Efficient bounds for oriented chromosome inversion distance

John Kececioglu* David Sankoff†

Abstract We study the problem of comparing two circular chromosomes that have evolved by chromosome inversion, assuming that the order of corresponding genes is known, as well as their orientation. Determining the minimum number of inversions is equivalent to finding the minimum of reversals to sort a signed circular permutation, where a reversal takes an arbitrary substring of elements and reverses their order, as well as flipping their sign. We show that tight bounds on the minimum number of reversals can be found by simple and efficient algorithms.

Keywords Genome rearrangements, chromosome inversion, reversal distance, sorting by signed reversals

1 Introduction

With the advent of genome sequencing in molecular biology, there is an increasing interest in the development of algorithms for comparing chromosomes in terms of high-level mutational events. In this paper we consider the comparison of two circular chromosomes from two related organisms on the basis of the order of their common genes, and in terms of chromosome inversions. An *inversion* replaces an arbitrary region of the chromosome with the reverse complement sequence. This has the effect of reversing the order of the genes within the region, as well as complementing the sequence for each gene.

We model this comparison problem as that of determining the minimum number of reversals to transform one signed circular permutation into another. The permutations represent the order of corresponding genes, and the sign of an element represents whether or not the sequence for the gene in one chromosome is reverse complemented in the other chromosome.

*Department of Computer Science, University of California, Davis, CA 95616, USA. Electronic mail: kece@cs.ucdavis.edu. Research supported by a U.S. Department of Energy Human Genome Distinguished Postdoctoral Fellowship.

†Centre de recherches mathématiques, Université de Montréal, C.P. 6128, succ. A, Montréal, Québec H3C 3J7, Canada. Electronic mail: sankoff@ere.umontreal.ca. Research supported by grants from the Natural Sciences and Engineering Research Council of Canada, and the Fonds pour la formation de chercheurs et l'aide à la recherche (Québec). David Sankoff is a fellow of the Canadian Institute for Advanced Research.

We write a permutation π as a list $(\pi_i\ \pi_2\ \cdots\ \pi_n)$ of elements, where $\pi_i = \pi(i)$. A *signed permutation* is just an ordinary permutation, except that elements may be positively or negatively signed. In a *circular permutation*, any cyclic shift of a permutation is considered to be equivalent. We adopt the convention that for a signed circular permutation, element 1 is always in the first position, and is positive. A *reversal* ρ of interval $[i, j]$ is the permutation

$$\rho = (1\ 2\ \cdots\ i{-}1\ {-}(j)\ {-}(j{-}1)\ \cdots\ {-}(i)\ j{+}1\ \cdots\ n).$$

Applying ρ to a permutation π by the composition $\pi \circ \rho$ has the effect of reversing the order of the elements in π in interval $[i, j]$, and flipping their sign.

The *reversal distance* between two n element permutations σ and τ, denoted by $d(\sigma, \tau)$, is the length ℓ of a shortest series of reversals $\rho_1, \rho_2, \ldots, \rho_\ell$ such that

$$\sigma \circ \rho_1 \circ \rho_2 \circ \cdots \circ \rho_\ell = \tau.$$

Since we can always take one permutation to be the *identity permutation* $\iota = (1\ 2\ \cdots\ n)$, the problem is equivalent to finding the minimum number of reversals to transform a permutation π into ι. This formulation is called *sorting by reversals*. We denote the minimum number of reversals to sort π by $d(\pi)$.

Kececioglu and Sankoff [7, 8] studied the problem of sorting an unsigned linear permutation by reversals, and developed the first approximation algorithm for the problem. Their algorithm runs in $O(n^2)$ time and is guaranteed to use no more than twice the minimum number of reversals. They also developed a family of lower bounds on $d(\pi)$ in terms of the cycles of a graph whose evaluation required linear programming. Using these bounds in the context of a branch-and-bound procedure, they were able to solve to optimality permutations of up to 30 elements, and bound the exact value to within 2 reversals for permutations of up to 50 elements.

Bafna and Pevzer [1] subsequently found an improved approximation algorithm with a performance guarantee of $\frac{7}{4}$, and presented an approximation algorithm for signed reversals with a guarantee of $\frac{3}{2}$. They also resolved a conjecture due to Holger Gollan [7] that for every n there is an n-element permutation requiring $n - 1$ reversals.

This paper applies the techniques developed in [8] to the case of signed circular permutations. This is important for the study of organisms with circular chromosomes when the orientation of genes, as well as their order, can be determined. In this context a tremendous simplification occurs. The lower bound can be evaluated in linear time, without the need for linear programming, and it can be given a simple combinatorial proof. The greedy approximation algorithm is also shown to perform remarkably well in this context. We first review the greedy algorithm in Section 2, and then develop a simplified lower bound in Section 3. Section 4 develops a more parsimonious branch-and-bound algorithm. We also prove in this section one of the first theorems concerning an optimal solution, namely that there is always an optimal series in which a quantity called the number of breakpoints is non-increasing. In Section 5 we show that for every n there is a signed permutation requiring at least $n - 1$ reversals, which gives a tight bound on the diameter for the signed problem and proves that the greedy algorithm is essentially worst-case

optimal. Section 6 studies the empirical performance of these algorithms on random permutations, and permutations generated by random reversals. Suprisingly, the observed average difference between the greedy approximation and the lower bound in these experiments remained less than 1 reversal for signed permutations with up to 10,000 elements, and the maximum observed difference was less than 4 reversals. The tightness of these bounds allowed us to solve most problems on permutations of up to 250 elements to optimality in less than an hour of computation, and failing that, to determine the minimum to within 1 reversal.

Throughout the paper we use n to denote the number of elements in a permutation. Unless otherwise stated, the term "permutation" will refer to signed circular permutations.

2 Upper bounding the distance

In this section we review for completeness the greedy algorithm of Kececioglu and Sankoff [8] for sorting by reversals, and adapt it to the context of signed circular permutations.

The key idea is that of a *breakpoint*. A breakpoint of a permutation π is a pair $(i, i \oplus 1)$ of consecutive positions in π such that $\pi_{i \oplus 1} \neq \pi_i \oplus 1$. Here $\oplus$ is the usual operation of addition, except that $n \oplus 1 = 1$, and $(-1) \oplus 1 = (-n)$. Similarly we use $\ominus$ for ordinary subtraction, except that $1 \ominus 1 = n$, and $(-n) \ominus 1 = (-1)$. A *strip* is a maximal run of elements between breakpoints.

We use $\Phi(\pi)$ to denote the *number of breakpoints* of π. A reversal can change $\Phi(\pi)$ by removing 2, 1, or 0 breakpoints, or by creating 1 or 2 breakpoints. We call a reversal that removes i breakpoints an *i-reversal*, where $i \in \{2, 1, 0, -1, -2\}$.

To sort a permutation π, we must reduce the number of breakpoints from $\Phi(\pi)$ to $\Phi(\iota) = 0$. This suggests the following *greedy heuristic*: at any step, choose a reversal that removes the most breakpoints. The algorithm of Figure 1 uses this rule, with the twist that it favors reversals that leave negative elements.

```
procedure Greedy(π) begin
    i := 0
    while π contains a breakpoint do begin
        i := i + 1
        Let ρ_i be a reversal that removes the most breakpoints of π,
            resolving ties in favor of reversals that leave negative elements.
        π := π ∘ ρ_i
    end
    return i, ρ_1 ρ_2 ··· ρ_i
end
```

Figure 1 The greedy algorithm.

The following results were proved in [8] for the greedy algorithm in the context of unsigned permutations. They carry over to signed permutations, and we state them without proof.

Lemma 1 *Every permutation with a negative element has a 1-reversal or a 2-reversal. Consequently the greedy algorithm performs a 0-reversal only when the permutation has no negative elements. Furthermore, if every reversal that removes a breakpoint of π leaves a permutation with no negative elements, then π has a 2-reversal.*

Theorem 1 *The greedy algorithm sorts any permutation π with a negative element in at most $\Phi(\pi)-1$ reversals, and any permutation without a negative element in at most $\Phi(\pi)$ reversals. Consequently*

$$d(\pi) \leq \Phi(\pi).$$

A consequence of Theorem 1 [7] is that the greedy algorithm is an approximation algorithm for sorting by reversals with a worst-case peformance ratio of 2. The algorithm can be implemented to run in time $O(n+\Phi^2(\pi)) = O(n^2)$, using $O(n)$ space [8]. Bafna and Pevzner [1] subsequently found a more elaborate algorithm that uses the greedy algorithm as a subroutine to guarantee a performance ratio of $\frac{3}{2}$.

In Section 6 we present extensive experiments indicating that, in contrast to what was observed for unsigned permutations [8], for *signed* permutations the simple greedy algorithm may yield satisfactory bounds, as it appears to have good performance in terms of its *difference* from the optimum, even for very large problems.

3 Lower bounding the distance

In this section we derive a lower bound on $d(\pi)$ using the notion of a cycle graph introduced in [7]. We show that, in contrast to unsigned permutations, the bound has a remarkably simple proof, and can be evaluated in $O(n)$ time.

With each permutation π we associate an undirected graph $G(\pi)$. Its vertices are as follows. For every pair $(i, i \oplus 1)$ of consecutive positions in π, there is a vertex v_i in G. In effect, vertices lie between positions of π. We call π_i the left value for v_i, and $\pi_{i\oplus 1}$ the right value for v_i. Similarly we call v_i the right vertex for value π_i, and v_i the left vertex for value $\pi_{i\oplus 1}$.

Figure 2 shows the construction of edges for G. Let x and y be the left and right values of vertex v_i. There is an edge from v_i to a vertex associated with $x \oplus 1$ and to a vertex associated with $y \ominus 1$. Consider the vertex associated with $x \oplus 1$. Either value $x \oplus 1$ or $-(x \oplus 1)$ appears in π. If value $x \oplus 1$ appears, v_i is joined by an edge to the vertex on the left of $x \oplus 1$. If value $(x \oplus 1)$ appears, v_i is joined to the vertex on the right of $-(x \oplus 1)$. In a similar manner v_i is joined to a vertex associated with $y \ominus 1$, as shown in the figure. We note that a pair of vertices may

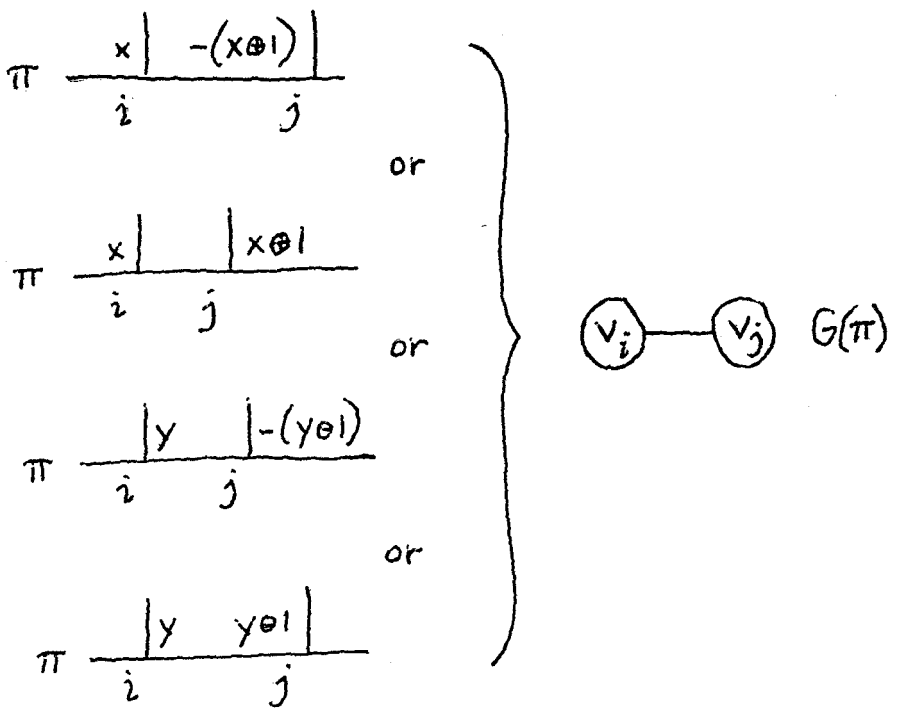


Figure 2 Construction of graph $G(\pi)$.

have two parallel edges between them, which we treat as a cycle of length two, and that a vertex may have a self-loop.

A significant difference between this construction and the one given in [7] for unsigned permutations is that every vertex has degree two. This implies G has a particularly simple structure: it consists of vertex-disjoint cycles. This radically simplifies the form of our lower bound, as well as its proof.

In the following we denote the *number of cycles* of $G(\pi)$ by $\Psi(\pi)$.

Theorem 2 *For every signed circular permutation π on n elements,*

$$d(\pi) \geq n - \Psi(\pi).$$

Proof Consider the effect on $G(\pi)$ of an arbitary reversal $[i+1, j]$. This changes values only at vertices v_i, v_{i+1}, ..., v_j.

At an interior vertex, a vertex other than v_i and v_j, the reversal exchanges the left value with the right value and negates their signs. In our construction, this does not affect the cycle passing through the vertex.

At the end vertices v_i and v_j, the reversal exchanges the right value of v_i with the left value of v_j, and negates their sign. This has three possible effects, as shown

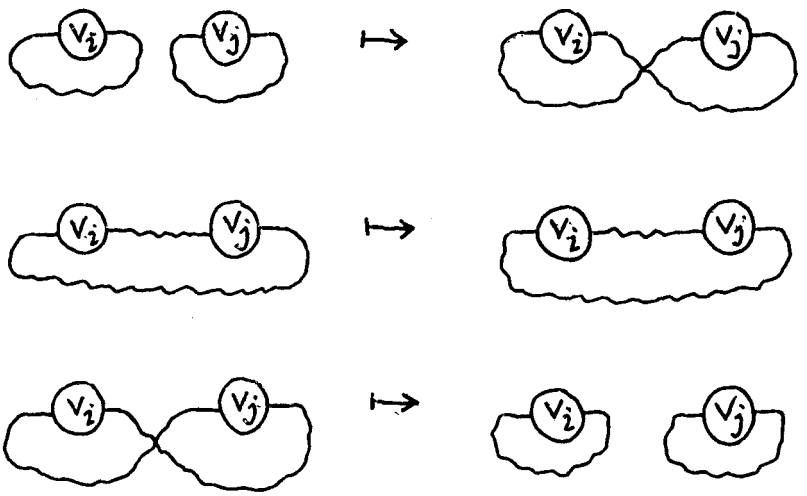


Figure 3 The effect of a reversal $[i+1, j]$ on $G(\pi)$.

in Figure 3. In each case the number of cycles in G either decreases by one, remains the same, or increases by one.

A series of reversals that sorts π must change the number of cycles from $\Psi(\pi)$ to $\Psi(\iota)$. A reversal can change $\Psi(\pi)$ by at most one. Hence any series that sorts π requires at least $|\Psi(\pi) - \Psi(\iota)|$ reversals, which is $n - \Psi(\pi)$. □

Bafna and Pevzner [1] have also observed that their lower bound, which is in terms of edge-disjoint Eulerian cycles that alternate in color in an edge-colored graph, simplifies as well in the context of signed permutations [1].

Since $G(\pi)$ consists of vertex-disjoint cycles, $\Psi(\pi)$ is simply the number of connected components, which yields the algorithm of Figure 4.

```
procedure LowerBound(π) begin
    Let n be the number of elements of π.
    Construct G(π) and count the number k of connected components.
    return n − k
end
```

Figure 4 The lower bound algorithm.

This trivial $O(n)$ time algorithm gives an astonishingly tight lower bound. In

our experiments of Section 6, it never differed from the greedy approximation, hence from the minimum, by more than 4 reversals on random permutations with 10 to 10,000 elements. To partially explain the closeness of the lower bound to the greedy approximation, we note the following:

- The bound counts 1- and 2-reversals in a series equally, since for both, $\Delta\Psi = +1$.
- The bound does not count 0-reversals in a series, since for such a reversal $\Delta\Psi = 0$ or -1. However, we note that by Lemma 1, a 0-reversal is performed by the greedy algorithm only when every element is positive, which is an extremely rare event.
- The bound does not count (-1) or (-2)-reversals, since for both, $\Delta\Psi = -1$. Such reversals however are never needed to achieve the minimum, as we prove in the next section.

4 Determining the exact distance

We can use these bounds to determine the minimum number of reversals by the branch-and-bound algorithm of Figure 5. The algorithm explores a tree of reversals depth-first, using the lower bound of Section 3 to prune subtrees, and can be implemented to run in $O(mn)$ time for a tree of m nodes, using $O(\Phi^2(\pi)) = O(n^2)$ space. It differs from the branch-and-bound algorithm for unsigned permutations [7] in that the tree does not contain reversals that cut strips. This reduces the branching factor significantly, from $\binom{n}{2}$ to $\binom{\Phi(\pi)}{2}$.

We now prove that the algorithm is correct, and note that the following is one of the first theorems regarding the structure of an optimal solution.

Theorem 3 *Every permutation has an optimal solution that does not cut strips.*

Proof We show that any series of reversals that cuts strips while sorting a permutation π, can be transformed into an equivalent series that also sorts π, but does not cut strips, without increasing the length of the series. Since we can apply this to a shortest series, there is an optimal solution that does not cut strips. The argument uses induction on both the length of the series and the number of reversals that cut strips. A series of one reversal that sorts a permutation cannot cut any strips, so the basis holds.

In general, consider a series that sorts π and cuts strips. Let ρ be the last reversal that cuts a strip and σ be the permutation to which ρ applies. To simplify matters, we assume that only the left end of ρ cuts a strip. (If ρ cuts a strip on the right, we can apply the argument to the right end as well.) Reversal ρ then has the

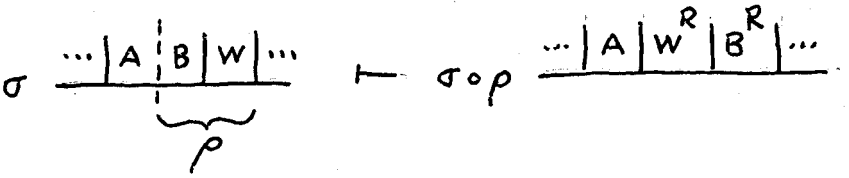
```
global d*, r*[1..n], r[1..n]

procedure BranchAndBound(π) begin
    d*, r* := UpperBound(π)
    Search(π, 0)
    return d*, r*
end

procedure Search(π, d) begin
    if π is the identity permutation then
        d*, r* := d, r
    else
        for every reversal ρ that does not cut a strip,
            considering reversals in order of decreasing ΔΦ, and
            resolving ties in favor of reversals that leave negative strips,
        do
            if d + 1 + LowerBound(π ∘ ρ) < d* then begin
                r[d + 1] := ρ
                Search(π ∘ ρ, d + 1)
            end
end
```

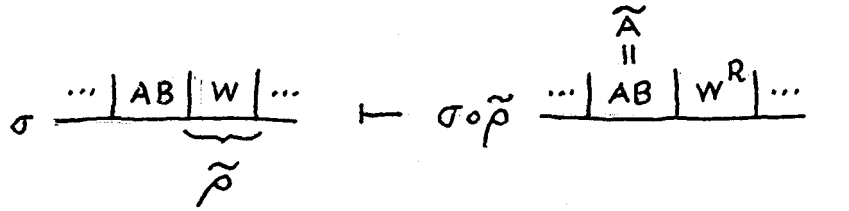
Figure 5 The branch-and-bound algorithm.

following form.



Here A, B, and W represent substrings of σ, and W^R represents the string formed by reversing W and negating its elements. Strings A and B form a strip, while W is an arbitrary substring of strips.

Instead of cutting strip AB with ρ, we perform the following reversal $\widetilde{\rho}$.



All reversals after ρ are then simulated by the following scheme. No reversal

after ρ cuts a strip, so B^R remains intact in all subsequent permutations. Delete B^R from these permutations, and give strip AB in $\sigma \circ \widetilde{\rho}$ the new name $\widetilde{A}$. If after deleting B^R we identify $\widetilde{A}$ in $\sigma \circ \widetilde{\rho}$ with strip A in $\sigma \circ \rho$, permutations $\sigma \circ \widetilde{\rho}$ and $\sigma \circ \rho$ are identical. Thus, after performing this deletion and renaming, the endpoints of the reversal following ρ can be mapped onto $\sigma \circ \widetilde{\rho}$. Continuing in this manner, we can map all subsequent reversals onto the transformed permutations.

In the identity permutation, either configuration AB or B^RA^R appears. In the first case, A must be reversed an even number of times by reversals following ρ, and in the second case, A must be reversed an odd number of times. Thus, when we finish simulating the remaining reversals on $\sigma \circ \widetilde{\rho}$, the strip $\widetilde{A}$, which will be reversed the same number of times as A, will be oriented at the conclusion as it should be in the identity permutation. In short, the transformed series sorts π.

It may happen in the course of the simulation that a reversal which formerly did not cut a strip is transformed into a reversal that now cuts a strip. The length of the suffix of the series that contains such a reversal has decreased by at least one, however, since $\widetilde{\rho}$ does not cut a strip. By induction, this suffix can be transformed into a series that does not cut strips. At this point the total number of reversals that cut strips in the series has decreased by one. By induction on the number of such reversals, the entire series can be transformed into a series free of reversals that cut strips. □

To compute the initial upper bound, we improve the greedy approximation using look-ahead. Instead of constructing a series by choosing a reversal that immediately removes the most breakpoints, we construct a series by looking ahead k reversals, finding a series of length k that removes the maximum number of breakpoints, and performing these k reversals. The best series of length k can be found by branch-and-bound just as in Figure 5. The only difference is that the search is stopped at a depth of k, and we are maximizing the number of breakpoints eliminated, rather than minimizing the length of a solution.

To carry this out, we need an upper bound on the total number of breakpoints that can be eliminated in a given number of reversals, and a way of obtaining a good initial series of k reversals. We find a good initial series by calling the algorithm recursively: to find a series of length k that eliminates a lot of breakpoints, we find a best series of length $\lceil \frac{k}{2} \rceil$ and follow it by a best series of length $\lfloor \frac{k}{2} \rfloor$. At the bottom of the recursion, we use the simple greedy algorithm. We call this approach logarithmic *bootstrapping*, as it bootstraps itself to a good initial solution in a logarithmic number of levels.

An upper bound on the number of breakpoints that can be eliminated in k reversals is obtained using the idea of a *cycle packing*, introduced in [8] in the context of unsigned permutations. A cycle packing for a permutation π is a set of cycles of $G(\pi)$ that are vertex-disjoint. We say the size of a cycle is its number of vertices minus one, and the size of a packing is the sum of the sizes of its cycles. A *k-packing* is a packing of size at most k.

Theorem 4 *Let c be the size of a maximum k-packing for π. Then the number*

of breakpoints of π that can be eliminated in k reversals is at most

$$\min\{k + c, \Phi(\pi)\}.$$

Since $G(\pi)$ consists of vertex-disjoint cycles, a maximum k-packing for a signed permutation can be found in $O(n)$ time with a simple greedy procedure. Again this in contrast to the situation for unsigned permutations, where bounding the size of a maximum packing required linear programming [7].

5 Bounding the diameter

Much of the theoretical work related to sorting by reversals has been concerned with bounds on the diameter [5, 3, 6]. The *diameter* $D(n)$ of the set S_n of n-element permutations, with respect to reversal distance, is the maximum number of reversals required to sort an n-element permutation:

$$D(n) = \max_{\pi \in S_n} d(\pi).$$

We now show that the lower bound of Section 3, together with the greedy algorithm, give a tight bound on the diameter.

Theorem 5 *For signed circular permutations, and all n,*

$$n-1 \leq D(n) \leq n.$$

Proof By Theorems 1 and 2 we know that for any permutation π on n elements,

$$n - \Psi(\pi) \leq d(\pi) \leq \Phi(\pi).$$

For circular permutations, $\Phi(\pi)$ attains a maximum value of n, which gives the upper bound on $D(n)$. We prove the lower bound on $D(n)$ by demonstrating a permutation π for every n for which $\Psi(\pi)$ is 1. The form of the permutation depends on whether n is odd or even.

For odd n, consider

$$\omega_n = (n\ n{-}1\ \cdots\ 2\ 1).$$

Recall that $G(\omega_n)$ has a vertex between every consecutive pair of positions in ω_n. Number the vertices so that vertex v_i contains values $(i \oplus 1, i)$. In $G(\omega_n)$, v_i is adjacent to two vertices: those containing values $(\cdot, i \oplus 2)$ and $(i \ominus 1, \cdot)$, which are $v_{i\oplus 2}$ and $v_{i\ominus 2}$. Consider following edges from v_2. We will visit

$$v_2, v_4, v_6, \ldots, v_{n-1}, v_1, v_3, v_5, \ldots, v_n, v_2.$$

This is every vertex of $G(\omega_n)$. Hence $G(\omega_n)$ consists of a single cycle when n is odd.

For even n, consider

$$\zeta_n = \left(-(\tfrac{n}{2}{+}1)\ \ \tfrac{n}{2}{+}2\ \ -(\tfrac{n}{2})\ \ \tfrac{n}{2}{+}3\ \ \cdots\ \ n{-}2\ \ -4\ \ n{-}1\ \ -3\ \ n\ \ -2\ \ 1\right).$$

We group the vertices of $G(\zeta_n)$ into two sets and name them $\{v_2, v_3, \ldots, v_{\frac{n}{2}+1}\}$ and $\{w_2, w_3, \ldots, w_{\frac{n}{2}+1}\}$. In general vertex v_i contains values $(\cdot, -i)$, and vertex w_i contains values $(-i, \cdot)$. We note the following three properties of $G(\zeta_n)$:

(i) w_i is joined by an edge to v_i for $2 \leq i \leq \frac{n}{2} + 1$,
(ii) v_i is joined by an edge to w_{i+1} for $2 \leq i \leq \frac{n}{2}$, and
(iii) $v_{\frac{n}{2}+1}$ is joined by an edge to w_2.

Together these imply $G(\zeta_n)$ is a single cycle. □

Theorem 5 implies that for signed linear permutations, $n - 2 \leq D(n) \leq n - 1$. Moreover these bounds are tight, as shown in the next section.

6 Computational results

We studied the behavior of the bounds and $d(\pi)$ in experiments on all permutations of a fixed size, random permutations, and permutations generated by a fixed number of random reversals. We now summarize the results.

6.1 Exact distances for small permutations

Table 1 gives the distribution of $d(\pi)$ for small n, obtained by running our exact algorithm on all n-element permutations. The distribution differs from that of unsigned permutations [8] in two respects.

Table 1 The number of permutations on n elements at distance d from the identity.

	n							
d	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1
1	1	3	6	10	15	21	28	36
2		3	16	50	120	245	448	756
3		1	25	170	700	2,170	5,586	12,600
4				145	1,554	8,820	35,612	115,254
5				8	1,447	19,495	138,229	684,525
6					3	15,148	262,688	2,295,786
7						180	202,464	4,198,049
8							64	3,006,846
9								8,067

First, we note that the extremal permutations are numerous (those π for which $d(\pi) = D(n)$) in contrast to the unsigned case. For unsigned permutations, Holger Gollan conjectured in a talk in 1992 that the extremal permutation is unique up to taking its inverse, and took a large step towards a proof by positing its general form

(see [7] for the statement of the conjecture and the form of Gollan's permutation). Bafna and Pevzner [1] established the conjecture by applying their lower bound to this permutation.

Second, the diameter does not grow uniformly, again in contrast to the unsigned case [1]. The table shows that the bounds of Section 5 on the diameter are tight. We suspect, however, that the diameter does not hit the lower bound infinitely often, and conjecture that for all sufficiently large n, $D(n) = n$.

To aid characterization of an extremal permutation, Table 2 lists the extremal permutations consisting only of positive elements for $n \leq 6$. Since any permutation containing a negative element can be sorted in $n - 1$ steps by Theorem 1, only positive permutations are candidates for showing $D(n) = n$.

Table 2 Extremal positive permutations on n elements.

n			
3	4	5	6
1 3 2	1 2 4 3	1 5 2 4 3	1 6 2 4 3 5
	1 4 2 3	1 4 2 5 3	1 3 5 4 6 2
	1 4 3 2	1 5 4 3 2	1 3 2 4 6 5
	1 3 4 2	1 4 3 5 2	
	1 3 2 4	1 3 5 4 2	
		1 5 3 2 4	
		1 3 5 2 4	
		1 3 2 5 4	

6.2 Bounds for random permutations

To study the quality of the greedy approximation we compared it to the lower bound on random permutations. The results, shown in Table 3, are striking.

Unexpectedly, the *difference* remained bounded for n ranging from 10 to 10,000. This is quite different from the behavior observed for unsigned permutations. For random unsigned permutations the average *ratio* A/L varied between 1.2 and 1.3 for n from 10 to 100 [8].

Table 4 records the variance in L for the same range of n. The variance is small, and slow growing, but it is not sufficiently small to completely account for the tightness of the bounds. In the next experiments on randomly reversed permutations, for example, the difference is small even when the standard deviation in the lower bound exceeds 8 reversals.

Moreover, the concentration of distance about the mean for random permutations may be useful for detecting when the gene order between two organisms is sufficiently scrambled to suggest that they are unrelated. For example, Table 4 indicates that a measured distance of 45 inversions for 50 genes is around what

Table 3 Difference between the approximation A and the lower bound L for random permutations. For each n the sample size is 100.

	$A-L$	
n	mean	max
10	0.05	1
25	0.12	2
50	0.14	2
100	0.06	1
250	0.10	2
500	0.11	2
1,000	0.08	1
2,500	0.15	1
5,000	0.11	2
10,000	0.11	1

Table 4 Growth of the lower bound L for random permutations. For each n the sample size is 100.

	L				$n-L$
n	mean	min	max	dev	mean
10	7.82	5	9	0.91	2.18
25	22.38	17	24	1.32	2.62
50	47.02	43	49	1.36	2.98
100	96.64	91	99	1.65	3.36
250	246.25	241	249	1.64	3.75
500	496.12	492	499	1.73	3.88
1,000	995.56	990	999	1.92	4.44
2,500	2,494.94	2,488	2,498	2.14	5.06
5,000	4,994.53	4,988	4,999	2.13	5.47
10,000	9,994.23	9,988	9,998	2.15	5.77

one would expect for completely unrelated organisms. We note that the expected difference between n and $d(\pi)$ appears to be proportional to roughly $\log n$.

6.3 Bounds for permutations generated by random reversals

An input more typical than a random permutation would be one generated from the identity by a fixed number k of random reversals. Table 5 shows the observed difference between the approximation and the lower bound for a range of k and n. As can be seen, the difference remains quite small.

It is interesting that the difference observed is generally greatest when the permutation is large but the number of reversals is small. At this extreme, the ends of the reversals are likely to cut at disjoint locations, which causes a preponderance of 2-reversals in a solution. As the greedy algorithm does not distinguish between the 2-reversals available, in such a situation it is more likely to choose a "bad" 2-reversal—one that prevents other 2-reversals, or fails to set them up, by reversing one of their endpoints. Bafna and Pevzner [1] improved the performance ratio of the greedy algorithm by refining its choice in this situation, and it would be interesting to see if their improvement smooths out the behavior of the algorithm for the full range of k.

Comparison of L and k in Table 5 reveals that as k gets large relative to n, reversal distance underestimates the true number of reversals. What percentage of reversals can we recover by measuring reversal distance?

Table 6 indicates that for roughly $k < .5n$ reversal distance is a good measure of the true number of reversals. We have observed the same transition point for $n \leq 1,000$, and for unsigned permutations as well [7]. This suggests the following rule of thumb: a measurement of inversion distance d between two organisms should be based on a sample of more than $2d$ approximately equally-spaced markers.

6.4 Exact distances for large permutations

Our last experiments studied the behavior of the exact algorithm on large permutations. Table 7 summarizes the results for both permutations generated by k random reversals, and completely random permutations ($k = \infty$).

Except for one problem on 100 elements, all problems of up to 250 elements were solved to optimality. As can be seen from the table, whenever optimal solution was possible, the initial lower bound was tight.

Generally when there was a gap between the lower bound and the upper bound obtained with look-ahead, the branch-and-bound algorithm was unable to close the gap within the search limit. The one exception is a random problem on 500 elements where the exact algorithm closed the gap of 1 reversal in a search of around 29,000 nodes.

For all problems the series obtained with look-ahead was known at termination to be within 1 reversal of the optimum. The largest improvement obtained by look-ahead was a savings of 4 reversals on a series of length 104. Running times on a standard workstation varied from less than 1 minute for 50 element problems, to around 45 minutes for 500 element problems.

Table 5 Difference between the approximation A and the lower bound L for permutations generated by k random reversals. For each n and k the sample size is 100.

		L		$A-L$	
n	k	mean	dev	mean	max
25	5	4.95	0.30	0.32	2
	10	9.68	0.75	0.16	1
	25	18.66	1.74	0.11	2
50	5	5.00	0.00	0.19	2
	10	9.88	0.46	0.17	2
	25	24.18	1.06	0.25	2
	50	39.83	2.21	0.10	2
100	5	5.00	0.00	0.17	2
	10	10.00	0.00	0.44	4
	25	24.87	0.56	0.24	2
	50	48.90	1.24	0.25	2
	100	81.16	3.03	0.07	1
250	10	10.00	0.00	0.37	2
	25	24.98	0.20	0.28	2
	50	49.92	0.37	0.48	3
	100	99.49	0.93	0.27	3
	250	207.22	4.07	0.11	2
500	10	10.00	0.00	0.23	2
	25	25.00	0.00	0.43	4
	50	49.98	0.20	0.51	3
	100	99.93	0.33	0.54	3
	250	248.48	1.64	0.40	3
	500	415.71	6.72	0.16	2
1000	10	10.00	0.00	0.56	4
	25	25.00	0.00	0.54	4
	50	50.00	0.00	0.49	2
	100	100.00	0.00	0.50	3
	250	249.78	0.60	0.60	3
	500	498.12	1.67	0.44	3
	1000	835.58	8.66	0.09	1

Table 6 Difference between the true number of reversals and the lower bound L for permutations generated by k random reversals. The sample for each k is 100 permutations of 1,000 elements.

	L		$k-L$
k	mean	dev	mean
50	50.00	0.00	0.00
100	100.00	0.00	0.00
150	149.96	0.28	0.04
200	199.85	0.61	0.15
250	249.78	0.60	0.22
300	299.75	0.66	0.25
350	349.51	0.89	0.49
400	399.49	0.89	0.51
450	449.11	1.19	0.89
500	498.12	1.67	1.88
550	546.85	2.56	3.15
600	593.30	3.88	6.70
650	636.81	5.60	13.19
700	674.42	6.06	25.58
750	709.36	6.38	40.64
800	741.44	8.44	58.56
850	768.54	8.58	81.46
900	793.02	8.63	106.98
950	815.46	8.59	134.54
1000	835.58	8.66	164.42
⋮			⋮
450	449.11	1.19	0.89
460	459.08	1.33	0.92
470	468.60	1.55	1.40
480	478.78	1.29	1.22
490	488.33	1.69	1.67
500	498.12	1.67	1.88
⋮			⋮

Table 7 Behavior of the exact algorithm on permutations of n elements with k random reversals. Parameters are the exact algorithm value E, upper bound U, approximation A, lower bound L, and tree sizes T_E and T_U for the exact and upper bound algorithms. The sample size for each n and k is 10, with look-ahead 5 and maximum tree sizes of 10,000 and 100,000 for the upper bound and exact algorithms, respectively.

		$E-L$	$U-E$	$A-U$	T_U	T_E
n	k	max	max	max	mean	max
25	5	0	0	1	20	0
	10	0	0	1	322	0
	25	0	0	0	1,243	0
	∞	0	0	2	1,232	0
50	10	0	0	0	264	0
	25	0	0	1	7,526	0
	50	0	0	0	2,095	0
	∞	0	0	2	6,534	0
100	25	1	0	2	6,020	100,000
	50	0	0	1	9,593	0
	100	0	0	0	7,207	0
	∞	0	0	1	3,020	0
250	25	0	0	2	2,148	0
	50	0	0	2	9,510	0
	100	0	0	2	10,000	0
	∞	0	0	0	9,483	0
500	25	1	1	1	105	100,000
	50	0	0	3	3,604	0
	100	1	0	4	10,000	100,000
	∞	0	1	1	9,000	28,931

7 Conclusions

While computing the reversal distance between signed permutations appears to be hard, good bounds can be obtained quite efficiently. The greedy algorithm yields an upper bound in $O(n^2)$ time, and the lower bound can be evaluated in $O(n)$ time.

As well as being simple and easy to implement, these methods yield bounds that are extremely tight. For random and randomly reversed permutations we have not observed them to differ by more than 4 reversals in extensive trials involving permutations of up to 10,000 elements. Coupled with a branch-and-bound algorithm using look-ahead, we could solve most problems of up to 250 elements to optimality in less than an hour, and failing this, determine the reversal distance to within 1 reversal.

This success is due by and large to the tightness of the lower bound, and suggests that if one is simply interested in the reversal distance between two organisms and not in an actual series of reversals, for instance when constructing an evolutionary

tree, one might simply evaluate the lower bound, in linear time. This phenomena has been observed for other optimization problems, such as in the traveling salesman problem, where the *value* of an optimal solution can be determined quite closely and easily by a suitable relaxation, such as to a matching problem, yet finding a *solution* near that value requires a tremendous amount of effort. Indeed, it would be interesting to examine how well a shortest series of reversals recovers the actual reversals in a random series.

Given that the lower bound relaxes all information concerning how reversals in a series overlap, and that the greedy algorithm forms a series based on extremely local and naive decisions, it is a mystery why the bounds they yield are so tight, and in particular why their *difference* is so small. Can one prove that the expected difference between the greedy approximation and the lower bound is a slowly growing function? Our experimental results suggest that this difference grows more slowly than $\log n$, and may in fact be bounded by a constant. Fortunately the lower bound and the greedy algorithm have a simple form, which provides some hope for a satisfying analysis.

References

[1] Bafna, Vineet and Pavel A. Pevzner. Genome rearrangements and sorting by reversals. In Proceedings of the 34th Annual IEEE *Symposium on Foundations of Computer Science*, 148–157, November 1993.

[2] Chrobak, M., T. Szymacha, and A. Krawczyk. A data structure useful for finding Hamiltonian cycles. *Theoretical Computer Science* 71, 419–424, 1990.

[3] Cohen, David S. and Manuel Blum. On the problem of sorting burnt pancakes. Manuscript, Computer Science Division, University of California at Berkeley, 1993.

[4] Fredman, M.L., D.S. Johnson, L.A. McGeoch, and G. Ostheimer. Data structures for traveling salesmen. In Proceedings of the 4th Annual ACM-SIAM *Symposium on Discrete Algorithms*, 145–154, 1993.

[5] Gates, William H. and Christos H. Papadimitriou. Bounds for sorting by prefix reversals. *Discrete Mathematics* 27, 47–57, 1979.

[6] Heydari, Mohammad H. *The Pancake Problem.* PhD dissertation, Department of Computer Science, University of Texas at Dallas, 1993.

[7] Kececioglu, John and David Sankoff. Exact and approximation algorithms for the inversion distance between two chromosomes. In Proceedings of the 4th Annual *Symposium on Combinatorial Pattern Matching*, Lecture Notes in Computer Science 684, Springer-Verlag, 87–105, June 1993. (An earlier version appeared as "Exact and approximation algorithms for the reversal distance between two permutations," Technical Report 1824, Centre de recherches mathématiques, Université de Montréal, Montréal, Canada, July 1992.)

[8] Kececioglu, John and David Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. To appear in *Algorithmica*, 1993.

[9] Sankoff, David, Guillame Leduc, Natalie Antoine, Bruno Paquin, B. Franz Lang, and Robert Cedergren. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *Proceedings of the National Academy of Science USA* 89, 6575–6579, 1992.

[10] Schöniger, Michael and Michael S. Waterman. A local algorithm for DNA sequence alignment with inversions. *Bulletin of Mathematical Biology* 54, 521–536, 1992.

[11] Watterson, G.A., W.J. Ewens, T.E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology* 99, 1–7, 1982.