On Computing the Breakpoint Reuse Rate in Rearrangement Scenarios

Anne Bergeron[1], Julia Mixtacki[2], and Jens Stoye[3]

[1] Dépt. d'informatique, Université du Québec à Montréal, Canada.
bergeron.anne@uqam.ca
[2] International NRW Graduate School in Bioinformatics and Genome Research, Universität Bielefeld, Germany. julia.mixtacki@uni-bielefeld.de
[3] Technische Fakultät, Universität Bielefeld, Germany.
stoye@techfak.uni-bielefeld.de

Abstract. In the past years, many combinatorial arguments have been made to support the theory that mammalian genome rearrangement scenarios rely heavily on breakpoint reuse. Different models of genome rearrangements have been suggested, from the classical set of operations that include inversions, translocations, fusions and fissions, to more elaborate models that include transpositions. Here we show that the current definition of *breakpoint reuse rate* is based on assumptions that are seldom true for mammalian genomes, and propose a new approach to compute this parameter. We explore the formal properties of this new measure and apply these results to the human-mouse genome comparison. We show that the reuse rate is intimately linked to a particular rearrangement scenario, and that the reuse rate can vary from 0.89 to 1.51 for scenarios of the same length that transform the mouse genome into the human genome, where a rate of 1 indicates no reuse at all.

1 Introduction

There has been ample evidence, since the birth of modern genetics, that the genomes of species have been often reshuffled, with large chunks of genetic material being moved [13]. Rearrangements such as inversions within one chromosome, translocations, fusions, and fissions of chromosomes have been regularly observed through the comparison of the genomes of close species. In the past two decades it became possible to compare more distant species, and the problem of explaining how to transform one genome into another with a sequence of rearrangements became a central problem of computational biology [11].

Computing the minimal number of inversions, translocations, fusions and fissions necessary to transform one genome into another is a solved and well understood mathematical problem [4–6, 8, 14]. A sequence of rearrangements that achieves this minimum is called a *parsimonious sorting scenario*.

The *breakpoint reuse rate* [1, 9, 12] measures how often regions of chromosomes are *broken* in these rearrangement scenarios. This parameter has traditionally a value between 1 and 2. Low or high values of the reuse rate have

been used in the literature to support different models of distributions of breaks along chromosomes [9, 12]. However, in this paper, we show that the reuse rate is extremely sensitive to both genome representation, and the rearrangement model.

In [9], the reuse rate is defined as a function of the length of rearrangement scenarios, but not the particular rearrangement operations used in it. Moreover, its computation is based on the assumptions that the compared genomes have the same number of chromosomes, and that they share the same set of telomere markers. In order to compute the reuse rate in the comparison of genomes that do not fit in this model, such as the mouse and human genomes, 'empty' chromosomes and additional telomere markers are added as necessary.

In this paper, we first argue that this approach necessarily yields abnormally high values of breakpoint reuse, and we then suggest a new approach to compute the reuse rate that uses all the information contained in a particular rearrangement scenario. We give lower and upper bounds for this measure, demonstrating that, depending on the particular scenario, a wide range of reuse rates can be inferred for the same data-set. Finally we apply our results to the human-mouse data-set studied in [9] and give a particular rearrangement scenario where the lower bound is actually achieved.

2 Preliminaries

Whole genomes are compared by identifying homologous segments along their DNA sequences, called *blocks*. These blocks can be relatively small, such as gene coding sequences, or very large fragments of chromosomes. The order and orientation of the blocks may vary in different genomes. In this paper, we assume that the genomes are compared with sufficiently large blocks such that each block occurs once and only once in each genome. For genomes that have linear chromosomes, a simple representation is to list the blocks in each chromosome, using minus signs to model relative orientation, such as in the following example:

Genome S: ∘ -2 -4 -5 -1 7 3 ∘ ∘ 8 6 10 9 11 12 ∘
Genome T: ∘ 1 2 3 4 5 ∘ ∘ 6 7 8 9 ∘ ∘ 10 11 12 ∘

Here, genome S has two chromosomes, and genome T has three. The unsigned symbol '∘' is used to mark ends of chromosomes. An *adjacency* in a genome is a sequence of two consecutive blocks, or a block and a '∘'. For example, in the above genomes, $(-2\ -4)$ is an adjacency of genome S, and $(5\ \circ)$ is an adjacency, also called a *telomere*, of genome T. Since a whole chromosome can be flipped, we always have $(a\ b) = (-b\ -a)$, and $(\circ\ a) = (-a\ \circ)$. Chromosomes can be represented by the set of their adjacencies.

Two genomes are said to be *co-tailed* if they have the same set of telomeres. Genomes that have only circular chromosomes are always co-tailed, since they do not have telomeres. Co-tailed genomes whose chromosomes are all linear have always the same number of chromosomes.

The *adjacency graph* of two genomes A and B is a graph whose vertices are the adjacencies of A and B, respectively called A-vertices and B-vertices, and such that for each block b there is an edge between adjacency $(b\ c)$ in genome A and $(b\ c')$ in genome B, and an edge between $(a\ b)$ in genome A, and $(a'\ b)$ in genome B. For example, the adjacency graph of genomes S and T from above is shown in Figure 1.

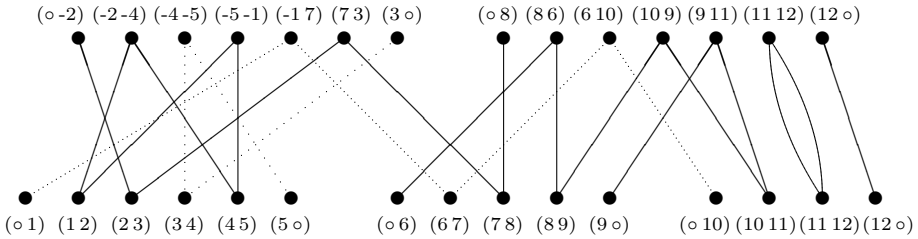

Fig. 1. The adjacency graph of genomes S and T. The S-vertices are on the top of the figure, and the T-vertices are on the bottom.

Since each vertex has at most two incident edges, the adjacency graph can be decomposed into connected components that are either cycles or paths. We classify paths as AA-paths, BB-paths and AB-paths depending on the type of vertices, A or B, of its extremities. For example, the adjacency graph of genomes S and T has the following connected components, described by the list of their adjacencies in genome T:

Two cycles, [(1 2)(4 5)] and [(11 12)].
One SS-path, [(2 3)(7 8)].
Two TT-paths, [(∘ 1)(6 7)(∘ 10)] and [(∘ 6)(8 9)(10 11)(9 ∘)].
Two ST-paths, [(3 4)(5 ∘)] and [(12 ∘)].

When two genomes share an adjacency, such as (11 12) or (12 ∘) for genomes S and T, then they belong to cycles of length 2 or paths of length 1 in the adjacency graph. We have:

Proposition 1 ([3]). *Two genomes are equal if and only if their adjacency graph has only cycles of length 2 and paths of length 1.*

We will refer to cycles of length more than 2, and paths of length more than 1, as *long* cycles and paths.

A *double-cut-and-join* (DCJ) rearrangement operation [3, 16] on genome A acts on two adjacencies $(a\ b)$ and $(c\ d)$ to produce either $(a\ d)$ and $(c\ b)$, or $(a\ -c)$ and $(-b\ d)$. When a, b, c and d are all blocks, the DCJ operation makes two cuts in genome A; these correspond, for linear chromosomes, to *inversions*,

translocations, and *circularization* when the two cuts act on the same chromosome and produce a circular chromosome. When one of a, b, c or d is a telomere, the operation makes one cut; these are special cases of inversions, translocations and circularizations that we will refer to as *semi*-operations. When there are two telomeres, the operation makes no cut, and it corresponds to a *fusion* of chromosomes, its inverse operation is a *fission* that makes one cut in a chromosome.

For a fixed set of rearrangement operations, the *distance* between genomes A and B is the minimum number of operations needed to rearrange – or *sort* – genome A into genome B. When all DCJ operations are allowed, we will denote this distance by $d_{DCJ}(A, B)$. The DCJ distance is easily computed from the adjacency graph [3]. It is given by:

$$d_{DCJ}(A, B) = N - (C + I/2) \tag{1}$$

where N is the number of blocks, C is the number of cycles, and I is the number of AB-paths. For example, the DCJ distance between our genomes S and T is given by $12 - (2 + 2/2) = 9$.

Rearrangement operations that do not create circular chromosomes are called *HP operations*, from the names of the authors of the first algorithm to compute the distance between genomes using these operations [5]. The corresponding distance is denoted by $d_{HP}(A, B)$. It is always greater or equal to the DCJ distance, and the difference is generally very small in real data. For example, the difference is 0 when the human and mouse genomes are compared with 281 blocks; and the difference is 1 when the human and chicken genomes are compared with 586 blocks.

A rearrangement operation is *sorting* if it lowers the distance by 1, and a sequence of sorting operations of length $d(A, B)$ is called a *parsimonious sorting scenario*. For the DCJ distance, it is easy to detect sorting operations since, by the distance formula (1), any operation that increases the number of cycles by 1, or the number of AB-paths by 2, is sorting. Note that no operation can modify simultaneously those two parameters [3].

3 Breakpoint reuse rate

Rearrangement scenarios have been used in the past years to assess whether some regions of chromosomes were particularly susceptible to break. The measure used to infer the existence of these regions is called the *breakpoint reuse rate*. Before formally defining this measure, it is important to note that most previous definitions relied on the assumption that *the compared genomes are co-tailed*. This assumption implies that genomes A and B have the same number of linear chromosomes, and that there are no long paths. Since all adjacencies are in cycles or short paths, it also implies that the adjacency graph has the same number of A-vertices and B-vertices.

Let b be the number of A- or B-vertices that are in long cycles, then the *breakpoint reuse rate* r for co-tailed genomes A and B is traditionally defined as:

$$r = 2d(A, B)/b.$$

This definition reflects the fact that, with co-tailed genomes, the only sorting operations are inversions and translocations that must make two *cuts* in the chromosomes since all adjacencies are in cycles.

A well known result from sorting theory [2, 5, 7] asserts that at least $\ell - 1$ inversions or translocations are needed in order to sort a cycle of length 2ℓ. We thus have the following lower bound on the breakpoint reuse rate for co-tailed genomes:

Proposition 2. *If the adjacency graph of genomes A and B has c long cycles of total length $L_c = 2\ell_1 + \ldots + 2\ell_c$ and no long paths, then the breakpoint reuse rate is at least*

$$2 - 4c/L_c.$$

Proof. The number of A- or B-vertices that are in long cycles is $b = \ell_1 + \ldots + \ell_c = L_c/2$, and the distance $d(A, B) \geq (\ell_1 - 1) + \ldots + (\ell_c - 1) = L_c/2 - c$, implying that

$$\begin{aligned} r \;=\; 2d(A,B)/b \;&\geq\; 2(L_c/2 - c)/(L_c/2) \\ &=\; (L_c - 2c)/(L_c/2) \\ &=\; 2 - 4c/L_c. \end{aligned}$$

□

Proposition 2 establishes a link between the number of cycles and the breakpoint reuse rate: a few long cycles imply a high reuse rate, and many shorter cycles imply a low reuse rate. This fact is crucial in evaluating the breakpoint reuse of genomes that have been artificially made co-tailed. The procedure eliminates all paths in the adjacency graph by closing them into cycles [5, 14]. Unfortunately, this process transforms all rearrangement operations that make less than two cuts into operations that makes two cuts. Clearly, this can lead to an overestimation of reuse rates.

In order to obtain more realistic measures, we first extend the definition of breakpoint reuse rate to arbitrary linear genomes.

Definition 1. *Consider a rearrangement scenario that transforms genome A into genome B. Let C be the total number of cuts made by the operations of the scenario, and b the number of B-vertices that are in long cycles or paths, then the* breakpoint reuse rate *r is defined by:*

$$r = C/b.$$

Note that this definition corresponds to the traditional breakpoint reuse rate when genomes are co-tailed. The reuse rate also depends on a particular scenario, and it is not necessarily symmetric. Indeed, for example, if genome B differs from genome A by a fusion of two chromosomes, then no cuts are necessary to transform A into B, yielding a breakpoint reuse rate of 0, but one cut is necessary to transform B into A, yielding a breakpoint reuse rate of 0.5. This example also shows that the breakpoint reuse rate can be less than 1 for general genomes.

4 Bounding the breakpoint reuse rate

Since the breakpoint reuse rate depends on a particular rearrangement scenario, it is interesting to produce lower and upper bounds that are independent of a particular parsimonious scenario. The existence of long paths in an adjacency graph yields opportunities to choose operations that use less than two cuts. On the other hand, long cycles often impose mandatory reuse, as we saw in Proposition 2. For lower bounds, the strategy will thus be to avoid operations that create long cycles in the adjacency graph, while the opposite strategy will yield upper bounds. In Section 5, we will show that these bounds are effectively reached with real data.

4.1 Lower bounds

We already saw that the number of cuts necessary to sort a cycle of length 2ℓ is at least $2(\ell-1)$, and there is no possibility to lower this number. In fact, in the DCJ model this bound is tight, while in the HP model certain sets of paths or cycles may require extra operations to be sorted [4, 5].

The situation is very different for paths. Consider first an AB-path of length $2\ell+1$ that begins with a telomere g_1 of genome B and ends with a telomere $g_{2\ell+1}$ of genome A, were we list all adjacencies of the path, with those of genome A underlined:

$$[(\circ\ g_1)\underline{(g_2\ g_1)}(g_2\ g_3)\cdots\underline{(g_{2\ell}\ g_{2\ell-1})}(g_{2\ell}\ g_{2\ell+1})\underline{(\circ\ g_{2\ell+1})}]$$

Consider the DCJ operation that acts on the adjacencies $(g_{2\ell}\ g_{2\ell-1})$ and $(\circ\ g_{2\ell+1})$ to produce the adjacency $(g_{2\ell}\ g_{2\ell+1})$ of genome B. This operation is sorting since it creates a cycle of length 2, and shortens the AB-path by 2. It also requires only one cut. We thus have:

Proposition 3. *The minimum number of cuts necessary to sort an AB-path of length $2\ell+1$ is ℓ.*

Proof. Applied iteratively, the above strategy clearly sorts the path with ℓ cuts. It thus remains to prove that it is impossible to reduce the number of cuts below ℓ. However, the path contains ℓ adjacencies $(g_{2i}\ g_{2i-1})$ of A, $1 \le i \le \ell$, that are not adjacencies of B. Removing these requires at least ℓ cuts. □

Note that, in order to have a sorting sequence that minimizes the number of cuts, it is necessary to create the adjacency of genome B that is next to the genome A telomere. Any other sorting DCJ operation on the AB-path will increase the number of cuts.

In the case of a BB-path of length 2ℓ, we have more choices. Indeed, any path of the form

$$[(\circ\ g_1)\underline{(g_2\ g_1)}(g_2\ g_3)\cdots(g_{2\ell-2}\ g_{2\ell-1})\underline{(g_{2\ell}\ g_{2\ell-1})}(g_{2\ell}\ \circ)]$$

can be cut anywhere between adjacencies of the form $(g_{2k}\ g_{2k-1})$. This is always a sorting operation since it creates two new AB-paths of lengths $2k+1$ and $2\ell-(2k+1)$. Using Proposition 3, we easily have:

Proposition 4. *The minimum number of cuts necessary to sort a BB-path of length 2ℓ is ℓ.*

Finally, AA-paths can be sorted with a minimum number of cuts using the following procedure. Consider an AA-path of the form

$$[\underline{(g_1\ \circ)}(g_1\ g_2)\underline{(g_3\ g_2)}\cdots\underline{(g_{2\ell-1}\ g_{2\ell-2})}(g_{2\ell-1}\ g_{2\ell})\underline{(\circ\ g_{2\ell})}].$$

Creating the adjacency $(g_1\ g_2)$ of genome B is possible by acting on the adjacencies $(g_1\ \circ)$ and $(g_3\ g_2)$ of genome A. It is sorting since it creates a cycle, and it requires only one cut. The same holds for the last adjacency of genome B, $(g_{2\ell-1}\ g_{2\ell})$. Note that if the length of the path is 2, then no cut is necessary and we just do a fusion. Thus we have:

Proposition 5. *The minimum number of cuts necessary to sort an AA-path of length 2ℓ is $\ell - 1$.*

When the full spectrum of DCJ operations is allowed, it is always possible to construct a sorting sequence that achieves the minimum number of cuts, but this is not guaranteed when we restrict the operations to HP operations, as the following example shows. Let genomes E and F be the following:

Genome E: ∘ -1 -2 3 ∘
Genome F: ∘ 1 2 3 ∘

The adjacency graph has two EF-paths, one of length 1, and one of length 5. Thus the minimal number of cuts is 2. However, creating the adjacency 1 2 in genome F requires the creation of a circular chromosome, which is forbidden in the HP model. Thus the only allowable operation is the inversion of block number 2, which requires two cuts, followed by the inversion of block number 1, which requires one cut.

4.2 Upper bounds

In order to compute the maximal number of cuts, we need to reverse the strategy of the preceding section and create cycles as large as possible.

Since sorting operations that act on an AB-path must split the path into a cycle and an AB-path, we choose to construct the largest possible cycle on a path of length $2\ell + 1$, splitting it in a path of length 1, and a cycle of length 2ℓ. We thus have:

Proposition 6. *The maximum number of cuts necessary to sort an AB-path of length $2\ell + 1$ is $2\ell - 1$.*

A BB-path of length 2ℓ can always be sorted with $2\ell - 1$ cuts, by creating a cycle of length $2\ell - 2$, requiring 2 cuts, and a path of length 2 that can be sorted with a fission. When there are only AA-paths in the adjacency graph, then the only sorting operations that can be applied to those must create a cycle, since

splitting the path always yields two AA-paths. Creating a cycle can be easily done by a fusion of the two telomeres of genome A that are at the ends of the path. This creates a cycle of length 2ℓ, thus requiring $2(\ell-1)$ cuts. On the other hand, it is possible to do better, in terms of maximizing the number of cuts, by pairing AA-paths with BB-paths in the following sense.

Consider an AA-path of length $2\ell_a$, and a BB-path of length $2\ell_b$:

$$[\underline{(g_1\ \circ)}(g_1\ g_2)\underline{(g_3\ g_2)}\dots\underline{(g_{2\ell_a-1}\ g_{2\ell_a-2})}(g_{2\ell_a-1}\ g_{2\ell_a})\underline{(\circ\ g_{2\ell_a})}]$$

$$[(\circ\ h_1)\underline{(h_2\ h_1)}(h_2\ h_3)\dots(h_{2\ell_b-2}\ h_{2\ell_b-1})\underline{(h_{2\ell_b}\ h_{2\ell_b-1})}(h_{2\ell_b}\ \circ)]$$

The strategy is to act on the adjacencies $(h_2\ h_1)$ and $(g_1\ \circ)$ of genome A to produce $(\circ\ h_1)$ and $(h_2\ -g_1)$, resulting in two AB-paths, one of length 1 that corresponds to the new common telomere $(\circ\ h_1)$, and the other of length $2\ell_a + 2\ell_b - 1$. Using Proposition 6, the total number of cuts of this strategy would be $(2\ell_a - 1) + (2\ell_b - 1)$. We thus have:

Proposition 7. *The maximum number of cuts necessary to sort a BB-path of length 2ℓ is $2\ell - 1$. The maximum number of cuts necessary to sort an AA-path of length 2ℓ is $2\ell-1$, when it can be paired with a BB-path, otherwise it is $2\ell-2$.*

Again, the maximal values given here are for the DCJ model. In the HP model, the actual maxima could be a bit higher, but this generally represents a very small fraction in real data. For the mouse and human genome presented in the next section, this cost is null.

4.3 Bounds of the Reuse Rate

Using the various bounds on the number of cuts of the preceding sections, we can now derive closed formulas for bounding the breakpoint reuse rate. We have:

Theorem 1. *Suppose that the adjacency graph of two genomes A and B contains c long cycles of total length L_c, m long AB-paths of total length L_m, p AA-paths of total length L_p, and q BB-paths of total length L_q, then the number of B-vertices that are in long cycles or paths is given by:*

$$b = (L_c + L_m + L_p + L_q + m + 2q)/2$$

and the breakpoint resuse rate r for all parsimonious DCJ sorting scenarios is bounded by

$$1 + \frac{(L_c/2 - (2c + m + p + q))}{b} \leq r \leq 2 - \frac{2c + 3m + p + 3q + \delta(p - q)}{b},$$

where $\delta = 1$ if $p > q$ and 0 otherwise. Moreover there exist sorting scenarios that meet these bounds.

Proof. We show the detailed contributions for long AB paths. All the three other cases are treated similarly.

Suppose that there are m long AB-paths of lengths $2\ell_1+1, 2\ell_2+1, \ldots, 2\ell_m+1$, and let L_m be the sum of these lengths. Then the number of B-vertices that belong to these paths is:

$$v = \sum_{i=1}^{m} (\ell_i + 1) = \frac{(L_m + m)}{2}.$$

By Proposition 3, the minimum number of cuts required to sort these paths is:

$$\sum_{i=1}^{m} \ell_i = \frac{(L_m - m)}{2} = v - m$$

and, by Proposition 6, the maximum number of cuts required is:

$$\sum_{i=1}^{m} (2\ell_i - 1) = L_m - 2m = 2v - 3m.$$

Adding similar contributions in B-vertices and cuts for cycles, AA-paths and BB-paths, we can easily derive the bounds of the statement. The fact that there exist sorting scenarios that meet the bounds is due to the fact that all DCJ operations that were used to derive the minimal and maximal number of cuts were sorting operations. □

5 Reuse rates in the mouse-human whole genome comparison

We now turn to test how these various computations behave with real data. We used the same mouse-human data that was analyzed in [9] to prove that there was extensive breakpoint reuse in mammalian evolution.

The data-set compares the order and orientation of 281 synteny blocks of the mouse and human genome, and is given in Appendix 1. The mouse genome M has 20 chromosomes, and the human genome H has 23. The adjacency graph has the following characteristics (see Appendix 2):

$c = 27$ long cycles: 24 of them of length 4 and one of length 6, 8 and 10 each, giving an overall length of $L_c = 120$,
$s = 4$ short MH-paths,
$m = 12$ long MH-paths: lengths ranging from 3 to 51,
$p = 12$ MM-paths: lengths ranging from 2 to 46,
$q = 15$ HH-paths: lengths ranging from 2 to 22.

By construction, the data-set has no short cycles since blocks adjacent in both genomes are merged together. The number of adjacencies in the human genome

that are in long cycles or paths is $b = 300 = 281 + 23 - 4$, and the number in the mouse genome is $297 = 281 + 20 - 4$.

The HP distance, as computed by GRIMM [15] after closing all the paths and adding empty chromosomes, is $d(M, H) = 246$. There are 300 adjacencies in both the modified human and mouse genomes being in long cycles, yielding a breakpoint reuse rate for these co-tailed genomes of $2 \cdot 246/300 = 1.64$.

The DCJ distance is also $246 = 281 - (27 + 16/2)$, since there are 27 cycles and $4+12 = 16$ MH-paths. Using the bounds of Theorem 1 to estimate the breakpoint reuse rates of scenarios that transform the mouse into the human genome according to the definition given in this paper, we get a reuse rate r between $1 - 33/300$ and $2 - 147/300$, or:

$$0.89 \leq r \leq 1.51.$$

Compared to the value 1.64 obtained by forcing the genomes to be co-tailed, we obtain strictly lower values of reuse rate for all possible sorting scenarios.

We also verified whether there exist scenarios using HP operations that attain the minimum number of cuts. The answer is yes, and one such scenario can be found in Appendix 3. This scenario uses the following operations:

26 inversions to sort 20 long cycles, making 52 cuts,
7 translocations to sort 7 long cycles, making 14 cuts,
15 fissions and 48 semi-operations to sort 15 HH paths, making 63 cuts,
12 fusions and 57 semi-operations to sort 12 MM paths, making 57 cuts,
81 semi-operations to sort 12 long MH paths, making 81 cuts.

6 Conclusion

In this paper, we generalized the notion of breakpoint reuse rate to genomes that are not necessarily co-tailed. This new measure can be applied to circular or linear genomes, or even to genomes that have both types of chromosomes. We gave lower and upper bounds for the number of cuts in a rearrangement scenario, yielding lower and upper bounds to the reuse rate of all possible parsimonious scenarios that transform a genome into another. We also showed that transforming genome data-sets in order to make them co-tailed can yield to an overestimation of breakpoint reuse rate.

Acknowledgments

We kindly acknowledge Glenn Tesler for providing us the original data of [10] in a user friendly format. J. S. would like to thank his children Ferdinand, Leopold and Balduin for their help verifying the correctness of the mouse-human sorting scenario by coding the genomes by integers written on 281 post-it notes. The resulting video can be found at `http://www.techfak.uni-bielefeld.de/~stoye/rpublications/SvH_small.mov`.

References

1. M. Alekseyev and P. A. Pevzner. Are there rearrangement hotspots in the human genome? *PLoS Comput. Biol.*, 3(11):e209, 2007.
2. V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversals. *SIAM J. Computing*, 25(2):272–289, 1996.
3. A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In *Proceedings of WABI 2006*, volume 4175 of *LNBI*, pages 163–173, 2006.
4. A. Bergeron, J. Mixtacki, and J. Stoye. HP distance via Double Cut and Join distance. In *Proceedings of CPM 2008*, volume 5029 of *LNCS*, pages 56–68, 2008.
5. S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of FOCS 1995*, pages 581–592, 1995.
6. G. Jean and M. Nikolski. Genome rearrangements: a correct algorithm for optimal capping. *Inf. Process. Lett.*, 104(1):14–20, 2007.
7. J. D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals with application to genome rearrangement. *Algorithmica*, 13(1/2):180–210, 1995.
8. M. Ozery-Flato and R. Shamir. Two notes on genome rearrangements. *J. Bioinf. Comput. Biol.*, 1(1):71–94, 2003.
9. P. Pevzner and G. Tesler. Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. USA*, 100(13):7672–7677, 2003.
10. P. Pevzner and G. Tesler. Transforming men into mice: The Nadeau-Taylor chromosomal breakage model revisited. In *Proceedings of RECOMB 2003*, pages 247–256, 2003.
11. D. Sankoff. Edit distances for genome comparison based on non-local operations. In *Proceedings of CPM 1992*, volume 644 of *LNCS*, pages 121–135, 1992.
12. D. Sankoff and P. Trinh. Chromosomal breakpoint reuse in genome sequence rearrangement. *J. Comput. Biol.*, 12(6):812–821, 2005.
13. A. H. Sturtevant. A crossover reducer in *Drosophila melanogaster* due to inversion of a section of the third chromosome. *Biologisches Zentralblatt*, 46(12):697–702, 1926.
14. G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, 65(3):587–609, 2002.
15. G. Tesler. GRIMM: Genome rearrangements web server. *Bioinformatics*, 18(3):492–493, 2002.
16. S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

Appendix 1: The Dataset

The mouse genome:

1: ○ -136 140 93 -95 -32 25 37 -38 39 -40 76 246 30 -29 33 -8 14 -11 10 -9 ○
2: ○ -161 162 -159 158 -157 156 -155 154 34 -35 36 -180 179 -178 -213 214 -24 28 259 -258 260 ○
3: ○ 141 139 -57 56 58 68 -201 55 -70 -7 -66 -5 ○
4: ○ 137 -142 -138 -97 146 153 148 145 4 -3 2 -1 ○
5: ○ 116 -115 120 124 18 62 -63 64 6 -267 195 -196 197 -113 -114 -119 105 118 200 ○
6: ○ 117 106 123 109 65 -67 -23 22 -21 -53 42 51 41 -167 -187 264 -188 189 ○
7: ○ 257 -255 254 -256 177 -210 212 211 -221 220 219 -218 -184 176 224 174 -175 -183 ○
8: ○ 250 205 126 -134 133 -132 -127 129 -71 130 -253 269 -69 -252 225 -226 227 12 -165 ○
9: ○ -185 251 110 -186 216 -215 -94 96 -217 -54 -48 -46 47 ○
10: ○ 101 -100 -98 99 27 -170 -266 -263 248 194 -193 192 -191 ○
11: ○ -268 112 -20 -85 -87 -80 84 231 -230 229 -228 -232 233 -234 237 -236 235 238 ○
12: ○ -17 16 -15 -121 -107 -122 207 209 -125 -108 ○
13: ○ -160 -13 -111 -89 88 -151 150 86 81 149 152 -72 -74 ○
14: ○ 50 -45 171 -49 43 -168 -172 208 206 198 -199 203 -128 -131 -202 204 ○
15: ○ -73 143 270 190 ○
16: ○ 223 -135 -265 59 61 -60 -52 261 ○
17: ○ -102 -103 104 -75 -222 91 262 -90 -92 44 -26 249 77 -240 19 239 ○
18: ○ 164 163 -166 243 -31 78 82 79 -83 241 245 242 -244 -247 ○
19: ○ 182 -181 -147 144 -169 173 ○
X: ○ -274 -275 273 281 -272 278 -279 280 -276 277 -271 ○

The human genome:

1: ○ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ○
2: ○ 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ○
3: ○ 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ○
4: ○ 62 63 64 65 66 67 68 69 70 71 ○
5: ○ 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 ○
6: ○ 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 ○
7: ○ 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 ○
8: ○ 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 ○
9: ○ 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 ○
10: ○ 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 ○
11: ○ 175 176 177 178 179 180 181 182 183 184 185 186 ○
12: ○ 187 188 189 190 191 192 193 194 195 196 197 ○
13: ○ 198 199 200 201 202 203 204 205 ○
14: ○ 206 207 208 209 210 ○
15: ○ 211 212 213 214 215 216 217 218 219 220 221 ○
16: ○ 222 223 224 225 226 227 ○
17: ○ 228 229 230 231 232 233 234 235 236 237 238 ○
18: ○ 239 240 241 242 243 244 245 246 247 ○
19: ○ 248 249 250 251 252 253 254 255 256 257 ○
20: ○ 258 259 260 ○
21: ○ 261 262 263 ○
22: ○ 264 265 266 267 268 269 270 ○
X: ○ 271 272 273 274 275 276 277 278 279 280 281 ○

Appendix 2: Cycles and paths

We list the cycles and paths by their sequences of adjacencies in the human genome.

27 long cycles:

[(1 2) (2 3)]
[(9 10) (10 11)]
[(15 16) (16 17)]
[(21 22) (22 23)]
[(34 35) (35 36)]
[(37 38) (38 39)]
[(62 63) (63 64)]
[(132 133) (133 134)]
[(155 156) (156 157)]
[(157 158) (158 159)]
[(178 179) (179 180)]
[(191 192) (192 193)]
[(195 196) (196 197)]
[(225 226) (226 227)]
[(228 229) (229 230)]
[(278 279) (279 280)]
[(79 80) (83 84)]
[(80 81) (86 87)]
[(93 94) (95 96)]
[(105 106) (117 118)]
[(106 107) (122 123)]
[(127 128) (131 132)]
[(138 139) (141 142)]
[(277 278) (271 272)]
[(218 219) (219 220) (220 221)]
[(273 274) (275 276) (276 277) (280 281)]
[(233 234) (235 236) (236 237) (234 235) (237 238)]

16 MH-paths:

[(∘ 1)]
[(238 ∘)]
[(260 ∘)]
[(∘ 271)]
[(270 ∘) (189 190)]
[(281 ∘) (273 273) (274 275)]
[(∘ 72) (74 75) (222 223)]
[(∘ 62) (18 19) (239 240)]
[(159 ∘) (162 163) (164 165)]
[(∘ 206) (208 209) (207 208) (171 172) (49 50)]
[(186 ∘) (110 111) (89 90) (92 93) (140 141)]
[(263 ∘) (265 266) (134 135) (126 127) (128 129) (203 204) (201 202) (68 69) (252 253) (268 269)]
[(∘ 105) (118 119) (199 200) (198 199) (202 203) (130 131) (253 254) (254 255) (256 257)]
[(61 ∘) (60 61) (59 60) (52 53) (41 42) (167 168) (172 173) (168 169) (43 44) (91 92) (261 262)]
[(197 ∘) (113 114) (119 120) (114 115) (112 113) (20 21) (53 54) (48 49) (42 43) (50 51) (45 46) (46 47) (47 48)]
[(∘ 144) (146 147) (152 153) (72 73) (142 143) (137 138) (97 98) (98 99) (99 100) (26 27) (44 45) (170 171) (27 28) (23 24) (66 67) (6 7) (267 268) (111 112) (12 13) (165 166) (242 243) (244 245) (241 242) (245 246) (76 77) (249 250)]

12 MM-paths:

[(73 74)]
[(190 191)]
[(136 137)]
[(182 183) (181 182)]
[(115 116) (116 117)]
[(161 162) (160 161)]

[(100 101) (101 102) (103 104) (102 103)]
[(200 201) (54 55) (216 217) (215 216) (185 186)]
[(17 18) (124 125) (108 109) (123 124) (120 121) (107 108)]
[(204 205) (250 251) (184 185) (217 218) (96 97) (145 146) (3 4) (4 5)]
[(163 164) (166 167) (187 188) (188 189) (264 265) (58 59) (67 68) (65 66) (5 6) (64 65) (109 110) (251 252) (224 225) (173 174)]
[(8 9) (33 34) (154 155) (153 154) (147 148) (180 181) (36 37) (25 26) (248 249) (193 194) (194 195) (266 267) (169 170) (144 145) (148 149) (81 82) (78 79) (82 83) (240 241) (77 78) (30 31) (29 30) (246 247)]

15 HH-paths:

[(∘ 187) (∘ 264)]
[(∘ 126) (205 ∘)]
[(∘ 41) (51 52) (∘ 261)]
[(14 ∘) (11 12) (227 ∘)]
[(40 ∘) (39 40) (75 76) (104 ∘)]
[(∘ 15) (121 122) (206 207) (∘ 198)]
[(∘ 222) (90 91) (262 263) (∘ 248)]
[(174 ∘) (175 176) (183 184) (∘ 175)]
[(125 ∘) (209 210) (211 212) (221 ∘)]
[(210 ∘) (177 178) (213 214) (212 213) (∘ 211)]
[(143 ∘) (269 270) (69 70) (7 8) (13 14) (∘ 160)]
[(87 ∘) (84 85) (230 231) (231 232) (232 233) (∘ 228)]
[(∘ 88) (88 89) (151 152) (149 150) (150 151) (85 86) (19 20) (∘ 239)]
[(247 ∘) (243 244) (31 32) (24 25) (214 215) (94 95) (32 33) (28 29) (258 259) (259 260) (∘ 258)]
[(71 ∘) (129 130) (70 71) (55 56) (56 57) (57 58) (139 140) (135 136) (223 224) (176 177) (255 256) (257 ∘)]

Appendix 3: A sorting scenario with minimum number of cuts

We list the adjacencies of the human genome in the order they are repaired.

16 inversions of one block:

(1 2) and (2 3)	(9 10) and (10 11)	(15 16) and (16 17)
(21 22) and (22 23)	(34 35) and (35 36)	(37 38) and (38 39)
(62 63) and (63 64)	(132 133) and (133 134)	(155 156) and (156 157)
(157 158) and (158 159)	(178 179) and (179 180)	(191 192) and (192 193)
(195 196) and (196 197)	(225 226) and (226 227)	(228 229) and (229 230)
(278 279) and (279 280)		

6 inversions to sort two cycles:

(218 219)	(219 220) and (220 221)	(233 234)	(235 236)
(236 237)	(234 235) and (237 238)		

4 inversions to sort the two interleaving cycles of chromosome X:

(273 274)	(275 276)	(277 278) and (271 272)	(276 277) and (280 281)

7 translocations that repair two adjacencies:

(79 80) and (83 84)	(80 81) and (86 87)	(93 94) and (95 96)
(105 106) and (117 118)	(106 107) and (122 123)	(127 128) and (131 132)
(138 139) and (141 142)		

15 fissions:

(∘ 126) and (205 ∘) (∘ 187) and (∘ 264) (14 ∘) (∘ 15) (40 ∘) (∘ 41) (71 ∘) (∘ 88) (143 ∘) (174 ∘) (∘ 248) (210 ∘) (221 ∘) (∘ 228) (∘ 258)

186 non-degenerate semi-operations:

(189 190)	(164 165)	(162 163)	(239 240)	(18 19)	(49 50)	(140 141)	(256 257)	(254 255)
(261 262)	(91 92)	(47 48)	(46 47)	(45 46)	(50 51)	(249 250)	(11 12)	(39 40)
(181 182)	(115 116)	(161 162)	(100 101)	(101 102)	(103 104)	(200 201)	(17 18)	(204 205)
(250 251)	(184 185)	(163 164)	(274 275)	(272 273)	(8 9)	(129 130)	(70 71)	(55 56)
(56 57)	(57 58)	(139 140)	(135 136)	(223 224)	(176 177)	(255 256)	(262 263)	(88 89)
(151 152)	(149 150)	(150 151)	(232 233)	(231 232)	(230 231)	(84 85)	(85 86)	(19 20)
(177 178)	(213 214)	(212 213)	(211 212)	(209 210)	(259 260)	(258 259)	(28 29)	(32 33)
(94 95)	(214 215)	(24 25)	(31 32)	(243 244)	(75 76)	(51 52)	(171 172)	(207 208)
(208 209)	(121 122)	(206 207)	(175 176)	(183 184)	(269 270)	(69 70)	(7 8)	(13 14)
(92 93)	(89 90)	(110 111)	(90 91)	(222 223)	(74 75)	(253 254)	(43 44)	(168 169)
(172 173)	(167 168)	(41 42)	(52 53)	(59 60)	(60 61)	(42 43)	(48 49)	(53 54)
(20 21)	(112 113)	(114 115)	(119 120)	(113 114)	(268 269)	(252 253)	(68 69)	(201 202)
(203 204)	(128 129)	(126 127)	(134 135)	(265 266)	(130 131)	(202 203)	(198 199)	(199 200)
(118 119)	(76 77)	(245 246)	(241 242)	(244 245)	(242 243)	(165 166)	(12 13)	(111 112)
(267 268)	(6 7)	(66 67)	(23 24)	(27 28)	(170 171)	(44 45)	(26 27)	(99 100)
(98 99)	(97 98)	(137 138)	(142 143)	(72 73)	(152 153)	(146 147)	(33 34)	(154 155)
(153 154)	(147 148)	(180 181)	(36 37)	(25 26)	(248 249)	(193 194)	(194 195)	(266 267)
(169 170)	(144 145)	(148 149)	(54 55)	(216 217)	(215 216)	(217 218)	(96 97)	(145 146)
(3 4)	(124 125)	(108 109)	(123 124)	(120 121)	(246 247)	(29 30)	(30 31)	(77 78)
(240 241)	(82 83)	(78 79)	(166 167)	(187 188)	(188 189)	(264 265)	(58 59)	(67 68)
(65 66)	(5 6)	(64 65)	(109 110)	(251 252)	(224 225)			

12 fusions:

(182 183)	(73 74)	(190 191)	(136 137)	(116 117)	(160 161)	(102 103)	(185 186)	(107 108)
(4 5)	(81 82)	(173 174)						