A Very Elementary Presentation of the Hannenhalli-Pevzner Theory

Anne Bergeron

LACIM, Université du Québec à Montréal,
C.P. 8888 Succ. Centre-Ville, Montréal, Québec, Canada, H3C 3P8.
bergeron.anne@uqam.ca

Abstract. In 1995, Hannenhalli and Pevzner gave a first polynomial solution to the problem of finding the minimum number of reversals needed to sort a signed permutation. Their solution, as well as subsequent ones, relies on many intermediary constructions, such as simulations with permutations on $2n$ elements, and manipulation of various graphs.
Here we give the first completely elementary treatment of this problem. We characterize *safe reversals* and *hurdles* working directly on the original signed permutation. Moreover, our presentation leads to polynomial algorithms that can be efficiently implemented using bit-wise operations.

1 Introduction

In the last ten years, beginning with [6], many papers have been devoted to the subject of computing the *reversal distance* between two permutations. A *reversal* $\rho(i, j)$ transforms a permutation

$$\pi = (\pi_1 \ldots \underline{\pi_i \ \pi_{i+1} \ \ldots \ \pi_j} \ldots \pi_n)$$

$$\text{to } \ \pi' = (\pi_1 \ldots \pi_j \ \ldots \ \pi_{i+1} \ \pi_i \ldots \pi_n).$$

and the reversal distance between two permutations is the minimum number of reversals that transform one into the other.

From a problem of unknown complexity, it graduated to an NP-Hard problem [2], but an interesting variant was proven to be polynomial [3]. In the *signed* version of the problem, each element of the permutation has a plus or minus sign, and a reversal $\rho(i, j)$ transforms π to:

$$\pi' = (\pi_1 \ldots -\pi_j \ldots -\pi_{i+1} -\pi_i \ldots \pi_n).$$

Permutations, and their reversals, are useful tools in the comparative study of genomes. The genome of a species can be thought of as a set of ordered sequences of genes – the ordering devices being the chromosomes –, each gene having an orientation given by its location on the DNA double strand. Different species often share similar genes that were inherited from common ancestors. However, these genes have been shuffled by mutations that modified the content

of chromosomes, the order of genes within a particular chromosome, and/or the orientation of a gene. Comparing two sets of similar genes appearing along a chromosome in two different species yields two (signed) permutations. It is widely accepted that the reversal distance between these two permutations faithfully reflects the evolutionary distance between the two species.

Computing the reversal distance of signed permutations is a delicate task since some reversals unexpectedly affect deep structures in permutations. In 1995, Hannenhalli and Pevzner proposed the first polynomial algorithm to solve it [3], developing along the way a theory of how and why some permutations were particularly resistant to sorting by reversals. It is of no surprise that the label *fortress* was assigned to specially acute cases.

Hannenhalli and Pevzner relied on several intermediate constructions that have been simplified since [4], [1], but grasping all the details remains a challenge. All the criteria given for choosing a *safe* reversal involve the construction of an associate permutation on $2n$ points, and the analysis of cycles and/or connected component of graphs associated with this permutation.

In this paper, we present a very elementary treatment of the sorting of the *oriented components* of a permutation, together with an elementary definition of the concept of *hurdle* that further simplifies the definition given in [4]. Our first algorithm is so simple that, for example, sorting a permutation of length 20, *by hand*, should be easy and straightforward.

The next section presents the basic algorithms. Section 3 contains the necessary links to the Hannenhalli-Pevzner theory, and the proofs of the claims of the Section 2. Finally, in the last section, we discuss complexity issues, and we give a *bit-vector* implementation of the sorting algorithm that runs in $\mathcal{O}(n^2)$.

2 Basic Sorting

The problem of sorting by reversal a signed permutation π is to find $d(\pi)$, its reversal distance from the identity permutation $(+1\ +2 \ldots +n)$. As usual, we will *frame* a permutation $\pi = (\pi_1\ \pi_2 \ldots \pi_n)$ with 0 and $n+1$, yielding the permutation: $(0\ \pi_1\ \pi_2 \ldots \pi_n\ n+1)$.

Given a signed permutation $\pi = (0\ \pi_1\ \pi_2 \ldots \pi_n\ n+1)$, an *oriented pair* (π_i, π_j) is a pair of adjacent integers, that is $|\pi_i| - |\pi_j| = \pm 1$, with opposite signs. For example, the oriented pairs of the permutation:

$$(\ 0\ +3\ +1\ +6\ +5\ -2\ +4\ +7\ )$$

are $(+1, -2)$ and $(+3, -2)$.

Oriented pairs are useful in the sense that they indicate reversals that create consecutive elements. For example, the pair $(+1, -2)$ induces the reversal:

$$(\ 0\ +3\ +1\ \underline{+6\ +5\ -2}\ +4\ +7\ )$$

$$(\ 0\ +3\ +1\ +2\ -5\ -6\ +4\ +7\ )$$

creating the consecutive sequence $+1\ +2$.

In general, the reversal induced by an oriented pair (π_i, π_j) will be,

$$\begin{array}{l} \rho(i, j-1), \text{ if } \pi_i + \pi_j = +1, \text{ and} \\ \rho(i+1, j), \text{ if } \pi_i + \pi_j = -1. \end{array}$$

Note that reversals that create consecutive pairs of integer are always induced by oriented pairs. Such a reversal is called an *oriented* reversal. We define the *score* of an (oriented) reversal as the number of oriented pairs in the resulting permutation. For example, the score of the reversal:

$$(\ 0\ \underline{+3\ +1\ +6\ +5}\ -2\ +4\ +7\ )$$

$$(\ 0\ -5\ -6\ -1\ -3\ -2\ +4\ +7\ )$$

is 4, since the resulting permutation has 4 oriented pairs. Computing the score of a reversal is tedious but elementary, and we will discuss efficient algorithms to do so in Section 4. The fact that oriented reversals have a beneficial effect on the ordering of a permutation suggests a first sorting strategy:

Algorithm 1 As long as π has an oriented pair, choose the oriented reversal that has maximal score.

For example, the two oriented pairs of the permutation:

$$(\ 0\ \underline{+3\ +1\ +6\ +5}\ -2\ +4\ +7\ )$$

are $(+1, -2)$, $(+3, -2)$, and their score are respectively 2 and 4. So we choose the reversal induced by $(+3, -2)$, yielding the new permutation:

$$(\ 0\ -5\ -6\ -1\ \underline{-3\ -2}\ +4\ +7\ ).$$

This permutation has now four oriented pairs $(0, -1)$, $(-3, +4)$, $(-5, +4)$ and $(-6, +7)$, all of which have score 2, except $(-3, +4)$. Acting on this pair yields:

$$(\ 0\ -5\ -6\ \underline{-1}\ +2\ +3\ +4\ +7\ ).$$

which has four oriented pairs. Note here that the score of the pair $(0, -1)$ is 0. The corresponding oriented reversal would produce a permutation with no oriented pair, and the algorithm would stop, in this case with an unsorted permutation. Fortunately, the pair $(-1, +2)$ has a positive – and maximal – score, and we get, in a similar way, the last two necessary reversals to sort the permutation:

$$(\ 0\ -5\ \underline{-6\ +1\ +2\ +3\ +4}\ +7\ )$$

$$(\ 0\ \underline{-5\ -4\ -3\ -2\ -1}\ +6\ +7\ )$$

$$(\ 0\ +1\ +2\ +3\ +4\ +5\ +6\ +7\ )$$

Interestingly enough, this elementary strategy is sufficient to optimally sort most random permutations and almost all permutations that arise from biological data. The strategy is also optimal, and we will prove in the next section the following claim.

Claim 1: If the strategy of Algorithm 1 applies k reversals to a permutation π, yielding a permutation π', then $d(\pi) = d(\pi') + k$.

The output of Algorithm 1 will be a permutation of positive elements. Most reversal applied to such permutations will create oriented pairs, but the choice of an optimal reversal is delicate. We discuss this problem in the next paragraph.

2.1 Sorting positive permutations

Let π be a signed permutation with only positive elements, and assume that π is *reduced*, that is π does not contain consecutive elements. Suppose also that π is framed by 0 and $n+1$ and consider, as in [4], the circular order induced by setting 0 to be the successor of $n+1$.

Define a *framed interval* in π as an interval of the form:

$$i \;\; \pi_{j+1} \; \pi_{j+2} \; \ldots \pi_{j+k-1} \;\; i+k$$

such that all integers between i and $i+k$ belong to the interval $[i \ldots i+k]$. For example, consider the permutation:

$$(\; 0\; 2\; 5\; 4\; 3\; 6\; 1\; 7\;).$$

The whole permutation is a framed interval by construction. But we have also the interval: 2 5 4 3 6, which can be reordered as 2 3 4 5 6, and, by circularity, the interval 6 1 7 0 2, which can be reordered as 6 7 0 1 2, since 0 is the successor of 7.

Definition 1 If π is reduced, a *hurdle* in π is a framed interval that properly contains no framed interval.

Claim 2 Hurdles as defined in Definition 1 are the same hurdles that are defined in [3] and [4].

When a permutation has only one or two hurdles, one reversal is sufficient to create enough oriented pairs in order to completely sort the permutation with Algorithm 1. Two operations are introduced in [3], the first one is *hurdle cutting* which consist in reversing one internal element, say π_{j+1}, of a hurdle:

$$i \; \underline{\pi_{j+1}} \; \pi_{j+2} \; \ldots \pi_{j+k-1} \; i+k.$$

This reversal is sufficient to sort all the interval using Algorithm 1. For example, the following permutation contains only one hurdle:

$$(\; 0\; 2\; 4\; 3\; 1\; 5\;).$$

The reversal of element 2 cuts the hurdle, and the resulting permutation

$$(0 \; -2 \; 4 \; 3 \; 1 \; 5)$$

can be sorted with 4 reversals by Algorithm 1.

The second operation is *hurdle merging*, which acts on the end points of two hurdles:

$$i \ldots \underline{i+k \ldots i'} \ldots i'+k'$$

and does the reversal $\rho(i+k, i')$. If a permutation has only two hurdles, merging them will produce a permutation that can be completely sorted by Algorithm 1.

Thus, for example, merging the two hurdles in the permutation

$$(0 \; 2 \; 5 \; 4 \; 3 \; 6 \; 1 \; 7).$$

yields the permutation:

$$(0 \; 2 \; 5 \; 4 \; 3 \; -6 \; 1 \; 7).$$

which can be sorted in 5 reversal using Algorithm 1.

Merging and cutting hurdles in a permutation that contains more than 2 hurdles must be managed carefully. Indeed, cutting some hurdles can create new ones!

Definition 2 A *simple* hurdle is a hurdle whose cutting decreases the number of hurdles. Hurdles that are not simple are called *super hurdles*.

For example, the permutation (0 2 5 4 3 6 1 7) has two hurdles. Cutting the hurdle 2 5 4 3 6 yields the permutation,

$$(0 \; 2 \; 3 \; 4 \; 5 \; 6 \; 1 \; 7)$$

which, by collapsing the sequence 2 3 4 5 6 is reduced to:

$$(0 \; 2 \; 1 \; 3),$$

which has only one hurdle. However, the permutation (0 2 4 3 5 1 6 8 7 9) contains two hurdles, and if one cuts the hurdle 2 4 3 5, the resulting reduced permutation will be

$$(0 \; 2 \; 1 \; 3 \; 5 \; 4 \; 6)$$

which still has two hurdles.

The following algorithm is adapted from [4], and is discussed originally in [3].

Algorithm 2 If a permutation has $2k$ hurdles, $k \geq 2$, merge any two non-consecutive hurdles. If a permutation has $2k+1$, $k \geq 1$, then if it has one simple hurdle, cut it; If it has none, merge two non-consecutive hurdles, or consecutive ones if $k = 1$.

Together with Algorithm 1, Algorithm 2 can be used to optimally sort any signed permutation. This completes the first part of the paper, and, in the next section, we turn to the task of proving our various claims.

3 Selected Results from the Hannenhalli-Pevzner Theory

The exposition of the complete results of the Hannenhalli-Pevzner theory is beyond the scope of this paper, and the reader is referred to the original paper [3], or the book on computational molecular biology by Pevzner [5]. Instead, we will show the soundness of our algorithms by directly using the *overlap graph* introduced in [4].

The first step in the construction of the overlap graph is to simulate a signed permutation on n elements with an unsigned permutation on $2n$ elements. Each positive element x in the permutation is replaced by the sequence $2x-1$ $2x$, and each negative element $-x$ by the sequence $2x$ $2x-1$. For example, the permutation:

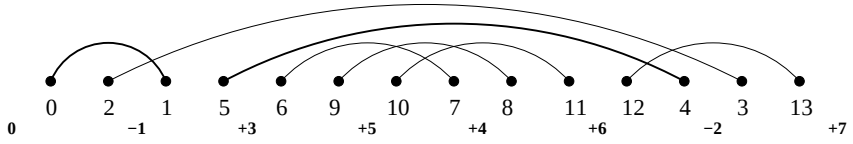
$$\pi = (\ 0\ -1\ +3\ +5\ +4\ +6\ -2\ +7\ )$$

becomes:

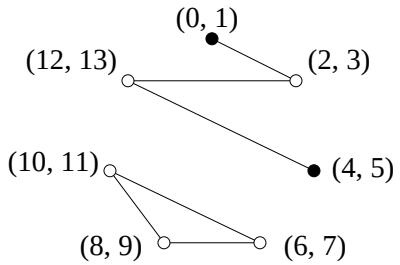
$$\pi' = (\ 0\ \ 2\ \ 1\ \ 5\ \ 6\ \ 9\ \ 10\ \ 7\ \ 8\ \ 11\ \ 12\ \ 4\ \ 3\ \ 13\ )$$

Reversals $\rho(i,j)$ of π are simulated by unsigned reversals $\rho(2i-1,2j)$ in π'.

The *overlap graph* associated with a permutation π has n vertices labeled by $(0,1),(2,3),\ldots,(2n,2n+1)$, with an edge between two vertices (a,b) and (c,d) iff, in the unsigned permutation, the interval corresponding to the positions of a and b overlaps – without proper containment – the interval corresponding to the positions of b and d. For example, if one draws arcs joining the end points of the pairs $(0,1),(2,3),\ldots,(2n,2n+1)$ in the permutation π':



The overlap graph can then be easily drawn by tracing an edge for each intersecting arcs in the above diagram, yielding:



There is a natural bijection between the vertices of the overlap graph and pairs of adjacent integers (π_i,π_j). in the original permutation. Indeed, a pair of

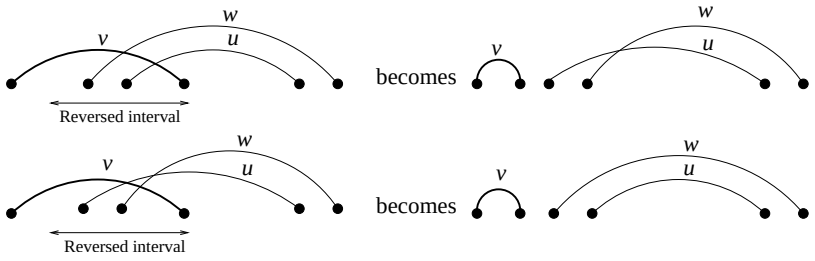
adjacent integers will generate four consecutive integers in the unsigned permutation: $2x-1$, $2x$, $2x+1$, and $2x+2$. The vertex $(2x, 2x+1)$ is associated with the pair (π_i, π_j). For example, the oriented pair $(3, -2)$ in π corresponds to the vertex $(4, 5)$ in the overlap graph. Vertices corresponding to oriented pairs are naturally called *oriented vertices*, and are denoted by solid dots in the overlap graph. Moreover, we will refer to the *reversal induced by a vertex* meaning the reversal induced by the oriented pair corresponding to the vertex. The following facts, mostly from [4], pinpoint the important relations between a signed permutation and its overlap graph.

Fact 1: *A vertex has an odd degree iff it is oriented.*

Proof. Let $2x-1$, $2x$, $2x+1$, and $2x+2$, be the four integers associated with the oriented pair (π_i, π_j). Since π_i and π_j have different signs, the positions of $2x$ and $2x+1$ will not have the same parity in the unsigned permutations. Thus, the interval between $2x$ and $2x+1$ has an odd length, implying that it overlaps an odd number of other intervals. On the other hand, any interval that overlaps an odd number of intervals must have an odd length. Therefore the positions of its end points must have different parities, implying that the corresponding pair of adjacent integers is oriented. ■

Fact 2: *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.*

Proof. The reversal corresponding to an oriented vertex v has the effect of collapsing the associated interval, thus v will become isolated. Let u and w be two intervals overlapping v, meaning that exactly one of their end points lies in the interval spanned by v. The reversal induced by v will reverse these two points. Here, a picture is worth a thousand words:



■

Fact 3: *If one performs the reversal corresponding to an oriented vertex v, each vertex adjacent to v will change its orientation.*

Proof. Since v is oriented, it has an odd number $2k+1$ of adjacent vertices. Let w be a vertex adjacent to v, with j neighbors also adjacent to v. With the

reversal, w will loose $j+1$ neighbors, and gain $2k-j$ new ones. Thus the degree of w will change by $2k-2j-1$, changing its orientation. ■

Fact 4: *The score of the oriented reversal corresponding to an oriented vertex v is given by:*

$$T+U-O-1$$

where T is the total number of oriented vertices in the graph, U is the number of unoriented vertices adjacent to v, and O is the number of oriented vertices adjacent to v.

Proof. This follows trivially from the preceding facts. ■

We now state a basic result that is proven, in different ways, both in [3] and [4]. Define an *oriented component* of the overlap graph as a connected component that contains at least one oriented vertex. A *safe* reversal is a reversal that does not create new unoriented components, except for isolated vertices.

Proposition 1 (Hannenhalli and Pevzner). *Any sequence of oriented safe reversals is optimal.*

The difficulties in sorting oriented components lie in the detection of safe reversals. Hannenhalli and Pevzner deal with the problem by computing several statistics on cycles and breakpoints of various graphs. Kaplan et al. solve it by searching for particular cliques in the overlap graph. The next theorem argues that the elementary strategy of choosing the reversal with maximal score is optimal, thus proving Claim 1.

Theorem 1. *An oriented reversal of maximal score is safe.*

Proof. Suppose that vertex v has maximal score, and that the reversal induced by v creates a new unoriented component C containing more than one vertex. At least one of the vertices in C must have been adjacent to v, since the only edges affected by the reversal are those between vertices adjacent to v. So, let w be a vertex formerly adjacent to v and contained in C, and consider the scores of v and w:

$$score(v) = T+U-O-1$$
$$score(w) = T+U'-O'-1$$

All unoriented vertices adjacent to v must be adjacent to w. Indeed, an unoriented vertex adjacent to v and not to w will become oriented, and connected to w, contrary to the assumption that C is unoriented. Thus, $U' \geq U$.

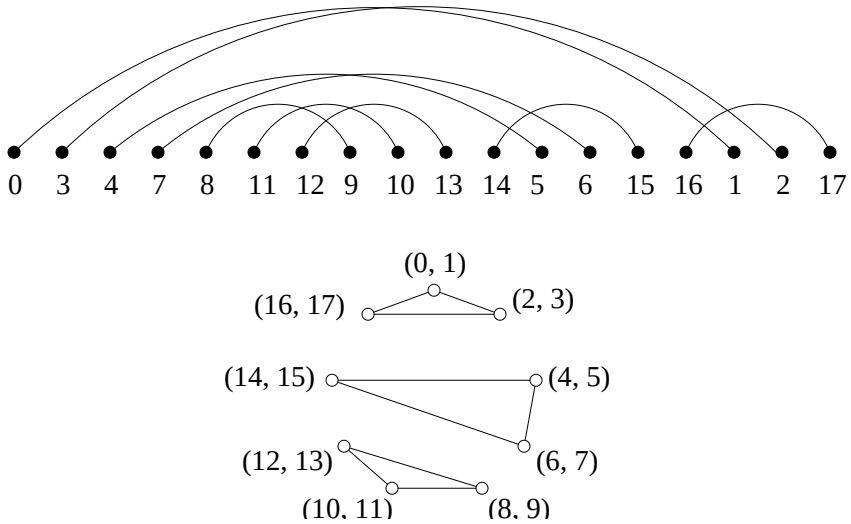
All oriented vertices adjacent to w must be adjacent to v. If this was not the case, an oriented vertex adjacent to w but not to v would remain oriented, again contradicting the fact that C is unoriented. Thus, $O' \leq O$.

Now, if both $O' = O$ and $U' = U$, vertices v and w have the same set of adjacent vertices, and complementing the subgraph of v and its adjacent vertices will isolate both v and w. Therefore, we must have that $score(w) > score(v)$, which is a contradiction. ■

3.1 Hurdles

In this section, we assume that π is a positive and reduced permutation. These assumptions are equivalent to say that the overlap graph has no oriented components – all of which can be cleared by Algorithm 1 –, and no isolated vertices.

Consider again the circular order, this time on the interval $[0..2n-1]$, induced by setting 0 to be the successor of $2n-1$. The *span* of a set of vertices X in the overlap graph is the minimum interval that contains, in the circular order, all the intervals of vertices in X. For example, the three connected components of the following overlap graph have spans $[4, 15] = [4\ 7\ 8\ 11\ 12\ 9\ 10\ 13\ 14\ 5\ 6\ 15]$, $[8, 13] = [8\ 11\ 12\ 9\ 10\ 13]$, and $[16, 3] = [16\ 1\ 2\ 17\ 0\ 3]$.



Hurdles are defined in [3] as unoriented components which are minimal with respect to span inclusion. Moreover, in [4], it is shown that the span of a connected component is always of the form $[2i, 2j-1]$. The following Lemmas and Theorem detail the relationships between connected components and framed intervals, substantiating the second claim of Section 1.

Lemma 1. *Framed intervals of the form $[i, j]$ in a permutation on n elements are in one-to-one correspondence with framed intervals of the form $[2i, 2j-1]$ in the corresponding unsigned permutation on $2n$ elements.*

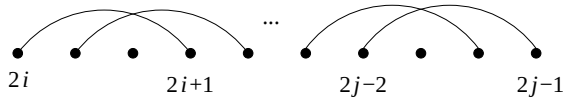
Proof. The end points i and j of a framed interval $[i, j]$ will be mapped, respectively, to the pairs $2i-1$, $2i$, and $2j-1$, $2j$. All integers between i and j appear in the interval $[i, j]$, if and only if all the integers between $2i$ and $2j-1$ appear in the interval $[2i, 2j-1]$.

Lemma 2. *Any framed interval $[2i, 2j-1]$ is the span of a union of connected components.*

Proof. If $[2i, 2j-1]$ is a framed interval, it contains exactly the integers between $2i$ and $2j-1$, thus the only arcs in this interval are: $(2i, 2i+1)$, $(2i+2, 2i+3), \ldots, (2j-2, 2j-1)$, and no other arc intersects this set. Therefore, the corresponding set of vertices is not connected to any other vertex. ■

Lemma 3. *The span $[2i, 2j-1]$ of a connected component is always a framed interval.*

Proof. If vertex $(2i, 2i+1)$ is connected to $(2j-2, 2j-1)$, there must be a sequence of intersecting arcs linking $2i$ to $2j-1$.



Any arc with only one end point between $2i$ and $2j-1$ would therefore intersect one of the arcs in the sequence, so there are none. Thus, if integer $2k$ is in the interval, then $2k+1$ is also in the interval, and if $i \leq k < j$, then $2k+2$ is also in the interval. ■

Theorem 2. *If π is reduced, an unoriented component is minimal iff its span is a framed interval that contains no other.*

Proof. By Lemma 3, the span of a connected component is always a framed interval. If the component is minimal with respect to span inclusion, by Lemma 2, its span cannot contain properly another framed interval.

On the other hand, a framed interval $[2i, 2j-1]$ that contains no other yields a single connected component C whose vertices endpoints are exactly the integers between $2i$ and $2j-1$. Thus the vertices of C are consecutive on the circle, and component C is minimal. ■

The main consequence of Theorem 2 is to give an elementary characterization of the concept of hurdles, that does not need the construction of the overlap graph.

4 Settling Scores

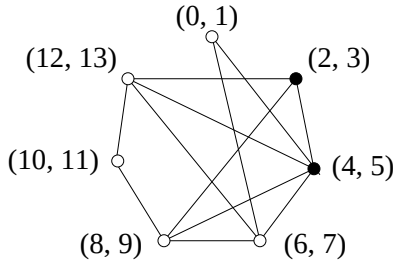
We now turn to the problem of computing the score of a reversal. In the following, we assume that the overlap graph of a permutation is explicitely represented as a bit matrix, and consider, for a vertex v, the bit vector $\boldsymbol{v}$ whose w^{th} coordinate is 1 iff vertex v is adjacent to vertex w.

We also assume that the score and parity of each vertex are stored in the vectors $\boldsymbol{s}$ and $\boldsymbol{p}$, respectively. It is only necessary to keep track of the *variable* part of the score that is, for both oriented and unoriented vertices, the expression $U_v - O_v$, where U_v is the number of unoriented vertices adjacent to v, and O_v is the number of oriented vertices adjacent to v.

As an example, consider the permutation

$$(0 \ +3 \ +1 \ +6 \ +5 \ -2 \ +4 \ +7)$$

its overlap graph. and its associated data structure.



	$(0,1)$	$(2,3)$	$(4,5)$	$(6,7)$	$(8,9)$	$(10,11)$	$(12,13)$
$(0,1)$	0	0	1	1	0	0	0
$(2,3)$	0	0	1	0	1	0	1
$(4,5)$	1	1	0	1	1	0	1
$(6,7)$	1	0	1	0	1	0	1
$(8,9)$	0	1	1	1	0	1	0
$(10,11)$	0	0	0	0	1	0	1
$(12,13)$	0	1	1	1	0	1	0
$\boldsymbol{p}$	0	1	1	0	0	0	0
$\boldsymbol{s}$	0	1	3	2	0	2	0

Clearly, initializing such a structure requires $\mathcal{O}(n^2)$ steps. Given it, finding the reversal with maximal score is trivial. The interesting part is the effect of a reversal on the structure.

Suppose that we choose to perform the reversal corresponding to the oriented vertex v. Since v will become unoriented and isolated, each vertex incident to v will automatically gain a point of score. Thus we first set:

$$\boldsymbol{s} \leftarrow \boldsymbol{s} + \boldsymbol{v}$$

Next, if w is a vertex incident to v, the vector $\boldsymbol{w}$ is changed according to:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} \oplus \boldsymbol{v}$$

where $\oplus$ is the *exclusive or* bit wise operation. Indeed, vertices adjacent to w after the reversal are either existing vertices that were not adjacent to v, or vertices that were adjacent to v but not to w. The exceptions to this rule are v and w themselves, and this problem can easily be solved by setting the diagonal bit $\boldsymbol{v}_v$ to 1, and $\boldsymbol{w}_w$ to 1 before computing the direct sum.

Now, if w is unoriented, each of its former adjacent vertices will *loose* one point of score, since w will become oriented, and each of its new adjacent vertices will *loose* one point of score. Note that a vertex that *stays* connected to w will loose a total of two points. We can thus write the effect of the change of orientation of w on the vector $\boldsymbol{s}$ of scores as:

$$\begin{aligned} \boldsymbol{s} &\leftarrow \boldsymbol{s} - \boldsymbol{w} \\ \boldsymbol{w}_w &\leftarrow 1 \\ \boldsymbol{w} &\leftarrow \boldsymbol{w} \oplus \boldsymbol{v} \\ \boldsymbol{s} &\leftarrow \boldsymbol{s} - \boldsymbol{w} \end{aligned}$$

where the subtractions are performed component wise on the vector of scores. If w is oriented, the losses are converted to gains, so the subtractions are converted to additions.

Finally, the parity vector $\boldsymbol{p}$ is updated by the equation:

$$\boldsymbol{p} \leftarrow \boldsymbol{p} \oplus \boldsymbol{v} \,.$$

The algorithm requires eventually $\mathcal{O}(n)$ *vector* operations for each reversal, and these operations can be implemented very efficiently using bit operations widely available in processors.

Kaplan et al., in [4], give an algorithm based on [1] that clears the hurdles from a permutation in less than $\mathcal{O}(n^2)$, and that can be used in conjunction with the above algorithm. But since we already have an extensive representation on the overlap graph, we can use it to keep track of the connected components, and to detect hurdles.

References

1. Piotr Berman, Sridhar Hannenhalli, *Fast Sorting by Reversal.* CPM 1996, LNCS 1075: 168-185.
2. Alberto Caprara, *Sorting by reversals is difficult.* RECOMB 1997, ACM Press: 75-83.
3. Sridhar Hannenhalli, Pavel A. Pevzner, *Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals.* JACM 46(1): 1-27 (1999).
4. Haim Kaplan, Ron Shamir, Robert Tarjan, *A Faster and Simpler Algorithm for Sorting Signed Permutations by Reversals.* SIAM J. Comput. 29(3): 880-892 (1999).
5. Pavel Pevzner, Computational Molecular Biology, MIT Press, Cambridge, Mass., 314 p., (2000).
6. David Sankoff, *Edit Distances for Genome Comparisons Based on Non-Local Operations.* CPM 1992, LNCS 644,: 121-135.