The Combinatorics of Tandem Duplication

Penso-Dolfin L\textsuperscript{1}, Wu T\textsuperscript{1}, and Greenman CD*\textsuperscript{1,2}

\textsuperscript{1}School of Computing Sciences, University of East Anglia, Norwich, UK, NR4 7TJ.
\textsuperscript{2}The Genome Analysis Center, Norwich Research Park, Norwich, UK, NR4 7UH.

Abstract

Tandem duplication is a rearrangement process whereby a segment of DNA is replicated and proximally inserted. A sequence of these events is termed an evolution. Many different configurations can arise from such evolutions, generating some interesting combinatorial properties. Firstly, new DNA connections arising in an evolution can be algebraically represented with a word producing automaton. The number of words arising from \( n \) tandem duplications can then be recursively derived. Secondly, many distinct evolutions result in the same sequence of words. With the aid of a bi-coloured 2d-tree, a Hasse diagram corresponding to a partially ordered set is constructed, for which the number of linear extensions equates to the number of evolutions generating a given word sequence. Thirdly, we implement some subtree prune and graft operations on this structure to show that the total number of possible evolutions arising from \( n \) tandem duplications is \( \prod_{k=1}^{n} (4^k - (2^k + 1)) \). The space of structures arising from tandem duplication thus grows at a super-exponential rate with leading order term \( O(4^{\frac{1}{2}n^2}) \).

1 Introduction

Tandem duplications occur when a region of DNA is duplicated and inserted adjacent to the original segment, such as portrayed in Figure 1A.

This biological process has long been known to be implicated in the formation of gene clusters [1], [2] and more recently has been implicated in the formation of amplicons in cancer [3], [4], [5], [6]. In both cases Darwinian selection may be acting to increase the number of copies of a target gene. In addition to the biological study of this process, there are a range of algorithmic and mathematical questions that are also of interest. These include identification and alignments of tandem duplications in data [7], [8], [9], [10], [11] and the construction of phylogenies describing their evolution [12], [13], [14]. In [13] this was done in a quite general context, where duplications and losses across multiple genomes were considered. In [14] tree operations were introduced that allowed a full exploration of tandem duplication trees; phylogenetic structures that describe tandem duplication evolution. A survey of algorithmic approaches can be found in [15]. The combinatorial nature of these rearrangement operations leads to some interesting combinatorics. The number of rooted and unrooted tandem duplication trees that arise from the tandem duplication of a loci of interest are explored in [16] and [17]. The space of permutations arising from a tandem duplication-loss model is characterized in [18] and [19].

These methods make a range of assumptions regarding the information that is available and the process that takes place. In particular, there are two issues that relate to the problem we consider.

Firstly, the genomic sequence information that is analyzed differs. In [13], the signed gene orders of several genomes are compared and explanatory phylogenetic evolutions derived. In [16] and [17] a single copy of a loci is analysed, and all the possible different evolutions that can take place counted. In the problem we consider, we also start with a single region of known (reference) sequence, and investigate the number of different possible evolutions that arise. Our approach differs from [16] and [17] with regard to the second issue.

This relates to the assumption that breakpoints can be reused. A breakpoint in this context can mean the gap between two contiguous loci, such as a pair of genes in a gene cluster, which can cover a wide

*Corresponding Author: Email C.Greenman@uea.ac.uk, Tel. +44 (0)1603 592300, Fax +44 (0)1603 593345

© 2015. This manuscript version is made available under the Elsevier user license
http://www.elsevier.com/open-access/userlicense/1.0/
Figure 1: A Tandem Duplication Process. A) Three structures i)-iii) arising from two tandem duplications on a reference of five regions; 1, 2, 3, 4, 5. B) Eleven possible evolutions with two tandem duplications. The example in A is highlighted by *. Underlined numbers are segments. Bold italicized numbers n indicate connections between segments formed in the n<sup>th</sup> tandem duplication.

region and be implicated in more than one duplication event with reasonable probability, or it can mean the precise end points of the duplicated region, which are less likely to be implicated on more than one occasion (for larger scale tandem duplications at least). Modern sequencing (paired-end) data can resolve breakpoints to the basepair level and reveal tandem duplications to great precision, such as with cancer data [3]. In such cases, when a tandem duplication occurs, two breakpoints are implicated in a presumably random process. The chance that precisely the same nucleotide positions are subsequently implicated in another TD is likely to be small and assuming unique breakpoint use is reasonable in these circumstances. The questions considered in this work are restricted to the case of unique breakpoint use. We now outline the main problem we consider.

In Figure 1A we start with five contiguous segments, labeled 1, 2, 3, 4 and 5. This is the original configuration and is termed the reference. The four reference positions between the segments represent breakpoint sites that demarcate where duplicated regions of tandem duplications may start or finish. We then have an initial tandem duplication, copying region 234 and inserting a new copy next to the first, to give sequence 123412345. Here we have used (not underlined, bold symbol) 1 to indicate our first connection between two segments not seen in the reference; the right side of segment 4 is connected to the left side of segment 2, as seen in Figure 1Aii. Note also that the left hand end of the duplicated region 234 implicates the breakpoint between segments 1 and 2, the right hand end implicates the breakpoint between segments 4 and 5. We have thus used two of the four breakpoints available. Next we have the second tandem duplication, copying region 42 to finally give 1234122412345. We now have another connection, labeled 2, between the right side of segment 2 and the left of segment 4, as seen in Figure 1Aiii. Note that we now have two copies of the connection labeled 1, which was also duplicated. The left hand end of the duplicated region represented by subword 412 implicates the reference position between 3 and 4, the right hand end implicates that between 2 and 3. We have thus implicated all four breakpoints between the five reference segments exactly once; unique breakpoint use.

In Figure 1B we see all 11 different ways that two tandem duplications can act on five segments with unique breakpoint reuse. Note that N tandem duplications will implicate 2N breakpoints and so 2N + 1 segments. We are then primarily interested in solving the following problem.

**Problem 1.1.** Count the number of different ways that an initial string of 2N + 1 segments can evolve under N tandem duplications, using each of the N breakpoints once.

To solve this involves a better understanding of the connections we have labeled. If we ignore all the labels representing segments, we get simpler sequences to consider. For example, the sequence [12345 → 123412345 → 1234122412345] becomes the simpler sequence [ε → 1 → 121], where ε denotes the empty word. Although this representation is simpler, it is not unique - five of the eleven cases in Figure 1B contain this sequence of connections. However, we will need to consider these sequences in more detail to solve Problem 1.1.

We then attack the problem as follows. Firstly, we formalize the representations by segments and connections given above. We then explore the size of the space of word sequences involving connection symbols. Each such word sequence will be seen to correspond to many different structures formed by tandem duplications. Thus, thirdly, we consider how to count the distinct cases that all correspond to a single sequence of words containing connection symbols. This involves counting linear extensions of a
suitable partially ordered set (poset). Fourthly, we combine these two pieces of information and provide an explicit formula to answer Problem 1.1. Concluding remarks complete the paper.

2 Representation

We now formalize the representations described in the examples above. We use the acronym TD to denote tandem duplication. In the following $N$ denotes the total number of TDs that take place. Now, we have words that involve two classes of symbols:

**Definition 2.1.** We define $i$ to be a segment symbol. This represents a copy of the DNA segment originally in the $i^{th}$ reference position, where $i \in \{1, \ldots, 2N + 1\}$.

**Definition 2.2.** We define $i$ to be a connection symbol. These symbols always have a segment symbol on either side. In any subword of the form $\min$, $i$ represents a connection between the right hand side of the DNA segment represented by $\m$, and the left hand side of the DNA segment represented by $\n$, formed during the $i^{th}$ TD, where $i \in \{1, \ldots, N\}$.

From the examples in Figure 1B we see that each TD duplicates a region represented by a subword of contiguous segments. We then construct words with the following automaton [20]:

**Definition 2.3.** A TD evolution $U$ is defined as a sequence of TD words $[U_0 \rightarrow U_1 \rightarrow \ldots \rightarrow U_N]$ generated as follows. We initialize the sequence with TD word $U_0 = 1 \cdot 2 \cdot \ldots \cdot (2N + 1)$. We obtain a TD word $U_n$ from $U_{n-1}$ as follows. Write $U_{n-1} = X \cdot Y \cdot Z$ as any product of three non-empty subwords that each begin and end with a segment symbol. Then $U_n = X \cdot Y \cdot n \cdot Y \cdot Z$. The value $n$ is the TD number. We define $n_a$ as the value of the rightmost symbol of $X$ and $n_b$ as the value of the rightmost symbol of $Y$. The values $n_a$ and $n_b$ are referred to as breakpoint numbers. We have a TD evolution provided $(1_a, 1_b, 2_a, 2_b, \ldots, N_a, N_b)$ are $2N$ distinct values that permute $(1, 2, \ldots, 2N)$. We let $U_N$ denote the set of TD evolutions arising from $N$ TDs.

Note that connection symbol $n$ has the last symbol of $Y$ on the left side and the first symbol of $Y$ on the right side. We also know that $Y$ must begin and end with a segment symbol by Definition 2.3, so $n$ is bordered by segment symbols on either side, as required by Definition 2.2.

**Example 2.1.** In the evolution * of Figure 1B we have: $U = [12345 \rightarrow 123412345 \rightarrow 1234122412345]$. The second TD duplicates the region represented by subword $\mathbf{412}$ in the TD word $123412345$. The subwords $X$, $Y$ and $Z$ are then $\mathbf{123}$, $\mathbf{412}$ and $345$, respectively. We then find that $2_a$, the value of the rightmost symbol in $X$, is 3, and $2_b$, the value of the rightmost symbol in $Y$, is 2. Note that these two values demarcate breakpoints of the duplicated region; the $3^{rd}$ breakpoint is implicated by the left end of the duplicated region represented by subword $\mathbf{412}$, between segments 3 and 4, and the $2^{nd}$ breakpoint is implicated by the right end of the duplicated region between segments 2 and 3.

The values $n_a$ and $n_b$ then have the following useful interpretation.

**Definition 2.4.** The reference position between segments $i$ and $i+1$, labeled as $i$, is defined as the $i^{th}$ breakpoint number. We also define breakpoint numbers $0_a = 0$ and $0_b = 2N + 1$ to be the reference positions at the start of segment 1 and the end of segment $2N + 1$, respectively.

**Lemma 2.1.** The left and right ends of the set of duplicated segments in the $n^{th}$ TD are at reference positions $n_a$ and $n_b$, respectively.

**Proof.** From the iteration in Definition 2.3, we note that the segment symbols start in consecutive reference order, and the only disruption occurring to this ordering are to the symbols adjacent to connection symbols. We also know from Definition 2.3 that the set of symbols in the subword $Y$ are duplicated. This implicates two breakpoints. The first is between the last segment represented in $X$ and the first segment represented in $Y$. These two consecutive segments must then be in consecutive reference order. The rightmost segment in $X$ is defined to be $n_a$, and so the left most segment in $Y$ is $n_a + 1$. The breakpoint number between these segments is $n_a$. Similarly, the second breakpoint is between the last segment represented in $Y$ and the first segment represented in $Z$. The rightmost segment in $Y$ is $n_b$. The rightmost breakpoint between these segments is therefore the reference position between segments $n_b$ and $n_b + 1$, which has label $n_b$, as required.
We can now formalize Problem 1.1 with the following equivalent task:

**Problem 2.1.** Determine the size of the set $U_N$.

The following construction of connection evolutions is required to solve this problem.

**Definition 2.5.** Restricting a TD evolution $U = [U_0 \rightarrow U_1 \rightarrow \ldots \rightarrow U_N]$ to connection symbols induces a connection evolution $E = [E_0 \rightarrow E_1 \rightarrow \ldots \rightarrow E_N]$, where each member of the sequence is a connection word. We define $E_N$ to be the set of possible connection evolutions that arises from $N$ TDs.

**Example 2.2.** In the evolution * of Figure 1B we have TD evolution: $U = [12345 \rightarrow 123412345 \rightarrow 1234122412345]$ which becomes connection evolution $E = [1 \rightarrow 1 \rightarrow 21]$ when the segment symbols are removed. Note that across all eleven TD evolutions in Figure 1B, only three corresponding connection evolutions arise: $[\epsilon \rightarrow 1 \rightarrow 21]$, $[\epsilon \rightarrow 1 \rightarrow 12]$ or $[\epsilon \rightarrow 1 \rightarrow 21]$. In general there are less connection evolutions than TD evolutions.

The following observation will prove useful in relating connection evolutions to TD evolutions:

**Lemma 2.2.** Let $E_i$ be any connection word in a connection evolution $E$, induced from a corresponding TD word $U_i$ as described in Definition 2.5. Let $m$ and $n$ be consecutive connection symbols in $E_i$. Then $m$ and $n$ are separated by segment symbols $m_a + 1 \cdot m_a + 2 \cdot \ldots \cdot n_b - 1 \cdot n_b$ in $U_i$ that are in consecutive reference order. If $m$ is the first connection symbol in $E_i$, in $U_i$ it is preceded by segment symbols $1 \cdot 2 \cdot \ldots \cdot m_b$. If $n$ is the last connection symbol in $E_i$, in $U_i$ it is followed by segment symbols $n_a + 1 \cdot n_a + 2 \cdot \ldots \cdot 2N + 1$.

**Proof.** From the iteration in Definition 2.3, we firstly again note that the segment symbols start in consecutive reference order, and the only disruption occurring to this ordering are to the symbols adjacent to connection symbols. More specifically, the word $U_n = X \cdot Y \cdot n \cdot Y \cdot Z$ is formed during the $n^{th}$ TD from $U_{n-1} = X \cdot Y \cdot Z$. The symbol to the left of $n$, the last symbol of $Y$, is segment $n_b$ by definition. The symbol to the right of $n$, is the first symbol of $Y$. Now the last symbol of $X$ is $n_b$ by definition. This is adjacent to $Y$, which is also a segment symbol, which being in consecutive reference order must be $n_a + 1$. We thus find that if we have consecutive symbols $mn$ in a connection word, in the corresponding TD word, we have the segment symbol $m_a + 1$ to the right of $m$ and $n_b$ to the left of $n$. These must run in consecutive reference order giving the segments stated. Noting from Definition 2.3 that the first and last segments are always $1$ and $2N + 1$ completes the Lemma.

This has the following important consequence.

**Corollary 2.1.** Every TD evolution $U$ has a unique signature $(E, \pi(2N))$ where:

i) $E$ is the corresponding connection evolution from Definition 2.5, and

ii) $\pi(2N) = (1_a, 1_b, 2_a, 2_b, \ldots, N_a, N_b)$ is a permutation that characterizes the order of possible breakpoint numbers $(1, 2, \ldots, 2N)$.

**Example 2.3.** If we take the last word of the connection evolution $[\epsilon \rightarrow 1 \rightarrow 21]$ and the breakpoint number permutation $(1_a, 1_b, 2_a, 2_b) = (1, 4, 3, 2)$ then by Lemma 2.2, between connection symbols 1 and 2 we have segment symbols $1_a + 1 \cdot \ldots \cdot 2_b = 2$ in the corresponding TD evolution. Similarly between connection symbols 2 and 1 we have segment symbols $2_a + 1 \cdot \ldots \cdot 1_b = 4$ in the corresponding TD evolution. Before the first copy of connection symbol 1 we have segment symbols $1 \cdot \ldots \cdot 1_b = 1234$, and after the second copy of connection symbol 1 we have segment symbols $1_a + 1 \cdot \ldots \cdot 2_b + 1 = 2435$. For the full TD word we then obtain $1234122412345$. Note that the permutation $(1_a, 1_b, 2_a, 2_b) = (1, 4, 3, 2)$ is a formal way of capturing the ordering $1_a < 2_b < 2_a < 1_b$ of the distinct breakpoints. Note furthermore that not all breakpoint number orderings are possible. For example, if we attempt to use $(1_a, 1_b, 2_a, 2_b) = (3, 4, 1, 2)$, then between connection symbols 1 and 2 we try to place $1_a + 1 \cdot \ldots \cdot 2_b = 4 \cdot \ldots \cdot 2$, which is not an increasing sequence of segment symbols.

**Proof.** (of Corollary 2.1) If we have a given connection evolution $E = [E_1 \rightarrow E_2 \rightarrow \ldots \rightarrow E_N]$ and a permutation $\pi(2N)$ that results in some ordering of breakpoint numbers $(1_a, 1_b, 2_a, 2_b, \ldots, N_a, N_b)$, then for any connection symbol $m$ in any connection word $E_m$ we can use Lemma 2.2 to identify the segment symbols that lie on either side of $m$ in the corresponding TD word $U_m$. Repeating this for all the words in the connection evolution results in a well defined TD evolution. Conversely, any two distinct TD evolutions cannot therefore map to the same signature $(E, \pi(2N))$ and the representation is unique.
Thus to solve Problem 2.1 we need to know firstly what different connection evolutions \( E \in \mathcal{E}_N \) are possible, and secondly, for each such connection evolution, how may different ways we can order breakpoint numbers \( (1_a, 1_b, 2_a, 2_b, ..., N_a, N_b) \). This gives us three problems of increasing complexity; firstly, how to count the number of connection evolutions, secondly how to count the TD evolutions that share a specific connection evolution, and thirdly, how to count the total number of TD evolutions and solve Problem 2.1. We consider these in turn.

3 Counting Connection Evolutions

We next consider how many connection evolutions can arise from \( N \) TDs; the size of the set \( \mathcal{E}_N \). For example, a first TD always produces word 1, a second TD can produce words 12, 21 or 121, giving 3 evolutions in total. Note that two of the final three words have length 2 and one has length 3. In general we have the following result.

**Theorem 3.1.** If \( w_{m,n} \) is the number of connection evolutions \( E = [E_0 \to ... \to E_n] \) arising from \( n \) TDs such that connection word \( E_n \) has length \( m \), we have the following recursion,

\[
w_{m,n} = \sum_{k=\left\lfloor \frac{m}{2} \right\rfloor}^{m-1} (2k - m + 2)w_{k,n-1}
\]

where we have initial values \( w_{i,0} = \begin{cases} 1 & \text{if } i=0 \\ 0 & \text{if } i \geq 1 \end{cases} \).

**Proof.** If we have a connection word \( E_n \) with \( k \) symbols then the subword \( Y \) of any corresponding TD word \( U_n \) that is duplicated (as described in definition 2.3) can be chosen to contain any number of them, so we can duplicate \( r \in \{0, 1, ..., k\} \) of those symbols. Furthermore there are \( k-r+1 \) sets of \( r \) consecutive symbols in \( E_n \) that we can choose to duplicate. Note that a TD duplication copies \( r \) connection symbols and also introduces one new connection symbol, resulting in a connection word with \( m = k + r + 1 \) symbols. Then \( k = m - r - 1 \) for \( r \in \{0, 1, ..., k\} \) and any connection word of length \( m \) can derive from a connection word of length \( k \in \{\left\lfloor \frac{m}{2} \right\rfloor, ..., m - 1\} \). Lastly, we note that there are \( (m-k-1) + 1 = 2k - m + 2 \) ways to do this.

**Example 3.1.** In Figure 2 we see a graph representation of the possibilities, where values \( w_{m,n} \) are equivalently obtained by taking products of the edge values along paths to the associated node, from the node labeled 1, and summing. For example, the node labeled 5 in the third column of nodes corresponds to \( w_{5,3} \) and has two paths, one with product \( 1 \cdot 2 \cdot 1 \), the other with \( 1 \cdot 1 \cdot 3 \) and we find \( w_{5,3} = 2 + 3 = 5 \), five connection words of length five; 12312, 21321, 13121, 12321 and 12131.

It is natural to attempt to find a general formula for the number of words arising from \( n \) TDs by constructing a generating function from this recursion. However, this approach did not prove fruitful suggesting a closed form expression for the connection evolution count is not forthcoming.

The counts of connection evolutions arising from the first few TDs can be seen in Table 1.

4 Counting Evolutions with Posets

We are now interested in the number of TD evolutions that all correspond to a single connection evolution \( E \). From Corollary 2.1 we see that this is equivalent to counting the number of possible orders of the \( 2N \) TDs, suggesting a closed form expression for the connection evolution count is not forthcoming.

![Figure 2: Schematic of number of possible TD words. Numbers at nodes indicate the length of TD words. Numbers on edges indicate the number of choices.](image-url)
breakpoint numbers \((1_a, 1_b, 2_a, 2_b, \ldots, N_a, N_b)\) that arise from TD processes with connection evolution \(E\), as described in Definition 2.3. In order to do this, in section 4.1 we first describe a visual zig-zag representation which is a useful way of seeing the breakpoint numbers accumulate through a TD evolution. In section 4.2 we then use 2d-trees to encapsulate the choices of the different orders of the breakpoint numbers. Finally, in section 4.3 we will use this to construct Hasse diagrams, for which the number of linear extensions will provide the desired count.

### 4.1 Zig-zag plots

Now from Lemma 2.2 we see that if we have consecutive symbols \(mnk\) in a connection word, then in the corresponding TD word, \(m\) and \(n\) are separated by segment symbols \(m_a + 1\) to \(n_b\). By Lemma 2.1, relative to the reference, these segments stretch from breakpoint \(m_a\) to \(n_b\). In a zig-zag plot we represent these as a horizontal (solid) line across the interval \([m_a, n_b]\). Similarly, the segment symbols in the TD word between connection symbols \(n\) and \(k\) are represented by interval \([n_a, k_b]\). The segments represented by the two intervals \([m_a, n_b]\) and \([n_a, k_b]\) are connected together in the TD word by connection symbol \(n\). In the zig-zag diagram this is a (dashed) line from the right end of the line representing interval \([m_a, n_b]\) to the left end of the line representing interval \([n_a, k_b]\). In Figure 3A we have such an example. The ordered breakpoint numbers are arranged along the top. The plots correspond to the connection evolution given in Figure 3E. For example, in Figure 3Ai we have the plot corresponding to connection word \(121\). Each connection symbol is represented by a dashed line connecting the four intervals \([0_a, 1_b], [1_a, 2_b], [2_a, 1_b]\) and \([1_a, 0_b]\) together. In general we have the following construction.

**Definition 4.1.** A zig-zag plot is defined to represent a TD structure as follows. Let \(n^{(1)} \cdot n^{(2)} \cdot \ldots \cdot n^{(K)}\) be a connection word in a TD evolution, along with a corresponding ordering of breakpoint numbers \(n_a^{(k)}\) and \(n_b^{(k)}\), where \(k \in \{1, 2, \ldots, K\}\). Construct an initial interval \([0_a, n_b^{(1)}]\), internal intervals \([n_a^{(1)}, n_b^{(2)}], [n_a^{(2)}, n_b^{(3)}], \ldots, [n_a^{(K-1)}, n_b^{(K)}]\) and a final interval \([n_a^{(K)}, 0_b]\). Each interval \([x, y]\) is plotted horizontally (solid line) from reference coordinate \(x\) to \(y\), and the breakpoint numbers \(x\) and \(y\) are labeled on a horizontal axis. The vertical coordinates of each interval descend down the page in sequence (the scale is not important). The right end of the \(k^{th}\) (\(k \leq K\)) interval is connected to the left end of the \((k + 1)^{th}\) interval in this sequence of intervals with a dashed line, representing connection symbol \(n^{(k)}\).

**Lemma 4.1.** There is a one-one correspondence between a TD evolution and its zig-zag representation.

**Proof.** If we have a TD evolution on \(N\) TDs, then from Corollary 2.1 we have a corresponding signature \((E, \pi(2N))\). The permutation \(\pi(2N)\) provides an ordering of breakpoint numbers, and the word evolution \(E\) provides us with connection words from which the zig-zag representation is constructed as described in Definition 4.1.

Conversely, if we have a zig-zag plot, the endpoints of intervals in Definition 4.1 are labeled on the horizontal axis with corresponding breakpoint numbers, so we obtain an ordering of the breakpoint numbers \(\pi(2N)\) from the horizontal coordinates of the plotted intervals. Each dashed line of the zig-zag representation connects a point with horizontal coordinate \(n_b^{(k)}\) to \(n_a^{(k)}\), from which we can recover the corresponding connection symbol \(n^{(k)}\). Extracting these symbols down the structure (in the order they were plotted) then recovers the connection words. We have thus recovered the signature \((E, \pi(2N))\) which is equivalent to the TD evolution by Corollary 2.1, as required.

Each zig-zag construction is thus just a visual representation of (and equivalent to) a TD word.

### 4.2 2d-trees

We now consider how a 2d-tree representation, which generalizes the notion of trees, can allow us to obtain the different possible orders of the breakpoint numbers. Trees can be characterized as connected graphs
such that each node has a single parental node, apart from a single root node. We can define an nd tree to be a graph such that all nodes (except root nodes) have n parental nodes. This kind of graph has been applied to data forms arising from search algorithms [22], [23] and have seen other applications in genetics as recombination graphs [24], and pedigree graphs [25], for example.

Consider then how the zig-zag plots build up in the structures in Figures 3Ai-v. We start with a single set of reference segments; the single interval \([0_a, 0_b]\) in Figure 3Ai. The node 0\(_a\) is assigned a type, \(a\) (coloured red), indicating it is the left end of an interval. Node 0\(_b\) is assigned a type, \(b\) (coloured blue) indicating a right end of an interval. These labels are associated with the top two nodes of the 2d-tree in Figure 3B. These are bridged by an edge which will represent their ordering in the reference: \(0_a < 0_b\). Now we have four TDs to introduce, resulting in eight breakpoint numbers to be placed between \(0_a = 0\) and \(0_b = 9\).

We then next consider the first TD event. This involves the duplication of a specific single set of contiguous segments (coloured green in Figure 3Ai), and implicates breakpoint numbers 1\(_a\) and 1\(_b\), by Lemma 2.1, resulting in the zig-zag diagram of Figure 3Aii. Now the two positions 1\(_a\) and 1\(_b\) are both bound between the coordinates of 0\(_a\) and 0\(_b\), so the only restriction is \(0_a < 1_a < 1_b < 0_b\). To encapsulate these choices in the 2d-tree representation, we have two nodes representing coordinates 1\(_a\) and 1\(_b\). These nodes both have edges connected to two parental nodes 0\(_a\) and 0\(_b\). We have edges of type \(a\) (red) from node 0\(_b\) to 1\(_a\) and 1\(_b\) representing the fact that 0\(_b\) is a lower bound of 1\(_a\) and 1\(_b\). Similarly we have edges of type \(b\) (blue) from 0\(_b\) to 1\(_a\) and 1\(_b\), representing the fact that 0\(_b\) is an upper bound of 1\(_a\) and 1\(_b\). The black edge is a third class of edge, termed a fence, and connects 1\(_a\) to 1\(_b\), representing the restriction \(1_a < 1_b\).

Our second TD then duplicates the segments in the red portion in Figure 3Aii, which includes the first connection, forming two breakpoints 2\(_a\) and 2\(_b\). The breakpoint 2\(_a\) is on the upper interval \([0_a, 1_b]\) of Figure 3Aii and so must lie between positions 0\(_a\) and 1\(_b\). These are its two parental nodes. The blue edge from
node $1_b$ to $2_a$ indicates $1_b$ is an upper bound of $2_a$. The red edge from $0_a$ to $2_a$ indicates $0_a$ is a lower bound of $2_a$. Similarly, breakpoint $2_b$ forms on interval $[1_a, 0_b]$ and has parental nodes $1_a$ and $0_b$.

The status of major (solid) and minor (dashed) is assigned to each pair of parental edges to a node, where major and minor refer to the parental nodes with higher and lower TD numbers, respectively. For example, $2_a$ has parents $1_b$ and $0_a$, the TD numbers satisfy $1 > 0$, so the edge from $1_b$ is the major, and that from $0_a$ is the minor. This distinction will later be important.

We then proceed through the TDs building up the representations. In general we have the following:

**Definition 4.2.** A 2d-tree is constructed as follows. All nodes $n_a$ are designated type $a$ (coloured red), and $n_b$ designated type $b$ (coloured blue). If the breakpoint represented by $n_a$ (resp. $n_b$) lies on the interval $[u_a,v_b]$ in the zig-zag diagram we have a type $a$ (red) edge from $u_a$ to $n_a$ (resp. $n_b$), and a type $b$ (blue) edge from $v_b$ to $n_a$ (resp. $n_b$).

If $u > v$, the edge from $u_a$ is designated class major (solid), the edge from $v_b$ is class minor (dashed). This is reversed if $u < v$.

If $n_a$ and $n_b$ are formed on the same segment (meaning no connections are duplicated) we have $n_a < n_b$ which we represent with an edge of class fence (black) between nodes representing breakpoint numbers $n_a$ and $n_b$ (this always includes $0_a$ and $0_b$).

Note that the choice of major and minor is ambiguous for the first TD. Both $1_a$ and $1_b$ are placed on the same interval $[0_a,0_b]$ so have parental nodes $0_a$ and $0_b$ that have equal TD-number 0. It will prove consistent to define them as follows: $1_a$ has major (resp. minor) parental nodes $0_b$ (resp. $0_a$), $1_b$ has major (resp. minor) parental nodes $0_a$ (resp. $0_b$). Note that in all other cases either a type $a$ node $n_a$ is placed on $[u_a,v_b]$, where $n > \{u,v\}$ resulting in new interval $[n_a,v_b]$, with $n \neq v$, or a type $b$ node $n_b$ is placed on $[u_a,v_b]$, where $n > \{u,v\}$ resulting in new interval $[u_a,n_b]$, with $n \neq u$. Thus apart from the initial interval, the TD numbers of any interval are distinct and the major/minor is well defined.

Note also that it is only when the fence edges are removed that each node has two parents, each edge connects a parent node to a daughter node, and we have a 2d-tree. Inclusion of fence edges implies structures such as Figure 3B take a more general form than a 2d-tree. For convenience we use the phrase 2d-tree with that understanding in mind.

We introduced fences for the situation where $n_a$ and $n_b$ form on the same segment. This means that the $n^{th}$ TD does not duplicate any connections. In terms of the connection words, this corresponds to a step where no connection symbols are duplicated; no symbol is duplicated in the following step of a connection evolution: $[121 \rightarrow 3121]$, for example.

We have seen that a zig-zag representation is equivalent to the TD evolution $U$ it is based upon, which is in turn equivalent to the signature $(E, \pi(2N))$; the induced connection evolution $E$ and ordering of breakpoint numbers. However, the corresponding 2d-tree does not contain the same information and is not equivalent. For example, if we take the connection evolution $[e \rightarrow 1 \rightarrow 12 \rightarrow 12312]$, the last word contains two copies of the pair $12$, corresponding to two copies of the interval $[1_a,2_b]$ between connection symbols 1 and 2. If both breakpoints $4_a$ and $4_b$ from the next TD are placed in either of these intervals during the fourth TD, the same 2d-tree results, even though the TD evolutions are different.

We next describe how to use the 2d-tree structures to count the number of TD evolutions that correspond to the same connection evolution.

### 4.3 Linear Extensions

Although we counted the number of connection evolutions with relative ease in section 3, there maybe several different TD evolutions that correspond to a single connection evolution. We see in Figure 4A, for example, the three TD evolutions that have a corresponding connection evolution $[e \rightarrow 1 \rightarrow 12]$, where the second connection follows one copy of the first. These three cases have the following explanation in terms of breakpoint ordering. Once the first duplication has occurred, the two breakpoints $2_a$ and $2_b$ associated with the second TD need to be positioned. Now, the first TD requires $1_a < 1_b$ resulting in intervals $[0_a,1_b]$ and $[1_a,0_b]$ (see Figure 3Aii, for example). To obtain the connection word $12$ from connection word 1 we find that we must not copy the first connection, and both $2_a$ and $2_b$ must lie on the second segment $[1_a,0_b]$, so we have $1_a < 2_a < 2_b$. We then find that the three cases depend on whether $1_b$ is less than, in between, or greater than $2_a$ and $2_b$. The three evolutions in Figure 4Ai-iii then correspond to the three orders $1_a < 2_a < 2_b < 1_b$, $1_a < 2_a < 1_b < 2_b$ or $1_a < 1_b < 2_a < 2_b$. 
These distinct orders represent possible breakpoint number orders, subject to the restrictions \( 1_a < 1_b \) from the first TD, and \( 1_a < 2_a < 2_b \) from the second TD. Articulating these restrictions more generally requires the construction of a suitable partially ordered set (poset) [26]. A poset is a set of elements with some order relationships between the elements. Posets are usually represented by a Hasse diagram. This is a directed graph where nodes represent the poset elements, and a directed edge between two nodes indicates an order relation between the two corresponding elements. Any single ordering of the elements that satisfies such a set of restrictions is known as a linear extension. The Hasse diagram for any connection evolution can be readily constructed from the corresponding 2d-tree as follows.

**Lemma 4.2.** If the direction of the type \( b \) (blue) edges are reversed in the 2d-tree, and fences are directed from \( n_a \) to \( n_b \) whenever they occur, a Hasse diagram with single source node \( 0_a \) and single sink node \( 0_b \) is obtained.

**Example 4.1.** For example, in Figure 3B we see the 2d-tree corresponding to the connection evolution given in Figure 3E. In Figure 3C we see the same graph except the blue edge directions have been reversed, and the three fences are directed. Note that all fully extended, directed paths lead from \( 0_a \) to \( 0_b \). Any linear extension, such as \( 0_a < 3_a < 4_a < 1_a < 2_a < 2_b < 4_b < 3_b < 1_b < 0_b \) at the top of Figure 3A, is satisfied by this Hasse diagram.

**Proof.** (of Lemma 4.2) When we add any node \( x \in \{ n_a, n_b \} \) to the 2d-tree, it has two parental nodes \( u_a \) and \( v_b \). By construction, the node \( x \) represents a breakpoint that is placed on the segment \([u_a, v_b] \) with leftmost reference position \( u_a \) and rightmost position \( v_b \), thus we have the ordering \( u_a < x < v_b \) in terms of reference position. Now the type \( a \) edge directed from \( u_a \) to \( x \) represents the ordering \( u_a < x \). We then select direction of the edges in the Hasse diagram to represent increasing reference position. Now \( x < v_b \), so we require a directed edge from \( x \) to \( v_b \), which is obtained by reversing the direction of the type \( b \) edge in the 2d-tree from \( v_b \) to \( x \). Finally we note that if we have a fence, we are adding two position \( n_a \) and \( n_b \) to the same segment. We then have the additional ordering \( n_a < n_b \) which is represented by the addition of a direction from \( n_a \) to \( n_b \).

Counting the number of different TD evolutions associated with a given connection evolution then reduces to counting the number of linear extensions associated with the corresponding poset. Although finding any single linear extension from a poset can be achieved in polynomial time [27], counting the number of linear extensions is known to be \#P-complete [28] and in general is slow to implement [26]. However, for the problem we have, we will show that restricting the Hasse diagram to major edges and fence edges (that is, removing the minor edges) contains all the ordering information. This simplified topology will enable us to obtain a closed form expression for the number of linear extensions.

For any connection evolution \( E \), we will refer to the graph obtained from 2d-tree \( T(E) \) by selecting just the major and fence edges as the major graph, \( T_{maj}(E) \). The following result describes how major and minor status relates to the segments of the form \([u_a, v_b] \) involved in the TD process. It will be used to explain why the subgraph \( T_{maj}(E) \) contains all the ordering information of the Hasse diagram.

**Lemma 4.3.** If \([u_a, v_b] \) is any interval from a zig-zag plot arising in a connection evolution then either:
Figure 5: Major and minor edge structure. A) The addition of new nodes preserves major-minor structure. B) The nesting structure of a branch of a major tree. C) The general major-minor structure.

A) Nodes \(u_a\) and \(v_b\) are connected by a single directed major edge from the node with TD number \(\min(u, v)\) to node with TD number \(\max(u, v)\). The positions satisfy the single linear extension \(u_a < v_b\).

Or:

B) Nodes \(u_a\) and \(v_b\) are connected by a minor directed edge from the node with TD number \(\min(u, v)\) to that with TD number \(\max(u, v)\). Furthermore there exist nodes with TD numbers in the order \(\min(u, v) < n_1 < n_2 < ... < n_I < \max(u, v)\) that are connected in a chain of major edges in the same order such that:

i) If \(u > v\), all internal nodes are type \(a\) (red) and the positions satisfy the single linear extension, 
\[(n_1)_a < (n_2)_a < ... < (n_I)_a < u_a < v_b,\]
ii) If \(u < v\), all internal nodes are type \(b\) (blue) and the positions satisfy the single linear extension, 
\[u_a < v_b < (n_I)_b < ... < (n_2)_b < (n_1)_b.\]

Proof. We prove this by induction. Initially we start with a single interval \([0_a, 0_b]\) and the first TD results in two intervals \([0_a, 1_a]\) and \([1_a, 0_b]\) (such as in Figure 3Aii). Now node 1\(_a\) has major parental node 0\(_a\) and 1\(_b\) has major parental node 0\(_b\). Thus each of these segments has a single major edge connecting the corresponding nodes and so satisfy the conditions of the lemma.

For the induction we next assume that any interval \([u_a, v_b]\) satisfies the conditions of the lemma for all \(u, v < m\). For each segment we thus have either a single major edge connecting nodes \(u_a\) and \(v_b\), or a minor edge connecting them along with a chain of major edges. We then introduce the \(m^{th}\) TD duplicating a region with endpoints \(m_a\) and \(m_b\). We need to check all resulting segments satisfy the Lemma. We have four cases to check.

Case I: The entire interval \([u_a, v_b]\) is duplicated or unmodified; then the poset graph is unchanged between nodes \(u_a\) and \(v_b\) and we have nothing to do.

Case II: The breakpoint \(m_a\) lies in \([u_a, v_b]\). We thus obtain a new interval \([m_a, v_b]\). A new node \(m_a\) then has major and minor parents with TD number \(\max(u, v)\) and \(\min(u, v)\). We then have two possibilities depending on whether \(u\) and \(v\) are connected by a major or minor edge.

Case IIa: If they are connected by a major edge then we see that if \(u < v\) then we have a new major edge from \(m_a \rightarrow v_b\), and interval \([m_a, v_b]\) satisfies criterion A of the Lemma. If \(u > v\), then we have a minor edge \(m_a \rightarrow v_b\) and a chain of two major edges \(v_b \rightarrow u_a \rightarrow m_a\), which satisfy \(u_a < m_a < v_b\), and interval \([m_a, v_b]\) matches criterion Bi of the Lemma.

Case IIb: Now \(u\) and \(v\) are connected by a minor edge, along with a chain of major edges as described in the theorem. Then if \(u < v\) we have a single major edge \(v_b \rightarrow m_a\), and the conditions of the theorem are met. If \(u > v\) we have a single minor edge \(v_b \rightarrow m_a\) and major edge \(u_a \rightarrow m_a\) with order \(u_a < m_a < v_b\). If we combine this condition with the inductive hypothesis of the theorem; \((n_1)_a < (n_2)_a < ... < (n_I)_a < u_a < v_b\), we obtain \((n_1)_a < (n_2)_a < ... < (n_I)_a < u_a < m_a < v_b\), which again has the correct structure.

Case III: If the breakpoint \(m_b\) lies in \([u_a, v_b]\), a parallel set of reasoning to case II applies.

Case IV: If both breakpoints \(m_a\) and \(m_b\) lie in \([u_a, v_b]\), we obtain intervals \([u_a, m_b]\) and \([m_a, v_b]\). These are the same segments as cases II and III and the same arguments apply to both segments.

We now use this result to describe the inheritance nature of major and minor edges.

**Corollary 4.1.** If any node has a major parental node of type \(a\) (resp. \(b\)), its minor parent is the most recent common ancestor (in the major graph) of opposite type \(b\) (resp. \(a\)).
Example 4.2. Consider the branch in Figure 5C. Node $a_3$ has a major type $a$ parental node $a_2$. The most recent type $b$ ancestor of $a_3$ is node $b_6$, which is its minor parent. Node $b_2$ has a major type $b$ parental node $b_4$, we have to go back to node $a_1$ for its most recent type $a$ ancestor, its minor parental node.

Proof. (of Corollary 4.1) Now by Lemma 4.3 any two nodes $u_a$ and $v_b$ bridging an interval $[u_a, v_b]$ are linked by a major or a minor edge. If a new node $x \in \{u_a, v_b\}$ corresponding to a new breakpoint in this interval is formed, $u_a$ and $v_b$ are the major and minor parents, in some order. We have four cases to check:

Case I: $(u < v$, major edge from $u_a$ to $v_b)$. Then $x$ has minor parent $u_a$ and major parent $v_b$. The minor parent $u_a$ is then connected to $x$ by the chain of major edges $u_a \to v_b \to x$. Node $x$ has a major parent of type $b$ and the minor parent $u_a$ is the most recent ancestor of type $a$ in the major graph (see Figure 5Ai).

Case II: $(u > v$, major edge from $v_b$ to $u_a)$. Analogous to Case I; swap $u$ and $v$, and swap $a$ and $b$ in argument (see Figure 5Aii).

Case III: $(u < v$, minor edge from $u_a$ to $v_b$). Then by Lemma 4.3 minor node $u_a$ is connected to major $v_b$ by a chain of major edges of the form $u_a \to (n_1)_b \to (n_2)_b \to \ldots \to (n_j)_b \to v_b$ for some internal nodes of type $b$. Now node $x$ has major parental node $v_b$ so there is also a major edge $v_b \to x$. Together we have the chain of major edges $u_a \to (n_1)_b \to (n_2)_b \to \ldots \to (n_j)_b \to v_b \to x$. We then find $x$ has a major of type $b$ and the minor $u_a$ is the most recent ancestor of type $a$ in the major graph (see Figure 5Aiii).

Case IV: $(u > v$, minor edge from $v_b$ to $u_a)$. Analogous to Case III; swap $u$ and $v$, and swap $a$ and $b$ in argument (see Figure 5Aiv). 

We can now explain the sense in which minor edges can be removed from the Hasse diagram. Specifically, we find that any set of nodes connected by a directed chain of major edges has a single ordering. More precisely:

Corollary 4.2. Consider any single directed chain of major edges connecting nodes $\{a_i, b_j : i = 1, \ldots, I, j = 1, \ldots, J\}$ where $a_i$ are nodes of type $a$ and $b_j$ are nodes of type $b$. Suppose furthermore that these nodes are in some order such that $a_i$ is an ancestor of $a_{i+1}$ for $i = 1, 2, \ldots, I - 1$, and $b_j$ is an ancestor of $b_{j+1}$ for $j = 1, 2, \ldots, J - 1$. These nodes have a single linear extension of the form:

$a_1 < a_2 < \ldots < a_I < b_J < \ldots < b_2 < b_1$.

Thus as we follow any single path down the major tree, the types $a$ and $b$ of the nodes can be intermixed. However, the TD numbers of the $a$ nodes increases down the path, as does the TD numbers of the $b$ nodes. Furthermore, the reference positions of the $a$ nodes increase and $b$ nodes decrease towards each other (see Figure 5B for an example).

Proof. Now consider any sub-chain of nodes connected by major edges of the form $a_1 \to b_1 \to b_2 \to \ldots \to b_n$. Then $b_{i+1}$ has major parent $b_i$ (of type $b$), so $b_{i+1} < b_i$. Also, $b_1$ has major parent $a_1$ (of type $a$) so $a_1 < b_1$. We also know that $b_2, \ldots, b_n$ all have $a_1$ as their minor parent by Corollary 4.1, so $a_1 < b_i$ for $i = 2, 3, \ldots, I$. Together we then have the single order $a_1 < b_n < \ldots < b_2 < b_1$. If the chain then continues as a chain of type $a$ nodes $b_n \to a_1' \to a_2' \to \ldots \to a_m'$, we similarly find that $a_1' < a_2' < \ldots < a_m' < b_n$. However, $a_1'$ has minor parent $a_1$ by Corollary 4.1 so $a_1 < a_1'$. We then find that these two orders combine into the single order $a_1 < a_1' < a_2' < \ldots < a_m' < b_n < \ldots < b_2 < b_1$. Thus we find that as we move down a chain of nodes connected by major edges, the $a$ and $b$ nodes lie in one single nested structure where the $a$ nodes are increasing and the $b$ nodes are decreasing in reference position as we move down the major graph; a single linear extension.

Lemma 4.4. i) Suppose $K$ branches descend from a single node $z$ in the major graph, such that the $k^{th}$ branch contains $m_k$ descendant nodes, and none of the $K$ daughter nodes of $z$ are connected by a fence.

Then the number of linear extensions involving the associated $m + 1$ breakpoints is $\binom{m}{m_1, \ldots, m_K} \prod_k \phi_k$, where $m = \sum_{k=1}^{K} m_k$, and $\phi_k$ is the number of linear extensions associated with the $m_k$ nodes in branch $k$.

ii) Suppose two of the branches descending from a single node $z$ in the major graph contain $m_1$ and $m_2$ descendant nodes, respectively, and the two daughter nodes of $z$ in these branches are connected by a fence.
Then the number of linear extensions involving the associated \( m + 1 \) breakpoints is \((\binom{m}{m_1} - 1)\phi_1\phi_2\), where \( m = m_1 + m_2 \), and \( \phi_1 \) and \( \phi_2 \) are the number of linear extensions associated with the \( m_1 \) and \( m_2 \) nodes in the respective branches.

**Proof.** i) We have \( \phi_k \) linear extensions associated with branch \( k \). If we select one linear extension from each branch, we have, by Corollary 4.2, \( K \) orderings of the form:

\[
(x^{(k)}_{i_1})_a < (x^{(k)}_{i_2})_a < ... < (x^{(k)}_{i_{m_k}})_b < (x^{(k)}_{i_{m_k+1}})_b
\]

Here \( (x^{(k)}_{i_j})_{a/b} \) are the breakpoints represented by the nodes in branch \( k \). Now node \( z \) is the common ancestor of the \( K \) branches and so arises from the earliest TD. Then by Corollary 4.2 either \( z = (x^{(1)}_{i_1})_a = ... = (x^{(K)}_{i_1})_a \) is the left most node and is of type \( a \), or \( z = (x^{(1)}_{i_{m_1+1}})_b = ... = (x^{(K)}_{i_{m_K+1}})_b \) is the right most node and is of type \( b \) (in Figure 5B for example, the red node from the earliest TD is type \( a \) and has the lowest position). Now node \( z \) is fixed in position and common to all \( K \) branches. Any pair of nodes from different branches are unrestricted relative to each other. Any pair of nodes within a branch \( k \) have one relative order from the linear extension selected from the \( \phi_k \) possibilities of that branch. We then need to count the number of ways of intercalating \( m_1 \) nodes from branch 1, with \( m_2 \) nodes from branch 2, through to \( m_K \) nodes from the last branch. There are \( \binom{m}{m_1,m_2,...,m_K} \) ways to do this.

ii) We now consider the case of a fence between two daughter nodes \( n_a \) and \( n_b \) of \( z \), which results in the extra condition \( n_a < n_b \). We have an ordering from each branch. By Corollary 4.2, if \( z \) is of type \( a \) they will take the form:

\[
(z)_a < (x^{(1)}_{i_1})_a < ... < (x^{(1)}_{i_{m_1-1}})_b
\]

\[
(z)_a < (x^{(2)}_{i_1})_a < ... < (x^{(2)}_{i_{m_2-1}})_b < n_b
\]

Here \( (x^{(1)}_{i_j})_{a/b} \) and \( (x^{(2)}_{i_j})_{a/b} \) are the breakpoints represented by the nodes descending from \( n_a \) and \( n_b \), respectively. Now there are \( \binom{m_1}{m_1} \) ways to interlace these two orders. Furthermore, precisely one of these interlacements contradicts the extra condition \( n_a < n_b \), and that is:

\[
(z)_a < (x^{(2)}_{i_1})_a < ... < (x^{(2)}_{i_{m_2-1}})_b < n_b < n_a < (x^{(1)}_{i_1})_a < ... < (x^{(1)}_{i_{m_1-1}})_b
\]

We subtract this single order from the count \( \binom{m}{m_1} \) to give the desired result.

The case where \( z \) is of type \( b \) is similar with the same conclusion.

Finally we put this information together to count the number of linear extensions arising from the 2d-tree.

**Theorem 4.1.** Let the nodes \( 0_a, 0_b \) and daughter edges be removed from the major graph. For each node \( x \) remaining let \( x_1, ..., x_K \) denote the number of nodes that are present in each of \( K \) descending branches. If any pair of daughter nodes are connected by a fence, they contribute a factor \((y_1 + y_2) - 1\), where \( y_1 \) and \( y_2 \) count the number of nodes descending down each branch connected by the fence. These two branches are then treated as a single branch with \( y_1 + y_2 \) daughter nodes. We then associate the number \( m(x) = \left(\begin{array}{c}
x \\
x_1, ..., x_K
\end{array}\right)\) with node \( x \). The number of distinct evolutions is then the product of these terms across nodes and fences.

**Proof.** The TD process starts with interval \([0_a, 0_b]\) which produces two intervals \([0_a, 1_b]\) and \([0_b, 1_a]\) after the first TD. All future segments produced will always have at least one parental node with a TD number greater then 0 so the only major edge from \( 0_a \) leads to \( 1_b \) and the only major edge from \( 0_b \) leads to \( 1_a \). Then \( 0_a \) and \( 0_b \) both have single branches descending. Now, applying Lemma 4.4 to any node with a single descending branch containing \( n \) nodes results in a combinatorial term of the form \( \frac{n!}{n!} = 1 \). The combinatorial factors from \( 0_a \) and \( 0_b \) can thus be ignored. For the remaining nodes we see from Lemma 4.4 that the orders \( \phi_m \) associated with nodes in individual branches are multiplied into the combinatorial terms (such as \( \binom{m_1, ..., m_K}{m} \)) associated with the parental node. We thus multiply the terms of the form \( \binom{m_1, ..., m_K}{m} \) from nodes and \( \binom{m_1, ..., m_K}{m} - 1 \) from fences.
Example 4.3. Consider the connection evolution $E = [\epsilon \rightarrow 1 \rightarrow 121 \rightarrow 3121 \rightarrow 3124121]$ with 2d-tree in Figure 3B. Once $0_0$ and $0_0$ are removed we have two fences corresponding to TD numbers 1 and 3. The restriction to major and fence edges then results in the graph in Figure 3D. The upper fence has two nodes attached to one side and six nodes to the other. This results in a count $(\binom{8}{3}) - 1 = 27$. We note that node 1b has three branches descending; one fenceless branch with two nodes, and two branches bridged by a fence; one and two nodes down each branch. The latter two branches with the fence then have $(\binom{3}{1}) - 1 = 2$ orders and are then treated as a single branch of three nodes. There are then $(\binom{5}{2}) = 10$ ways of interlacing the five positions from the remaining branch with two nodes and amalgamated branch with three nodes. The total number of linear extensions, and so TD-evolutions, associated with connection evolution $E$ is then $27 \cdot 2 \cdot 10 = 540$.

Note that in the proof we saw that a node with a single descending branch containing $n$ nodes results in a combinatorial factor $\frac{n!}{2}$; one. This is true in general and explains why combinatorial terms from nodes with one descending branch were ignored in this example.

We thus now can count both the number of TD words, and the number of distinct evolutions for each word. We next consider how to combine this information and count the total number of evolutions for a given number of TDs.

5 The Size of TD Space

We have seen that a TD evolution can be represented as an automaton on words. Furthermore, the number of TD evolutions represented by any single connection evolution can be obtained from the corresponding major graph using the methods of the previous section. This naturally leads to the problem of determining the total number of TD evolutions. For example, in Figure 4 we see all eleven evolutions that arise from two TDs; three evolutions corresponding to word 12, three corresponding to 21 and five corresponding to 121. The aim of this section is to prove our main discovery:

Theorem 5.1. The number $N_n$ of distinct evolutions arising from $n$ TDs is given by:

$$N_n = \prod_{k=1}^{n} (4^k - (2k + 1))$$

Thus $N_2 = (4^1 - (2(1) + 1)) \cdot (4^2 - (2(2) + 1)) = 11$, in agreement with Figure 4, for example. The first few terms in this series can be seen in the bottom row of Table 1.

5.1 A Motivating Example

Before constructing a proof of Theorem 5.1, we discuss a motivating example. Recall that $\mathcal{E}_n$ is the set of word evolutions on $n$ TDs. Consider the following examples (we drop the first empty word $\epsilon$ for convenience).

$$E = [1 \rightarrow 121 \rightarrow 3121 \rightarrow 3124121]$$
$$E^+ = [2 \rightarrow 232 \rightarrow 4232 \rightarrow [4235232]]$$
$$E' = [1 \rightarrow 12 \rightarrow 12312 \rightarrow 1412312 \rightarrow 1412352312]$$

The first two word evolutions both use four symbols; $E, E^+ \in \mathcal{E}_4$. These only differ in the labeling of TDs; all we have done is increase each symbol in $E$ by 1 to get $E^+$. In $E'$ we have a word evolution involving one more TD; $E' \in \mathcal{E}_5$.

There are two things to note. Firstly, if we delete the symbol 1 in $E'$ we recover evolution $E^+$. That is, conversely, introducing a new first TD event to $E \in \mathcal{E}_4$ results in $E' \in \mathcal{E}_5$. This suggests we can generate TD evolutions in general by the repeated introduction of initial TDs. This leads to the following definition:

Definition 5.1. If a new first TD is introduced to word evolution $E \in \mathcal{E}_n$, the resulting evolution $E' \in \mathcal{E}_{n+1}$ is called an induced evolution.
Secondly, the major graph of $E'$ is given in Figure 3H. Although we can form this directly from the word evolution $E'$ using the 2d-tree construction from the previous section, we note that Figure 3H is a subgraph of the 2d-tree from the original evolution $E$ (Figure 3B). This suggests we can get the major trees of induced evolutions from the 2d-trees of the originating evolutions.

This implies in general that there may be a connection between $E_{n-1}$ and $E_n$, both in terms of word evolutions, and in terms of major graphs. We need to explore both of these links in more detail.

Firstly we observe that for any word evolution there are a range of ways that a new first TD can be introduced. For example, take the trivial TD-Evolution $E = [1]$, and increase the symbols by 1: $E^+ = [2]$. We can introduce a new first TD in three ways: $E' = [1 \rightarrow 12]$, $E' = [1 \rightarrow 21]$ or $E' = [1 \rightarrow 121]$. Note that all three word evolutions reduce back to evolution $E^+ = [2]$ if all copies of connection symbol 1 are deleted. We will show something stronger in general; each single word evolution $E \in \mathcal{E}_{k-1}$ leads to a unique subset of induced evolutions $\mathcal{E} \subset \mathcal{E}_k$.

We will secondly show that all the major graphs for the word evolutions of $\mathcal{E}$ can be obtained from the 2d-tree for $E$. Now for any individual word evolution $E$, we can use the major graph $T_{maj}(E)$ to count the number of associated TD evolutions using Theorem 4.1. We will extend this and show that the number of TD evolutions corresponding to $\mathcal{E}$ is equal to the number of TD evolutions corresponding to $E$ multiplied by a constant factor $4^n - (2n + 1)$. Applying this observation recursively to the spaces $\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_n$ will then be seen to result in Theorem 5.1.

5.2 Induced Evolutions

For induced evolutions to be a useful concept, we must establish that any word evolution $E' \in \mathcal{E}_{n+1}$ can be uniquely represented as an induced evolution from some word evolution $E \in \mathcal{E}_n$.

**Lemma 5.1.** Let $D(E)$ be the process where we remove all copies of TD symbol 1 from word evolution $E$ and reduce each connection symbol in value by 1. This process has the following properties:

i) If $E \in \mathcal{E}_{n+1}$, then $D(E) \in \mathcal{E}_n$ is a valid word evolution.

ii) For any word evolution $E \in \mathcal{E}_n$, there exists a word evolution $E' \in \mathcal{E}_{n+1}$ such that $D(E') = E$.

**Proof.** i) Any connection evolution $E$ starts with trivial connection word 1. The next TD in $E$ results in connection evolution $[1 \rightarrow 12]$, $[1 \rightarrow 21]$ or $[1 \rightarrow 121]$. For all three choices, removing the initial connection symbol 1 from the evolution leaves us the single symbol 2, which becomes 1 when the symbols are reduced in value by 1, thus we obtain the correct initial word for $D(E)$. Now the word evolution is constructed by the TD word automaton as a mapping of the form $AXB \rightarrow AX(n+1)XB$, for possibly empty subwords $A$, $X$ or $B$, for the $(n+1)^{th}$ TD. If we remove all copies of the symbol 1 from the subwords $A$, $X$ and $B$, and reduce all symbols by 1, to give $A'$, $X'$ and $B'$, respectively, we get a mapping of the form $A'X'B' \rightarrow A'X'nX'B'$ which is a valid step in the $n^{th}$ iteration of the TD word automaton, as required.

ii) For any evolution $E = [X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow ... \rightarrow X_n]$ from $\mathcal{E}_n$ we simply construct $E' = [1 \rightarrow 1X'_1 \rightarrow 1X'_2 \rightarrow 1X'_3 \rightarrow ... \rightarrow 1X'_n]$ where word $X'_i$ is obtained from $X_i$ by increasing the value of each symbol by 1. This is a valid word evolution in $\mathcal{E}_{n+1}$. Then applying $D$ to $E'$ recovers $E$, as required.

This allows us to partition the space $\mathcal{E}_n$ as follows:

**Corollary 5.1.** Let $\mathcal{E}(E)$ denote the set of induced evolutions from $E$. Then:

i) For any two evolutions $E_1, E_2 \in \mathcal{E}_n$, the two corresponding sets of induced evolutions do not overlap; $\mathcal{E}(E_1) \cap \mathcal{E}(E_2) = \phi$.

ii) The set of induced evolutions satisfies the relation, $W_{n+1} = \bigcup_{E \in W_n} \mathcal{E}(E)$.

**Proof.** i) We have shown from Lemma 5.1i that deletion of symbol 1 creates a well defined mapping $D : \mathcal{E}_{n+1} \rightarrow \mathcal{E}_n$. Conversely, therefore, we therefore cannot have distinct word evolutions $E_1, E_2 \in \mathcal{E}_n$ that produce the same induced evolution $E'$ when a new first TD is introduced; $\mathcal{E}(E_1)$ and $\mathcal{E}(E_2)$ are thus distinct.

ii) We know from Lemma 5.1i that for any $E \in \mathcal{E}_n$, $\mathcal{E}(E) \in \mathcal{E}_{n+1}$. This implies that $\bigcup_{E \in \mathcal{E}_n} \mathcal{E}(E) \subset \mathcal{E}_{n+1}$.

Conversely, from Lemma 5.1i we know that $\bigcup_{E \in \mathcal{E}_n} \mathcal{E}(E) \supset \mathcal{E}_{n+1}$. \qed
Thus we can generate all of the word evolutions in $E_{n+1}$ as a disjoint union of induced evolutions from $E_n$.

We wish to construct the major graphs $T_{maj}(E')$ of all the induced word evolutions $E' \in E$ from the 2d-tree $T(E)$ of the original evolution. To do this we need to relate the positions of new symbol 1 in the new evolution $E'$ to the nodes of the 2d-tree $T(E)$. In all that follows $X$ represents unspecified subwords in a word evolution. We have the following definition.

**Definition 5.2.** Let $Z = \{1_a, 1_b, 2_a, 2_b, 3_a, 3_b, ..., n_a, n_b \}$ be the node labels for a 2d-tree $T(E^+)$, where $E^+$ is the connection evolution after the TD numbers have been increased by 1 in some connection evolution $E$. For any evolution $E'$ induced from $E$, a 1-nodeset $N \subseteq Z$ is defined as follows:

i) If the word $XmX$ in word evolution $E^+$ becomes word $X1mX$ in induced evolution $E'$, then $m_a \in N$.

ii) If the word $XmX$ in word evolution $E^+$ becomes $Xm1X$ in induced evolution $E'$, then $m_a \in N$.

iii) $1_a, 1_b \in N$

**Example 5.1.** In Figure 3F,G we have evolutions:

$$E^+ = [2 \rightarrow 232 \rightarrow 4232 \rightarrow 4235232]$$
$$E' = [1 \rightarrow 12 \rightarrow 12312 \rightarrow 1412312 \rightarrow 1412352312]$$

Now 2 in $E^+$ becomes 12 in $E'$, so $2_b \in N$. Similarly, $X3X$ becomes $X31X$ so $3_a \in N$ (see italic symbols above). We see $X4X$ becomes $X141X$, the symbol 4 picking up a 1 either side in the induced evolution, so that $4_a, 4_b \in N$. Finally we note that 5 remains isolated from the symbol 1 so $5_a, 5_b \notin N$. Thus $N = \{1_a, 1_b, 2_a, 3_a, 4_a, 4_b\}$.

Now each 1-nodeset is a subset of the node labels for the 2d-tree. We find these sets have the following tree like structure:

**Lemma 5.2.** Let $T(E)$ be the 2d-tree for a connection evolution $E$, and $N$ be the 1-nodeset corresponding to an induced evolution $E'$. Then if $x \in N$,

i) If $x$ is not a root node, its parents are in $N$.

ii) If $x$ is the parental node of a fence, at least one of the daughter nodes must be in $N$.

Conversely, any set of nodes $N$ from $T(E)$ satisfying i) and ii) is a 1-nodeset for some induced evolution $E'$.

Thus the 1-nodesets have the tree like property that for any node belonging to the 1-nodesets, all its ancestors are also present. In particular, the root nodes belong to $N$. Consider the example above; $N = \{1_a, 1_b, 2_a, 3_a, 4_a, 4_b\}$, these are the (solid) nodes in Figure 3H which satisfy these criterion. The two roots $1_a$ and $1_b$ are in $N$. There is a fence between $2_a$ and $2_b$, which have parental nodes $1_a, 1_b$ that are members of node set $N$. At least one of $2_a, 2_b$ must therefore be in $N$, and in this case $2_b$ is.

**Proof.** (of Lemma 5.2) Consider the $n^{th}$ TD in evolution $E$. We have two cases to consider.

**Case I:** The TD is not a fence. Then we have a node $n_a$ with parents $u_a$ and $v_a$, and node $n_b$ with parents $u'_b$ and $v'_b$. We also have step $XuvXu'_b'X \rightarrow XuvXu'_b'X$ in the corresponding connection evolution $E$, where the connections from $v$ to $u'$ (inclusive) are duplicated. Note that subword $XuvX$ represents somatic connections across the region containing the breakpoint $n_a$, and $Xu'_b'X$ similarly covers breakpoint $n_b$.

We consider changes to the parts $XuvX$ and $Xu'_b'X$ of word $XuvXu'_b'X$ when symbol 1 is introduced in the induced evolution separately.

**Case Ia:** If we have word $XuvX$ after symbol 1 has been introduced into $E'$, then by Definition 5.2, $u_a, v_b \notin N$. We then find we have evolution step $XuvX \rightarrow XuvXnXv$ in $E'$ and so symbol $n$ is not adjacent left of symbol 1 and we find that $n_a \notin N$. That is, if the major parent of $n_a$ is not in $N$, $n_a$ cannot be in $N$.

If we have $Xu1vX$ after symbol 1 has been introduced, then by Definition 5.2, $u_a, v_b \notin N$. That is, the parents of $n_a$ are in $N$. Now we have two possibilities. Firstly, we can have evolution step $Xu1vX \rightarrow Xu1vXnXv$ in $E'$, where the connection 1 is not duplicated. In this case we find symbol $n$ is not adjacent to a 1 so by Definition 5.2, $n_a \notin N$. Secondly, we can have evolution step $Xu1vX \rightarrow Xu1vXn1vX$, where the somatic connection 1 is duplicated. In this case we find symbol $n$ is adjacent left of symbol 1 so by Definition 5.2, $n_a \notin N$. Thus if the parents of $n_a$ are in $N$, $n_a$ may or may not be in $N$ depending upon the choice of the induced evolution.
Note that the converse is also true and if the parents of \( n_a \) are in \( N \), we select the evolution step depending on whether \( n_a \) is in \( N \).

**Case Ib:** The argument for node \( n_b \), which depends upon \( Xu'v'X \), is analogous, with the same conclusions.

**Case II:** Consider the case that the \( n^{th} \) TD results in a fence.

In that case we have a step \( Xuuv \rightarrow XuvX \) in \( E \). Then if we have corresponding step \( Xuuv \rightarrow XuvX \) in induced evolution \( E' \) we find that \( u_a, v_b \notin N \) by Definition 5.2 and both \( n_a, n_b \notin N \).

Alternatively we may find that we have a step of the form \( XulvX \rightarrow X \) in \( E' \). Then parental nodes \( u_a, v_b \in N \) and we have three possibilities to consider.

We may have \( XulvX \rightarrow XulvX \). Here the 1 is not duplicated, but we find \( n \) is to the right of a 1 and so \( n_b \in N \).

We may have \( XulvX \rightarrow XulvX \). Here the 1 is not duplicated, but we find \( n \) is to the left of a 1 and so \( n_a \in N \).

Lastly, we may have \( XulvX \rightarrow XulvX \). Here the 1 is duplicated and both \( n_a, n_b \in N \).

Thus when the parent nodes of a fence are in \( N \), at least one of the daughter nodes \( n_a \) or \( n_b \) must be.

Conversely, when the parent node and one or more of \( n_a \) or \( n_b \) are in \( N \), we select the corresponding evolutionary step. □

We can now show how to construct the major graph \( T_{maj}(E') \) from the parental 2d-tree \( T(E) \).

**Lemma 5.3.** For any evolution \( E' \) induced from \( E \), let \( N \) be the corresponding 1-nodeset obtained from \( T(E) \). Let \( T(E) \) be the 2d-tree corresponding to \( E \). The major graph \( T_{maj}(E') \) is constructed as follows.

i) Select all nodes from \( T(E) \) and increase each TD number in the node labels by 1.

ii) If any type \( a \) (resp. \( b \)) node (that is not a root) is a member of \( N \), select the parental edge of the same type, \( a \) (resp. \( b \)).

iii) If any type \( a \) (resp. \( b \)) node (that is not a root) is immediately adjacent (but not in) \( N \), select the parental edge from the opposite type, \( b \) (resp. \( a \)).

iv) If any node \( n_a \) (resp. \( n_b \)) is neither a member of, or immediately adjacent to, \( N \), select the major edge of \( n_a \) from \( T(E) \).

v) If \( n_a \) and \( n_b \) are connected by a fence in \( T(E) \) (for TD number \( n \geq 2 \)), select the fence if and only if \( n_a \notin N \) or \( n_b \notin N \).

vi) Place a fence between 1a and 1b and swap these two node labels.

**Example 5.2.** Consider again the original 2d-tree \( T(E) \) in Figure 3B, where \( E \) is the evolution in Figure 3E.

We wish to construct and the major graph \( T_{maj}(E') \) (of induced evolution \( E' \)) given in Figure 3H by applying the Lemma. We found the 1-nodeset corresponding to evolution \( E' \) previously as \( N = \{1_b, 1_a, 2_b, 3_a, 4_a, 4_b\} \), the black nodes in Figure 3H. Then to construct \( T_{maj}(E') \) we take the nodes of \( T(E) \) and first increase the TD numbers by 1, swap the two labels with TD number 1, and place a fence between them. Node \( 2_a \notin N \) is adjacent to \( N \) so we select the edge from the node of opposite type \( b \) type by Lemma 5.3ii. This was \( 0_b \), which is now mapped to \( 1_a \), so we select edge \( 1_a \rightarrow 2_a \). Node \( 2_b \in N \), so we select the parental edge of same node \( b \). This was also \( 0_b \), so we select the edge from mapped node \( 1_a \rightarrow 2_b \). By Lemma 5.3v, we furthermore select the fence between nodes \( 2_a \) and \( 2_b \). Now \( 3_a \in N \) thus we select the edge from type \( a \) parent, the node \( 1_b \) (mapped from \( 0_a \)). Node \( 3_b \) is not in or adjacent to \( N \) so we select the major edge from \( T(E) \); \( 2_a \rightarrow 3_b \).

Now \( 4_a, 4_b \in N \) so we select the edges from parental nodes of same type; \( 1_b \rightarrow 4_a \) and \( 2_b \rightarrow 4_b \) parental node \( 2_a \). Nodes \( 5_a, 5_b \) are adjacent to \( N \) so we select its parent edges of opposite type; \( 2_b \rightarrow 5_a \) and \( 3_a \rightarrow 5_b \).

Observe that the differences between \( T(E) \) and \( T_{maj}(E') \) are a form of subtree prune and graft operations [29]; when the major edge is swapped for the minor edge we are pruning from the major parental node and grafting to the minor parental node.

**Proof. (of Lemma 5.3)**

i) All the breakpoints from evolution \( E \) remain in evolution \( E' \) so we inherit the representative breakpoints. The introduction of a new first TD increases each TD number by 1.

ii) Consider the case that we have a type \( a \) node \( n_a \in N \). Then we have evolution step \( XuuvX \rightarrow XuvXnXvX \) in \( E^+ \), and \( n_a \) has major and minor parents \( u_a \) and \( v_b \) in \( T(E) \) (in some order, depending upon whether \( u > v \)). This evolution step becomes \( XuuvX \rightarrow XuuvXn1vX \) in \( E' \). Then the connection symbol
1 is duplicated and breakpoint \( n_a \) occurs between connections \( u \) and 1. The major and minor parents are then \( u_a, 1_b \) in \( T(E') \). Now \( u > 1 \), so the major parent of \( n_a \) in \( T(E) \) is the node \( u_a \). Thus if we have a type \( a \) node \( n_a \in N \), we select the parental edge of the same type; \( u_a \rightarrow n_a \), irrespective of whether it was the major or minor in \( T(E) \). The argument for \( n_b \) is analogous.

iii) Consider the case that \( n_a \) is adjacent to a node in \( N \), that is, its major and minor parents are in \( N \). Then we have evolution step \( XuvX \rightarrow Xu\nu XnvX \) in \( E \) that becomes \( Xu1vX \rightarrow Xu1\nu XnvX \) in \( E' \). This time, in the induced evolution, the major and minor parents of \( n_a \) are \( 1_a, v_b \). Now \( v > 1 \) so the major parent of \( n_a \) in \( E' \) is node \( v_b \). Thus if we have a type \( a \) node \( n_a \in N \), we select the parental edge of the opposite type; \( v_b \rightarrow n_a \), irrespective of whether it was the major or minor in the original evolution \( E \). The argument for \( n_b \) is analogous.

iv) Consider the case that \( n_a \) is neither adjacent to a node in, or a member of, \( N \). Then we have evolution step \( XuvX \rightarrow Xu\nu XnvX \) in \( E \) that becomes \( XuvX \rightarrow Xu\nu XnvX \) in \( E' \). Now the major/minor status of \( n_a \) does not change from the original. The argument for \( n_b \) is analogous.

v) If \( n_a \) and \( n_b \) are connected by a fence we have a step of the form \( XuvX \rightarrow Xu\nu X \) in \( E \). The corresponding step in the induced evolution \( E' \) takes one of four forms. Firstly, if \( Xu1vX \rightarrow Xu\nu X \) \( b \) is adjacent to \( 1 \) on both sides, so \( n_a, n_b \in N \). Note that the \( n \)th TD has duplicated symbol 1, so we do not have a fence in \( T_{maj}(E') \). Secondly, if we have \( Xu1vX \rightarrow Xu\nu X \) in \( E' \), then \( n_a \notin N \) and \( n_b \notin N \). Note that \( n \) has not duplicated the symbol 1 and we still have a fence. Thirdly, the evolution \( Xu1vX \rightarrow Xu\nu X \) in \( E' \) similarly preserves the fence, with \( n_a \neq N \) and \( n_b \notin N \). Finally, if we have \( XuvX \rightarrow Xu\nu X \) in \( E' \), the fence is preserved and \( n_a, n_b \notin N \). Thus \( T_{maj}(E') \) contains the fence if and only if at least one of \( n_a \notin N \) or \( n_b \notin N \) is true.

vi) Firstly note that the initial TD in any evolution must occur on the single reference segment, and so must be fence because there are no prior TDs to duplicate, thus we place a fence between nodes \( 1_a \) and \( 1_b \).

Consider the \( n \)th TD for some \( n \geq 2 \). Note that the only way that node \( n_a \) or \( n_b \) can have a parental node \( 0_a \) or \( 0_b \) is to have a step of the form \( X \rightarrow nX \) or \( X \rightarrow Xn \) in connection evolution \( E \). Consider first the step \( X \rightarrow nX \). Note that \( n \) must represent a fence because there are no symbols to the left of \( n \) which could have been duplicated. Then \( n_a \) and \( n_b \) have minor parents \( 0_a \) and some major parent \( v_b \). The induced evolution can then be in one of three forms.

Firstly, we have a corresponding step \( 1X \rightarrow n1X \) in \( E' \). In this case, from Definition 5.2 we find \( n_a \) is in \( N \) and so is connected to its type \( a \) parent \( 0_a \) by ii) above. Now because connection symbol \( n \) corresponds to a fence, \( n_b \) has the same parents as \( n_a \), so is adjacent to \( N \) in \( T(E) \). Then using iii) above we find \( n_b \) is connected to its type \( a \) parent, also \( 0_a \). However, constructing the major tree directly from the 2d-tree corresponding to connection evolution \( E' \), we find that connection symbol \( n \) represents a fence with major parent \( 1_b \). Thus to get an equivalent form from the original 2d-tree, we map \( 0_a \) to \( 1_b \).

The case for \( 1X \rightarrow 1nX \) is similar, resulting in a map from \( 0_b \) to \( 1_a \). For the third choice, the step becomes \( 1X \rightarrow 1n1X \) in \( E' \). We then find that \( n_a, n_b \in N \) by Definition 5.2 as \( n \) is adjacent to \( 1 \) on both sides. Thus in the major graph for \( E' \), \( n_a \) is connected to its type \( a \) parent \( 0_a \) and \( n_b \) is connected to its \( b \) parent \( u_b \). However, direct from \( E' \) we see that \( n_a \) has major parent \( 1_b \) and \( n_b \) has major parent \( u_b \), so again we map \( 0_b \) to \( 1_b \) for a consistent correspondence.

The argument using step \( X \rightarrow Xn \) and node \( n_a \) from \( E \) is entirely similar with parallel conclusions.

In summary, we now know that for any connection evolution \( E \in \mathcal{E}_{n-1} \) there is a unique subset of induced evolutions \( \mathcal{E}(E) \in \mathcal{E}_n \), each member \( E' \) of which corresponds to a 1-nodeset from the 2d-tree of \( E \). We can now use this to produce the major graph \( T(E') \) for the induced evolution using Lemma 5.3. We can then calculate the number of TD evolutions associated with each \( E' \) from Theorem 4.1. We thus need to sum the TD-Evolution counts across the set of 1-nodesets corresponding to \( \mathcal{E}(E) \). Whilst this is possible, leading to \( 4^n - (2n + 1) \) induced TD evolutions for each connection evolution \( E \), the proof relies on a more general space of graphs than we have considered so far, which we now introduce.

### 5.3 \( \beta \)-trees

Firstly we generalize the notion of the 2d-tree obtained from TDs.

**Definition 5.3.** A \( \beta \)-tree \( T \) is any directed graph such that:

i) All nodes and edges are classified as either type \( a \) or type \( b \)

ii) There is a root node \( (A) \) of type \( a \) and a root node \( (B) \) of type \( b \), and all directed edges point away from the roots.
iii) All other nodes have a type a parental node and a type b parental node. The two edges from the parental nodes are also of type a and b, respectively. Either the two parents are the two roots, or one parent is a descendant of the other. The edge from the more recent ancestor is the major, the other is the minor.

iv) A type a node and type b node may be linked by a fence if they have the same parental nodes, or are the two roots.

Thus the 2d-trees defined from TDs are β-trees, for example. Note that β-trees are more general; take Figure 6A,B, for example, they do not have an even number of nodes and cannot arise from a TD process, but satisfy the requirements of a β-tree.

Similar to the 2d-tree construction, the major graph \( T_{\text{maj}}(T) \) of a β-tree \( T \) is the graph obtained when the minor edges are removed.

Secondly, we generalize the notion of 1-nodesets.

**Definition 5.4.** A β-subtree \( \tau \) of a β-tree \( T \) is a subset of nodes from \( T \) such that:

i) The two root nodes are in \( \tau \).

ii) If a node in \( \tau \) is the parent of a fence, one of the two daughter nodes bridged by the fence must also be in \( \tau \).

iii) If a node is in \( \tau \), both parental nodes are also in \( \tau \).

**Example 5.3.** Consider Figure 6A,B. Here the original β-trees are in Figures 6A,Bi. The β-subtrees are indicated in Figure 6A,B iii-iv by the solid nodes. Note that the two roots are always in \( \tau \). These are the parents of the fence \( f \) in Figure 6B and so in agreement with Definition 5.4ii we find that at least one of the two nodes \( a_1 \) and \( b_1 \) bridged by \( f \) lies in \( \tau \).

The β-subtree \( \tau \) of a β-tree \( T \) can be used to define a modified major graph, analogously to the construction of \( T_{\text{maj}}(E') \) in Lemma 5.3, as follows:

**Definition 5.5.** For a β-tree \( T \) and β-subtree \( \tau \), the induced tree \( T(\tau) \) is the major graph obtained from \( T \) by the following operations:

i) Select all nodes from \( T \).
ii) For any node (that is not a root) in $\tau$ of type $a$ (resp. $b$), select the parental edge of same type $a$ (resp. type $b$).

iii) If any node (that is not a root) of type $a$ (resp. $b$) is immediately adjacent (but not in) $\tau$, select the parental edge of opposite type $b$ (resp. type $a$).

iv) If any node is neither a member of, or immediately adjacent to, $\tau$, select the major parental edge from $T$.

v) If two nodes are connected by a fence in $T$, select the fence if and only if one or both nodes are not in $\tau$.

Note that this definition differs from Lemma 5.3 in one important way. By Definition 5.4, any $\beta$-subtree contains both root nodes. By Definition 5.5v, any fence between the two root nodes is not selected in $T(\tau)$. However, we find by Lemma 5.3ii that $T_{maj}(E')$ will contain a fence between the two roots. We will later see that this difference has an important implication for the calculation of the total number of possible TD evolutions.

In order to introduce the main property of $\beta$-trees that will allow us to count TD evolutions we need to introduce some notation.

**Terminology**

- For any 2d-tree $T$ with major graph $T(\tau)$ corresponding to $\beta$-subtree $\tau$:
  - $T(\tau)$ is the graph obtained when the two root nodes of $T(\tau)$ are contracted together.
  - $C(\tau)$ is the product of combinatorial coefficients across nodes and fences of $T(\tau)$ given by Theorem 4.1.

- $S$ denotes the set of valid $\beta$-subtrees according to Definition 5.4.

- $\epsilon$ denotes the trivial $\beta$-subtree containing just the two root nodes.

- For any node or fence $x$ in $T$:
  - $N_x(\tau)$ is the number of nodes from $x$ and its descendants in $T$, attached to the $A$ root in $T(\tau)$.
  - $\tau_x$ denotes the restriction of $\tau$ to $x$ and its descendants.
  - $S_x$ denotes the set of possible subsets $\tau_x$.
  - $C_x(\tau)$ denotes the product of factors of $C(\tau)$ arising from $x$ and its descendants.
  - $c_x(\tau)$ denotes the single factor associated with node (or fence) $x$.
  - Terms with an underline added, such as $c_x(\tau)$, are the corresponding terms using $T(\tau)$ instead of $T(\tau)$.
  - When terms, such as $c_x(\tau)$ (and $C_x(\tau)$), only depend upon $x$ (and its descendants) in $T(\tau)$, we equivalently use notation $c_x(\tau_x)$ (and $C_x(\tau_x)$).

**Example 5.4.** In Figure 6Bii we have a $\beta$-tree. The graphs in each of Figures 6Biv-vi are the possible major graphs $T(\tau)$, where each $\beta$-subtree $\tau$ can be identified from the solid nodes. The counts $N_A(\tau)$ can be seen above the $A$ node for each graph. Node $b_1$ in the $\beta$-tree in Figure 6Bii has two daughter branches with one node each, so we write $c_{b_1}(\epsilon) = \binom{2}{1} = 2$. We trivially have $c_{b_2}(\epsilon) = c_{b_3}(\epsilon) = 1$ for leaf nodes $b_2$ and $b_3$; the descendants of $b_1$, so can also write $C_{b_1}(\epsilon) = c_{b_1}(\epsilon) \cdot c_{b_2}(\epsilon) \cdot c_{b_3}(\epsilon) = 2$.

We are finally in a position to describe the following fundamental result, which will allow us to determine the total number of TD evolutions.

**Theorem 5.2.** Let $T$ be any $\beta$-tree with $N$ nodes, and $\mathcal{S}$ the corresponding set of $\beta$-subtrees. Then for any $r \in \{1, 2, \ldots, N - 1\}$ we have:

$$\sum_{\tau \in S : N_A(\tau) = r} C(\tau) = C(\epsilon)$$  \hspace{1cm} (1)
Example 5.5. We see in Figure 6Bii the major graph $T(\epsilon)$, where the two root nodes have been contracted together. There are two non-trivial combinatorial terms. One from the fence $f$ below the root, which has two nodes descending the left side, three the right, resulting in combinatorial coefficient $c_f(\epsilon) = \binom{5}{2} = 10$, by Theorem 4.1. The other term comes from the daughter node $b_1$ to the right of the root, which has two daughter branches with one node each, resulting in combinatorial term $c_{b_1}(\epsilon) = \binom{3}{2} = 3$. All other nodes give coefficient 1. Together we get the factor given below the graph. Now there are two major graphs $T(\tau_1)$, $T(\tau_2)$ for which $N_A(\tau_1) = N_A(\tau_2) = 5$: the first graph in Figure 6Biv, which has combinatorial term $C(\tau_1) = 6$, and the fifth graph in Figure 6Bvi, which has combinatorial term $C(\tau_2) = 12$. These add up to the same value 18 we found for the graph with contracted roots, agreeing with Equation (1) for $r = 5$.

Proof. (of Theorem 5.2) We prove this by induction on the number of nodes in the $\beta$-tree.

Induction Initial Case

Firstly consider $\beta$-trees with $N = 2$ nodes in total. There are two root nodes $A$ and $B$ of type $a$ and $b$, respectively. There are only two possible $\beta$-trees depending on whether $A$ and $B$ are linked by a fence. Now any subtree from $S$ must contain the two roots by Definition 5.4i, so there is only the one subtree $\epsilon = \{A, B\}$ to consider. If there is a fence between $A$ and $B$ then by Definition 5.5v, the fence does not belong to $T(\epsilon)$. Thus for both cases $T(\epsilon)$ contains both root nodes and no edges. Now the only value that $r = 1,\ldots,N - 1 = 1$ can take is 1 and $N_A(\tau) = r = 1$: the number of nodes attached to node $A$ is 1. Now from Theorem 4.1, the combinatorial term associated with each node $A$ and $B$ is 1. Furthermore, the contracted tree $T(\epsilon)$ is a single node, which similarly has a combinatorial term of 1. We then find that:

$$\sum_{\tau \in S, N_A(\tau) = r} C(\tau) = 1.1 = 1 = C(\epsilon)$$

The result is therefore correct for $\beta$-trees with $N = 2$ nodes.

Inductive Assumption

Next we make the inductive hypothesis that the theorem is true for all $\beta$-trees with $N \leq K - 1$ nodes, for some $K > 2$. We now consider a $\beta$-tree $T$ with $N = K$ nodes. We have two root nodes, $A$ and $B$. The daughter nodes from these two roots may be either type $a$ nodes from root node $B$, type $b$ nodes from root node $A$, or fences descending from both nodes, as portrayed in Figure 7Ai.

Note also that although the original $\beta$-tree $T$ can have type $a$ nodes with major parent $A$, or type $b$ nodes with major parent $B$, they can effectively be assumed to have opposite parentage. More specifically, if we have a daughter node $x_\alpha$ of type $a$ descending from either root node, then when any $\beta$-subtree $\tau$ not including $x_\alpha$ (such as $\epsilon$), we use $x_\alpha$ is attached to $B$ in $T(\tau)$ by Definition 5.5iii. Also, for any $\beta$-subtree $\tau$ including $x_\alpha$ (such as the entire nodeset from $T$), we find $x_\alpha$ is attached to $A$ in $T(\tau)$ by Definition 5.5ii. When the two roots are contracted in $T(\epsilon)$ we find node $x_\alpha$ attached to the single root. Thus the choice of which root to use as the major parent of $x_\alpha$ has no affect on the validity of Equation (1) and we take the root $B$ as stated. The argument for daughter node $x_b$ is similar. This is equivalent to assuming $T = T(\epsilon)$.

Although there may be any number of these type of branches descending from the roots, for the sake a simpler exposition we provide the proof just for the four branches drawn in Figure 7Ai. The generalization is relatively straightforward (see comment at end of proof).

Now we require a sum over $\beta$-subtrees $\tau$ such that $N_A(\tau) = r$. That is, we require $r$ nodes in the component of $T(\tau)$ attached to root node $A$. We suppose that in the original tree $T$ (see Figure 7Ai) there are $n_a$, $n_b$ and $n_f$ nodes contained in each of the two branches containing nodes $a$ and $b$, and the two branches containing the fence $f$, respectively, where $n_f = n_a' + n_b'$. We suppose that there are subsequently $r_a$, $r_b$ and $r_f = r_a' + r_b'$ of these nodes attached to $A$ in major tree $T(\tau)$, such as in Figure 7Aii. The count $r$ includes node $A$ so we require $r_a + r_b + r_f = r - 1$.

Now $C(\tau)$ is a product of terms across the nodes and fences, which can be split into $A$, $B$, the nodes in the two branches containing $a$ and $b$, and the two branches bridged by fence $f$. Recalling the terminology introduced above, we can split the combinatorial term for major tree $T(\tau)$ as:

$$C(\tau) = c_A(\tau) \cdot c_B(\tau) \cdot C_a(\tau_a) \cdot C_b(\tau_b) \cdot C_f(\{\tau_a', \tau_b'\})$$

20
Note that $\tau_a$, $\tau_b$, $\tau_a'$ and $\tau_{b'}$ are the subsets of $\tau$ the include nodes $a$, $b$, $a'$ and $b'$ and their descendants, respectively. Thus when we have trivial $\beta$-subtree $\tau = \varepsilon$, we find these subsets are empty; $\tau_a = \tau_b = \tau_a' = \tau_{b'} = \phi$.

The left hand side of Equation (1) can then be split into sums across the four branches as follows:

$$\sum_{\{\tau \in S: \frac{r_a + r_b + r_f}{r_a - 1}\} = n_a} C(\tau) = \sum_{\{\tau \in S: \frac{r_a + r_b + r_f}{r_a} = n_a\} = \tau_a} c_A(\tau) \cdot \sum_{\{\tau_b \in S_b: \frac{r_b}{r_b} = n_b\} = \tau_b} C_b(\tau_b) \cdot \sum_{\{\tau_{b'} \in S_{b'}: \frac{r_{b'}}{r_{b'}} = n_{b'}\} \sim (\tau_a, \tau_{b'}) \neq \phi} C_f(\{\tau_a', \tau_{b'}\}) \quad (2)$$

Now the sum is restricted to $\beta$-subtrees with $r - 1$ nodes in the branches descending from $A$. We have three branches from node $A$ with node counts $r_a$, $r_b$ and $r_f = r_a' + r_{b'}$, where the two branches containing $a'$ and $b'$ are treated as a single branch in accordance with Theorem 4.1. Thus we find that we associate node $A$ with the multinomial coefficient:

$$c_A(\tau) = \binom{r - 1}{r_a, r_b, r_f} \quad (3)$$

We similarly find we have:

$$c_B(\tau) = \binom{K - r - 1}{n_a - r_a, n_b - r_b, n_f - r_f} \quad (4)$$

We thus have expressions for two terms in Equation (2). To calculate the remaining terms we show that each branch corresponds to a smaller $\beta$-tree ($< K - 1$ nodes) which will enable us to use the inductive hypothesis. We have three cases to consider.

**Case I: Dealing with Type a Branches**

Instead of the full $\beta$-tree $T$ (represented in Figure 7Ai), consider the $\beta$-tree in Figure 7Bi which we obtain by removing all branches except the branch containing $a$, removing edge $B \rightarrow a$ and contracting nodes $A$ to $a$ together. We call the resulting $\beta$-tree $T'$. The corresponding major graphs for $T$ and $T'$ are represented in Figures 7A,Bii.

Now every $\beta$-subtree $\tau'$ of $T'$ can be written as $\tau' = \{\tau_a, B\}$ for some $\tau_a \in S_a$. This correspondence applies for every $\tau_a \neq \phi$. For this single case, $\tau_a = \phi$ we find the root $a$ for $T'$ is missing from $\{\tau, B\}$, and we do not have a valid $\beta$-subtree of $T'$. We thus treat the two cases of $\tau_a = \phi$ and $\tau_a \neq \phi$ separately.

**Case Ia ($\tau_a = \phi$)**. Now the major branch of 2d-tree $T$ containing $a$ is unmodified in $T(\tau)$. Thus all $n_a$ nodes in the branch containing the $a$ node are in one component connected to the root $B$ node, and so $N_a(\tau_a) = r_a = 0$. Now for any $\tau_a \neq \phi$, node $a$ is attached to the root $A$ in $T(\tau)$ by Definition 5.5ii, so $r_a > 0$. Thus the only case with $r_a = 0$ is $\tau_a = \phi$. For this case we note that $C_a(\tau_a) = C_a(\varepsilon)$, and the following equation holds true.

$$\sum_{\{\tau_a \in S_a: \frac{N_a(\tau_a)}{N_a(\tau_a)} = r_a\} = \tau_a} C_a(\tau_a) = C_a(\varepsilon) \quad (5)$$

**Case Ib ($\tau_a \neq \phi$)** We next verify Equation (5) for values $r_a \neq 0$.

Now for $\tau_a \neq \phi$ we have well defined $\beta$-subtrees $\tau' = \{\tau_a, B\}$. Furthermore, the descendants of root $a$ in the major graph $T'(\tau')$ match the descendants of node $a$ in major graph $T(\tau)$, and we find that the combinatorial term associated to tree $T'(\tau')$ will be precisely $C_a(\tau_a)$. Noting that $T'$ has at least one less node than $T$, we can apply the inductive hypothesis using Equation (1) to $T'$ and hence derive Equation (5) for the remaining cases where $r_a > 0$.

**Case II: Dealing with Type b Branches**

By a symmetric argument on the branch with node $b$ we obtain an analogous equation of the form:

$$\sum_{\{\tau_b \in S_b: \frac{N_b(\tau_b)}{N_b(\tau_b)} = r_b\} = \tau_b} C_b(\tau_b) = C_b(\varepsilon) \quad (6)$$
Case III: Dealing with Daughter Fences

We are interested in the remaining combinatorial term $C_f(\{\tau_{a'}, \tau_{b'}\})$ from Equation (2) that we have yet to examine. This corresponds to the two branches containing nodes $a'$ and $b'$, and the fence $f$ between them. We thus define $\beta$-tree $T'$ as the restriction of $T$ to these two branches (see Figure 7Ci). If we also remove the fence $f$, we get the $\beta$-tree in Figure 7Di, which we call $T''$. We use terms, such as $C'$ and $C''$, for example, to refer to combinatorial terms associated to $T'$ and $T''$.

The reason for doing this is because the combinatorial terms of the two sets of induced major graphs are closely related, which we will exploit. For example, in Figure 6A we see a $\beta$-tree with a fence and in 6B we see the same $\beta$-tree with the fence removed. The combinatorial terms (in square brackets below each graph in Figures 6A,B iii-vi) are identical in all cases except when either $\tau_{a'} = \phi$ or $\tau_{b'} = \phi$ is empty.

First consider $T''$. For the graph in Figure 7Di we have $n_{b'}$ nodes descending from node $A$ and $n_{a'}$ from node $B$. Now because there is no fence $f$ present in $T''$ we have two separate branches; one from root $A$ down the branch containing node $b'$, the other from root $B$ down the branch containing node $a'$. We can then apply the same methods as Cases I and II above to conclude Equation (1) is valid for $T''$ (Figure 7D).

This gives us:

\[
\sum_{\{\tau_{a'}, \tau_{b'}\} \in S_{a'}, \tau_{b'} \in S_{b'}, N_f((\tau_{a'}, \tau_{b'})) = r_f} C''(\tau_{a'}, \tau_{b'}) = C_{a'}(\varepsilon) \cdot C_{b'}(\varepsilon) \cdot \left(\frac{n_{a'} + n_{b'}}{n_{a'}}\right)
\]

Here we have used the fact that the components $C_{a'}(\varepsilon)$ and $C_{b'}(\varepsilon)$ derived from the $\beta$-tree $T''$ (corresponding to the two triangles in Figure 7Diii) are identical to the components $C_{a'}(\varepsilon)$ and $C_{b'}(\varepsilon)$ derived from the original $\beta$-tree $T$ (see Figure 7Ai iii). The combinatorial term $\left(\frac{n_{a'} + n_{b'}}{n_{a'}}\right)$ arises because when the two root nodes are contracted together the single resulting node has two descending branches containing $n_{a'}$ and $n_{b'}$ nodes (see Figure 7Diii), and we then apply Theorem 4.1.

Now we want the corresponding sum to Equation (7) for tree $T'$. We have four cases to consider depending on whether $\tau_{a'}$ or $\tau_{b'}$ are empty.

Case IIIi $(\tau_{a'}, \tau_{b'} \neq \phi)$ Now if both subsets $\tau_{a'}$ and $\tau_{b'}$ are non-empty, we find from Lemma 5.3v that node $a'$ is attached to root $A$ and node $b'$ is attached to root $B$, and the fence is not part of $T'$. This results in the identical situation to $T''$, where there was no fence $f$ in the first place. We then find that:

\[
\sum_{\{\tau_{a'}, \tau_{b'}\} \neq \phi, N_f((\tau_{a'}, \tau_{b'})) = r_f} C'(\{\tau_{a'}, \tau_{b'}\}) = \sum_{\{\tau_{a'}, \tau_{b'}\} \neq \phi, N_f((\tau_{a'}, \tau_{b'})) = r_f} C''(\{\tau_{a'}, \tau_{b'}\})
\]

An example of this can be seen in Figure 6A,Bvi, where the combinatorial terms (in square brackets) are equal between the two groups.

Now from Equation (2) we are interested in the subsets $\tau_{a'}$ and $\tau_{b'}$ such that the number of nodes either bridged by, or descending from, fence $f$ is equal to $r_f = n_{a'} + r_{b'}$ for some value $r_f$. We have just seen that when $\tau_{a'}$ and $\tau_{b'}$ are both non-empty, the two sums in Equation (5) corresponding to trees $T'$ and $T''$ are equal for all values of $r_f$. For the remaining three cases, where at least one of $\tau_{a'}$ and $\tau_{b'}$ is empty, we will see that there is a constant difference between the sums arising from trees $T'$ and $T''$. Furthermore, the value $r_f$ will be seen to arise in exactly one of these three cases.

Case IIIii $(\tau_{a'}, \tau_{b'} = \phi$ and $r_f = n_{b'}$) The $\beta$-subtree $\tau = \varepsilon$ is trivial and there are no changes to the major graph. We then find there are $n_{b'}$ nodes present in the branch descending from $A$ in $T''(\tau)$. We thus find this case applies if $r_f = n_{b'}$. Now this situation does not apply to $T'$ (when the fence $f$ is present). The parental nodes of $f$ (the roots in this case) lie in $\tau = \varepsilon$, so one of the two nodes bridged by $f$ must lie in the $\beta$-subtree $\tau$ by Definition 5.4ii. We thus find that although $\varepsilon \in S'$ is a valid $\beta$-subtree for $T''$, $\varepsilon \notin S'$ is not a valid $\beta$-subtree for $T'$ (Figure 6A,Bii only has a contribution for the fenceless graph, for example). For $T''$, the trivial conditions $\tau_{a'}, \tau_{b'} = \phi$ result in a single induced major tree (corresponding to Figure 7Di), with major edges that match the original $\beta$-tree $T$, and we obtain combinatorial term $C_f'(\varepsilon) = C_{a'}(\varepsilon) \cdot C_{b'}(\varepsilon)$.

We thus find:
Figure 7: A) The general form of a 2d-tree. Triangles indicate a 2d-tree substructure. Dashed lines indicate possible presence of a fence. B) Reduction to a single branch. C) Reduction to a descending fence. D) The graphs of C with the fence removed. i) 2d-trees. ii) Trees $T(\tau)$. iii) Trees $\overline{T(\tau)}$ after root node contraction.

$$\sum_{\{\tau_\alpha' \neq \phi, \tau_\beta' \neq \phi, \ N_f(\{\tau_\alpha', \tau_\beta'\}) = r_f\}} C'(\{\tau_\alpha', \tau_\beta'\}) = \sum_{\{\tau_\alpha' \neq \phi, \tau_\beta' \neq \phi, \ N_f(\{\tau_\alpha', \tau_\beta'\}) = r_f\}} C''(\tau_\alpha', \tau_\beta') - C'_{\alpha'}(\varepsilon) \cdot C_{\beta'}(\varepsilon) \quad (9)$$

**Case III** (3) $(\tau_\alpha' = \phi, \tau_\beta' \neq \phi$ and $r_f < n_{\beta'})$ Now $\tau_\alpha'$ is trivial, so the arm descending from $B$ containing node $\alpha'$ is unchanged from the original major graph for both trees $T'$ and $T''$, and all $n_{\alpha'}$ nodes remain in the component of the major graph containing $B$ ($r_{\alpha'} = 0$). After the changes induced by $\tau_\beta'$, the other arm splits with $r_{\beta'}$ nodes belonging to the component of the major tree containing $A$, and $n_{\beta'} - r_{\beta'}$ nodes belonging to the component containing $B$. Thus in total there are $r_f = r_{\beta'}$ nodes from the original two branches that end up in the component of the major graph containing $A$, for some $r_{\beta'} \in \{1, 2, ..., n_{\beta'} - 1\}$. This case will thus apply provided $r_f < n_{\beta'}$. Now the combinatorial term from the unmodified $\alpha'$ branch matches those from the original $\beta$-tree; $C'_{\alpha'}(\varepsilon)$. Now in the tree $T''$ (without the fence $f$) the branch containing node $\beta'$ can be treated with the inductive hypothesis, like Case II above, and we find that:

$$\sum_{\{\tau_\alpha' = \phi, \tau_\beta' \neq \phi, \ N_f(\{\tau_\alpha', \tau_\beta'\}) = r_f\}} C''(\tau_\alpha', \tau_\beta') = C'_{\alpha'}(\varepsilon) \cdot \sum_{\{\tau_\beta' \neq \phi, \ N_f(\{\tau_\alpha', \tau_\beta'\}) = r_f\}} C''_{\beta'}(\tau_\beta') \cdot \left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}}\right) = C'_{\alpha'}(\varepsilon) \cdot C_{\beta'}(\varepsilon) \cdot \left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}}\right) \quad (10)$$

Here we pick up a combinatorial factor $\left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}}\right)$ from the two branches descending from node $B$. Now for the tree $T'$ (with fence $f$), the only combinatorial factor that differs between any pair of induced major trees $T'(\tau)$ and $T''(\tau)$, is the combinatorial term from node $B$ in $T''(\tau)$, which becomes the fence factor for $f; \left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}}\right) - 1$ in $T'(\tau)$. That is:

$$\frac{C''(\{\tau_\alpha', \tau_\beta'\})}{\left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}}\right) - 1} = \frac{C'(\{\tau_\alpha', \tau_\beta'\})}{\left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}}\right) - 1}.$$ 

Substituting this into Equation (10) gives us:

$$\sum_{\{\tau_\alpha' = \phi, \ N_f(\{\tau_\alpha', \tau_\beta'\}) = r_f\}} C'(\{\tau_\alpha', \tau_\beta'\}) = C'_{\alpha'}(\varepsilon) \cdot C_{\beta'}(\varepsilon) \cdot \left(\frac{n_{\beta'} - r_{\beta'} + n_{\alpha'}}{n_{\alpha'}} - 1\right) \quad (11)$$
Then subtracting Equation (11) from Equation (10) reveals the same constant difference observed in the previous case:

$$
\sum_{\{\tau_a' = \phi \atop \tau_b' \neq \phi \atop N_f(\{\tau_a', \tau_b'\}) = r_f\}} C'(\{\tau_a', \tau_b'\}) = \sum_{\{\tau_a' = \phi \atop \tau_b' \neq \phi \atop N_A = r_f+1\}} C''(\{\tau_a', \tau_b'\}) - C'_{a'}(\varepsilon) \cdot C'_{b'}(\varepsilon) \quad (12)
$$

**Case III iv:** ($\tau_b' = \phi$, $\tau_a' \neq \phi$ and $r_f > n_{b'}$) The argument is analogous to Case III ii and the same difference is obtained where we find:

$$
\sum_{\{\tau_a' \neq \phi \atop \tau_b' = \phi \atop N_f(\{\tau_a', \tau_b'\}) = r_f\}} C'(\{\tau_a', \tau_b'\}) = \sum_{\{\tau_a' = \phi \atop \tau_b' = \phi \atop N_A = r_f+1\}} C''(\{\tau_a', \tau_b'\}) - C'_{a'}(\varepsilon) \cdot C'_{b'}(\varepsilon) \quad (13)
$$

Thus in all three cases (III i-iii) the difference between the tree with and without the fence is $C'_{a'}(\varepsilon) \cdot C'_{b'}(\varepsilon)$. Furthermore, for any single value of $r_f$, one of these three cases is applicable. We thus find, using Equations (9), (12) and (13) with (8) that:

$$
\sum_{\{\tau_a' \in S_a' \atop \tau_b' \in S_b' \atop N_f(\{\tau_a', \tau_b'\}) = r_f\}} C'(\{\tau_a', \tau_b'\}) = \sum_{\{\tau_a' \in S_a' \atop \tau_b' \in S_b' \atop N_f(\{\tau_a', \tau_b'\}) = r_f\}} C''(\{\tau_a', \tau_b'\}) - C'_{a'}(\varepsilon) \cdot C'_{b'}(\varepsilon)
$$

Then substituting this into Equation (7) gives us:

$$
C'(\{\tau_a', \tau_b'\}) = C'_{a'}(\varepsilon) \cdot C'_{b'}(\varepsilon) \cdot (n_{a'} + n_{b'}) - C'_{a'}(\varepsilon) \cdot C'_{b'}(\varepsilon)
$$

(14)

Now $C'(\{\tau_a', \tau_b'\})$ matches the combinatorial term from the fence and its descendants, $C_f(\{\tau_a', \tau_b'\})$. Furthermore $C'_{a'}(\varepsilon)$, $C'_{b'}(\varepsilon)$ and $(n_{a'} + n_{b'}) - 1$ match the terms in the graph $\mathcal{T}$ obtained from the branch containing node $a'$, the branch containing node $b'$, and fence $f$ (by Theorem 4.1), and so equal $C'(\varepsilon)$. Thus we find that:

$$
\sum_{\{\tau_a' \in S_a \atop \tau_b' \in S_b' \atop N_f(\{\tau_a', \tau_b'\}) = r_f\}} C_f(\{\tau_a', \tau_b'\}) = C_f(\varepsilon)
$$

(15)

**Completing the Induction**

Thus finally substituting Equations (3), (4), (5), (6) and (15) into Equation (2) we find that we have:

$$
\sum_{\{\tau \in S : N_A(\tau) = r\}} C(\tau) = C_{a'}(\varepsilon) \cdot C_{b'}(\varepsilon) \cdot C_f(\varepsilon) \cdot \sum_{\{\tau_a, \tau_b, \tau_f : r_a + r_b + r_f = r - 1\}} \binom{r-1}{n_a-r_a, n_b-r_b, n_f-r_f} \binom{K-r-1}{r_a-r_a, n_b-r_b, n_f-r_f}
$$

(16)

Then applying the multinomial version of the Vandermonde identity [30] results in:

$$
\sum_{\{\tau \in S : N_A(\tau) = r\}} C(\tau) = C_{a'}(\varepsilon) \cdot C_{b'}(\varepsilon) \cdot C_f(\varepsilon) \cdot \binom{K-2}{n_a, n_b, n_f}
$$

(17)

However, the multinomial coefficient is identical to the combinatorial term we get if nodes $A$ and $B$ are contracted to a single root. In Figure 7Aii we see two branches containing nodes $n_a$ and $n_b$, these have combinatorial terms equal to $C_{a'}(\varepsilon)$ and $C_{b'}(\varepsilon)$. We also have fence $f$ bridging the two branches containing nodes $n_{a'}$ and $n_{b'}$. Application of Theorem 4.1 to $f$ for the root contracted graph $\mathcal{T}(\varepsilon)$ (given in Figure 7Aiii) returns precisely the term $\binom{K-2}{n_a, n_b, n_f}$. We thus find that we have all the coefficients of $C(\mathcal{T})$ and Equation (1) is obtained.
If we have more than one branch descending from the root nodes, the only change to the argument above is that we sum over a greater number of \( r_i \) values. The Vandermonde identity still applies and the same result is obtained.

### 5.4 Proving the Main Result

Finally, we can use this inductive relationship to determine the number of different evolutions that arise from a TD process, and prove our main result.

**Proof.** (Proof of Theorem 5.1) Let \( E \in \mathcal{E}_{n-1} \) be a connection evolution on \( n - 1 \) TDs with 2d-tree \( T \), and \( \mathcal{E}(E) \subset \mathcal{E}_n \) the corresponding subset of induced evolutions. Let \( \mathcal{N}(E) \) and \( \mathcal{N}(\mathcal{E}(E)) \) denote the number of TD evolutions corresponding to connection evolution \( E \), and set of induced connection evolutions \( \mathcal{E}(E) \), respectively. Now for every induced evolution \( E' \) we know that there corresponds a 1-nodeset \( \tau \) such that the major graph \( T_{maj}(E') \) corresponding to \( E' \) is obtained from Lemma 5.3. By Definition 5.4, \( \tau \) is also a \( \beta \)-subtree, and we have an induced major graph \( T(\tau) \). We also know from Theorem 5.2 that for any induced major graph \( T(\tau) \) we can sum \( C(\tau) \) over the \( \beta \)-subtrees \( \tau \) to obtain \( C(\varepsilon) \).

Now the difference between \( T(\tau) \) and the major graph \( T_{maj}(E) \) is that the latter has a fence between the two root nodes, this difference arising from Lemma 5.3vi.

Now \( E' \) is a connection evolution on \( n \) TDs so \( T_{maj}(\tau) \) has \( 2n \) nodes. Furthermore, \( T_{maj}(E') \) can have \( r \) nodes at the type \( a \) root, for some \( r = \{1, 2, ..., 2n - 1\} \), along with \( 2n - r \) nodes at the type \( b \) root. Then given the extra fence between the roots, by Theorem 4.1, the number of TD evolutions associated with \( T_{maj}(E') \) is given by \((\binom{2n}{r} - 1)C(\tau)\). Then, using Theorem 5.2, the total number of TD evolutions induced by \( E \) is given by:

\[
\mathcal{N}(\mathcal{E}(E)) = \sum_{r=1}^{2n-1} \sum_{\{\tau \in \mathcal{S} : N_A(\tau) = r\}} \left(\binom{2n}{r} - 1\right)C(\tau) = \sum_{r=1}^{2n-1} \left(\binom{2n}{r} - 1\right) \cdot \sum_{\{\tau \in \mathcal{S} : N_A(\tau) = r\}} C(\tau) \\
= \sum_{r=1}^{2n-1} \left(\binom{2n}{r} - 1\right) \cdot C(\varepsilon) \\
= C(\varepsilon) \cdot \left((\sum_{r=0}^{2n} \binom{2n}{r} - 2) - (2n - 1)\right) \\
= C(\varepsilon) \cdot (2^{2n} - (2n + 1))
\]

Now by Theorem 4.1 \( C(\varepsilon) \) is the number of TD evolutions associated with connection evolution \( E \). Thus we have:

\[
\mathcal{N}(\mathcal{E}(E)) = \mathcal{N}(E) \cdot (4^n - (2n + 1)) \tag{16}
\]

Furthermore, by Corollary 5.1, the set of induced evolutions from \( \mathcal{E}_{n-1} \) gives rise to a disjoint union of \( \mathcal{E}_n \). Thus summing Equation (16) across all \( E \in \mathcal{E}_{n-1} \) gives \( \mathcal{N}_n = \mathcal{N}_{n-1} \cdot (4^n - (2n + 1)) \). Starting a recursion from the single TD-Evolution with \( \mathcal{N}_1 = 1 \) then proves the theorem.

### 6 Conclusions

We have seen in Table 1 from our main result that the number of different evolutions increases with uncompromising velocity. This is primarily a theoretical result involving some interesting combinatorical approaches, however, it does have some biological relevance. In particular, the vast number of evolutions means that beyond five or six tandem duplications it is at present unrealistic to explore this space using an exhaustive search. Therefore, heuristic methods might be required to compare any observations, such as copy number information [31], [32], to the set of possible TD structures to determine evolutions that may explain the data.

The intricate methods utilized in this work largely parallel those used to examine breakage fusion cycles [32]. These are a distinct form of rearrangement which suggests there may be a more general space in which rearrangements operate and these methods apply. In contrast, the solution to the tandem duplication problem when breakpoint reuse is allowed ([16], [17]) does not require as much machinery, suggesting more efficient methods to this class of problems may be possible. Furthermore, tandem duplication and breakage fusion cycles are leading candidate rearrangements in the formation of large scale copy number increases.
such as those found in amplicons in cancer. A generalization of these methods to the combined space of these rearrangement processes may help to better understand their evolution.

In this study we have treated the process in a strictly discrete manner, treating the breakpoint numbers as a discrete count of the number of DNA segments that lie on one side of the breakpoint. However, one could consider TD as a continuous process, where breakpoints occur as a random process on the real line (or stretch of DNA) and investigate the relative likelihoods of different structures arising, as has been done with breakage fusion bridge cycles in [32].

The methods above and in [32] can be viewed as mathematical operations on the real line. It would seem plausible that other duplication mechanisms beyond those found in biological rearrangements would yield to similar analyses, which may shed light on the applicability of these methods which link combinatorics, general automaton on symbolic algebra and duplicating mappings on intervals.

Acknowledgements

This research was supported in part at KITP by the National Science Foundation under 264 Grant No. PHY11-25915.

References

[1] Ohno S., 1970, Evolution by gene duplication, Springer-Verlag.
[2] Nye TMW, 2009, Modelling the evolution of multi-gene families, Stat Methods Med Res, 18(5), 487-504.
[3] McBride DJ, Etemadmoghadam D, Cooke SL, Alsop K, George J, Butler A, Cho J, Galapathithee D, Greenman CD, Howarth KD, Lau KW, Ng CK, Raine K, Teague J, Wedge DC, Australian Ovarian Cancer Study Group, Caubit X, Stratton MR, Brenton JD, Campbell PJ, Futreal PA, Bowtell DDL, (2012), Tandem duplication of chromosomal segments is common in ovarian and breast cancer genomes, J. Pathology, 227(4), 446 - 455.
[4] Raphael BJ, Pevzner PA. 2004. Reconstructing tumor amplisomes. Bioinformatics, 20: i265–i273.
[5] Raphael BJ, Volik S, Collins C, Pevzner PA. 2003. Reconstructing tumor genome architectures. Bioinformatics, 19, iii162–iii171.
[6] Zhang C, Leibowitz ML and Pellman D, 2013, Chromothripsis and beyond: rapid genome evolution from complex chromosomal rearrangements, Genes Dev., 27, 2513-2530.
[7] Benson G, An algorithm for finding tandem repeats of unspecified pattern size, in RECOMB ’98 Proceedings of the second annual international conference on Computational molecular biology, 20-29.
[8] Benson G, 1997, Sequence alignment with tandem duplication, J. Comput Biol., 4(3), 351-67.
[9] Benson G, 2005, Tandem cyclic alignment, Discrete Applied Mathematics, 146(2), 124–133.
[10] Benson G, 1999, Tandem repeats finder: a program to analyze DNA sequences, Nucl. Acid. Res., 27(2), 573-580.
[11] Kolpakov R, Bana G and Kucherov G, 2003, mreps: efficient and flexible detection of tandem repeats in DNA, Nucl. Acid. Res., 31(13), 3672-3678.
[12] Benson G and Dong L, 1999, Reconstructing the duplication history of a tandem repeat. Benson, in Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB-99), 44-53.
[13] Bertrand D, Lajoie M, and El-Mabrouk N, 2008, Inferring ancestral gene orders for a family of tandemly arrayed genes, J. of Comp. Biology, 15(8), 1063-1077.
[14] Bertrand D and Gascuel O, 2005, Topological rearrangements and local search method for tandem duplication trees, IEEE/ACM Transactions on Comp. Biol. and Bioinformatics, 2(1), 15-28.
[15] Rivals E, 2004, A survey on algorithmic aspects of tandem repeats evolution. Int. J. Found. Comput. Sci., 15(2), 225-257.
[16] Gascuel O, Hendy MD, Jean-Marie A, McLachlan R, 2003, The combinatorics of tandem duplication trees, Syst Biol., 52(1), 110-8.
[17] Yang J and Zhang L, 2004, On counting tandem duplication trees, Mol. Biol. Evol., 21(6), 1160-1163.
[18] Bouvel M and Rossin D, 2009, A variant of the tandem duplication - random loss model of genome rearrangement, Theoretical Computer Science, 410, 847-858.
[19] Bouvel M and Pergola E, 2010, Posets and permutations in the duplication-loss model: Minimal permutations with d descents, Theoretical Computer Science, 411, 2487-2501.
[20] Allouche J. and Shallit J. 2003, Automatic Sequences, Theory, Applications, Generalizations, Cambridge university press.
[21] Greenman CD, Pleasance ED, Newman S, Yang F, Fu B, Nik-Zainal S, Jones D, Lau KW, Carter N, Edwards PA, Futreal PA, Stratton MR, Campbell PJ, Futreal PA, Bowtell DDL, 2011, Estimation of rearrangement phylogeny for cancer genomes, Genome Research, 21(2):346-61.
[22] Bentley JL, 1975, Multidimensional binary search trees used for associative searching, Comm. ACM, 18, 509-517.
[23] Turtle H and Croft WB, 1991, Evaluation of an inference network-based retrieval model, ACM Transactions on Information Systems, 9(3), 187-222.

[24] Griffiths RC and Marjoram P, 1996, Ancestral inference from samples of DNA sequences with recombination, J. Comp. Biol., 3(4), 479-502.

[25] Kirkpatrick B, Reshefy Y, Finucane H, Jiangx H, Zhu B and Karp RM, 2012, Comparing pedigree graphs, J. of Comp. Biology, 19(9), 998-1014.

[26] Neggers J and Kim HS, 1998, Basic Posets, World Scientific.

[27] Karzanov A and Khachiyan L, 1991, On the conductance of order Markov chains, Order, 8, 7-15.

[28] Brightwell G and Winkler P, 1991, Counting linear extensions, Order, 8(3), 225-242.

[29] Semple C and Steel M, 2003, Phylogenetics, Oxford University Press.

[30] Zeng J, 1996, Multinomial convolution polynomials, Discrete Math., 160, 219-228.

[31] Kinsella M and Bafna V, 2012, Combinatorics of the breakage-fusion-bridge mechanism, J. of Comp. Biology, 19 662-678.

[32] CD Greenman, SL Cooke, J Marshall, MR Stratton, PJ Campbell, 2013, Modelling Breakage-Fusion-Bridge cycles as a stochastic paper folding process, arXiv:1211.2356.