Appendix: Counting Sorting Scenarios and Intermediate Genomes for the Rank Distance*

João Paulo Pereira Zanetti[1][0000−0002−9955−7751], Leonid Chindelevitch[2][0000−0002−6619−6013], and João Meidanis[1][0000−0001−7878−4990]

[1] Institute of Computing
University of Campinas
Av. Albert Einstein, 1251, Campinas SP, Brazil
{joao.zanetti,meidanis}@ic.unicamp.br
[2] School of Computing Science
Simon Fraser University
8888 University Drive, Burnaby BC, Canada
leonid@sfu.ca

A Proofs for Section 3.2 — Sorting

Lemma 1. *Given a genome A, and an operation X applicable to A, we have $d(A, A + X) = r(X)$.*

Proof. If X is a cut, the graph $BG(A, A + X)$ has $2n - 2$ extremities in 0-paths and 2-cycles, and one 1-path formed by the adjacency cut from A. Therefore, $d(A, A + X) = 2n - 2c(A, A + X) - p(A, A + X) = 1$.

If X is a join, the graph $BG(A, A + X)$ is similar to the case above, but the one-path correspond to the adjacency joined in $A + X$.

If X is a double swap, it cuts two adjacencies, and adds two new adjacencies using the same four extremities. The graph $BG(A, A+X)$ has one 4-cycle in the place of two 2-cycles of $BG(A, A)$, and therefore $d(A, A + X) = 2$. □

Corollary 2. *Given two genomes A and B,*

$$d(A, B) \leq w(A, B).$$

Proof. There is a scenario $\mathcal{X} = X_1, X_2, \ldots, x_k$ such that $w(\mathcal{X}) = w(A, B)$. Considering the scenario $\mathcal{X}$, we have

$$w(A, B) = \sum_{i=1}^{k} r(X_k).$$

Because of Lemma 1, we have

$$w(A, B) = \sum_{i=1}^{k} d(A + \sum_{j=1}^{i-1} X_j, A + \sum_{j=1}^{i} X_j).$$

* JPPZ is supported by FAPESP grant 2017/02748-3. LC is supported by an NSERC Discovery Grant and a Sloan Foundation Fellowship. JM is supported by FAPESP grant 2018/00031-7.

Finally, the triangle inequality gives us

$$w(A, B) \geq d(A, B).$$

□

Lemma 3. *If $Ax \neq x$, and $Bx = x$, then cutting the adjacency $\{x, Ax\}$ in A is always a sorting operation.*

Proof. In the breakpoint graph $BG(A, B)$, the node corresponding to the extremity x is an end to a path. Let P be this path.

Let X be the cut of adjacency $\{x, Ax\}$. The graph $BG(A + X, B)$ has all the same components of $BG(A, B)$, except for P. Instead of P, there are two paths. The first is a path with all the nodes of P except for x. It has the same type as P. The second is a proper 0-path with the node x. Therefore, $d(A + X, B) = d(A, X) - 1$. □

Lemma 4. *If A and B have the same free ends, there is a sorting double swap.*

Proof. If A and B have the same free ends, then the components of $BG(A, B)$ are 0-paths and cycles. For a cycle, there is always at least one double swap that splits the cycle in two smaller ones, decreasing the distance by 2. □

Theorem 5. *Given two genomes A and B,*

$$d(A, B) = w(A, B).$$

Proof. Lemmas 3 and 4 give us an outline for a sorting algorithm:

1. Apply cuts to A and B until both genomes have the same free ends
2. Apply double swaps until all the remaining components are sorted

Let $\mathcal{X}$ be a sorting scenario obtained with the procedure above. Because all operations in $\mathcal{X}$ are sorting, and reduce the distance by exactly their weight, we have $w(\mathcal{X}) = d(A, B)$.

Since $w(A, B) \leq w(\mathcal{X})$, by Corollary 2, $d(A, B) = w(A, B)$. □

B Proofs for Section 3.3 — Counting Scenarios for Paths

Given a path with length k, the sizes of the optimal scenarios fall in a limited range. The longest operation scenarios for a k-path are the ones with only cuts and joins, making up a total of k operations. Since the double swaps have twice the weight of a cut or join, scenarios with more double swaps are shorter. Their minimum length is defined in the following lemma.

Lemma 6. *Let A and B be two genomes over the same genes such that the only big component of $BG(A, B)$ is a k-path P. The minimum length of a scenario that sorts A into B is $\left\lfloor \frac{k}{2} \right\rfloor + 1$.*

Proof. We will show this by induction on k. Since P is a big path, we know that $k \geq 1$. A path of length 1 is always sorted with one operation, either a cut or a join. A path of length 2 is always sorted with two operations, a cut and a join.

Let us now treat the case $k \geq 3$. Assuming that any path with length $k' < k$ has a minimum length of $\lfloor k'/2 \rfloor + 1$ for its scenarios, we are going to compute the minimum possible length ℓ for the scenarios of a k-path.

There are three different types of operations that can be applied to P: double swaps, cuts, and joins. A double swap produces a cycle of length $2x < k$ and a path of length $k - 2x$, where $x \geq 1$. The length of any scenario for the cycle is $x - 1$, and by the induction hypothesis, the length of a solution for the path is at least $\left\lfloor \frac{k-2x}{2} \right\rfloor + 1$. Therefore, we have

$$\begin{aligned} \ell &\geq 1 + x - 1 + \left\lfloor \frac{k-2x}{2} \right\rfloor + 1 \\ &= 1 + x + \left\lfloor \frac{k}{2} \right\rfloor - x \\ &= \left\lfloor \frac{k}{2} \right\rfloor + 1. \end{aligned}$$

If the operation applied is a cut, the result is two paths, with lengths $x < k$ and $k - x - 1$, where $x \geq 0$. Here we will make use of a property of the floor function: $\lfloor a + b \rfloor \leq \lfloor a \rfloor + \lfloor b \rfloor + 1$ [2, Theorem 4.1], so that we have

$$\begin{aligned} \ell &\geq 1 + \left\lfloor \frac{x}{2} \right\rfloor + 1 + \left\lfloor \frac{k-x-1}{2} \right\rfloor + 1 \\ &\geq \left\lfloor \frac{k-1}{2} \right\rfloor + 2 \\ &\geq \left\lfloor \frac{k}{2} \right\rfloor + 1. \end{aligned}$$

The last option is a join. In this case, the resulting component is a cycle with $k+1$ edges, and it requires a scenario with $\frac{k+1}{2} - 1$ operations. Noting that k is odd in this case, we have

$$\begin{aligned} \ell &= 1 + \frac{k+1}{2} - 1 \\ &= \left\lfloor \frac{k}{2} \right\rfloor + 1. \end{aligned}$$

For all three types of operations, ℓ has the lower bound we are looking for of $\left\lfloor \frac{k}{2} \right\rfloor + 1$, proving the lemma. □

With this set of optimal operations, and the range for the length of a scenario, we arrive at three recurrences, one for each type of path. Since the cuts and double swaps are only applied to dashed edges, the indices are different according to the type of the path, and the paths obtained after splitting also have different types.

First, consider balanced paths. We assume, without loss of generality, that the first edge (zero-indexed), is from A. That means that the dashed edges are the ones with even indices. When one of them is cut, one of the sub-paths is balanced, while the other begins and ends with solid edges. When a double swap is applied, the remaining path is always the same type as the original one, in this case, balanced. Thus we can write the following recurrence $S_b(k, \ell)$ for the number of scenarios of length ℓ that solve a balanced path with k edges. The base case is an isolated vertex ($k = 0$) which does not need any operations ($\ell = 0$).

$$
\begin{aligned}
S_b(0,0) =& 1 \\
S_b(k,\ell) =& \sum_{i=0}^{\frac{k}{2}-1} \sum_{j=i+1}^{2i} S_b(2i,j) S_s(k-2i-1, \ell-j-1) \binom{\ell-1}{j} + \\
& \sum_{x=0}^{\frac{k}{2}-1} \sum_{z=1}^{\frac{k}{2}-x-1} z^{(z-2)} S_b(k-2z, \ell-z) \binom{\ell-1}{z-1}
\end{aligned}
$$

The dashed paths are processed similarly. Their dashed edges are even indexed, but both sub-paths after a cut are balanced. We then get the recurrence $S_d(k, \ell)$ for the number of scenarios of length ℓ that solve a dashed path with k edges. The base case is a single dashed edge ($k = 1$) which requires a cut ($\ell = 1$).

$$
\begin{aligned}
S_d(1,1) =& 1 \\
S_d(k,\ell) =& \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \sum_{j=i+1}^{2i} S_b(2i,j) S_b(k-2i-1, \ell-j-1) \binom{\ell-1}{j} + \\
& \sum_{x=0}^{\lfloor \frac{k}{2} \rfloor} \sum_{z=1}^{\lfloor \frac{k}{2} \rfloor - x} z^{(z-2)} S_d(k-2z, \ell-z) \binom{\ell-1}{z-1}
\end{aligned}
$$

On the other hand, solid paths have odd-numbered dashed edges, and a cut always results in two solid paths. Furthermore, as we saw before, this type of path has the extra option of joining the ends and making it a $(k + 1)$-cycle, adding another term to the sum. So, the third recurrence $S_s(k, \ell)$ counts the number of scenarios of length ℓ that solve a solid path with k edges. The base case is a single solid edge ($k = 1$) which always requires joining its ends ($\ell = 1$).

$$
\begin{aligned}
S_s(1,1) =& 1 \\
S_s(k,\ell) =& \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor - 1} \sum_{j=i+1}^{2i+1} S_s(2i+1,j) S_s(k-2i, \ell-j-1) \binom{\ell-1}{j} + \\
& \sum_{x=1}^{\lfloor \frac{k}{2} \rfloor} \sum_{z=1}^{\lfloor \frac{k}{2} \rfloor - x} z^{(z-2)} S_s(k-2z, \ell-z) \binom{\ell-1}{z-1} + \\
& \left(\frac{k+1}{2} \right)^{\left(\frac{k+1}{2}-2\right)}
\end{aligned}
$$

Notice how the recurrences have similar structures. We now show how it is possible to use a single recurrence for paths in general. First, we prove that $S_d(k,\ell) = S_s(k,\ell)$, for every k and ℓ.

Lemma 7. *Given two genomes A and B, if $\mathcal{L} = [X_1, X_2, \ldots, X_\ell]$ is an optimal operation scenario from A to B, then $\mathcal{L}' = [-X_\ell, -X_{\ell-1}, \ldots, -X_1]$ is an optimal operation scenario from B to A.*

Proof. Let X_i be an arbitrary operation in $\mathcal{L}$. It is a basic operation, either a cut, a join, or a double swap. In $\mathcal{L}'$, we have the same operations as in $\mathcal{L}$, but negated, and in reverse order.

All we have to prove is that $B - X_\ell - X_{\ell-1} - \ldots - X_i$ is a genome, for every i such that $1 \leq i \leq \ell$. But $B - X_\ell - X_{\ell-1} - \ldots - X_i = A + X_1 + X_2 + \ldots + X_{i-1}$, which is a genome by hypothesis.

From this, we conclude that the operation $-X_i$ is also a basic operation with the same rank as X_i, and it can be applied to $B - X_\ell - X_{\ell-1} - \ldots - X_{i+1}$. Applying the same reasoning for every operation in $\mathcal{L}$ (and $\mathcal{L}'$), we conclude that $\mathcal{L}'$ is an operation scenario, with $w(\mathcal{L}') = w(\mathcal{L}) = d(A,B) = d(B,A)$. □

Lemma 8. *The number of optimal scenarios with ℓ operations for a dashed path with k edges equals the number of such scenarios for a solid path with k edges.*

Proof. Let A and B be two genomes such that the graph $BG(A,B)$ is a dashed path with $2k+1$ vertices, starting and ending with edges from the source genome A.

Let us reverse the direction of the sorting. To sort B into A, we get the graph $BG(B,A)$ that is also a path with $2k+1$ vertices, starting and ending with edges from the target genome, A, so now it is a solid path. From Lemma 7, we know that for every optimal operation scenario $\mathcal{L} = [X_1, X_2, \ldots, X_\ell]$ from A to B, there is an optimal scenario $\mathcal{L}' = [X_\ell, X_{\ell-1}, \ldots, X_1]$ from B to A. Therefore, for every scenario sorting the dashed $2k+1$-path in $BG(A,B)$ with ℓ operations, there is a corresponding scenario sorting the solid $2k+1$-path in $BG(B,A)$, also with ℓ operations. □

Because of Lemma 8, we know that $S_d(k,\ell) = S_s(k,\ell)$, for every k and ℓ. That is, we can compute the recurrence for unbalanced (odd) paths, independent of the genome most represented in them. As we now have a recurrence for odd paths only, and another exclusively for even ones, we can further simplify them into a single recurrence relation for the number $S_p(k,\ell)$ of optimal sorting scenarios of length ℓ for a path with k edges.

$$\begin{aligned} S_p(0,0) =& 1 \\ S_p(k,\ell) =& \sum_{i=0}^{\lceil k/2\rceil-1} \sum_{j=i+1}^{2i} S_p(2i,j) S_p(k-2i-1, \ell-j-1) \binom{\ell-1}{j} + \\ & \sum_{x=0}^{\lceil k/2\rceil-1} \sum_{z=1}^{\lceil k/2\rceil-x-1} z^{(z-2)} S_p(k-2z, \ell-z) \binom{\ell-1}{z-1} \end{aligned}$$

With $S_c(k)$ and $S_p(k,\ell)$, we can count the total number of scenarios between any two genomes, as in the main text.

C Extremal Numbers of Scenarios for Paths of Length k

In this section, we show that although the general solution of the recurrence in the previous section is not likely to have a closed form, the values corresponding to the smallest ($l = \lfloor \frac{k}{2} \rfloor + 1$) and largest ($l = k$) number of operations, respectively, do have a closed form. The former can be expressed as a simple linear combination of perfect powers, while the latter are the so-called zigzag numbers.

We first introduce these numbers and their properties, then state our result.

Definition 9. *Let n be a positive integer and let S_n be the group of permutations of n objects, numbered from 1 to n. A permutation $\pi \in S_n$ is called **alternating** [4] if it alternatively increases and decreases, i.e. $\pi(1) < \pi(2) > \pi(3) < \pi(4) \ldots$ or $\pi(1) > \pi(2) < \pi(3) > \pi(4) \ldots$.*

Definition 10. *Let $k = 2m$ be an even positive integer. Let S_m be the number of alternating permutations on $k = 2m$ elements that are not equivalent under reversal. S_m is called the m-th **secant number** or the m-th **zig number**). These numbers satisfy the property [4]*

$$\sec(x) = \sum_{m=0}^{\infty} S_m \frac{x^{2m}}{(2m)!} = 1 + \frac{1}{2}x^2 + \frac{5}{24}x^4 + \frac{61}{720}x^6 + \ldots.$$

Definition 11. *Let $k = 2m-1$ be an odd positive integer. Let T_m be the number of alternating permutations on $k = 2m-1$ elements that are not equivalent under*

reversal. T_m is called the m-th ***tangent number*** *or the m-th* ***zag number****). These numbers satisfy the property [4]*

$$\tan(x) = \sum_{m=1}^{\infty} T_m \frac{x^{2m-1}}{(2m-1)!} = x + \frac{1}{3}x^3 + \frac{16}{120}x^5 + \frac{272}{5040}x^7 + \dots.$$

Proposition 12. *Let $k = 2m$ be an even positive integer. Then $S_p(k, \lfloor \frac{k}{2} \rfloor + 1) = S_p(2m, m+1) = m^m - 2(m-1)^m$, and $S_p(k,k) = S_p(2m, 2m) = S_m$.*

Proposition 13. *Let $k = 2m-1$ be an odd positive integer. Then $S_p(k, \lfloor \frac{k}{2} \rfloor + 1) = S_p(2m-1, m) = (m-1)^{m-1}$, and $S_p(k,k) = S_p(2m-1, 2m-1) = T_m$.*

Proposition 14. *Let k be a positive integer. Then $S_p(k, \lfloor \frac{k}{2} \rfloor + 1) \leq S_p(k,l) \leq S_p(k,k)$ for $l \in \left[\lfloor \frac{k}{2} \rfloor + 1, k\right]$, so the first (last) value is the minimum (maximum).*

Corollary 15. *Let A and B be two genomes whose breakpoint graph $BG(A,B)$ has p big cycles with lengths $2\ell_1 + 2, \dots, 2\ell_p + 2$, and q big paths with lengths $k_1, \dots, k_q$, with the latter arranged in decreasing order. Then, for any $1 \leq r \leq q$, the number of optimal scenarios transforming A into B is bounded below by*

$$C \sum_{\ell'_1 = \underline{k}_1}^{k_1} \cdots \sum_{\ell'_r = \underline{k}_r}^{k_r} \binom{L + L' + \underline{k}_{r+1} + \ldots + \underline{k}_q}{\ell_1, \ldots, \ell_p, \ell'_1, \ldots, \ell'_r, \underline{k}_{r+1}, \ldots, \underline{k}_q} \prod_{j \leq r} S_p(k_j, \ell'_j) \prod_{j > r} S_p(k_j, \underline{k}_j),$$

and it is bounded above by

$$C \sum_{\ell'_1 = \underline{k}_1}^{k_1} \cdots \sum_{\ell'_r = \underline{k}_r}^{k_r} \binom{L + L' + k_{r+1} + \ldots + k_q}{\ell_1, \ldots, \ell_p, \ell'_1, \ldots, \ell'_r, k_{r+1}, \ldots, k_q} \prod_{j \leq r} S_p(k_j, \ell'_j) \prod_{j > r} S_p(k_j, k_j),$$

where $C := \prod_{i=1}^{p} S_c(2\ell_i + 2)$, $\underline{k}_j := \lfloor k_j/2 \rfloor + 1$ for $1 \leq j \leq q$, $L := \sum_{i=1}^{p} \ell_i$, and $L' := \sum_{i=1}^{r} \ell'_i$.

Using Corollary 15, we can now control the amount of memory required for the computation of the number of scenarios, at the expense of sometimes ending up with upper and lower bounds rather than the exact number. The computation scheme takes as input a memory limit N (we use $N = 10^9$) and works as follows.

First, recognizing that the number of possible values for ℓ'_j is $\lceil k_j/2 \rceil$ for $1 \leq j \leq q$, we can find the largest r for which the product $\lceil k_1/2 \rceil \cdot \lceil k_2/2 \rceil \cdot \ldots \cdot \lceil k_r/2 \rceil$ does not exceed the memory limit N. This index r will be the cutoff point for the formulas in Corollary 15; everything beyond it will be bounded above (using the upper limit for both the multinomial coefficient and the second argument to S_p) and below (using the lower limit for both of them).

Second, we compute both summations in Corollary 15, which have no more than N terms by construction. For numerical accuracy reasons, all the computations are performed in logarithmic space, using the well-known identities

$$\log(xy) = \log(x) + \log(y) \text{ and } \log(x+y) = \log(x) + \log(1 + \exp(\log(y) - \log(x))).$$

This way of computing the summations simultaneously ensures that there is no loss of numerical accuracy (especially when the terms vary by many orders of magnitude) and the results can be represented using floating-point numbers, no matter how large they get (i.e. avoiding overflow). At the end of the computation, the results are converted back to real space and returned.

When the memory limit N is large enough to have $r = q$, the results are equal and represent the exact number of scenarios transforming A into B; otherwise they are upper and lower bounds computed by using at most N memory cells.

D Proofs for Section 4 — Counting Intermediate Genomes for Paths

For paths, we once again have more options than when sorting by DCJ. In order to prove the needed results for paths in $BG(A, C)$, we will label their nodes. Given an alternating path P with m vertices in $BG(A, C)$, we will label its vertices $v_1, v_2, \ldots, v_m$. We say that an edge $\{v_i, v_{i+x}\}$ has *span* x with respect to P, that is, the span of an edge linking two vertices v_i and v_{i+x} of P is the distance between these vertices in P.

Lemma 16. *Let A and C be two genomes over the same genes such that the unique big component of $BG(A, C)$ is the path $v_1, \ldots, v_m$. If B is an intermediate genome such that $BG(B, C)$ has a cycle containing the edge $\{v_i, v_{i+2k}\}$, this cycle has at least one other edge of even span.*

Proof. To form an alternating cycle of length $2l$ in $BG(B, C)$, there are l edges from C and l edges from B. Since B is an intermediate between A and C, the cycle cannot contain any extremities from other components in $BG(A, C)$. The edges from C are fixed, disjoint, and each of them is incident on one even-indexed extremity and one odd-indexed extremity. Therefore, in the cycle, there are l even vertices and l odd vertices.

To complete the cycle, the l edges from B have to cover all $2l$ vertices. The edge $\{v_i, v_{i+2k}\}$ is incident on two vertices of the same parity. The remaining $l-1$ edges have to cover $l-2$ vertices of one parity, and l of the opposite parity. Therefore, by the Pigeonhole principle, at least one edge from B is incident on two vertices of the opposite parity from i. This edge also has even span. □

Lemma 17. *If A and C are two genomes over the same genes and the unique big component of $BG(A, C)$ is a path, then no intermediate genome B between A and C has an adjacency with even span.*

Proof. Let $\mathcal{L} = [X_1, X_2, \ldots, X_\ell]$ be a sorting scenario where the genome $B_j = A + X_1 + X_2 + \ldots + X_j$ has the adjacency $\{v_i, v_{i+2k}\}$. Assume without loss of generality that B_j is the first genome generated by $\mathcal{L}$ that has an even span adjacency. The operation X_j is either a join of v_i and v_{i+2k} or a double swap creating $\{v_i, v_{i+2k}\}$ and another adjacency.

If X_j is a double swap, there are two possibilities for it, illustrated in Figure 1. For both of these options, the double swap merely reverses a part of

the component, and the number of paths and cycles in $BG(B_{j-1}, C)$, where $B_{j-1} = B_j - X_j$, is the same as in $BG(B_j, C)$. Thus, X_j is not a sorting operation, a contradiction.

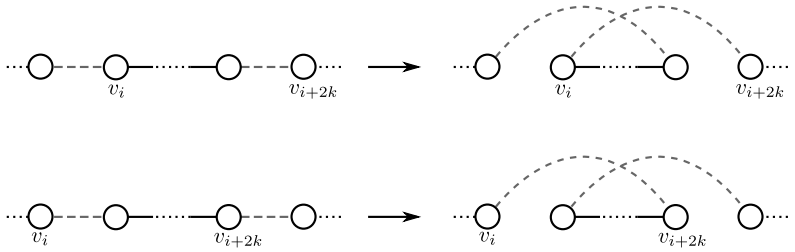

Fig. 1. Two possible configurations for a double swap that creates the adjacency $v_i v_{i+2k}$. The breakpoint graph before the swap is shown on the left, and the breakpoint graph after the swap is shown on the right. The two dashed edges are replaced by $v_i v_{i+2k}$ and another adjacency. In both cases, it is possible to notice that the number of cycles and paths is not affected by the swap, it merely reverses a part of a component.

If X_j is a join, it joins the ends of a solid path P in $BG(B_{j-1}, C)$, closing a cycle with the edge $\{v_i, v_{i+2k}\}$. According to Lemma 16, this cycle has another dashed edge of even span. This edge was already present in P in $BG(B_{j-1}, C)$, which contradicts the hypothesis that B_j is the first genome generated by $\mathcal{L}$ that has an even distance adjacency.

Therefore, intermediate genomes cannot have any even-span edges. □

With the help of Lemma 17, we characterize the intermediate genomes of a path in $BG(A, C)$.

Theorem 18. *If A and C are two genomes over the same set of genes and $BG(A, C)$ has a unique big component that is the k-path $v_1, v_2, \ldots, v_{k+1}$ and small components, a matching B is an intermediate genome between A and C if and only if:*

- *B has the same edges as A and C outside $\{v_1, \ldots, v_{k+1}\}$;*
- *all edges in B involving vertices $v_1, \ldots, v_{k+1}$ have odd span;*
- *the edges in B do not cross one another, that is, there are no two edges $\{v_i, v_j\}$ and $\{v_{i'}, v_{j'}\}$ such that $i < i' < j < j'$;*
- *for every edge $\{v_i, v_j\}$ in B, with $i < j$, all vertices $v_{i+1}, \ldots, v_{j-1}$ are saturated in B (i.e., belong to some adjacency in B).*

Proof. Let us begin by showing that an intermediate genome satisfies the conditions. The first condition ensures that all small components of $BG(A, C)$ are present in both $BG(A, B)$ and $BG(B, C)$. From Lemma 17, we know that only edges with odd span are possible in an intermediate.

At any point during sorting, if an operation X were to add an edge that crosses another one, this would be an operation that recombines two separate components, and hence cannot be optimal. Thus, there cannot be any crossing edges.

As for the fourth and last condition, if an edge $\{v_i, v_j\}$ has span greater than 1, it means that $\{v_i, v_j\}$ was created by a double swap or a join, and all vertices $v_{i+1}, \ldots, v_{j-1}$ are part of a cycle. As we saw in Section 3.2, all vertices in a cycle have to be part of an adjacency in all intermediates.

Now, it is necessary to show that any matching B that fulfills all four conditions is an intermediate between A and C. We will do so using induction on the number of edges of B.

The base case is when B has no edges other than the common edges of A and C. The matching B is an intermediate between A and C, because with a path it is always possible to start at A, reach B by cutting all dashed edges incident to any of $v_1, \ldots, v_m$, and then join at all C-edges incident to $v_1, \ldots, v_m$, optimally arriving at C.

In the general case, B has one or more edges incident to $v_1, \ldots, v_m$. Cut an edge of B with maximum span, obtaining a genome B', with $d(B, B') = 1$. Genome B' satisfies all four conditions and, therefore, by the induction hypothesis, is an intermediate genome between A and C. Note that $d(B, A)$ is either $d(B', A) + 1$ or $d(B', A) - 1$, and the same applies to $d(B, C)$. However, B cannot be closer than B' to both A and C, since this would contradict the triangle inequality. We just have to show that B is not farther than B' from both A and C.

Let $\{v_i, v_j\}$ be the edge of B cut to generate B'. We know that this edge has odd span, and therefore, the edges $\{v_i, v_{i+1}\}$ and $\{v_{j+1}, v_j\}$ are either both dashed edges or both solid edges. Suppose they are dashed edges. In this case, the cut from B to B' cut a cycle in $BG(B, A)$, making $d(B, A) < d(B', A)$, or $d(B, A) = d(B', A) - 1$. On the other hand, $\{v_i, v_j\}$ is part of a path in $BG(B, C)$, and the cut splits this path, making $d(B, C) = d(B', C) + 1$.

In the case where $\{v_i, v_{i+1}\}$ and $\{v_{j+1}, v_j\}$ are solid edges, the same reasoning applies, and we have $d(B, A) = d(B', A) + 1$, and $d(B, C) = d(B', C) - 1$. In both cases, $d(B, A) + d(B, C) = d(B', A) + d(B', C) = d(A, C)$, and that makes B an intermediate. □

The characterization of path intermediates given by Theorem 18 allows us to enumerate all possible intermediates using a recursion. Given a path with length k formed by the vertices $v_1, v_2, \ldots, v_{k+1}$, consider all possible adjacencies incident to v_1. They are $\{v_1, v_2\}, \{v_1, v_4\}$, and so on (pairs involving v_1 and an even-indexed vertex).

There are four different cases. The first case is when v_1 has no adjacency. In this case, the number of intermediates is the number of intermediates for the path that goes from v_2 to v_{k+1}. Therefore, there are $I_p(k-1)$ intermediates.

Then, we consider the adjacency with the smallest span $\{v_1, v_2\}$. There is no vertex "under" the adjacency that needs to be considered, that is, no vertex between v_1 and v_2, so the number of intermediates is $I_p(k-2)$.

Another case is when considering the adjacency with the largest possible span, $\{v_1, v_{2\lceil k/2\rceil}\}$. The vertices $v_2, \ldots, v_{2\lceil k/2\rceil-1}$ are "under" the adjacency, and, according to Theorem 18, the number of intermediates for them is equivalent to the number of intermediates of a cycle that covers $2\lceil k/2\rceil-2$ vertices, $I_c(2\lceil k/2\rceil-2)$. There is at most one vertex after $v_{2\lceil k/2\rceil}$, so no adjacency can exist "outside" $\{v_1, v_{2\lceil k/2\rceil}\}$ and contribute to the number of intermediates. Thus, the total number of intermediates for this case is $I_c(2\lceil k/2\rceil - 2)$.

Finally, for each possible adjacency $\{v_1, v_{2i}\}$, with $1 < i < \lceil k/2\rceil$, the number of intermediates containing this adjacency is the number of intermediates for $v_2, \ldots, v_{2i-1}$ times the number of intermediates for $v_{2i+1}, \ldots, v_{k+1}$. According to Theorem 18, the first factor is equivalent to the number of intermediates of a cycle that covers $v_2, \ldots, v_{2i-1}$. The second factor equals the number of intermediates of the remaining path $v_{2i+1}, \ldots, v_{k+1}$. Therefore, for every adjacency $\{v_1, v_{2i}\}$, $1 < i < \lceil k/2\rceil$, there are $I_c(2i-2)I_p(k-2i)$ intermediates.

The base cases are paths with lengths 0, 1, or 2. A 0-path (a single vertex) means both genomes are equal, so there is only one intermediate. A path with one edge has two intermediates, with and without that adjacency. A path with two edges can have either of the adjacencies, or none of them, adding up to three intermediates.

From this construction, we arrive at the following recurrence $I_p(k)$ for the number of intermediates of a path with length k:

$$
\begin{aligned}
&I_p(0) = 1\\
&I_p(1) = 2\\
&I_p(2) = 3\\
&I_p(k) = I_p(k-1) + I_p(k-2) + I_c(2\lceil k/2\rceil - 2) + \sum_{i=2}^{\lceil k/2\rceil-1} I_c(2i-2)I_p(k-2i)
\end{aligned}
$$

With the help of the On-line Encyclopedia of Integer Sequences , we were able to relate I_p to the sequence A001405 [3] in a way that suggests the closed formula for $I_p(k)$ with $k \geq 0$:

$$I_p(k) = \binom{k+1}{\lfloor (k+1)/2\rfloor}.$$

We will now show that this formula satisfies the recurrence for I_p.

Theorem 19. *For $k \geq 0$,*

$$I_p(k) = \binom{k+1}{\lfloor (k+1)/2\rfloor}.$$

Proof. We are given that $I_p(k) = k+1$ for $1 \leq k \leq 3$ and, for all $k > 3$, we have

$$I_p(k) = I_p(k-1) + I_p(k-2) + I_c(2\lceil k/2\rceil - 2) + \sum_{i=2}^{\lceil k/2\rceil-1} I_c(2i-2)I_p(k-2i),$$

where $I_c(2j) = \frac{1}{j+1}\binom{2j}{j} := C_j$ is the j-th Catalan number.

We define $I_p(-1) := 1$, which allows us to rewrite the recurrence relation as

$$I_p(k) = I_p(k-1) + \sum_{i=1}^{\lceil k/2 \rceil} C_{i-1} I_p(k-2i).$$

We shift indices and define $J_k := I_p(k-1)$, so $J_0 = J_1 = 1$, $J_2 = 2$, and

$$J_{k+1} = J_k + \sum_{i=0}^{\lceil k/2 \rceil - 1} C_i J_{k-2i-1}. \tag{1}$$

In this notation, our goal is to prove that $J_k = \binom{k}{\lfloor k/2 \rfloor}$ for all $k \in \mathbb{N}$. We do this by strong induction on k. We use generating series and Pascal's identity to write

$$j_1(x) := \sum_{k \geq 0} \binom{2k}{k} x^k = \frac{1}{\sqrt{1-4x}}$$

$$c(x) := \sum_{k \geq 0} C_k x^k = \frac{2}{1+\sqrt{1-4x}}$$

$$j_2(x) := \sum_{k \geq 0} \binom{2k+1}{k} x^k = \frac{2}{\sqrt{1-4x}} - \frac{2}{1+\sqrt{1-4x}} = \frac{2}{\sqrt{1-4x}(1+\sqrt{1-4x})}$$

where the first two identities are known from the literature [1] and the last one follows from

$$\binom{2k+1}{k} = \binom{2k}{k} + \binom{2k}{k-1} = \binom{2k}{k} + \left(\binom{2k}{k} - C_k\right) = 2\binom{2k}{k} - C_k. \tag{2}$$

Next we assume that $J_n = \binom{n}{\lfloor n/2 \rfloor}$ for all values up to $n \leq k$, and prove it for $n = k+1$. We look separately at the case where k is even and the case where k is odd. We use the notation $[x^n]f(x)$ to denote the coefficient of x^n in the power series corresponding to $f(x)$.

If k is even, say $k = 2l$, consider the right-hand side of equation (1) and note that, by interpreting the product as a convolution and using identity (2), we get

$$\begin{aligned}
&\sum_{i=0}^{\lceil k/2 \rceil - 1} C_i J_{k-2i-1} = \sum_{i=0}^{l-1} C_i J_{2(l-i)-1} = \\
&= [x^{2l-1}] x c(x^2) j_2(x^2) = [x^{2l}] x^2 c(x^2) j_2(x^2) = \\
&= [x^{2l}] \frac{4x^2}{\sqrt{1-4x^2}(1+\sqrt{1-4x^2})^2} = \\
&= [x^{2l}] \frac{1}{\sqrt{1-4x^2}} - [x^{2l}] \frac{2}{1+\sqrt{1-4x^2}} = \\
&= \binom{2l}{l} - C_l = \binom{2l+1}{l} - \binom{2l}{l}.
\end{aligned}$$

Thus $J_{2l} = \binom{2l}{l}$ implies $J_{2l+1} = \binom{2l+1}{l}$, proving the desired statement for k even.

On the other hand, if k is odd, say $k = 2l + 1$, then, by using the same technique, we get

$$\begin{aligned}
&\sum_{i=0}^{\lceil k/2 \rceil - 1} C_i J_{k-2i-1} = \sum_{i=0}^{l} C_i J_{2(l-i)} = \\
&= [x^{2l}] c(x^2) j_1(x^2) = [x^{2l}] \frac{2}{\sqrt{1-4x^2}(1+\sqrt{1-4x^2})} = \\
&= [x^{2l}] j_2(x^2) = \binom{2l+1}{l} = \\
&= 2\binom{2l+1}{l} - \binom{2l+1}{l} = \binom{2l+2}{l+1} - \binom{2l+1}{l}.
\end{aligned}$$

Thus, $J_{2l+1} = \binom{2l+1}{l}$ implies $J_{2l+2} = \binom{2l+2}{l+1}$, proving the desired statement for k odd.

Since the base case is clearly true for $k = 0$ and $k = 1$, the statement holds in general. □

E Bijection Between Sperner Families and Intermediates of a Path

We present below two algorithms that implement the correspondence between intermediate genomes and subsets of extremities. As mentioned in Section 4, when two genomes differ by just a k-path in their breakpoint graph, the number of intermediates equals the number of extremity subsets with $\lfloor (k+1)/2 \rfloor$ elements.

Algorithm 1: Algorithm to transform a set with $\lfloor (k+1)/2 \rfloor$ distinct elements between 0 and k into an intermediate of a path of length k (0-based).

```
Data: Subset S.
Result: The set E of adjacencies of the corresponding intermediate.
T ← empty stack
E ← {}
for i from 0 to k do
    if i ∈ S then
        T.push(i)
    else
        if T is not empty then
            x ← T.pop()
            E ← E ∪ {{x, i}}
return E
```

Algorithm 2: Algorithm to transform an intermediate of a path of length k (0-based) into a set with $\lfloor (k+1)/2 \rfloor$ distinct elements between 0 and k.

Data: The sets E of adjacencies and T of telomeres of an intermediate.
Result: The corresponding subset of extremities.
$S \leftarrow \{\}$
for $\{x, y\} \in E$ **do**
  $S \leftarrow S \cup \{\min(x, y)\}$
$T' \leftarrow \lfloor (k+1)/2 \rfloor - |S|$ largest telomeres of T
return $S \cup T'$

References

1. Lehmer, D.H.: Interesting series involving the central binomial coefficient. American Mathematical Monthly **92**, 449–457 (1985)
2. Niven, I., Zuckerman, H.S., Montgomery, H.L.: An introduction to the theory of numbers. John Wiley & Sons (2008)
3. Sloane, N.J.A. (ed.): The On-Line Encyclopedia of Integer Sequences. Published electronically at http://oeis.org (accessed on 2017-11-21)
4. Stanley, R.P.: A survey of alternating permutations, Contemporary Mathematics, vol. 531, pp. 165–196. American Mathematical Society (2010)