Hurdles and Sorting by Inversions: Combinatorial, Statistical, and Experimental Results

Krister M. Swenson, Yu Lin, Vaibhav Rajan, and Bernard M.E. Moret

Laboratory for Computational Biology and Bioinformatics, EPFL (Ecole Polytechnique Fédérale de Lausanne), and Swiss Institute of Bioinformatics, Lausanne, Switzerland
{krister.swenson,yu.lin,vaibhav.rajan,bernard.moret}@epfl.ch

Abstract. As data about genomic architecture accumulates, genomic rearrangements have attracted increasing attention. One of the main rearrangement mechanisms, inversions (also called reversals), was characterized by Hannenhalli and Pevzner and this characterization in turn extended by various authors. The characterization relies on the concepts of breakpoints, cycles, and obstructions colorfully named hurdles and fortresses. In this paper, we study the probability of generating a hurdle in the process of sorting a permutation if one does not take special precautions to avoid them (as in a randomized algorithm, for instance). To do this we revisit and extend the work of Caprara and of Bergeron by providing simple and exact characterizations of the probability of encountering a hurdle in a random permutation. Using similar methods we provide the first asymptotically tight analysis of the probability that a fortress exists in a random permutation. Finally, we study other aspects of hurdles, both analytically and through experiments: when are they created in a sequence of sorting inversions, how much later are they detected, and how much work may need to be undone to return to a sorting sequence.

1 Introduction

The advent of high-throughput techniques in genomics has led to the rapid accumulation of data about the genomic architecture of large numbers of species. As biologists study these genomes, they are finding that genomic rearrangements, which move single genes or blocks of contiguous genes around the genome, are relatively common features: entire blocks of one chromosome can be found in another chromosome in another species. The earliest findings of this type go back to the pioneering work of Sturtevant on the fruit fly [11, 12]; but it was the advent of large-scale sequencing that moved this aspect of evolution to the forefront of genomics.

The best documented type of rearrangement is the *inversion* (also called reversal), in which a block of consecutive genes is removed and put back in (the same) place in the opposite orientation (on the other strand, as it were). The most fundamental computational question then becomes: given two genomes, how efficiently can such an operation as inversion transform one genome into the other? Since an inversion does not affect gene content (the block is neither shortened nor lengthened by the operation), it makes sense to view these operations as being applied to a signed permutation of the set $\{1, 2, \ldots, n\}$.

Hannenhalli and Pevzner [6, 7] showed how to represent a signed (linear) permutation of n elements as a *breakpoint graph* (also called, more poetically, a diagram of reality and desire [10]), which is a graph on $2n+2$ vertices (2 vertices per element of the permutation to distinguish signs, plus 2 vertices that denote the extremities of the permutation) with colored edges, where edges of one color represents the adjacencies in one permutation and edges of the other color those in the other permutation. In such a graph, every vertex has indegree 2 and outdegree 2 and so the graph has a unique decomposition into cycles of even length, where the edges of each cycle alternate in color. Hannenhalli and Pevzner introduced the notions of *hurdles* and *fortresses* and proved that the minimum number of inversions needed to convert one permutation into the other (also called "sorting" a permutation) is given by the number of elements of the permutation plus 1, minus the number of cycles, plus the number of hurdles, and plus 1 if a fortress is present. Caprara [5] showed that hurdles were a rare feature in a random signed permutation. Bergeron [2] provided an alternate characterization in terms of *framed common intervals* and went on to show that *unsafe inversions*, that is, inversions that could create new obstructions such as hurdles, were rare [3] when restricted to adjacency creating inversions. Kaplan and Verbin [8] capitalized on these two findings and proposed a randomized algorithm that sorts a signed permutation without paying heed to unsafe inversions, finding that, in practice, the algorithm hardly needed any restarts to provide a proper sorting sequence of inversions, although they could not prove that it is in fact a proper Las Vegas algorithm.

In this paper, we extend Bergeron's result about the possibility of creating a hurdle by doing an inversion. Her result is limited to inversions that create new adjacencies, but these are in the minority: in a permutation without hurdles, any inversion that increases the number of cycles in the breakpoint graph is a candidate. Using Sankoff's *randomness hypothesis* [9], we show that the probability that *any* cycle-splitting inversion is unsafe is $\Theta(n^{-2})$. We then revisit Caprara's complex proof and provide a simple proof, based on the framed intervals introduced by Bergeron, that the probability that a random signed permutation on n elements contains a hurdle is $\Theta(n^{-2})$. Finally, we show that this approach can be extended to prove that the probability such a permutation contains a fortress is $\Theta(n^{-15})$. Our results are elaborated for circular permutations, but simple (and by now standard) adaptations show that they also hold for linear permutations.

Framed common intervals considerably simplify our proofs; indeed, our proofs for hurdles and fortresses depend mostly on the relative scarcity of framed intervals. Our results add credence to the conjecture that an algorithm choosing random sorting inversions is a Las Vegas algorithm, i.e., that it returns a sorting sequence with high probability after a constant number of restarts. Because this result suggests that the probability of failure of such an algorithm is $O(1)$ when working on a permutation of n elements, one could consider designing an algorithm that runs faster by using a stochastic, rather than a deterministic, data structure, yet remains a Las Vegas algorithm. Indeed, how fast a signed permutation can be sorted by inversions remains an open question: while we have an optimal linear-time algorithm to compute the number of inversions needed [1], computing one optimal sorting sequence takes subquadratic time—$O(n\sqrt{n \log n})$, either stochastically with the algorithm of Kaplan and Verbin or deterministically with the similar approach of Tannier and Sagot [14].

2 Preliminaries

Let Σ_n denote the set of signed permutations over n elements; a permutation π in this set will be written as $\pi = (\pi_1 \pi_2 \ldots \pi_n)$, where each element π_i is a signed integer and the absolute values of these elements are all distinct and form the set $\{1,2,\ldots,n\}$. Given such a π, a pair of elements (π_i, π_{i+1}) or (π_n, π_1) is called an *adjacency* whenever we have $\pi_{i+1} - \pi_i = 1$ (for $1 \leq i \leq n-1$) or $\pi_1 - \pi_n = 1$; otherwise, this pair is called a *breakpoint*. We shall use Σ_n^0 to denote the set of permutations in which every permutation is entirely devoid of adjacencies. Bergeron *et al* [3] proved the following result about $|\Sigma_n^0|$.

Lemma 1. *[3] For all $n > 1$, $\frac{1}{2}|\Sigma_n| < |\Sigma_n^0| < |\Sigma_n|$.*

For any signed permutation π and the identity $I = (12 \ldots n)$, we can construct the breakpoint graph for the pair (π, I). Since there is one-to-one mapping between π and the corresponding breakpoint graph for (π, I), we identify the second with the first and so write that π contains cycles, hurdles, or fortresses if the breakpoint graph for (π, I) does; similarly, we will speak of other properties of a permutation π that are in fact defined only when π is compared to the identity permutation.

A *framed common interval* (FCI) of a permutation (made circular by considering the first and last elements as being adjacent) is a substring of the permutation, $as_1s_2 \ldots s_kb$ or $-bs_1s_2 \ldots s_k-a$ such that

- for each i, $1 \leq i \leq k$, $|a| < |s_i| < |b|$, and
- for each l, $|a| < l < |b|$, there exists a j with $|s_j| = l$, and
- it is not a concatenation of substrings satisfying the previous two properties.

So the substring $s_1s_2 \ldots s_k$ is a signed permutation of the integers that are greater than a and less than b; a and b form the *frame*. The framed interval is said to be common, in that it also exists, in its canonical form, $^+a^+(a+1)^+(a+2)\ldots{}^+b$, in the identity permutation. Framed intervals can be nested. The *span* of an FCI is the number of elements between a and b, plus two, or $b-a+1$. A *component* is comprised of all elements inside a framed interval that are not inside any nested subinterval, plus the frame elements. A *bad component* is a component whose elements all have the same sign.

In a circular permutation, a bad component A *separates* bad components B and C if and only if every substring containing an element of B and an element of C also has an element of A in it. We say that A *protects* B if A separates B from all other bad components. A *superhurdle* is a bad component that protects another bad component. A *fortress* is a permutation that has an odd number (larger than 1) of hurdles, all of which are superhurdles. The smallest superhurdles are equivalent to intervals $f = {}^+(i)^+(i+2)^+(i+4)^+(i+3)^+(i+5)^+(i+1)^+(i+6)$ or the reverse $f' = {}^-(i+6)^-(i+1)^-(i+5)^-(i+3)^-(i+4)^-(i+2)^-(i)$. A *hurdle* is a bad component that is not protected by a superhurdle.

We will use the following useful facts about FCIs; all but fact 3 follow immediately from the definitions.

1. A bad component indicates the existence of a hurdle.
2. To every hurdle can be assigned a unique bad component.
3. FCIs never overlap by more than two elements. Two FCIs can only overlap at their endpoints [4] and at most both the endpoints of an FCI can overlap with other FCIs.
4. An interval shorter than 4 elements cannot be bad.

3 The Rarity of Hurdles and Fortresses

In this section, we provide asymptotic characterizations in $\Theta(\,)$ terms of the probability that a hurdle or fortress is found in a signed permutation selected uniformly at random. Each proof has two parts, an upper bound and a lower bound; for readability, we phrase each part as a lemma and develop it independently. We begin with hurdles; the characterization for these structures was already known, but the original proof of Caprara [5] is very complex.

Theorem 1. *The probability that a random signed permutation on n elements contains a hurdle is* $\Theta(n^{-2})$.

Lemma 2 (Upper bound for shorter than $n-1$). *The probability that a random signed permutation on n elements contains a hurdle spanning no more than $n-2$ elements is* $O(n^{-2})$.

Proof. Fact 4 tells us that we need only consider intervals of at least four elements. Call $F_{\leq n-2}$ the indicator random variable corresponding to the event that an FCI spanning no more than $n-2$ and no less than four elements exists. Call $F(i)_{\leq n-2}$ the indicator random variable corresponding to event that such an FCI exists with a left endpoint at π_i. We thus have $F_{\leq n-2} = 1$ if and only if there exists an i, $1 \leq i \leq n$, with $F(i)_{\leq n-2} = 1$. Note that $F(i)_{\leq n-2} = 1$ implies either $\pi_i = a$ or $\pi_i = {}^{-}b$ for some FCI. Thus we can write

$$Pr\big(F(i)_{\leq n-2} = 1\big) \leq \sum_{l=4}^{n-2} \frac{1}{2(n-1)} \binom{n-2}{l-2}^{-1} \tag{1}$$

since $\frac{1}{2(n-1)}$ is the probability the right endpoint matches the left endpoint (π_l is -a or b if π_i is -b or a respectively) of an interval of span l and $\binom{n-2}{l-2}^{-1}$ is the probability that the appropriate elements are inside the frame. We can bound the probability from (1) as

$$\begin{aligned} Pr\big(F(i)_{\leq n-2} = 1\big) &\leq \frac{1}{2(n-1)} \sum_{l=2}^{n-4} \binom{n-2}{l}^{-1} \\ &\leq \frac{1}{n-1} \sum_{l=2}^{\lceil n/2 \rceil - 1} \binom{n-2}{l}^{-1} \\ &\leq \frac{1}{n-1} \left(\sum_{l=2}^{\sqrt{n}} \left(\frac{l}{n-2} \right)^{l} + \sum_{l=\sqrt{n}+1}^{\lceil n/2 \rceil - 1} \binom{n-2}{l}^{-1} \right) \end{aligned} \tag{2}$$

where the second term is no greater than

$$\sum_{l=\sqrt{n}+1}^{\lceil n/2 \rceil - 1} \binom{n-2}{l}^{-1} \leq \sum_{l=\sqrt{n}+1}^{\lceil n/2 \rceil - 1} \left(\frac{1}{2} \right)^{\sqrt{n}+1} \in O(1/n^2) \tag{3}$$

and the first term can be simplified

$$\begin{aligned}\sum_{l=2}^{\sqrt{n}}\left(\frac{l}{n-2}\right)^l &= \sum_{l=2}^{4}\left(\frac{l}{n-2}\right)^l + \sum_{l=5}^{\sqrt{n}}\left(\frac{l}{n-2}\right)^l \\ &\leq \sum_{l=2}^{4}\left(\frac{l}{n-2}\right)^l + \sum_{l=5}^{\sqrt{n}}\left(\frac{n}{n-2}\frac{\sqrt{n}}{n}\right)^5 \\ &\in O\Big(3\times\frac{16}{(n-2)^2}+\sqrt{n}\,n^{-5/2}\Big) = O(n^{-2}). \end{aligned} \tag{4}$$

To compute $Pr(F_{\leq n-2})$ we use the union bound on $Pr(\bigcup_{i=1}^{n} F(i)_{\leq n-2})$. This removes the factor of $\frac{1}{n-1}$ from (2) yielding just the sum of (4) and (3) which is $O(n^{-2})$. The probability of observing a hurdle in some subsequence of a permutation can be no greater than the probability of observing a FCI (by fact 2). Thus we know the probability of observing a hurdle that spans no more than $n-2$ elements is $O(n^{-2})$.

We now proceed to bound the probability of a hurdle that spans $n-1$ or n elements. Call intervals with such spans *n-intervals*. For a bad component spanning n elements with $a = i$, there is only a single $b = (i-1)$ that must be a's left neighbor (in the circular order), and for a hurdle spanning $n-1$ elements with $a = i$, there are only two configurations ("$^{+}$(i-2) $^{+}$(i-1) $^{+}$i" and its counterpart "$^{+}$(i-2) $^{-}$(i-1) $^{+}$i") that will create a framed interval. Thus the probability that we see an n-interval with a particular $a = i$ is $O(1/n)$ and the expected number of n-intervals in a permutation is $O(1)$.

We now use the fact that a bad component is comprised of elements with all the same sign. Thus the probability that an n-interval uses all the elements in its span (i.e., there exist no nested subintervals) is $O(2^{-n})$. Call a bad component that does not use all of the elements in its span (i.e., there must exist nested subintervals) a *fragmented* interval.

Lemma 3 (Upper bound for *n*-intervals). *The probability that a fragmented n-interval is a hurdle is $O(n^{-2})$.*

Proof. We divide the analysis into three cases where the fragment-causing subinterval is of span

1. $n-1$,
2. 4 through $n-2$, and
3. less than 4.

The existence of a subinterval of span $n-1$ precludes the possibility of the frame elements from the larger n-interval being in the same component, so there cannot be a hurdle using this frame. We have already established that $Pr(F_{\leq n-2})$ is $O(n^{-2})$. Thus we turn to the third case. If an interval is bad, then the frame elements of any fragmenting subinterval must have the same sign as the frame elements of the larger one. If we view each such subinterval and each element not included in such an interval as single characters, we know that there must be at least $n/3$ signed characters. Since the signs of the characters are independent, the probability that all characters have the same sign is $1/2^{O(n)}$ and is thus negligible.

Thus the probability of a bad n-interval is $O(n^{-2})$. Now using fact 4 we conclude that the probability of existence of a hurdle in a random signed permutation on n elements is $O(n^{-2})$.

Lemma 4 (Lower bound). *The probability that a signed permutation on n elements has a hurdle with a span of four elements is* $\Omega(n^{-2})$.

Proof. Call h_k the hurdle with span four that starts with element $4k+1$. So the subsequence that corresponds to h_k must be $^{+}(4k+1)^{+}(4k+3)^{+}(4k+2)^{+}(4k+4)$ or $^{-}(4k+4)^{-}(4k+2)^{-}(4k+3)^{-}(4k+1)$. We can count the number of permutations with h_0, for instance. The four elements of h_0 are contiguous in $4!(n-3)!2^n$ permutations of length n. In $c = 2/(4!2^4)$ of those cases, the contiguous elements form a hurdle, so the total proportion of permutations with h_0 is

$$c\frac{4!(n-3)!2^n}{n!2^n} \in \Omega\Big(\frac{1}{n^3}\Big).$$

Similarly, the proportion of permutations that have both h_0 and h_1 is

$$F_2 = c^2\frac{(4!)^2(n-6)!2^n}{n!2^n} \in O\Big(\frac{1}{n^6}\Big)$$

and, therefore, the proportion of permutations that have at least one of h_0 or h_1 is

$$2 \times c\frac{4!(n-3)!2^n}{n!2^n} - F_2. \tag{5}$$

We generalize (5) to count the proportion of permutations with at least one of the hurdles h_0,h_1,...,$h_{\lfloor n/4 \rfloor}$; this proportion is at least

$$\left\lfloor \frac{n}{4} \right\rfloor \times c\frac{4!(n-3)!2^n}{n!2^n} - \binom{\lfloor n/4 \rfloor}{2} F_2 \tag{6}$$

which is $\Omega(n^{-2})$ since the second term is $O(n^{-4})$.

Now we turn to the much rarer fortresses.

Theorem 2. *The probability that a random signed permutation on n elements includes a fortress is* $\Theta(n^{-15})$.

Lemma 5 (Upper bound). *The probability that a random signed permutation on n elements includes a fortress is* $O(n^{-15})$.

Proof. We bound the probability that at least three superhurdles occur in a random permutation by bounding the probability that three non-overlapping bad components of length seven exist. We divide the analysis into three cases depending on the number l of elements spanned by a bad component.

1. For one of the three FCIs we have $n-14 \leq l \leq n-11$.

2. For one of the three FCIs we have $17 \leq l \leq n-15$.
3. For all FCIs we have $7 \leq l < 17$.

As we did in Lemma 2 (equation 1), we can bound the probability that we get an FCI of length l starting at a particular position by

$$Pr(F_l = 1) \leq \frac{1}{2(n-1)} \binom{n-2}{l-2}^{-1}. \tag{7}$$

In the first case the probability that the FCI is a superhurdle is $O(n^{-11} \cdot 2^{-n})$ if the FCI is not fragmented and $O(n^{-15})$ if it is (using the same technique as for the proof of Lemma 3). In the second case the probability is at most

$$n \sum_{l=17}^{n-15} F_l = n \sum_{k=15}^{n-17} \frac{1}{2(n-1)} \binom{n-2}{k}^{-1}$$

which, by the same reasoning used for equation 2 to derive $O(n^{-2})$, is $O(n^{-15})$. Thus the first two cases both give us an upper bound of $O(n^{-15})$.

Fact 3 tells us that any pair of FCIs can overlap only on their endpoints. Thus, if we first consider the probability of finding a smallest FCI, we know that no other FCI will have an endpoint inside it. So the probability of having a second FCI, conditioned on having a smaller first one, is dependent only on the size of the first. The same reasoning extends to the probability of having a third conditioned on having two smaller FCIs. Since each of the three FCIs spans less than seventeen elements, the probability of each FCI appearing is at most $n \sum_{l=7}^{17} F_k = O(n^{-5})$, and the probability of there being at least three of them is $O(n^{-15})$.

We now turn to the lower bound. Consider the probability of the existence, among random permutations, of a permutation with exactly three superhurdles spanning seven elements each. A lower bound on this probability is a lower bound on the probability of existence of a fortress in a random permutation.

Lemma 6 (Lower bound). *The probability that a random signed permutation on n elements includes a fortress is* $\Omega(n^{-15})$.

Proof. Denote by $F_{3,7}(n)$ the number of permutations on n elements with exactly 3 superhurdles spanning 7 elements each. To create such a permutation, choose a permutation of length $n-18$ (with zero adjacencies and without hurdles), select three elements, and extend each of these three elements to a superhurdle, renaming the elements of the permutation as needed. That is, replace element $+i$ by the framed interval of length 7 $f = {}^{+}(i)^{+}(i+2)^{+}(i+4)^{+}(i+3)^{+}(i+5)^{+}(i+1)^{+}(i+6)$ and rename all the elements with magnitude j to have magnitude $j+6$ (for those with $|j| > |i|$). After extending the three selected elements, we get a permutation on n elements where there are exactly 3 superhurdles each spanning 7 elements.

From Lemma 1 and the results about the rarity of hurdles from the previous section, we have

$$F_{3,7}(n) > \frac{(n-18)!2^{n-18}}{2}\left(1 - O(n^{-2})\right)\binom{n-18}{3}$$

where $\frac{(n-18)!2^{n-18}}{2}(1-O(n^{-2}))$ is a lower bound for the number of permutations of length $n-18$ (with zero adjacencies and without hurdles) and $\binom{n-18}{3}$ is the number of ways to choose the elements for extension. Therefore we have

$$\begin{aligned}\frac{F_{3,7}(n)}{n!2^n} &> \frac{(n-18)!2^{n-18}}{2}\left(1-O(n^{-2})\right)\binom{n-18}{3}\frac{1}{n!2^n} \\ &\in \Omega(n^{-15}) \end{aligned} \tag{8}$$

4 On the Proportion of Unsafe Cycle-splitting Inversions

We now build on results from the previous section to show that the results of Bergeron *et al.* [3] about oriented inversions can be extended to so-called cycle-splitting inversions. Our results depend on a conjecture of Sankoff and Haque [9], which we support with a new experiment.

Denote the two vertices representing a permutation element π_i in the breakpoint graph by π_i^- and π_i^+ (π° can denote either). Think of embedding the breakpoint graph on a circle as follows: we place all $2n$ vertices on the circle so that:

1. π_i^+ and π_i^- are adjacent on the circle,
2. π_i^- is clockwise-adjacent to π_i^+ if and only if π_i is positive, and
3. a π_i° is adjacent to a π_{i+1}° if and only if π_i and π_{i+1} are adjacent in π.

For two vertices $v_1 = \pi_i^\circ$ and $v_2 = \pi_j^\circ$ ($i \neq j$) that are adjacent on the circle, add the edge (v_1, v_2)—a reality edge (also called a black edge); also add edges (π_i^+, π_{i+1}^-) for all i and (π_n^+, π_1^-)—the desire edges (also called gray edges). The breakpoint graph is just as described in [6], but its embedding clarifies the notion of orientation of edges, which plays a crucial role in our study of unsafe inversions.

In the breakpoint graph two reality edges on the same cycle are *convergent* if a traversal of their cycle visits each edge in the same direction in the circular embedding; otherwise they are *divergent*. Any inversion that acts on a pair of divergent reality edges splits the cycle to which the edges belong; conversely, no inversion that acts on a pair of convergent reality edges splits their common cycle. (An inversion that acts upon a pair of reality edges in two different cycles simply merges the two cycles.)

An inversion can be denoted by the set of elements in the permutation that it rearranges; for instance, we can write $r = \{\pi_i, \pi_{i+1}, \ldots, \pi_j\}$. The permutation obtained by applying an inversion r to a permutation π is denoted by $r\pi$. Thus, using the same r, we have $r\pi = (\pi_1 \ldots \pi_{i-1}{}^{-}\pi_j \ldots {}^{-}\pi_i\pi_{j+1} \ldots \pi_n)$. We call a pair (π, r) *unsafe* if π does not contain a hurdle but $r\pi$ does. A pair (π, r) is *oriented* if $r\pi$ contains more adjacencies than π does. A pair (π, r) is *cycle-splitting* if $r\pi$ contains more cycles than π does. (When π is implied from the context, we call r unsafe, oriented, or cycle-splitting, respectively, without referring to π.) Note that every oriented inversion is a cycle-splitting inversion. An inversion r on a permutation π is a *sorting* inversion whenever we have $d(r\pi) = d(\pi) - 1$.

Let π be a random permutation without hurdles and r a randomly chosen oriented inversion on π. Bergeron *et al.* [3] proved that the probability that the pair (π, r) is unsafe

is $O(n^{-2})$. However, not every sorting inversion for a permutation without hurdles is necessarily an oriented inversion; on the other hand, it is necessarily a cycle-splitting inversion. The result in [3] thus applies only to a small fraction of all sorting inversions. We now proceed to study *all* inversions that can increase the cycle count. We show that, under Sankoff's randomness hypothesis (stated below), the proportion of these inversions that are unsafe is $O(n^{-2})$.

In [9], Sankoff and Haque built graphs by effectively fixing desire edges, one to each vertex, and then randomly connecting each vertex to exactly one reality edge. Equivalently we can view this process as fixing reality edges and then randomly connecting each vertex to exactly one desire edge. However, the orientation of a reality edge in the breakpoint graph is not independent of the orientation of the other reality edges. Thus, in this random generation process (where they are independent), it is possible to generate a graph that does not correspond to a permutation. Sankoff and Haque [9] proposed a *Randomness Hypothesis* in this regard; it states that the probabilistic structure of the breakpoint graph is asymptotically independent of whether or not the generated graph is consistent with a permutation. In the randomly constructed graphs, every reality edge induces a direction independently and each direction has a probability of $\frac{1}{2}$, so the expected number of reality edges with one orientation equals that with the other orientation. Our own experiments support the randomness hypothesis in this respect, as illustrated in Figure 1, which shows the number of edges inducing a clockwise orientation on a cycle of length 500 from 2000 random permutations of length 750. Observations (the vertical bars) match a binomial distribution (the black dots). The ran-

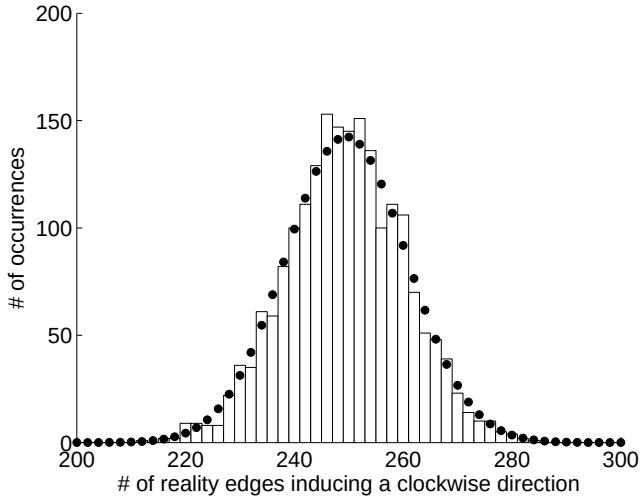

Fig. 1. The number of edges inducing a clockwise direction in cycles of length 500, taken from random permutations. Black dots are the expected values from the binomial distribution while white bars are experimental values.

domness hypothesis is important inasmuch as it is considerably simpler to analyze a random breakpoint graph than a random signed permutation.

We proceed by analyzing the random breakpoint graph to show that the number of cycle-splitting inversions over all permutations of length n is $\theta(n^2)|\Sigma_n|$. By showing that the total number of unsafe cycle-splitting inversions over all permutations is at most $|\Sigma_n|$, we conclude that a random inversion applied to a random permutation creates a hurdle with probability $O(n^{-2})$.

The number of cycle-splitting inversions in a permutation π equals the number of pairs of divergent reality edges in the breakpoint graph for π. Consider a cycle containing L reality edges and let k of them share the same orientation; the number of pairs of divergent reality edges in this cycle is then $k(L-k)$. Thus, under the randomness hypothesis, the expected number of pairs of divergent reality edges for a cycle containing L reality edges is

$$\sum_{k=0}^{L} \binom{L}{k} \left(\frac{1}{2}\right)^L k(L-k) = \frac{1}{4}L(L-1)$$

by the binomial theorem. We also know that the maximum number of pairs of divergent reality edges for a cycle with L reality edges is $\frac{1}{4}L^2$. Thus at least half of the cycles with L reality edges have at least $\frac{1}{4}L^2 - \frac{1}{2}L$ pairs of divergent reality edges (for $L > 2$).

Using the randomness hypothesis, Sankoff and Haque [9] have shown that in a random breakpoint graph (with $2n$ vertices) the expected number of reality edges in the largest cycle is $\frac{2}{3}n$. Since the maximum number of reality edges in the largest cycle is n, at least half the random breakpoint graphs have a cycle with at least $\frac{1}{3}n$ reality edges. So, for all random breakpoint graphs, at least $\frac{1}{4}$ of them have at least $\frac{1}{36}n^2 - \frac{1}{6}n$ pairs of divergent reality edges. Hence, under the randomness hypothesis, the number of pairs (π, r), where r is a cycle-splitting inversion of π, is $\Theta(n^2)|\Sigma_n|$.

Let $H_n \in \Sigma_n$ be the subset of permutations over n elements where each permutation contains one or more hurdles. Given a permutation $h \in H_n$, at most $\binom{n}{2}$ pairs of (π, r) can yield this specific h. Since $|H_n| = \Theta(\frac{1}{n^2}|\Sigma_n|)$, the number of unsafe pairs (π, r) is $O(|\Sigma_n|)$ and thus so is the number of unsafe cycle-splitting pairs. Therefore, under the randomness hypothesis, for a random permutation $\pi \in \Sigma_n$, if r is a cycle-splitting inversion on π, the probability that r is unsafe is $O(n^{-2})$. Unlike the result from Bergeron about oriented inversions, this result is conditioned on Sankoff's randomness hypothesis, which remains to be proved. All experimental work to date appears to confirm the correctness of that hypothesis; and under this hypothesis, our result extends that of Bergeron from a small fraction of candidate inversions to all candidate (i.e., cycle-splitting) inversions.

5 Experimental Results

In this section we focus on statistical properties of hurdles as they arise in the context of algorithms for sorting by inversions, in particular, the algorithms of Kaplan and Verbin [8] and of Tannier and Sagot [14], or any algorithm that chooses cycle-splitting inversions uniformly at random. Through various experiments we study the behavior

of unsafe inversions in random sorting sequences; our results suggest that randomized approaches to sorting by inversions should work very well in practice.

Select uniformly at random a pair (π, r) made of a length i permutation π and an oriented inversion r on π. From Bergeron *et al.* [3] we know that (π, r) is unsafe with probability $O(i^{-2})$. Applying r to π and gluing the resulting adjacencies gives a new permutation π', which is a sample drawn from the uniform distribution on the set of permutations Σ^0_{i-1} if r creates one adjacency or on the set of permutations Σ^0_{i-2} if r creates two adjacencies. Now we can apply a random oriented inversion r' to π'. Note, however, that the probability that (π', r') is unsafe need not be $O((i-1)^{-2})$ (or $O((i-2)^{-2})$). The reason is that we cannot uniformly select a random *pair* of permutation and oriented inversion by first uniformly selecting a permutation and then uniformly selecting an oriented inversion on that permutation. Indeed, we can select a pair only in the very first step since the pair fixes the permutation (but not the permutation and an inversion) in the second step.

To study the effect of random selection of pairs rather than random selection of permutation followed by random selection of inversion, and to see how these selections are affected, if at all, by restricting the choice of inversions to oriented inversions or allowing any cycle-splitting inversion, we define four randomized algorithms as follows.

- Algorithm RO: uniformly select a random permutation π, then in every iteration uniformly select and apply a random oriented inversion r.
- Algorithm ROp: uniformly select a pair of permutation and oriented inversion (π, r), apply r on π, then in every subsequent iteration uniformly select and apply a random oriented inversion r.
- Algorithm RC: uniformly select a random permutation π, then in every iteration uniformly select and apply a random cycle-splitting inversion r.
- Algorithm RCp: uniformly select a pair of permutation and cycle-splitting inversion (π, r), apply r on π, then in every subsequent iteration uniformly select and apply a random cycle-splitting inversion r.

Each of these algorithms applies a randomly selected inversion at each step. A single run in each algorithm stops in a positive permutation that may or may not be the identity. If it is not the identity, then at least one unsafe inversion was used; our experiments as described below are designed to characterize these unsafe inversions.

All of our experiments are on random permutations with no adjacencies nor hurdles and with lengths between 20 and 500. For each length, we tested each of the randomized algorithms on $100,000$ runs.

Table 1 lists the failure rates for the four algorithms. (A run of the algorithm fails when it stops at a positive permutation that is not the identity.) The failure rates of algorithms RO and ROp are close to 37%, which matches the experimental results of Kaplan and Verbin [8], whose algorithm is essentially algorithm RO. Given any permutation without adjacencies, these two randomized algorithms always apply an oriented inversion to create adjacencies; the adjacencies are then glued to get a shorter permutation without adjacencies for the next iteration. If the probability of selecting an unsafe inversion is $\Theta(i^{-2})$ at each step for permutations of length i without adjacencies (modulo some dependencies as one progresses through the sorting), the upper

Table 1. *The failure rates for algorithm RO, algorithm ROp, algorithm RC and algorithm RCp*

Length	20	50	100	200	500
Algorithm RO	36.45%	36.90%	37.1%	36.7%	37.4%
Algorithm ROp	36.33%	37.06%	37.4%	37.4%	36.7%
Algorithm RC	37.39%	40.31%	41.4%	42.3%	41.5%
Algorithm RCp	37.45%	40.15%	41.0%	41.5%	42.3%

bound for the overall probability of failure at completion for a permutation of length n is $O(\sum_{i=2}^{n} i^{-2}) = O(1)$.

Table 1 also shows that the two approaches—uniformly selecting a random pair of permutation and oriented (cycle-splitting) inversion in algorithm ROp (RCp) and uniformly selecting a permutation and then uniformly selecting an oriented (cycle-splitting) inversion in algorithm RO (RC)—have very similar failure rates. Thus choosing a random pair of permutation and oriented (or cycle-splitting) inversion in the first step does not change the behavior of the randomized algorithm. (However, we currently have no explanation to offer for the lower failure rate for oriented inversions than for cycle-splitting inversions.)

To deal with the failed instances, Kaplan and Verbin [8] proposed a random-restart strategy; under this strategy, one keeps restarting until the algorithm succeeds. Table 2 shows the average number of restarts needed for our four algorithms. The results for oriented inversions are once again in accord with the experimental results of Kaplan and Verbin.

We now study the failed instances for algorithms RO and RC in detail. A run of any of the randomized algorithms fails if it applies at least one unsafe inversion. Figure 2 shows the distribution of the first unsafe inversion for algorithm RC on permutations of length 100. (The distribution of the first unsafe inversion for algorithm RO is identical.) We analyze random permutations (generated by applying d uniformly chosen inversions and filtering out those with distance less than d) at seven different distances from the identity ($d = 20, 40, 60, 80, 90, 95, 100$). Note that, for a distance d, the entire distribution lies to the left of d in the horizontal axis, since the first unsafe inversion has to occur within the first d inversions. We find that the first unsafe inversion, if any, usually occurs in the second half of the sorting sequence, indeed, among the last few inversions. Thus, for any permutation, the randomized approach constructs most of the sorting sequence and fails, if at all, when the permutation is nearly sorted.

Table 2. *Average number of trials needed to sort with random-restart strategy*

Length	20	50	100	200	500
Algorithm RO	0.61	0.59	0.59	0.59	0.59
Algorithm ROp	0.60	0.60	0.58	0.58	0.59
Algorithm RC	0.65	0.69	0.71	0.72	0.75
Algorithm RCp	0.66	0.69	0.72	0.73	0.74

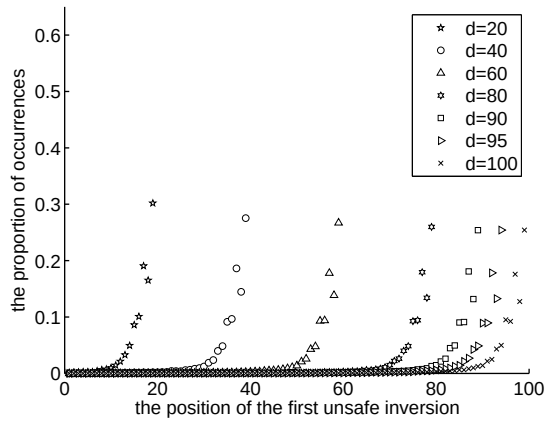

Fig. 2. The distribution of the first unsafe inversion for algorithm RC on permutations of length 100.

Define the *gap length* as the number of inversions applied after the first unsafe inversion and until the algorithm gets stuck. The gap length can be also viewed as the minimum number of inversions over which to backtrack in order to correct a failed instance. Figure 3 shows the distribution of the gap length for algorithm RC on the same permutations of length 100. It is interesting to point out that permutations at different distances from the identity ($d = 20, 40, 60, 80, 90, 95, 100$) have essentially the same distribution of gap length. The gap length (if any) is usually very small, which implies that backtracking a minimum number of steps to replace the unsafe inversion with a safe one should outperform a method that does a full restart.

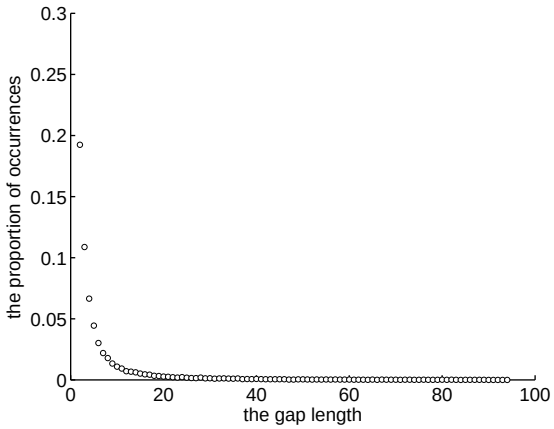

Fig. 3. The distribution of the gap length for algorithm RC on permutations of length 100.

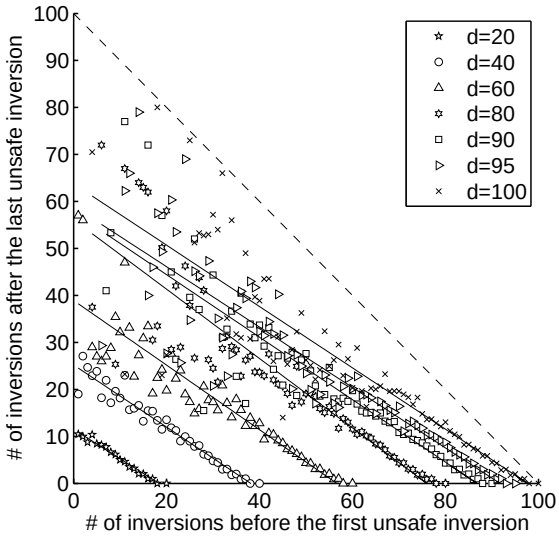

Fig. 4. The number of inversions after the last unsafe inversion, as a function of the number of inversions before the first unsafe inversion.

Tannier and Sagot [14] showed that not all of the inversions applied after the first unsafe inversion have to be undone in a backtracking approach. They showed that inversions applied after the *last* unsafe inversion are valid sorting inversions and, in fact, form the *tail* of an optimal sorting sequence. Thus both the head (before the first unsafe inversion) and the tail (after the last unsafe inversion) can be re-used. As the correction can itself introduce unsafe inversions, their algorithm uses successive iterations, in which lists of sorting inversions are appended to the head and other lists are prepended to the tail, until the entire sorting sequence is constructed.

Figure 4 shows the average number of inversions after the last unsafe inversion ($n_{>l}$) as a function of the number of inversions before the first unsafe inversion ($n_{<f}$). The distributions are for permutations of length 100 and for different distances ($d = 20, 40, 60, 80, 100$), under algorithm RC. (The distributions for algorithm RO are nearly identical.) The diagonal (drawn only for distance 100) shows the ideal situation where $n_{>l} + n_{<f} = d$ (which can occur only when there is no unsafe inversion). Each of the other lines is a linear regression fit of the data for a fixed distance. These lines are not perfect diagonals, but they come close, indicating that, for most permutations, nearly all inversions occur before the first unsafe inversion or after the last unsafe inversion. The approach of Tannier and Sagot will thus give most of the sorting sequence in a single iteration for a majority of instances. Put in structural, rather than algorithmic, words, valid sorting inversions make up a very large proportion of random inversions, even when the failed sequence selects an unsafe inversion very early.

6 Conclusions

We have both extended and simplified results of Bergeron and Caprara on the expected structure of signed permutations and their behavior under inversions. These extensions demonstrate the mathematical power of the framed common interval framework developed by Bergeron and the potential uses of the randomness hypothesis proposed by Sankoff and Haque to bind the asymptotic properties of valid and randomized breakpoint graphs. Our results confirm the evasive nature of hurdles (and, even more strongly, of fortresses); indeed, these structures are both so rare and, more importantly, so hard to create accidentally that they can be ignored in almost all cases. (Of course, if a permutation does have a hurdle, that hurdle must be handled if we are to sort the permutation; but handling initial hurdles takes only linear time—the cost comes when attempting to avoid creating a new one, i.e., when testing cycle-splitting inversions for safeness.) Moreover, our experimental results regarding the first occurrence of an unsafe inversion, the gap length, and the fraction of a failed sorting sequence that can be reused all suggest possible paths for improving on the results of Kaplan and Verbin and of Tannier and Sagot for sorting by inversions.

7 Acknowledgments

A preliminary version of this work [13] appeared in the proceedings of the 6th RECOMB Workshop on Comparative Genomics (RECOMB-CG'08).

References

1. D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comput. Biol.*, 8(5):483–491, 2001. A preliminary version appeared in WADS'01, pp. 365–376.
2. A. Bergeron. A very elementary presentation of the Hannenhalli–Pevzner theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.
3. A. Bergeron, C. Chauve, T. Hartman, and K. Saint-Onge. On the properties of sequences of reversals that sort a signed permutation. *JOBIM*, pages 99–108, June 2002.
4. A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. In *Proc. 9th Int'l Conf. Computing and Combinatorics (COCOON'03)*, volume 2697 of *Lecture Notes in Computer Science*, pages 68–79. Springer Verlag, Berlin, 2003.
5. A. Caprara. On the tightness of the alternating-cycle lower bound for sorting by reversals. *J. Combin. Optimization*, 3:149–182, 1999.
6. S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Ann. ACM Symp. Theory of Comput. (STOC'95)*, pages 178–189. ACM Press, New York, 1995.
7. S. Hannenhalli and P.A. Pevzner. Transforming mice into men (polynomial algorithm for genomic distance problems). In *Proc. 36th Ann. IEEE Symp. Foundations of Comput. Sci. (FOCS'95)*, pages 581–592. IEEE Press, Piscataway, NJ, 1995.
8. H. Kaplan and E. Verbin. Efficient data structures and a new randomized approach for sorting signed permutations by reversals. In *Proc. 14th Ann. Symp. Combin. Pattern Matching (CPM'03)*, volume 2676 of *Lecture Notes in Computer Science*, pages 170–185. Springer Verlag, Berlin, 2003.

9. D. Sankoff and L. Haque. The distribution of genomic distance between random genomes. *J. Comput. Biol.*, 13(5):1005–1012, 2006.
10. J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishers, Boston, MA, 1997.
11. A.H. Sturtevant and G.W. Beadle. The relation of inversions in the x-chromosome of drosophila melanogaster to crossing over and disjunction. *Genetics*, 21:554–604, 1936.
12. A.H. Sturtevant and Th. Dobzhansky. Inversions in the third chromosome of wild races of drosophila pseudoobscura and their use in the study of the history of the species. *Proc. Nat'l Acad. Sci., USA*, 22:448–450, 1936.
13. K.M. Swenson, Y. Lin, V. Rajan, and B.M.E. Moret. Hurdles hardly have to be heeded. In *Proc. 6th RECOMB Workshop Comp. Genomics (RECOMB-CG'08)*, volume 5267 of *Lecture Notes in Computer Science*, pages 241–251. Springer Verlag, Berlin, 2008.
14. E. Tannier and M. Sagot. Sorting by reversals in subquadratic time. In *Proc. 15th Ann. Symp. Combin. Pattern Matching (CPM'04)*, volume 3109 of *Lecture Notes in Computer Science*, pages 1–13. Springer Verlag, Berlin, 2004.

Appendix: Cover Letter

This work, as noted in the acknowledgments, appeared in a preliminary form in the RECOMB-CG'08 proceedings; its submission here to the Journal of Computational Biology as part of selected papers from that conference includes fairly extensive corrections as well as 40% entirely new material.

We received three referee reports for our RECOMB-CG'08 submission. The third report was just 5 lines and applauded our results; the second report was strongly supportive, but noted that our discussion assumed circular genomes and did not really make that clear; and the first summarized our contribution, but regretted that we did not investigate the structure of actual genomes. We had already responded to the criticism of the second reviewer for the final version in the proceedings, which, like this manuscript, is focused on circular genomes, but makes this explicit and also takes particular care to discuss the linear case whenever there is a significant difference (which occurs only twice in the paper). The first referee's comment about testing real genomes for the presence of hurdles cannot be easily addressed, for three reasons. First, most genomes are not simple signed permutations, but have gene families, raising issues of duplication/loss models and orthologies, which are outside the scope of this paper. Second, the few genomes where such duplications might not present a problem, such as animal mitochondria or the chloroplasts of terrestrial plants, may have evolved under transpositions as well as inversions (almost certainly so for mitochondria) and hurdles are specific to the inversion framework; and third, our focus is on randomly chosen inversions, whereas inversions we could observe or infer in real genomes are the product of millions of years of selection, with selection biases that we are only starting to understand. We fully agree with the first referee that such a study would be very interesting and informative, but in our view it requires understanding and tools that we still lack, although we are pleased to see that the referee views our work here as contributing to these tools and this understanding.

What is new in this extended manuscript comes in two flavors: amendments and improvements to what had already appeared in the proceedings and entirely new material. As to the first flavor, we have rewritten Section 4 (which extends Bergeron's results from oriented inversions to cycle-splitting inversions) to make it clearer as well as to locate precisely where and how the Randomness Hypothesis of Sankoff is being used. We have also edited in a fairly minor way the writing throughout the first four sections, as well as added text to reflect the presence of the new material. As to the second flavor, Section 5, entitled "Experimental Results" is entirely new and accounts for slightly over 4 of the 14.5 (15.5 counting the bibliography) pages of this manuscript—in other words, we have added about 40% new material as compared to the RECOMB-CG version. This new section ties our results to the two sorting algorithms that attempt to ignore hurdles—the randomized algorithm of Kaplan and Verbin and its clever deterministic version by Tannier and Sagot. Since the focus of our work in both versions is the accidental creation of hurdles in the process of sorting a permutation by inversions, and since our RECOMB-CG results indicated that such accidental creation should be very rare, it only made sense to examine in some detail what happened in these rare cases and how much damage could be expected. Our experiments confirm the intuition built from our theoretical results and indicate that, indeed, sorting methods based on

random selection of oriented or cycle-splitting inversions should do very well on almost every input permutation. Together, these results strongly suggest that an algorithm that runs asymptotically faster than that of Tannier and Sagot (the current fastest, at least in asymptotic terms) on almost all input permutations can be designed.