    else
      /* *at least one new unoriented component (Theorem 1, Cases 2 and 3)* */
      Add components in P to hurdle graph; Label their types, and relabel their neighbors as needed;
      Count new number of hurdles (h') and determine whether a fortress exists in new permutation (f');
      if $h' + f' = h + f$ **then**
        /* *either Case 2 or Case 3 must apply* */
        `append`$(M, \rho(e_{i,j,k_1}, e_{i,j,k_2}))$;
      end
    end
  end
  return M;
end

FIG. 6. Algorithm `get_revs_split_cycles`. Note that cases 2 and 3 of Theorem 1 are not handled separately. It is simplest just to test whether the sum $\Delta h + \Delta f = 0$, as will be true iff ρ is a sorting reversal (see the proof of Theorem 1). In this algorithm, and in the ones that follow, the notation $\rho(e_{i,j,k}, e_{i',j',k'})$ is used to indicate the reversal that acts on black edges $e_{i,j,k}$ and $e_{i',j',k'}$.

Input: All $e_{i,j,k}, o_{i,j,k}$, and the values of h, s, and f from `find_all_sorting_reversals`.
Output: A list M of all sorting reversals that cut hurdles (or pseudohurdles).
begin
  Initialize list M;
  /* *Theorem 2, Case 1* */
  if $f = 0$ **then**
    $H \leftarrow \{i \mid$ component i is a simple hurdle$\}$;
    if $s = 2a + 1$ *and* $h = 2a + 2$ *(for positive integer a)* **then return** M; /* *avoid a fortress* */
  end
  /* *Theorem 2, Case 2* */
  else $H \leftarrow \{i \mid$ component i is a pseudohurdle or a single protector$\}$;
  /* *enumerate all pairs of edges belonging to the same cycle of a component in H* */
  foreach e_{i,j,k_1}, e_{i,j,k_2} *such that* $i \in H$ *and* $k_1 \neq k_2$ **do**
    `append`$(M, \rho(e_{i,j,k_1}, e_{i,j,k_2}))$;
  end
  return M;
end

FIG. 7. Algorithm `get_revs_cut_hurdles`.

Input: All $e_{i,j,k}, o_{i,j,k}$, and the values of h and s from `find_all_sorting_reversals`.
Output: A list M of sorting reversals that merge separate components, assuming there is not a fortress.

```
begin
    Initialize list M;
    Find all separating hurdles: let S_i = {j | component j is a benign component whose
    separating hurdle is component i};
    H ← {i | component i is a hurdle};
    /* enumerate all pairs of hurdles */
    foreach i, k ∈ H such that i ≠ k do
        /* avoid a DSH (Theorem 4, Case 1c) */
        if hurdles i and k form a double superhurdle then continue ;
        /* avoid a fortress (Theorem 4, Case 1d) */
        else if (s = 2a + 1, h = 2a + 3 [for positive integer a], and hurdles i and k are both
        simple hurdles) or (s = 2a + 2, h = 2a + 3 [for positive integer a], and one of hurdles i
        and k is a superhurdle) then
        continue ;
        else
            foreach j ∈ {i} ∪ S_i do
                /* j is i or is separated by i */
                foreach l ∈ {k} ∪ S_k do
                    /* l is k or is separated by k; Theorem 4, Case 1 must apply to l and j */
                    foreach e_{j,y1,z1}, e_{l,y2,z2} do
                        append(M, ρ(e_{j,y1,z1}, e_{l,y2,z2}));
                    end
                end
            end
        end
    end
    return M;
end
```

FIG. 8. Algorithm `get_revs_merge_nofort`.

Test data fell into three classes. The first class consisted of pairs of random signed permutations, such that one member of each pair had been "scrambled" with respect to the other by a specified number of random reversals. The second class was similar to the first except that permutations were scrambled not with random reversals but with a procedure designed to introduce unoriented components (to exercise the parts of the program that are active only when multiple unoriented components are present). The third class consisted of hand-picked pairs of permutations representing special cases unlikely to appear in the other two classes (configurations involving fortresses, long hurdle chains, double superhurdles, and the like). A total of several thousand pairs of permutations were produced. Correctness testing was performed by comparing the output of program `find-all-sr` with that of a control called program `find-all-bf`. The latter program finds all sorting reversals of one permutation with respect to another by brute force—that is, by considering all $\binom{n+1}{2}$ "neighbors" of one permutation and computing the reversal distance of each neighbor from the other permutation. Program `find-all-bf` directly uses the well-tested code for reversal distance by Bader *et al.* (2001), and because it is also very simple, is believed to be highly reliable. Program `find-all-sr` was not confirmed to be correct until on all test cases it produced identical results to program `find-all-bf`. Performance testing focused on two types of comparisons: the performance of `find-all-sr` versus that of `find-all-bf`, and the performance of `find-all-sr` when using the connected-components version of the `detect` algorithm versus that when using the bitwise version. All testing was performed on a Sony laptop with a 700 MHz Pentium III processor and 128 MB of RAM,

Input: All $e_{i,j,k}, o_{i,j,k}$, and the values of h and s from `find_all_sorting_reversals`.

Output: A list M of sorting reversals that merge separate components, assuming there is a fortress.

begin

Initialize list M;

Find all separating hurdles: let $S_i = \{j \mid$ component j is a benign component whose separating hurdle is component $i\}$, and let $S_\neg = \{j \mid$ component j is a benign component that has no separating hurdle$\}$;

$H \leftarrow \{i \mid$ component i is a hurdle$\}$;

$U \leftarrow \{j \mid$ component j is an unoriented component$\}$;

Label each unoriented component with its hurdle chain: let $\gamma_j = i$ if $j \in U$, $i \in H$, and j belongs to the hurdle chain of i; let $\gamma_j = -1$ if j belongs to no hurdle chain;

Identify the anchor of each chain: let $\alpha_i = j$ if $j \in U$ is anchor of chain of $i \in H$;

Initialize mark[i] to 0 for all $i \in H$;

foreach $i \in H$ **do**

`mark`[i] $\leftarrow 1$; /* *mark i as visited* */

/* *other hurdles and the benign comps they separate, excl. DSH partners (Theorem 4, Case 2a)* */

$V \leftarrow \{j \mid k \in H$ and `mark`[k] $= 0$ and k is not a double superhurdle partner of i and $j \in \{k\} \cup S_k\}$;

/* *the anchor of i's chain (Theorem 4, Case 2b and Lemma 9, Case 3a)* */

$W \leftarrow \{\alpha_i\}$;

/* *prot. nonhurdles not in i's chain (Theorem 4, Case 2b and Lemma 9, Case 3b)* */

$X \leftarrow \{j \mid j \in U$ and $j \notin H$ and $\gamma_j \neq i\}$;

/* *benign comps that have no sep. hurdle and are not separated from one component in i's chain by another (Theorem 4, Case 2b and Lemma 9, Case 3c)* */

$Y \leftarrow \{j \mid j \in S_\neg$ and $\nexists k, l$ such that $\gamma_k = i, \gamma_l = i$, and k separates j from $l\}$;

/* *DSH partners of i and the benign comps they separate (Theorem 4, Case 2b and Lemma 9, Case 3d)* */

$Z \leftarrow \{j \mid k$ is a double superhurdle partner of i and `mark`[k] $= 0$ and $j \in \{k\} \cup S_k\}$;

foreach $j \in \{i\} \cup S_i$ **do**

/* *j is i or is separated by i* */

foreach $l \in V \cup W \cup X \cup Y \cup Z$ **do**

/* *l is such that Theorem 4, Case 2 applies to j and l* */

foreach $e_{j,y_1,z_1}, e_{l,y_2,z_2}$ **do**

`append`$(M, \rho(e_{j,y_1,z_1}, e_{l,y_2,z_2}))$;

end

end

end

end

return M;

end

FIG. 9. Algorithm `get_revs_merge_fort`. The array "`mark`" simply ensures that pairs of hurdles are not processed twice.

running the Linux operating system (RedHat 7.0). The most extensive testing was performed using test data of the first class, as the presence of multiple unoriented components did not appear to change performance significantly. I ran tests for values of n between 25 and 100 and numbers of random reversals (a parameter called r) between 0% and 100% of n.

Figure 10 shows results for $n = 100$ and $0 < r \leq 100$, which are typical of what was observed. Plots are shown for `find-all-bf` and both versions of `find-all-sr`. As would be expected, the brute

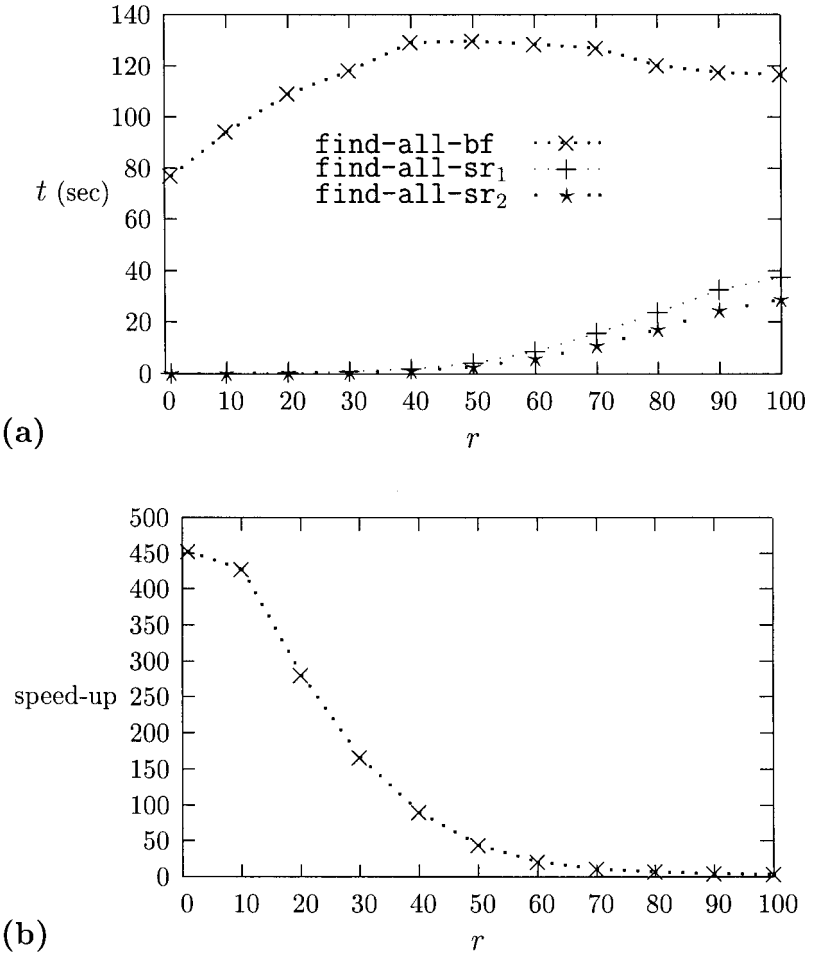

FIG. 10. (**a**) Running times of programs `find-all-bf` (an implementation of the brute-force algorithm) and `find-all-sr` (an implementation of Algorithm 5) for $n = 100$ and various values of r (the number of random reversals by which one of the permutations has been "scrambled"). Plotted are total times required to process 500 pairs of permutations. Results are shown for `find-all-sr` using both the connected-components (`find-all-sr`$_1$) and bitwise (`find-all-sr`$_2$) versions of `detect`. (b) Speed-up of `find-all-sr`$_2$ with respect to `find-all-bf` for the same experiment.

force algorithm shows approximately constant performance[6] for all values of r, and the implementations of Algorithm 5 perform significantly better at small r than at large r. Note that Algorithm 5 still performs approximately three times as fast as the brute force approach even when $r = n$, when it should be closest to its worst-case performance. Note also that the bitwise implementation of `detect` outperforms the connected-components implementation consistently, for all values of r. Figure 10 plots the speed-up of `find-all-sr` with respect to `find-all-bf`. In this experiment, a speed-up of over 400 times is achieved for $r \leq 10$. These results are particularly encouraging because small values of r are often of greatest interest when solving higher-level genome rearrangement problems with real biological data. Figure 11 shows the number of sorting reversals for $0 < r \leq 100$. When r is close to n, as one might expect, a significant fraction of all possible reversals are sorting reversals; but even when $r = 0.5n$, hundreds of sorting reversals are possible.

[6]The slightly better performance of `find-all-bf` at small r is probably the result of improvements in the speed of distance calculations due to the presence of numerous trivial components.

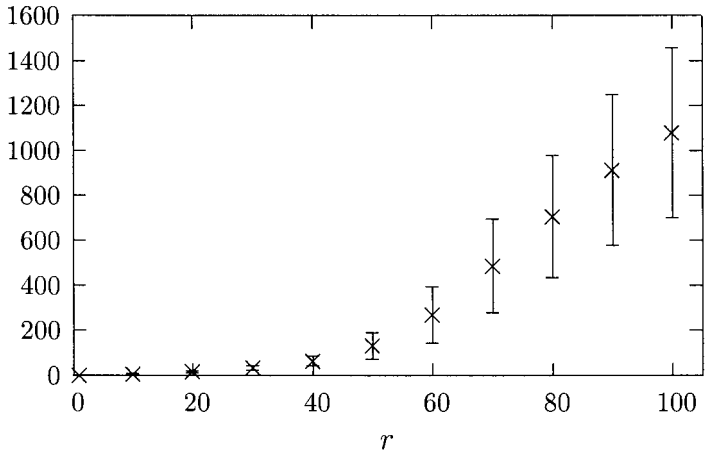

FIG. 11. Average number of sorting reversals for the experiment shown in Fig. 10. Error bars indicate one standard deviation.

7. SUMMARY

An algorithm has been presented that enumerates all sorting reversals of one signed permutation with respect to another, and consequently, allows the enumeration of all minimum-length sequences of sorting reversals. The algorithm is based on a combination of the original theory of Hannenhalli and Pevzner with more recent insights by Setubal and Meidanis. It depends on the characterization of several new types of unoriented components and makes use of a simple graph that succintly captures many of the important relationships among unoriented components (the hurdle graph). The algorithm follows from an exhaustive classification scheme for all reversals and a case-by-base analysis of each class. It has been derived by first solving a simpler version of the problem, the "Fortress Free Model" (*FFM*), then adjusting the solution to accommodate fortresses. The solution of the *FFM* is fairly straightforward; accommodating fortresses is more difficult, especially in the case of a reversal that affects multiple components (Theorem 4).

The final algorithm takes $O(n^3)$ or $O(n^4)$ time, depending on which version is used of the routine to detect the introduction of new unoriented components (`detect`). In asymptotic terms, it offers little improvement over a very simple brute-force solution, that of enumerating each neighbor of one permutation and testing its distance from the other permutation, which requires $\Theta(n^3)$ time. In practice, however, the new algorithm performs dramatically better; a speed-up of more than 400 times was observed for permutations of size 100, when the distances between permutations were small. The algorithm is significantly faster when using the "bitwise" version of `detect` than when using the "connected-components" version.

In both real and simulated data, hurdles are very rare (unless they are intentionally introduced into simulated data). When no hurdle exists, the problem of *ASR* is dramatically simplified, because only reversals that split cycles can be sorting reversals. Thus, Theorem 1 and the algorithms of Figs. 5 and 6 are sufficient to solve the problem. An irony of problems of sorting-by-reversals is that while most theoretical difficulty stems from hurdles and fortress, these structures are almost completely absent in real data. An implementation of only the algorithms of Figs. 5 and 6 would be adequate for most practical purposes, and one of the algorithms of Figs. 5, 6, and 7 for nearly all practical purposes (such a program could simply abort when two or more hurdles were present).

While enumerating all minimum-length sequences of sorting reversals is of some theoretical interest, the algorithm presented here is probably more useful for solving problems in which additional constraints help to limit the size of the search space. For example, it has allowed for a large improvement in the performance of a simple branch-and-bound approach to the reversal median problem (Siepel, 2001). Recent work has supported the use of methods for phylogeny reconstruction that are based on multiple-permutation generalizations of the problem of sorting by reversals (Bourque and Pevzner, 2002; Moret *et al.*, 2002). The algorithm presented here may help to improve the efficiency of such methods.

A natural extension of the problem of enumerating all minimum-length sequences of reversals is to enumerate all sequences of reversals that exceed the minimum length by no more than a small constant k. This problem can be solved relatively easily if one can enumerate "neutral" reversals—that is, reversals that do not change the reversal distance—as well as sorting reversals. As it turns out, the results presented in this paper can be extended to the case of neutral reversals without too much trouble (Siepel, 2001).

APPENDIX

Here, a complete proof for Lemma 9 is presented, beginning with several intermediate results.

Lemma A1. *A reversal ρ that affects a single oriented component eliminates a fortress and is a sorting reversal iff there exists a fortress, ρ acts on divergent edges of the same cycle, and ρ introduces at least one new unoriented component such that the number of hurdles increases by exactly one.*

Proof. If there exists a fortress, ρ acts on divergent edges of the same cycle, and ρ introduces at least one new unoriented component such that the number of hurdles increases by exactly one, then ρ eliminates the fortress ($\Delta f = -1$), because there can no longer be an odd number of hurdles; furthermore, $\Delta c = 1$ (because ρ acts on divergent edges of the same cycle) and $\Delta h = 1$ (as required explicitly), so $\Delta d = -1$ and ρ is a sorting reversal. Thus, the claim in the "if" direction is proved. Now consider the converse claim, that ρ eliminates a fortress and is a sorting reversal only if the criteria of the lemma are met. Suppose ρ is a sorting reversal that affects a single oriented component and eliminates a fortress. Clearly there must exist a fortress. In addition, because $\Delta f = -1$ and $\Delta d = -1$, it must be true that $\Delta h = \Delta c$. Furthermore, $\Delta h \geq 0$, because ρ affects a single oriented component, so either $\Delta h = \Delta c = 0$ or $\Delta h = \Delta c = 1$ (recall that $\Delta c \in \{-1, 0, 1\}$). It is impossible, however, that $\Delta h = 0$, because ρ can eliminate a fortress only by changing the number of hurdles or by converting simple hurdles to superhurdles; and here it cannot convert a simple hurdle to a superhurdle, because it affects only a single oriented component. Thus, $\Delta h = \Delta c = 1$. If $\Delta c = 1$, then ρ must act on divergent edges of the same cycle, and if $\Delta h = 1$, then ρ must introduce at least one new unoriented component (again because it affects only a single oriented component). Thus, all of the criteria of the lemma must be met. ■

Lemma A2. *A reversal ρ that affects a single unoriented component eliminates a fortress and is a sorting reversal iff there is a fortress and ρ cuts a single protector or a pseudohurdle.*

Proof. First consider the claim that ρ eliminates a fortress and is a sorting reversal if there is a fortress and ρ cuts a single protector or a pseudohurdle. Suppose there is a fortress and ρ cuts a single protector or a pseudohurdle. By Lemma 8, ρ must leave the number of hurdles unchanged but replace a superhurdle with a simple hurdle. Thus, the fortress must be eliminated, because a fortress cannot exist if a simple hurdle is present. Furthermore, ρ must be a sorting reversal, because $\Delta c = 0$ (ρ must act on convergent edges of the same cycle), $\Delta h = 0$ (by Lemma 8), and $\Delta f = -1$ (as shown above). Now consider the converse claim, that ρ eliminates a fortress and is a sorting reversal only if there is a fortress and ρ affects a single protector or a pseudohurdle. Suppose to the contrary that ρ eliminates a fortress and is a sorting reversal but affects a component u that is (1) a superhurdle but not a single protector or (2) a protected nonhurdle but not a pseudohurdle (these are the only possible alternatives because a simple hurdle cannot exist in the case of a fortress). By Definitions 7 and 8, ρ must belong either to an unanchored hurdle chain or to an anchored hurdle chain of length greater than two. But an unanchored hurdle chain cannot exist, because it would require there to be an even number of hurdles (which is impossible if there is a fortress); thus, u must belong to an anchored hurdle chain of length greater than two. As a result, when ρ affects and orients u, an anchored hurdle chain of length at least two must remain, and such a chain must be terminated by a superhurdle. Because ρ can alter no other hurdle chain (it affects only a single unoriented component), the number of hurdles must remain unchanged and no superhurdle can have been converted to a simple hurdle; thus, the fortress must remain, and we have a contradiction. ■

Lemma A3. *A reversal ρ that affects multiple components eliminates a fortress and is a sorting reversal iff there is a fortress and ρ acts on black edges belonging to different components u and v such that either:*

1. *u and v are superhurdles or benign components with separating superhurdles, and u and v or their separating hurdles form a double superhurdle; or*
2. *u and v are such that ρ affects all members of exactly one superhurdle chain.*

Proof. First consider the claim that ρ eliminates a fortress if there is a fortress and one of the cases of the lemma applies. Suppose there is a fortress and u and v or their separating hurdles form a double superhurdle (note that all hurdles must be superhurdles, because there is a fortress). Then ρ must destroy two hurdles and cause a new one to emerge, as described in the proof of Lemma 6. Thus, $\Delta h = -1$, and as a result, ρ will destroy the fortress (an odd number of hurdles will be made even). Furthermore, $\Delta c = -1$ (ρ acts on edges of different cycles), so $\Delta d = -1$ and ρ is a sorting reversal. Suppose instead that there is a fortress and u and v are such that ρ affects all members of exactly one superhurdle chain, k. Then ρ must remove k from the hurdle graph, but can completely remove no other hurdle chain. The reason is that every member of k will become oriented and so will be removed from the graph; but every other chain will have at least one member that is not affected (note that all chains must be anchored, because there is a fortress). Every hurdle chain must have a hurdle at its terminus, so the number of hurdles will be decreased by exactly one and the fortress can no longer exist. Once again, $\Delta c = -1$, $\Delta h = -1$, and $\Delta f = -1$, and ρ is a sorting reversal.

Now consider the converse claim, that a reversal ρ that affects multiple components eliminates a fortress only if there is a fortress and one of the cases of the lemma applies. Suppose ρ affects multiple components, eliminates a fortress, and is a sorting reversal. Clearly ρ can eliminate a fortress only if there is a fortress. In addition, if ρ affects multiple components, it must act on black edges belonging to different components. Consequently, $\Delta c = -1$, and ρ is a sorting reversal only if $\Delta h = -1$. If $\Delta h = -1$, ρ must affect all members of at least one hurdle chain; if it affected all members of no hurdle chain, the number of hurdles could not decrease. Because no reversal can affect more than two hurdles, ρ can affect all members of no more than two hurdle chains. Thus, if $\Delta h = -1$, ρ either affects all members of one hurdle chain and introduces no new hurdles, or affects all members of two hurdle chains and introduces one new hurdle. As described in the proof of Lemma 6, ρ introduces a new hurdle only if u and v or their separating hurdles form a double superhurdle. Furthermore, if ρ affects the members of a double superhurdle, then it affects two hurdles and all unoriented components that separate them; thus, ρ affects all members of two hurdle chains and introduces one new hurdle. Therefore, ρ either affects all members of one hurdle chain or affects the members of a double superhurdle. Finally, any affected hurdle chain must be a superhurdle chain, because there is a fortress, so the claim is proved. ■

Lemma A4. *A reversal ρ eliminates all members of exactly one superhurdle chain iff it acts on black edges belonging to two components u and v such that the following is true. One of u and v is either a superhurdle or a benign component with a separating hurdle. Let u be this component, let the hurdle u' be either u or the separating hurdle of u, and let k be the hurdle chain of u'. One of the following is true of v:*

1. *v is the anchor of chain k,*
2. *v is a protected nonhurdle not belonging to chain k,*
3. *v is a benign component that has no separating hurdle, and no one component in chain k separates v from another component in chain k.*

Proof. First consider the claim that ρ eliminates all members of exactly one superhurdle chain if the criteria of the lemma are met. Let u, u', and k be defined as stated, and suppose one of the three cases of the lemma applies to v. Each case requires that every member of k either be equal to u, be equal to v, or separate u and v. (The third case is subtle: if v is a benign component that has no separating hurdle, and no one component in chain k separates v from another component in chain k, then v must

be separated from u' by the anchor of k; thus, for our purposes, v is much like a protected nonhurdle not belonging to chain k). Each case also requires that all members of no other chain separate u and v. Therefore, ρ does not eliminate all members of any other superhurdle chain. Now consider the converse claim, that ρ eliminates all members of exactly one superhurdle chain only if the criteria of the lemma are met. Suppose ρ eliminates all members of superhurdle chain k and does not eliminate all members of any other superhurdle chain. Then ρ must act on the black edges of two components u and v such that (without loss of generality) u is equal to, or separated from v by, the superhurdle u' of k, and v is equal to, or separated from u by, the anchor a of k. Therefore, u must either equal u' or be a benign component separated by u'. If v equals the anchor a, then the first case applies. Otherwise, v must be a superhurdle, a protected nonhurdle, or a benign component. However, v cannot be a superhurdle; if it were, all members of two superhurdle chains would be eliminated. If v is a protected nonhurdle (and is not a), then it must not belong to k; otherwise, a would not separate u and v. If v is a benign component, then it must not have a separating superhurdle (otherwise all members of two superhurdle chains would be eliminated), and no one component in k can separate v from another (otherwise a would not separate u and v). Thus, the criteria of the lemma must be met. ■

Lemma 9. *A reversal ρ will eliminate a fortress and be a sorting reversal iff there exists a fortress and one of the following is true:*

1. *ρ acts on divergent edges of the same cycle and introduces at least one new unoriented component such that the number of hurdles increases by exactly one.*
2. *ρ cuts a single protector or a pseudohurdle.*
3. *ρ affects two components u and v such that one of u and v is a superhurdle or a benign component that has a separating superhurdle. Let u be this component, let the hurdle u' be either u or the separating hurdle of u, and let k be the hurdle chain of u'. One of the following must be true of v:*
   - *(a) v is the anchor of chain k.*
   - *(b) v is a protected nonhurdle not belonging to chain k.*
   - *(c) v is a benign component that has no separating hurdle, and no one component in chain k separates v from another component in chain k.*
   - *(d) v is a superhurdle or a benign component with a separating superhurdle, and v or its separating superhurdle forms a double superhurdle with u'.*

Proof. Any reversal must affect a single oriented component, a single unoriented component, or multiple components. A reversal that affects a single oriented component eliminates a fortress and is a sorting reversal iff case 1 applies (Lemma A1), a reversal that affects a single unoriented component eliminates a fortress and is a sorting reversal iff case 2 applies (Lemma A2), and a reversal that affects multiple components eliminates a fortress and is a sorting reversal iff case 3 applies (Lemmas A3 and A4). ■

ACKNOWLEDGMENTS

Thanks to Bernard Moret, my adviser at the University of New Mexico, for encouragement and support, and to the anonymous reviewers of this paper for corrections and helpful suggestions.

REFERENCES

Bader, D.A., Moret, B.M.E., and Yan, M. 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Proc. 7th Int. Workshop on Algorithms and Data Structures*, *Lecture Notes in Computer Science*, vol. 2125, 365–376.

Bafna, V., and Pevzner, P.A. 1993. Genome rearrangements and sorting by reversals. *Proc. 34th Ann. IEEE Symposium on Foundations of Computer Science*, 148–157.

Bergeron, A. 2001. A very elementary presentation of the Hannenhalli–Pevzner theory. *Proc. 12th Ann. Symposium on Combinatorial Pattern Matching*, *Lecture Notes in Computer Science*, vol. 2089, 106–117.

Bergeron, A., and Strasbourg, F. 2001. Experiments in computing sequences of reversals. *Proc. 1st Int. Workshop on Algorithms in Bioinformatics*, *Lecture Notes in Computer Science*, vol. 2149, 164–174.

Berman, P., and Hannenhalli, S. 1996. Fast sorting by reversal. *Proc. 7th Ann. Symposium on Combinatorial Pattern Matching*, 168–185.

Blanchette, M., Kunisawa, T., and Sankoff, D. 1996. Parametric genome rearrangement. *Gene* 172, GC11–GC17.

Bourque, G., and Pevzner, P.A. 2002. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.* 12, 26–36.

Caprara, A. 1997. Sorting by reversals is difficult. *Proc. 1st Ann. Int. Conf. Computational Molecular Biology*, 75–83.

Hannenhalli, S., and Pevzner, P.A. 1995. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proc. 27th Ann. ACM Symposium on the Theory of Computing*, 178–189.

Kaplan, H., Shamir, R., and Tarjan, R.E. 1999. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.* 29(3), 880–892.

McLysaght, A., Seoighe, C., and Wolfe, K.H. 2000. High frequency of inversions during eukaryote gene order evolution. In Sankoff, D., and Nadeau, J.H., eds., *Comparative Genomics*, 47–58, Kluwer Academic Press, NY.

Moret, B.M.E., Siepel, A., Tang, J., and Liu, T. 2002. Inversion medians outperform break-point medians in phylogeny reconstruction from gene-order data. *Proc. 2nd Int. Workshop on Algorithms in Bioinformatics*, *Lecture Notes in Computer Science*, vol. 2452, 521–536.

Sankoff, D., and El-Mabrouk, N. 2002. Genome rearrangement. In Jiang, T., Xu, Y., and Zhang, M., eds., *Current Topics in Computational Molecular Biology*, 135–156, MIT Press, New Haven, CT.

Setubal, J., and Meidanis, J. 1997. *Introduction to Computational Molecular Biology*, PWS Publishing, Boston, MA.

Siepel, A. 2001. *Exact Algorithms for the Reversal Median Problem.* Master's thesis, University of New Mexico, Albuquerque, New Mexico. Available at *www.cse.ucsc.edu/~acs/masters-thesis.html*.

Address correspondence to:
Adam C. Siepel
Center for Biomolecular Science and Engineering
Baskin Engineering Building
University of California
Santa Cruz, CA 95064

E-mail: acs@soe.ucsc.edu