Genome Reconstruction and Combinatoric Analyses of Rearrangement Evolution

Luca Penso Dolfin

PhD thesis

School of Computing Sciences
University of East Anglia

January 2016

Research is to see what everybody else has seen,
and to think what nobody else has thought.

Albert Szent-Gyorgyi

Contents

List of Figures

List of Tables

Acknowledgements

At the end of this PhD experience, I have to thank many people who have been close to me during this three years in Norwich.
Thanks to Chris Greenman and Taoyang Wu for their excellent guidance, and for all that I have learnt in these years.
Thanks to my friends in Norwich, for always making me feel home, and to the older ones in Venice, for constantly reminding me where I come from.
Last, but not least, thanks to my family, for always supporting me in the good and in the bad time.

Index of terms and abbreviations

Allele:alternative forms of a gene, differing from one another in the DNA sequence [26] *Allelic ratio*: ratio between the intensity of the two alleles of a SNP, typically provided by a microarray experiment[42]. *Amplicon*: in cancer genetics, DNA material which has been amplified through some rearrangement mechanisms [17]. *ATP*: adenosine triphosphate [26].

Deletion: a mutation in which one or more base pairs is removed from a DNA sequence [26].

DNA: deoxyribonucleic acid [61]

Enzyme: a type of protein that speeds the rate of a specific biochemical reaction, making it fast enough to be compatible with life [70].

Eukaryote: organism whose cells have a nucleus and other specialed organelles. The genetic material typically consists of a nuclear genome, and a smaller mithocondrial genome. *Gene*: a region of DNA (deoxyribonucleic acid) coding either for the messenger RNA encoding the amino acid sequence in a polypeptide chain or for a functional RNA molecule [26].

Genome: the complete genetic constitution of an organism, a cell, or a virus [61].

Genotype: 1) the particular combination of alleles for a particular locus or 2) the set of genes possessed by an individual organism [37].

Heterozygous: an individual organism that possesses different alleles at a locus [37].

Haploid: condition of cells or organisms possessing a single chromosome complement, hence a single gene copy at each locus [37].

Haploinsufficiency: the appearance of a mutant phenotype in an individual cell or organism that is heterozygous for a normally recessive trait [90].

Helicase:an enzyme that unwinds and separates the two strands of the DNA double helix [31]. *homologous genes*: evolutionarily related genes, having descended from a gene in a common ancestor [90].

Homologous pair of chromosomes: two chromosomes that are alike in structure and size and that carry genetic information for the same set of hereditary characteristics. One chromosome of a homologous pair is inherited from the male parent and the other is inherited from the female parent [90].

Homologous recombination: exchange of genetic information between homologous DNA molecules [90].

Insertion: a mutation that occurs when one or more base pairs is added to a DNA sequence [90].

Inversion: a 180 reversal of the orientation of a part of a chromosome, relative to some standard chromosome [37].

Karyotype: description of the number, size and morphology of the chromosomes in a cell [100].

Locus: the position of a gene along a chromosome; often used to refer to the gene itself [37].

Marker: an allele that serves as a probe to follow a specific phenotype [31].

Mitosis: the process of nuclear division in Eukaryotic cells that occurs when a parent cell divides to produce two identical daughter cells [31].

mRNA: messenger RNA, the intermediate form of the genetic code between DNA and protein [31].

NAHR: Non-Allelic Homologous Recombination, a mechanism of recombination between non homologous sequences.

Oncogene: dominant-acting gene that stimulates cell division, leading to the formation of tumors and contributing to cancer; arises from mutated copies of a normal cellular gene (proto-oncogene) [90]. *Phenotype*: the outward appearance of an organism for a given characteristic [90].

read coverage/ read depth: for a genome, average number of times each base has been sequenced. For a particular region, fractional average of the read depth of each nucleotide [57].

Replication: DNA replication is a process by which a double-stranded DNA molecule is copied into two, identical DNA molecules [31].

RNA: ribonucleic acid [61].

SNP: Single Nucleotide Polymorphism [61].

tandem duplication: duplication of a chromosome segment that is adjacent to the original segment [90].

Telomeres: repetitive sequences found at both ends of a chromosome [32].

Translocation: movement of a chromosome segment to a non-homologous chromosome or

to a region within the same chromosome; also movement of a ribosome along mRNA in the course of translation [90].
tRNA: transfer ribonucleic acid (tRNA) is a type of RNA molecule that helps decode a messenger RNA (mRNA) sequence into a protein [31].
Tyrosine kinase: an enzyme that can transfer a phosphate group from an *ATP* molecule to a protein [25].

ABSTRACT

Cancer is often associated with a high number of large-scale, structural rearrangements. In a highly selective environment, some 'driver' mutations conferring clonal growth advantage will be positively selected, accounting for further cancer development. Clarifying their nature, as well as their contribution to the pathology is a major current focus of biomedical research. Next generation sequencing technologies can be used nowadays to generate high-resolution data-sets of these alterations in cancer genomes.
This project has been developed along two main lines: 1) the reconstruction of cancer aberrant karyotypes, together with their underlying evolutionary history; 2) the elucidation of some combinatorial properties associated with gene duplications.
We applied graph theory to the problem of reconstructing the final cancer genome sequence; additionally, we developed an algorithmic approach for the reconstruction of a multi-step evolution consistent with read coverage and paired end data, giving insights on the possible molecular mechanisms underlying rearrangements. Looking at the combinatorics of both tandem and inverted duplication, we developed an algebraic formalism for the representation of these processes. This allowed us to both explore the geometric properties of sequences arising by Tandem Duplication (TD), and obtain a recursion for the number of tandem duplications evolutions after n events. Such results are missing for inverted duplications, whose combinatorial properties have been nevertheless deeply elucidated. Our results have allowed: 1) the identification, through an original approach,

of potential rearrangement mechanisms associated with cancer development, and 2) the definition and mathematical description of the complete evolutionary space of specific rearrangement classes.

1 Introduction

Cancer cells typically show a collection of genetic mutations which distinguish them from normal human cells. The causal relation between some of these transformations and the development of the pathology has been widely described in the scientific literature (a review on some crucial discoveries is given in [103]). Clarifying the molecular mechanisms responsible for such changes, as well as their contribution to the pathology represents a crucial challenge of modern biomedical research. One particular type of cancer genetic modification is represented by rearrangements, that is mutations causing a change in the genome organisation and possibly in the number of copies of some genetic material. This thesis will focus on the combinatorial properties of some specific types of rearrangements, as well as on the challenge of reconstructing cancer genomic organisation using Next Generation Sequencing data.

Following is a short overview on some crucial concepts of cell biology, the different classes of rearrangements, as well as the scientific literature and methodologies related to this thesis.

After this introduction, the main research goals will be presented: firstly, an Eulerian graph approach (fully described in chapter two) and an algorithmic approach (chapter three) to the genome reconstruction challenge; secondly, a mathematical description of the combinatorial properties of two types of rearrangements: tandem duplication (chapter four) and inverted duplication (chapter five).

1.1 What is DNA?

The acronym *DNA* stands for *Deoxyribonucleic Acid*, that is the molecule used by all known living organisms as a source of instructions for their biological development and functioning throughout the whole life. Along with RNA and proteins, DNA is one of the

three major macromolecules essential for all known forms of life. This information consists of a sequence of just 4 different units, the *nucleobases Guanine, Adenine, Thymine* and *Cytosine* generally indicated by the letters G, A, T and C , respectively. These units are typically organised in couples of long *polymers* (that is, a macromolecule consisting of repeating structural units), forming two strands where alternating sugars (*deoxyribose*) and *phosphate groups* (PO_4^{3-}) belonging to the same chain are linked by *phosphodiester* bonds (Figure 1.1). The two strands interact to each other through hydrogen bonds between *nucleotides* (i.e. a nucleobases linked to a deoxyribose and a phosphate group), resulting in a superstructure which is generally described as a double-helix, as each chain wraps itself around the other. Just two types of interactions between these coupled helices are possible: A:T (linked by 2 hydrogen bonds) and G:C (3 hydrogen bonds)[112].

In each strand one backbone is labelled 3' (three prime) and the other 5' (five prime). The 3' end terminates at the hydroxyl (OH^-) group of the third carbon in the deoxyribose, and is known as the tail end. The two strands are *anti-parallel*, meaning that the 3' end of the former corresponds to the 5' end of the latter.
However, the whole genetic material (*genome*) of an Eukaryotic organism does not usually consist of a single molecule: it is rather organized in several, more complex structures called *chromosomes*. Long stretches of DNA are wrapped around proteins, the *histones*, which allow the double helix to be repetitively coiled and thus efficiently packed, saving much useful space inside cell. Chromosomes may either be located in the nucleus, if that structure is present, or otherwise in the space between the cell's organelles, the *cytoplasm*. Each chromosome may be even present in several copies. An organism carrying just one copy of each chromosome is called *haploid*. Similarly, in a *diploid* organism (like our species, *Homo sapiens*) the number of copies is 2, while the term *polyploid* generally refers to all conditions where 3 or more copies are present (*triploid,tetraploid,pentaploid,...*). Different copies of the same chromosome are said to be *homologous* to each other.

The total length of the genetic material is extremely variable across species, varying from a fraction of a *Megabase* (corresponding to 10^6 nucleotides) to several tens of *Gigabases*

(10^9). For instance, the haploid set of our nuclear human genome is made up of 23 chromosomes, corresponding to a total sequence of about $3.2x10^9$ bases [34]. As we are diploid organisms, it means that the total amount of DNA in each human cell is $6.4x10^9$ bases!

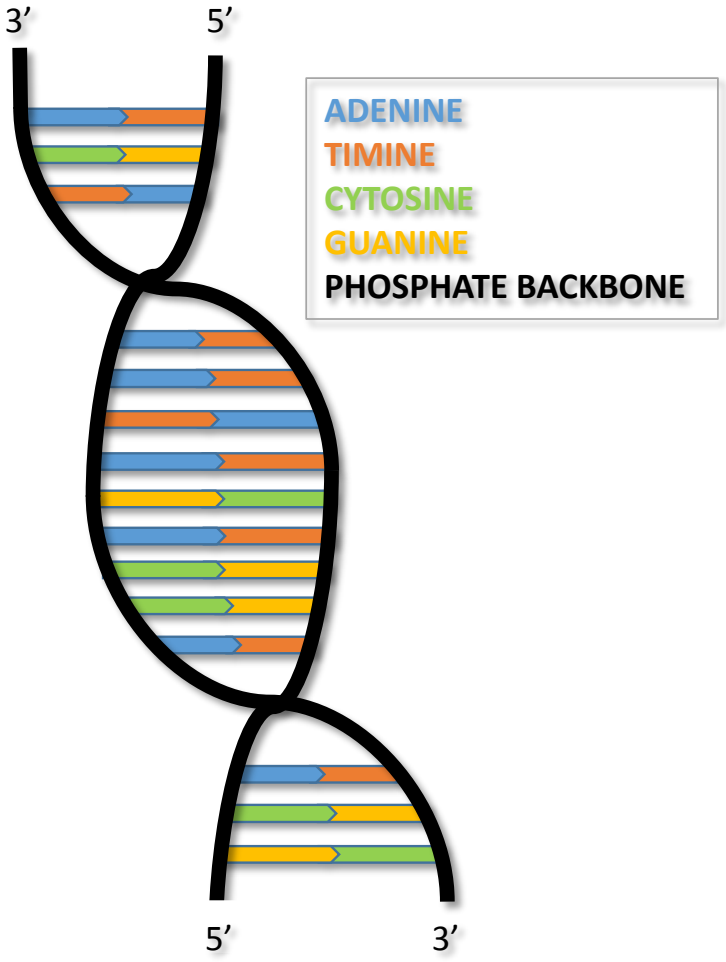

Figure 1.1: the DNA double helix structure.

1.2 Genetic code and protein synthesis

How is the genetic information encoded inside the DNA molecule? Like the letters of an alphabet, the sequence of its four nucleotides form a series of instructions for the cell's molecular machinery to build up RNA (*Ribonucleic Acid*, a molecule which is chemically very close to DNA itself) and proteins. The 'words' of this language (the *genetic code*) are all made up of three adjacent nucleotides, called *triplets* or *codons*. A DNA region (more technically defined as a *locus*) which is used for the synthesis of either a protein or an RNA molecule is called a *gene*[112]. These sequences, depending on the organism, may account for a different fraction of the whole genome, but in humans they represent just 2 % of it [33]. In the most common case of a gene being used for the synthesis of a protein, each triplet indicates a particular unit to be chosen, that is a specific *amino-acid*[61]. Thus, a sequence of codons corresponds to a chain of amino-acids(*polypeptide*) making up the whole protein.
In Eukaryotes, the process of *protein synthesis*[112] can be divided into three steps:

1. *transcription* of the so called *messenger-RNA* (*mRNA*)

2. *post-transcriptional modifications* of the same mRNA molecule

3. *translation* of its sequence into an amino-acid chain, which becomes a functional protein once it has acquired its proper three-dimensional structure

In the first step, a group of proteins called *transcription factors* binds to specific DNA positions, helping the enzyme *RNA Polymerase* to operate RNA synthesis according to the target gene genetic information. The two helices of DNA are separated by the enzyme *helicase*, disrupting all hydrogen bonds between them. RNA polymerase reads one helix (*antisense* or *negative* strand) from the 3' end to the 5', synthesizing a single-stranded mes-

senger RNA in the 5'-to-3' direction. The resulting RNA molecule is almost identical to the coding DNA sequence, a part from having all Thymines (T) replaced by another nucleobase, Uracil (U), and all sugar rings (*rybose*) carrying an additional oxygen atom bond to their second carbon atom . This single strand of mRNA exits the nucleus through nuclear pores, and migrates into the cytoplasm. The mRNA undergoes some post-transcriptional modifications, which include capping with 7-methyl-guanosine, tailing with an Adenine polymer(*poly-A*) and eventually *splicing* of *introns* (noncoding parts of the sequence, in contrast with the retained coding parts, called *exons*). The processed mRNA can eventually enter the latter step, occurring in the cytoplasm with the support of other protein complexes, the *Ribosomes*. mRNA is decoded to produce a specific polypeptide according to the rules specified by the codon-based genetic code. With the help of *transfer-RNAs* (t-RNA, molecules capable of providing the ribosome with all needed amino-acids) the ribosome uses the mRNA sequence as a template to operate the synthesis of a chain of amino-acids that will eventually form a complete protein.

1.3 Mitosis and DNA replication

The term *Mitosis* refers to the process of nuclear (*karyokinesis*) and cytoplasm division (*cytokinesis*) of a single eukaryotic cell, leading to the formation of two daughters which are both a perfect copy of their 'mother'. Five stages are generally distinguished [28]:

1. *Prophase*. At this stage the chromatin begins to condense, making all chromosomes much more visible, in a process that is going to continue until metaphase. In diploid organisms, homologous chromosomes are physically linked together in correspondence of the *centromere* site. Each partner of a couple is called *chromatid*.

2. *Prometaphase*. The nuclear envelope is fragmented into many small vesicles that will eventually be equally shared by the future daughter cells. Specific proteic polymers (the *microtubules*) can now access the chromosomes pairs, and will anchor to them at specific sites. Each chromatid is then tugged by microtubules toward the opposite cell pole with respect to its partner. These couples, however, have not been separated yet.

3. *Metaphase*. Chromosomes assume an even more compacted state, and centromeres of all chromosomes line up in a plane at the cell's equator. A complex checkpoint mechanism determines whether the plane is properly formed, in which case the cell can enter the next stage.

4. *Anaphase*. Sister chromatids are eventually separated and move toward opposite poles.

5. *Telophase and cytokinesis*. During the final stage, the chromosomes eventually reach the poles. Following this, cytokinesis occurs: the nuclear membrane of each daughter cell's nucleus is built up, while chromosomes begin to de-condense into their original conformation.

For the mitosis to give rise to two perfect copies of the ancestral cell, the whole genetic material must be duplicated. This process starts with the separation of the 2 DNA helices at specific, A-T rich sites, thanks to the activity of a group of proteins forming a *pre-replication complex*. The resulting structure is called *replication fork*. Each chain will be used as a template by the enzyme DNA polymerase, capable of linking each nucleotide with a new copy of its complementary unit (A with T, G with C); the 2 newly synthesized strands consist of a *leading* strand, and a *lagging* strand. The former is elongated in the same direction as the growing replication fork. An enzyme called *primase* synthesizes an RNA primer, that is a short nucleotide sequence carrying a free 3'-OH group, which binds

to the single stranded DNA and is used as a starting point for the copying activity of the DNA polymerase. Because the synthesis of the new strand is made in the same direction of the replication fork movement, which constantly extends to the adjacent nucleotides while the template is being copied, the leading strand replication is allowed to continue without interruptions. In the case of the lagging strand, however, its opposite orientation with respect to the leading helix means that the 3' to 5' elongation, typical of every newly synthesized strand, does not follow the movement of the replication fork. Therefore, replication can only proceed with a separated synthesis of short adjacent sequences, the *Okazaki fragments*, added one by one as soon as a new double helix region is included in the fork. Eventually, the process ends with the primers' removal, and the joining by a *DNA ligase* of the short Okazaki fragments linked to the lagging strand, building up together a complete complementary sequence.
DNA replication is not, however, a completely error-free process: mismatches in the two resulting double helices will be formed in Eukaryotes at a rate of about 1 per every 100,000 nucleotides [5]. Effectively acquired mutations are however strongly limited by the presence in the cell of efficient repair systems. Some mistakes are corrected even during replication, through a process known as *proofreading*, while others are removed later on during the *mismatch repair* process. After that, all incorrectly paired nucleotides that have not been found by the repair machinery will become permanent mutations. The cell, indeed, will no longer be able to recognize them as errors as soon as it enters a new cell division [5].

A newly synthesized strand may also loops out slightly, resulting in the addition, omission of some nucleotide base, in a mechanism called *strand slippage*(Figure 1.4). The presence of small repeated sequences makes a DNA locus particularly prone to this type of mutation. Double strand breaks might also occur as the result of exposure to chemical agents, irradiation or reactive oxygen species. Such breaks might be repaired through a variety of mechanisms, including a simple rejoining of the broken ends with little or no modifications (*non homologous end joining*, NHEJ) or a repair guided by homologous sequences (*homologous recombination*)[52]. However, we will see how mutations accumulated in cancer cells might lead to an inefficient DNA repair. This in turn would favour the appearance

of additional sequence alterations, possibly conferring a selective advantage in terms of survival and proliferation rates.

1.4 Cancer in the western world: from the classic age to the genomic era

The earliest description of cancer was made by Hippocrates (ca. 460 BC ca. 370 BC), who used to call these pathologies '*carcinos*': a term referring to the appearance of the cut surface of a solid malignant tumor, with "the veins stretched on all sides as the crab has its feet". Hippocrate's explanation of tumours was based on humoural theory, stating that all human pathologies are caused by imbalances in the quantity of the four basic substances (*humours*) making up our body. Specifically, he believed that cancer was the direct effect of an excess of black bile. While superficial tumours were to be removed, deep ones were to be left, thus allowing the patient to hold out for a longer time. These beliefs were generally supported until the Renaissance time, when the humoural explanation began to be contested by the studies of Andreas Vesalius (1514-1561) on dissected human cadavers [87]. Eighteenth Century's Enlightenment saw the definitive drop out of the humoural theory. Ren Descartes (1596-1650) attempted to link cancer and lymphatic system in his lymph theory. Considerable progress was achieved in the field of epidemiology: for instance, John Hill (1714-1775) first observed a higher incidence of nasal cancer in snuff users. The first hospital for the treatment of cancer was opened in 1740 in Rheims, France, and not much later, in 1792, the first cancer institute in Middlesex, England. However, poor results were to be experienced for a long time more, due to bad hygienic conditions and an evident lack of knowledge of the underlying biological mechanisms [87].
The primary role of DNA began to be clear in the 19th Century, thanks to the studies of David von Hansemann (1858−1920) [48] and Theodor Boveri (1862−1915) [20]. The observation under the microscope of unusual chromosomal aberrations in cancer cells led to the proposal that cancers are abnormal clones of cells, characterized and caused by

abnormalities of hereditary material. After the identification of DNA as the molecule of inheritance [6] and determination of its structure [113], eventually the central role of the genome in a cell's life became clear. Next came the demonstration that mutagens (agents causing DNA damage and mutations) are causally linked to cancer development [71]; moreover, increasingly refined analyses of aberrant chromosomes in tumor cells showed that specific and recurrent genomic abnormalities, such as the *translocation* (an exchange of DNA material) between chromosomes 9 and 22 in chronic myeloid leukaemia (known as the *Philadelphia translocation* [81, 96]), are associated with particular cancer types. Inserting a cancer human genome into *wild-type* (healthy) cells (a technique known as *transformation*) was shown to cause their conversion into cancer cells [66, 99]. Isolation of the specific DNA sequence responsible for this transforming activity led to the identification of the first naturally occurring, human cancer-causing sequence change, the single base G > T substitution that causes an amino-acid substitution (glycine → valine) in codon 12 of the *HRAS* gene [94, 104]. A new chapter of cancer research was about to begin, focusing on the discovery of genes underlying the development of the pathology.

More recently, the development of next-generation sequencing technologies has lead to an incredibly fast rise of available genomic information, giving the scientific community a great chance to explore the mechanisms underlying the cancer genome's evolution. Resulting DNA sequences can be analysed in unprecedented detail, allowing the appearance and constant growth of *somatic*(cancer-specific) mutations databases (for instance, [35]). However, evidence suggests that much information about these processes is still missing: further research is necessary for a complete, reliable explanation of the observed patterns of variation in the genomic organisation, as well as in the number of copies of some genomic regions.

1.5 The rapid evolution of an aberrant genome

A population of cancer cells is the result of an evolutionary history where stochastic mutational events progressively build up a richer and richer set of genetic variants. Resulting phenotypes will be subject to a mechanism of *Darwinian selection*, causing the decrease in frequency, and perhaps complete disappearance of cells bearing deleterious mutations; meanwhile, other mutants might acquire a proliferative advantage and higher survival rates, and their genetic background will quickly spread across the population. The typical result of this process is the continuous appearance, disappearance and size change of several subpopulations (*clones*) of cells sharing some particular genetic variants. Within the same patient, a small fraction of positively selected genotypes are likely to compete against each other, and just in some cases a single cell may collect enough advantageous mutations to proliferate very quickly, invade other tissues and cause a metastasis.
In the following sections, we will focus on one particular family of mutations typical of cancer genomes: the structural and phenotypic alterations caused by the amplification, loss or exchange between chromosomes of large fragments of DNA.

1.6 DNA loss and amplification

Copy number variation (that is, the change in the number of copies of a given chromosome or DNA region) can be seen as one of the major sources of diversity evolutionary forces can operate on. Loss of tumour suppressor genes (preventing the development of cancer) and amplification of oncogenes (favouring mechanisms such as cell proliferation) are commonly observed copy number changes associated with cancer evolution. Several molecular mechanisms underlying copy number variation have been described. In the following section, an overview on such phenomena will be made.

1.6.1 Breakage-fusion-bridge cycles

Intrachromosomal, clustered, relatively narrow peaks of amplification are consistent with the *breakage-fusion-bridge* (BFB) process [54], which was first proposed by Barbara McClintock [77]. Among all putative explanations of the observed patterns of intrachromosomal amplification, the BFB cycle is one of the most supported [106](Figure 1.2). According to this model, the starting point of the process is a DNA double strand break (DSB). In most cancer cells, given the poor functioning of the DNA checkpoints (that is, the surveillance mechanisms capable of delaying the cell cycle in the presence of DNA damage) and the repair machinery [68], a chromosome break doesn't prevent mitosis from starting: the cell cycle therefore proceeds through the pre-mitotic phases S and G2: at this stage, all DNA material has already been replicated, giving rise to 2 identical broken chromosomes. The repair machinery will act on the four uncapped DNA ends, either fusing them correctly to generate repaired chromosomes or fusing sister chromatids to generate palindromic chromosomes. These massive fragments may be either dicentric (with two centromeres) or *acentric* (without any centromere). In the former case, such aberrant chromosomes could be broken during chromosome segregation, because of microtubules moving them in opposite directions. The result may be the inheritance, in the daughter cells, of broken chromosomes with partially deleted or duplicated regions. The broken end would be involved in further BFB cycles, causing the accumulation of palindromic duplications of chromosomal regions and an amplified copy number of some genes (palindromic gene amplification). As long as broken ends are present, allowing for another sister chromatid fusion, these cycles can continue indefinitely, causing further amplification of genetic material.

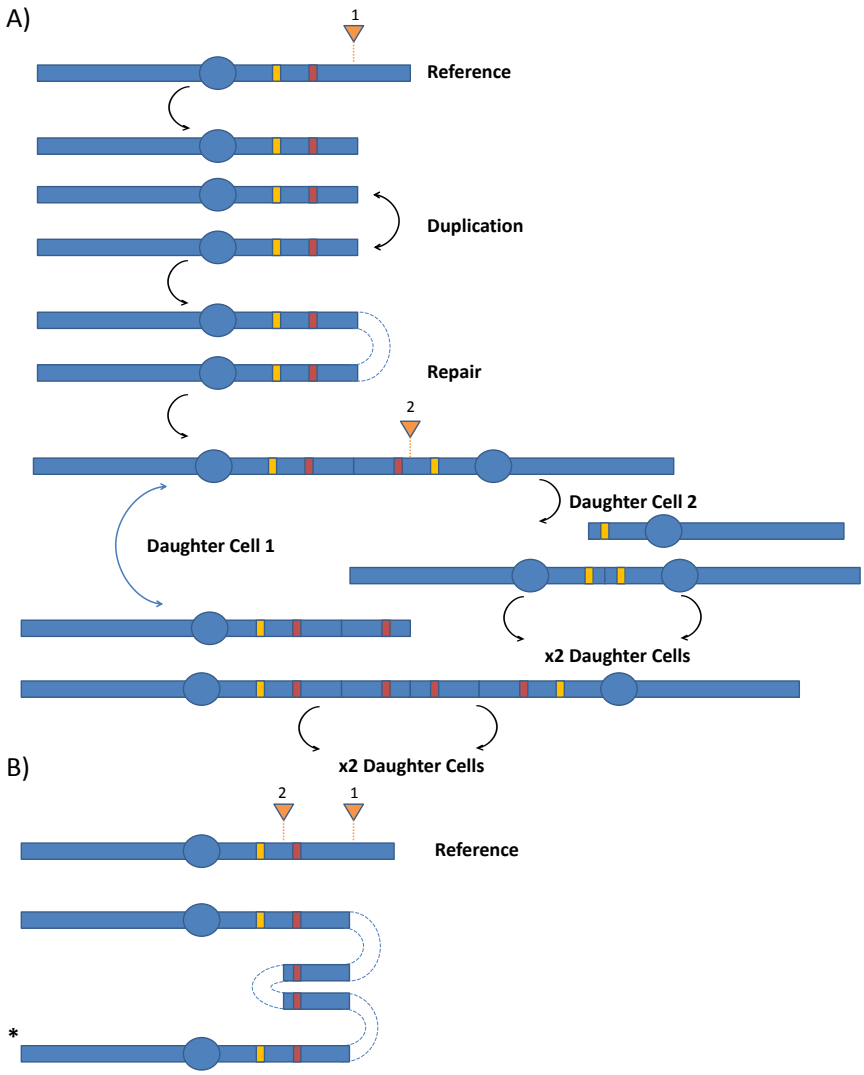

Figure 1.2: Schematic representation of a Breakage-Fusion-Bridge(BFB). A) A chromosome is represented, with the circle indicating a centromere, the red and yellow markers hypothetical genes duplicated and deleted during the process. A DNA break (whose position is given by orange triangles) and the following duplication and repair result in a palindromic chromosome with two centromeres. During cell division, microtubules attract these centromeres in opposite directions causing another break, and the cycle continues. B) The folded resulting structure (*) compared to the original reference genome. Based on [43].

1.6.2 Segmental duplications and deletions

As opposed to Breakage Fusion Bridge cycles, intrachromosomal (inside a chromosome) duplications are also observed in cancer genomes. Depending on the exact structural transformation, distinct mechanisms might underlie these rearrangements:

- Many additional, adjacent copies (*tandemly duplicated* segments) arise from *non-allelic homologous recombination* (NAHR, typically occurring between non-homologous sequences), *unequal crossing-over* between sister chromatids (Figure 1.3) or *replication slippage* (Figure 1.4A). The first two mechanisms may also cause deletions or inversion of the duplicated segment's interposing sequences [106]. The latter accounts for duplication (or deletions) of small sequences inside a replication fork (1-2 kb).

- *Fragment capture by double strand break*: an exogenous fragment can be inserted into a double strand chromosomal break.

Deletions, on the other hand, result in the loss of a particular DNA sequence. Causes include translocation events, unequal crossing over, replication slippage (Figure 1.4B) and chromosome shattering followed by non homologous repair (the so called *Non Homologous End Joining*, see Figure 1.7). The effects of these mutations may vary depending on the size and location of the event. The deletion of a centromere creates an aberrant acentric chromosome which will most likely be lost during cell division. Gene expression patterns may be affected as well and result in a phenotypic change. In the case of both homologous copies of a gene being transcribed in a (healthy) cell, the deletion of the entire coding sequence in one chromosome leads to a condition called *haploinsufficiency* [27]. Deletions of smaller sequences can lead to a modified, non functional gene; typically, in the case of tumour suppressor genes such events can have a drastic positive effect on tumour development.

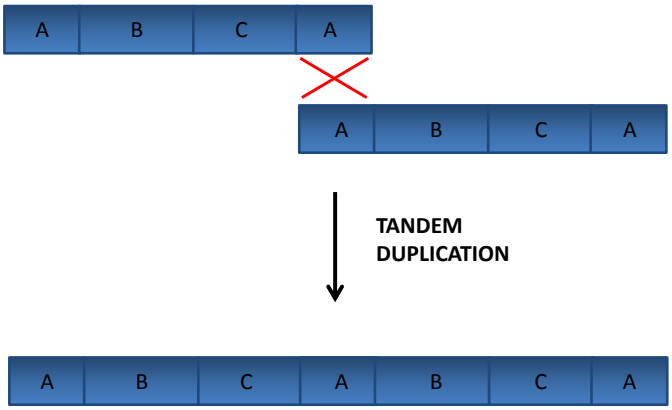

Figure 1.3: Schematic view of a tandem duplication arising from a recombination event (NAHR or unequal crossing-over) . The presence of sequence homology between either two sister chromatids can lead to a misalignment (region A in the figure) during crossing over, causing a copy number change. Alternatively, the same structural change might arise from the repair of a double strand break by NAHR, using a non-homologous (yet sharing some sequence similarity) template.

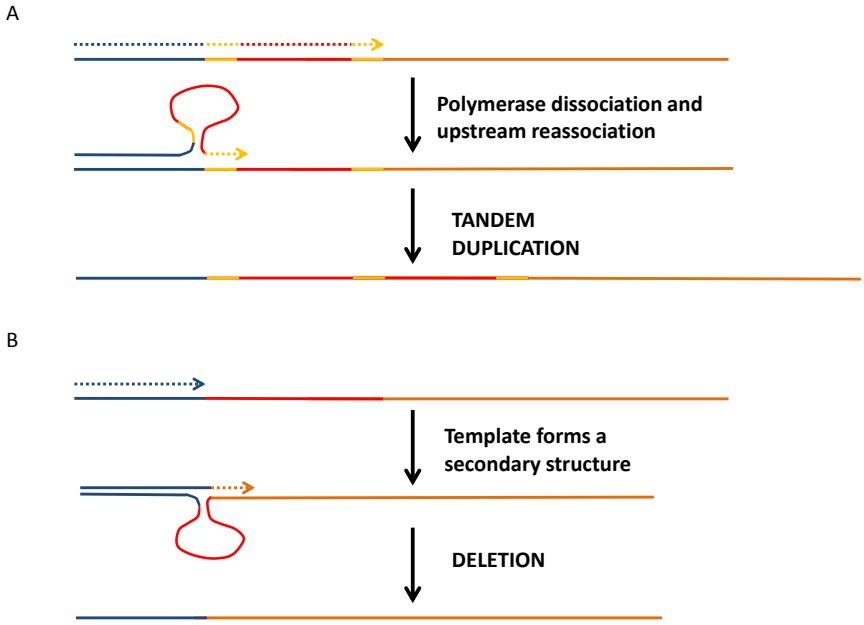

Figure 1.4: Replication slippage. A) During replication of a region containing some repeats (yellow labelled), the polymerase might dissociate from one copy of the repeat and re-associate on a different copy, located upstream. As shown in the figure, a portion of the template sequence might thus be copied again, leading to the duplication of the red segment. (B) Following the formation of a secondary structure in the lagging-strand template, DNA polymerase dissociates and aligns to another sequence. Replication continues leading to the deletion of the loop sequence (red).

1.6.3 Unbalanced translocations

A *translocation* is the movement of a DNA segment to another site, either within the same chromosome or to a non-homologous one. These molecular rearrangements are generally considered to be the primary cause of various cancers. Depending on the location of breakpoints, such events can cause the disruption or misregulation of a gene's typical function. Over the past few decades, clinical cytogeneticists have been able to link specific chromosome breakpoints to clinically defined cancers, including subtypes of leukemias,

lymphomas, and sarcomas. Virtually all of the translocations observed in tumors have arisen through somatic mutations, so they are not inherited in families.

A translocation between two chromosome is typically designated as follows: *t(chr a;chr b)(region a; region b)*, where *t* indicates that a translocation has occurred; the first set of parentheses indicates the two chromosomes involved in the translocation, and the second set contains the breakpoints on the chromosome arms. The event is referred as *unbalanced* if it causes the loss of some genetic material(Figure 1.5). *Loss of heterozygosity* (the loss of one parental allele) following an unbalanced translocation is the typical outcoming genetic alteration, possibly causing inactivation of a tumor suppressor gene in multistage human carcinogenesis [84] . Cytogenetic studies have shown that unbalanced translocation is a frequent chromosome alteration in a variety of human cancers, including acute leukemias and colorectal cancer [79, 85].

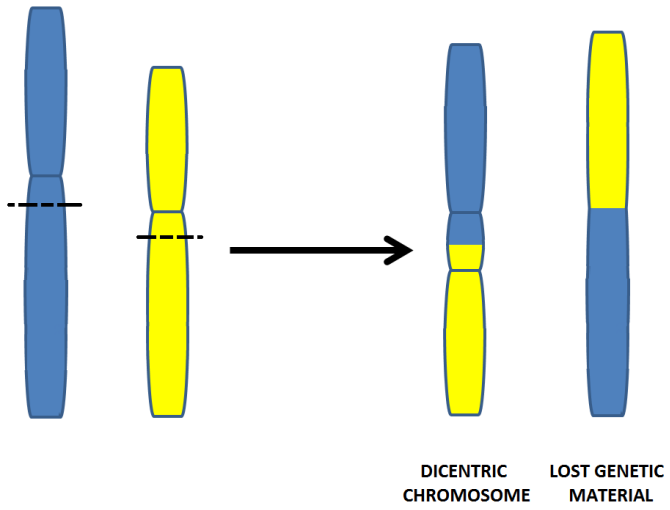

Figure 1.5: Schematic representation of an unbalanced translocation. Two breaks lead to the creation of a dicentric chromosome, while the left genetic material is often lost

1.7 Copy-number-preserving rearrangements

1.7.1 Inversions

An inversion occurs when a segment of a chromosome changes its orientation. The molecular mechanism simply consists of a single chromosome breakage followed by a rearrangement within itself. Inversions are of two types: *paracentric* and *pericentric*. depending on whether they do or do not include the centromere. Such type of rearrangement, compared copy number changing rearrangements and translocations, is generally less disruptive at the DNA sequence level, and thus less likely to have a phenotypic effect in carriers. However, a recent study on T-cell lymphomas has suggested that a recurrent inversion on chromosome 6 might cause the over-expression of the regulatory gene *Dlx5*, leading to an increased cell proliferation [105].

1.7.2 Balanced translocations

A translocation is referred as *balanced* if it does not lead to any loss of genetic material (Figure 1.6). An example is represented by the recurrent translocation between chromosomes 8 and 14 found in patients with Burkitt's lymphoma, which places the *MYC* proto-oncogene (normally located on chromosome 8) under the control of the powerful immunoglobin heavy chain gene (*IGH*) promoter on chromosome 14 [110]. The rearrangement induces an overexpression of *MYC* in lymphoid cells, where the *IGH* promoter is normally active. Another possible outcome of a balanced translocation is the creation of a new chimeric gene, built up as a puzzle of sequences originally belonging to separated loci. The aberrant gene products or the altered regulation of gene expression may eventually confer a selective advantage to the cancer cell: an example is given by the Philadelphia chromosome, where a translocation fuses the coding sequence of the *BCR* (breakpoint cluster region) gene on

chromosome 22 with the coding sequence of the *ABL* gene on chromosome 9 [3]. The *BCR-ABL* fusion protein encoded by the chimeric gene constitutively activates signalling pathways involved in cell growth and proliferation. In the case of the *ETV6-NTRK3* gene fusion in congenital fibrosarcoma, the new gene is produced by a translocation between chromosomes 12 and 15 (*t(12;15)*). In this chimeric sequence, the 5' region of ETV6 is fused to the 3' region of NTRK3, which is believed to affect NTRK3 *signal-transduction pathways* [63].

Consistent associations of many gene fusions with specific tumour types have been demonstrated, and the identification of such specificities raises the chances of developing therapeutic gene-targeting approaches.

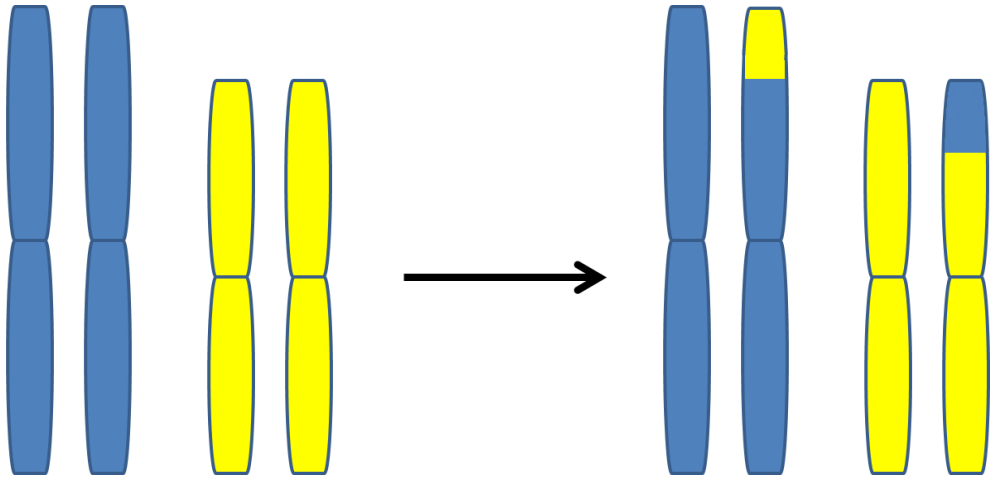

Figure 1.6: schematic representation of a reciprocal translocation.

1.8 Chromothripsis

The recent advent of high-throughput DNA sequencing has highlighted some crucial aspects of cancer genome evolution: collected evidence to date suggests that the classical, relatively simple rearrangements described so far are often grouped together into a unique

super-event, leading to dramatic structural and copy number changes. The term *chromothripsis* is generally used to indicate these 'catastrophic' events [102].

After the cell enters mitosis and while DNA replication is still occurring, one or more chromosomes might be shattered into many segments of variable size. Some of the resulting fragments might be lost, while others will be rejoined together through a microhomology-based mechanism: two double-strand breaks sharing a few nucleotides long regions of similarity will be stuck together by the repair machinery, in a process called *non-homologous-end-joining* (NHEJ) [55] (Figure 1.7). Chromosomes carrying complex, highly localized rearrangements are the typical outcome of this process.

Although the causes of this initial physical damage of chromosome is unknown, these clusters of double-strand breaks might well result from ionizing radiation. Alternatively, dicentric chromosomes formed by Breakage-Fusion-Bridge could break during mitosis, as they are pulled apart in opposite directions [102]. Rather than favouring the hypothesis of a multi-step evolution, next generation data often points toward this 'catastrophe' explanation. For instance, some segment copy number profiles, restricted to as few as two states, are hardly consistent with sequential, independent rearrangements [102]. Under this model, the number of different states observed would be expected to increase as the number of breakpoints rises. Tandem duplications lead to a copy number increase and when several such events overlap with one another, many segments are expected to be repetitively amplified under the progressive rearrangements model. Deletion events would clearly limit these increases in copy number, but the total set of rearrangements is very unlikely to lead to a simple profile with just two copy number states, particularly when the number of events is high. Other examples are regions of preserved heterozygosity which are spanned by multiple rearrangements (like deletions, duplications and inversions). A deletion occurring early in a successive series of rearrangements, would permanently eliminate heterozygosity between its two breakpoints. A progressive model is therefore compatible with chromothripsis only assuming that deletions have occurred late in the sequence of events: an unlikely scenario when the number of rearrangements is very high. The chromothripsis explanation is further supported by the observation that alternating regions of heterozygosity (retention of a DNA fragment) and loss of heterozygosity are the natural result of a rearrangement series caused by a unique event [102].

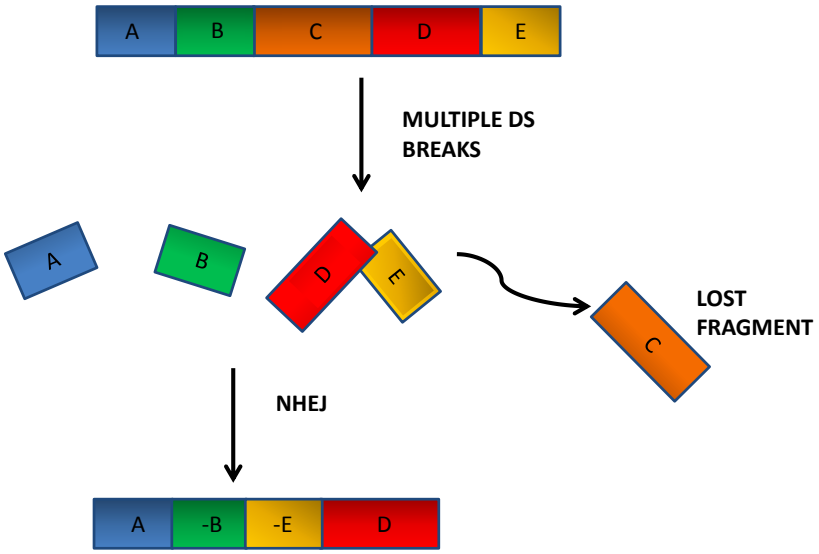

Figure 1.7: Schematic representation of a chromothripsis event: fragments produced by multiple DS breaks are then stuck together through a mechanism of non-homologous-end-joining (NHEJ).

1.9 Microhomology mediated break induced replication

Recently, experimental research with nuclease-induced breaks has shed light on a mechanism called Microhomology Mediated Break Induced Replication (MMBIR), occurring in cancer cells when a replication fork breakage exposes a single broken end. In these circumstances, repair by homologous recombination or nonhomologous end-joining, requiring a couple of broken ends, is not an option[51]. MMBIR has been suggested as a mechanism to repair broken ends in such situations, provided that stretches of single-stranded DNA are available and share microhomology with the single-strand end from the collapsed fork.

Such mechanisms might originate from a variety of situations, including stalled transcription complexes or at secondary structures in DNA caused by complex genomic architecture, as well as in promoter regions and replication origins. Following strand invasion, elongation of the broken end might continue for several kilobases, until a telomere is reached, or the extended end is dissociated and annealed with a different template, possibly repeating the process several times ([51, 101]) (see Figure 1.8). Consequences of MMBIR at the DNA sequence level include as different rearrangements as duplications, inverted duplications, deletions or unbalanced translocations([51]).

The possibility of template switching for any single-stranded DNA region sharing microhomology with the single stranded end involved in the mechanism would explain why MMBIR is imprecise and prone to generate chromosomal structural changes. Recent research on cancer genomic variation has highlighted the presence of microhomology associated with somatically acquired connections ([17, 64, 119, 120]), giving support to a central role of microhomology-mediated mechanisms in the formation of aberrant genomic architectures. Moreover, the observation of large duplications and deletions implies that mechanisms different from replication slippage within a single replication fork are responsible for such changes, making MMBIR a possible candidate.

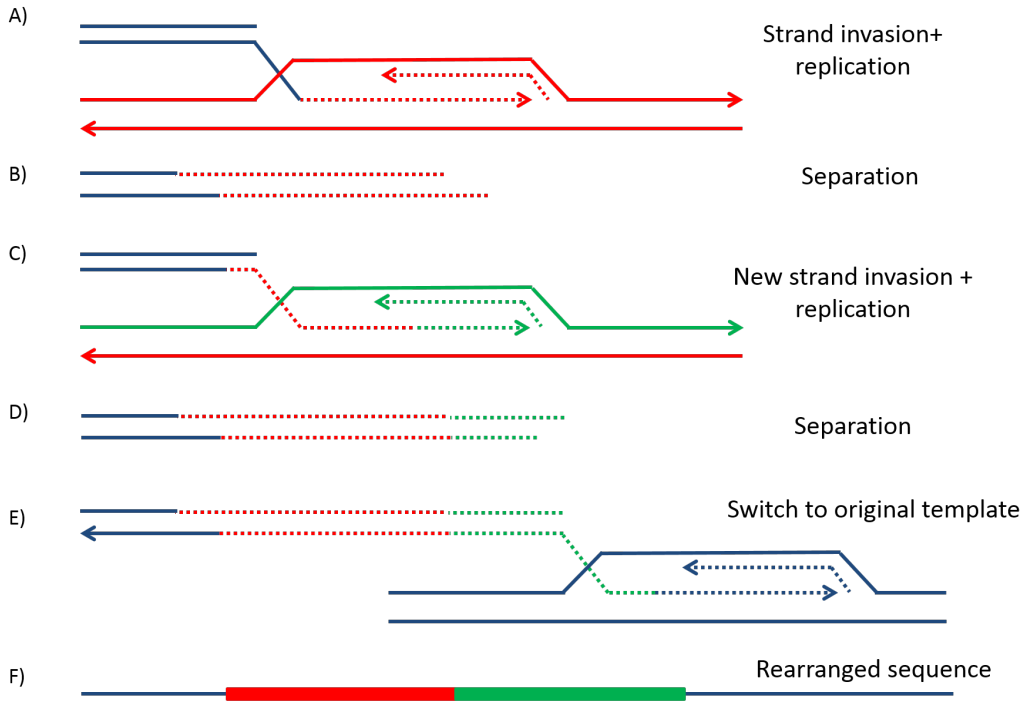

Figure 1.8: Schematic representation of a Microhomology-Mediated Break-Induced Replication, following the appearance of a single DS break. Different genomic loci, connected by microhomology junctions (arrow) are represented by distinct colours; arrows have a 5'-3' orientation. The broken arm of a collapsed replication fork forms a new fork in (A). The end is extended (A and C) and dissociated (B and D) several times reforming the fork on different templates . (E) The switch returns to the original sister chromatid (blue), forming a new replication fork and allowing for the completion of replication. The final product (F) contains sequence from different genomic regions. Each line represents a DNA nucleotide chain (strand). Based on [51].

1.10 Detecting rearrangements using Paired End Sequencing

Many different experimental techniques have been developed for the study of chromosomal rearrangements. *Fluorescent in situ hybridization* [91, 107], *chromosome painting* [58],

spectral karyotyping [97] and *comparative genome hybridization* [59] are significantly limited by the low resolution of the results. The problem of describing the architecture of a tumour genome accurately while limiting the costs can be solved using *paired end sequencing*(PES) [23, 36]. First, the tumour genome is cut into small, $\sim$ 500 bp long segments; then the DNA material is copied several times through *DNA amplification*. Such a step may be carried out through a cloning-based or cloning-free procedure. In the former method, DNA fragments are linked to specific sequences called *plasmids*, creating a circular DNA sequence which is later incorporated into *E. coli* cells (*transformation*). Rapid replication of this bacterium will result in an analogous amplification of the contained insert, giving a wide collection of identical fragments called a *DNA library*. In the cloning-free protocol, on the other hand, circular products are amplified through the *Polymerase Chain Reaction*(PCR) [29].

After amplification, both ends of each obtained fragment are sequenced: one read copies the forward strand, while the other uses the reverse strand, so that the resulting reads have opposite 5' to 3' directionality. All reads are then aligned to the reference human genome, using an alignment algorithm such as *BLASTN*[78] or *Bowtie2*[69]. Results will show a set of *concordant* pairs of reads (mapping at a distance of about 500 bps, pointing towards each other in opposite orientations) and a set of *discordant* couples, showing a significant difference in mapping distance and/or an opposite orientation (Figure 1.9). This also allows the detection of small insertions and deletions through the comparison between read pairs and the average insert size of the genomic library. Typically, all end sequences which don't map uniquely in the reference genome are discarded, avoiding in this way any ambiguity in the inference of rearrangement events [29].

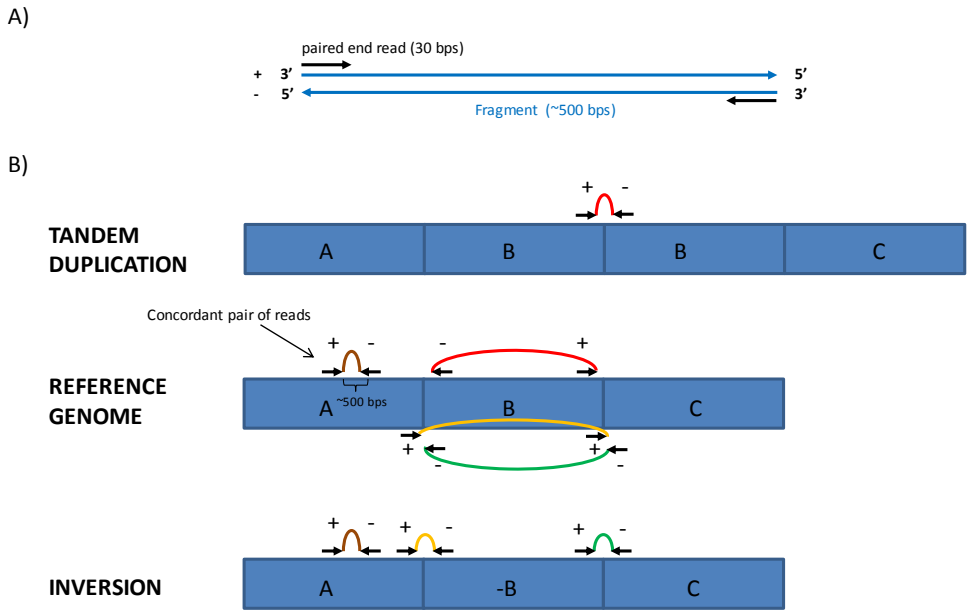

Figure 1.9: Sequencing and alignment of paired end sequences. A) The two ends of each fragment are sequenced, using opposite strands (forward,+ and reverse,-). B) Reads are mapped to the reference genome. The red labelled paired ends map on opposite +,- strands in the reference genome, similar to the rearranged one; however, their distance is much bigger in the reference, suggesting a tandem duplication. Similarly, distance for the green and the yellow pairs is bigger in the reference, as well as the alignment orientation. Together, these two pairs of reads support the presence of an inversion.

1.11 Evolutionary distance between genomes

The challenge of finding a reliable set of operations to transform one genome to another is a classical problem of computational biology. In 1995, a sorting by reversal approach was proposed by Hannenhalli and Pevzner [8]. In that study, a sequence of n genes in a

genome was represented by a signed permutation on {1, . . . , n} and a + or - sign is associated with every gene copy. In such a model, a reversal of a fragment always changes both the order and the signs of the elements within that fragment. The reconstruction of an evolutionary history is then equivalent to the *reversal distance problem*: finding the minimum number of reversals to transform a permutation of π of the original gene order into the identity signed permutation $(1, 2, ..., n)$. For this purpose, the paper introduced the notion of *breakpoint graph*: an edge coloured graph with n+2 vertices $(0, 1, ..n, n + 1)$, where black edges connect couples of nodes which are consecutive in π, and the other edges are grey labelled.

Further research lead to a new algorithm allowing for inversions, translocations, fissions and fusions [46, 47].

A successful attempt to simplify these algorithms was made by Yancopoulos *et al* [114] who invented the soon become popular *Double Cut and Join* (DCJ) operation. Their algorithm allows a simple and efficient computation of reversals, translocations,fusions, fissions and block interchanges (a generalised type of transposition).

In 2006, Hartman and Shamir noticed that the problem of sorting linear permutations by transpositions is equivalent to the problem of sorting circular permutations by transpositions. From this observation, they developed a more efficient algorithm to transform 2 and 3 cycle permutations into 1-cycle, thus sorting circular breakpoint graphs by transpositions and transreversals [49, 50].

The problem of accounting for different occurrence rates for each rearrangement type was later addressed by Bader and Ohlebusch [7] who took into account biologically realistic weights for their method.

In 2008, Gog *et al.* [40] presented the new program GENESIS, allowing the sorting of multi-chromosomal genomes by reversals, translocations, fusions and fissions in polynomial time. GENESIS includes the algorithm presented by Bader and Ohlebusch [7]; an improved version of the algorithm by Hannenhalli and Pevzner [46]; and a combination of the two previous algorithms plus the DCJ method [114].

1.12 The combinatorics of gene duplications

The gene duplication process is known to be implicated in the formation of gene clusters [82, 86] as well as amplicons in cancer [73, 92, 93, 115]. 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 these process, there is a range of algorithmic and mathematical questions that are also of interest. These include identification and alignments of tandem duplications in data [9, 10, 11, 72, 65] and the construction of phylogenies describing their evolution [12, 15, 14]. In [15] 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. A survey of algorithmic approaches can be found in [95]. The combinatorial nature of these rearrangement operations leads to some interesting questions. For instance, in [39] and [115] the problem of counting the number of rooted and unrooted tandem duplication trees was addressed. Other studies focused on the space of permutations arising from a tandem duplication-loss model [18] and [19].

These methods are based on a range of assumptions regarding the available information and the nature of the process. Among them, two issues are particularly important for the problem of counting the number of evolutions by duplication.

The first is the genomic sequence information. In [15], a comparison among genomes of the signed gene orders is carried out: ancestral orders which minimise the number of inversions are then inferred. [39] and [115], on the other hand, consider a single copy of a locus, providing the count of all possible distinct evolutions.

Secondly, in these works the important assumption that breakpoints can be reused is made. Here breakpoints can be seen as either gaps between two contiguous loci, such as a pair of genes in a gene cluster, which can cover a wide region and be implicated in more than one

duplication event with reasonable probability, or the precise end points of the duplicated region. In such cases, when a duplication occurs, the two breakpoints are implicated in a presumably random process. The probability of additional duplication events introducing a break at the same exact positions is likely to be small, and we can reasonably assume unique breakpoint use. The combinatorial questions considered in chapters four and five, as well as the reconstruction problem of chapter three, are based on this assumption.

1.13 Aim of the Study

Results of this thesis will be organised into 4 different chapters, presenting the distinct research challenges and approaches that we faced.

An Eulerian path approach for the reconstruction a cancer genome sequence

This study (presented in chapter 2) has focused on the problem of inferring the final genomic architecture of a cancer genome, using paired end and copy number data as the input information.

An algorithmic approach for the inference of a cancer genome evolution

The aim of this study (see chapter 3) was the reconstruction of multiple steps of a cancer genome evolution, looking for a series of rearrangements producing a final rearranged genome which is most consistent with paired end and copy number variation data. We generalized different classes of molecular double strand repair mechanisms through the use of a particular Cut and Join operation (which we have called Cut and Repair, CR), along with the unedited modelling of the Microhomology-Mediated Break-Induced Replication (MMBIR). Based on available paired end reads and copy number variation data, our al-

gorithm models each new somatically acquired connection supported by PES data as a 'rearrangement operation', thus constructing multiple steps of structural and copy number modifications of a genome, leading to a new aberrant karyotype. Solutions are then filtered based on the comparison between the pattern of copy number variation in the digital genome and read coverage data from the tumour sample.

The combinatorics of segmental duplication

We have developed an efficient algebraic representation of a tandem duplication (TD) evolution (see chapter 4), representing chromosomes as numeric vectors (*words*) where each TD-like connection is uniquely represented by a number. Based on such a representation, we have faced the following challenges:

- Describing the full space of TD evolutionary paths in terms of number of words of size m after k TD events

- Using Hasse diagrams, describing the combinatorial properties of the TD space; this allowed us to define a series of constraints for the number of different breakpoint orders which are consistent with a specific word.

Following this investigation, we addressed the same questions as above for a model of inverted duplication evolution (presented in chapter 5).

2 An Eulerian path approach for the reconstruction of a cancer genome sequence

We have seen in the introduction how Next Generation Sequencing technologies have recently allowed the generation of high-resolution data-sets of cancer genome rearrangements. There is a great biomedical interest in gaining knowledge about the resulting aberrant karyotypes, as well as the exact mechanisms responsible for such changes. While the latter point is better addressed in the following chapter three, here we will focus on the former question: how do we reconstruct chromosomes from copy number and rearrangement information.
We start by giving the following, crucial definition:
Definition 2.1. *We call* copy number vector *the word of integers* $[cp_1, cp_2...cp_n]$ *where each* cp_i *represents the number of copies of the* i^{th} *DNA segment.*

We are thus able to informally state our problem:
Problem 2.1. *Given a copy number vector and rearrangement information from a cancer sample, use graph theory to reconstruct the set of digital karyotypes which best explain such data.*

We will first go through some more crucial definitions, allowing us to better formalise problem 2.1. Next, we will present the different stages of the analysis in detail; conclusions will complete the chapter.

2.1 Graph theory and genome representation

Starting from our two types of input information (copy number vector and rearrangement data), we want to transform our challenge into a graph theory problem. We will be

using different types of graphs to represent the cancer genome as a path along nodes (each corresponding to a DNA segment) and edges (representing germ-line or somatic connections). First we give some important graph theory definitions.

Definition 2.2. *An* undirected graph *is a graph in which the edges are not associated with a particular direction. In such a graph, given two nodes A and B there will be no distinction between $A \leftarrow B$ and $B \rightarrow A$.*

Definition 2.3. *A* directed graph *is, conversely, a graph in which any particular edge is designated a forward or a backward direction (thus $A \leftarrow B$ and $B \rightarrow A$ are distinct edges).*

Definition 2.4. *A spanning tree T of a connected, undirected graph G is a tree that includes all of the vertices of G.*

Definition 2.5. *An* arborescence *(see Figure 2.4 for an example) is a directed tree in which, for a vertex u called the root and any other vertex v, there is exactly one directed path from u to v.*

Definition 2.6. *A* bidirected graph *is a graph in which both edge extremities lie either on the right or the left side of the node it touches, giving rise to* 4 *possible connections.*

Example 2.1. *Consider Figure 2.1A. Blue numbers* 1..5 *represent the node labels. A red edge connects the right side of node* 2 *with the right side of node* 3*. However, we also have a distinct edge connecting the right side of node* 2 *to the left side of node* 3*. Thus we find that for any couple of nodes we have more than one possible type of edge, depending on which node extremities are touched.*

Definition 2.7. *Given a graph $G = (V, E)$, two edges in V are called* multiple *or* parallel *if and only if they connect the same couple of nodes.*

Definition 2.8. *A* multigraph *is a graph which allows for multiple edges between two nodes.*

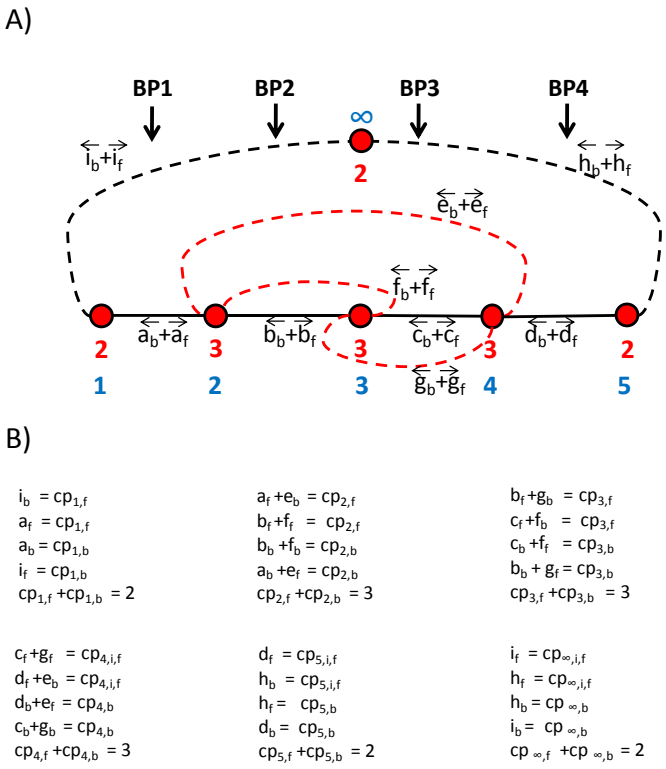

Figure 2.1: A) Schematic representation of a bidirected graph. Nodes represent DNA segments. Black edges represent wild type connections; coloured edges correspond to somatic connections. Blue numbers indicate segment labels; Red numbers represent the copy number vector across nodes. To each undirected edge k correspond two labels k_b, k_f. These refer to the variables shown in B), and represent the number of copies of edge k with either of the two possible orientations (arbitrarily assigned): backward (b) and forward (f). B) List of constraints for the construction of directed graphs. $cn_{k,b/f}$ stands for the number of backward/forward segments represented by node k, expected to sum up to the copy number of segment k.

Definition 2.9. *A* strongly connected component *is a subgraph of a directed graph G such that for every pair of vertices u, v in the subgraph, there is a directed path from u to v and a directed path from v to u.*

Definition 2.10. *A* directed cycle *in a directed graph is a sequence of vertices starting and ending at the same vertex such that, for each two consecutive vertices of the cycle, there exists an edge directed from the earlier vertex to the later one.*

Definition 2.11. *An* Eulerian cycle*, Eulerian circuit or Euler tour in a graph is a cycle*

that uses each edge exactly once.

Lemma 2.1. *[38] A directed graph has an Eulerian cycle if and only if :*

1. *for every vertex v in V, $indegree(v) = outdegree(v)$*

2. *all of its vertices (with non-zero indegree and outdegree) belong to a single strongly connected component*

The information about genome connections deriving from paired end sequencing experiments allows us to construct an initial bidirected graph (lacking the edge directionality and copy number information). An example of that is shown in Figure 2.2C, where four breakpoints define five DNA segments, represented by nodes 1 to 5. Alignment of paired end reads to the reference genome is used to identify breakpoints, as well as adding somatic connections to our graph. Consider Figure 2.2A-B as an example. If a pair of reads map at the expected distance (about 500 bps) and with the expected orientation (Figure 2.2A, pair connected by a brown curve) The rearrangement information can be then combined to our second piece of information, the copy number vector. This can be derived from the read depth (that is, the average number of times each base has been sequenced) observed across the DNA region of interest, using an appropriate algorithm such as ASCAT [111] or PICNIC [42]. I Figure 2.2D, we see that the read depth signal across segments 1 to 5 is used to infer the copy number vector [2 3 2 3 2]. This tells us about the number of copies of these five segments. Now, we want to use this information about the copy number to infer directionality of the edges, thus constructing a set of directed (multi)graphs consistent with the original bidirected one. These graphs then allow us to construct genome sequences by walking through each of them. Adding directionality information means we now have two nodes for each DNA segment: each of them representing a particular segment orientation, forward or backward. Because these graphs are consistent with the input data, any complete cancer genome sequence must contain every edge from one of such directed graphs, in exactly the same number of copies. This means that, if we make sure

that our graphs are Eulerian, the set of valid cancer genomes will correspond to the set of Eulerian cycles of all directed graphs. Looking at the graph in Figure 2.1A, however, we notice that nodes 1 and 5 can be indegree-outdegree balanced only with a further addition of node and edges: more precisely, we need an auxiliary node labelled ∞ and edges $\{(5,\infty),(\infty,-5),(-1,\infty),(\infty,1)\}$ to make sure that our graph is Eulerian.

Example 2.2. *Consider the directed graph of Figure 2.3. This graph can be derived from the bidirected graph of Figure 2.1A.*

We can now formalise Problem 2.1:

Problem 2.2. Eulerian reconstruction problem. *Given a bidirected graph G_b obtained from a cancer sample $\mathcal{C}$, and associated copy number vector, determine the complete set of consistent directed Eulerian graphs $G_{d,1}, G_{d,2}...G_{d,n}$ and construct all Eulerian cycles for each $G_{d,i}$.*

In the following sections we will go through all steps of the analysis required for the solution of Problem 2.2:

- Construction of all directed graphs (containing the edge directionality information which is missing in the original bidirected graph)

- Inference of all arborescences for each directed graph

- Construction of Eulerian cycles of every directed graph, using the arborescences obtained from the previous step

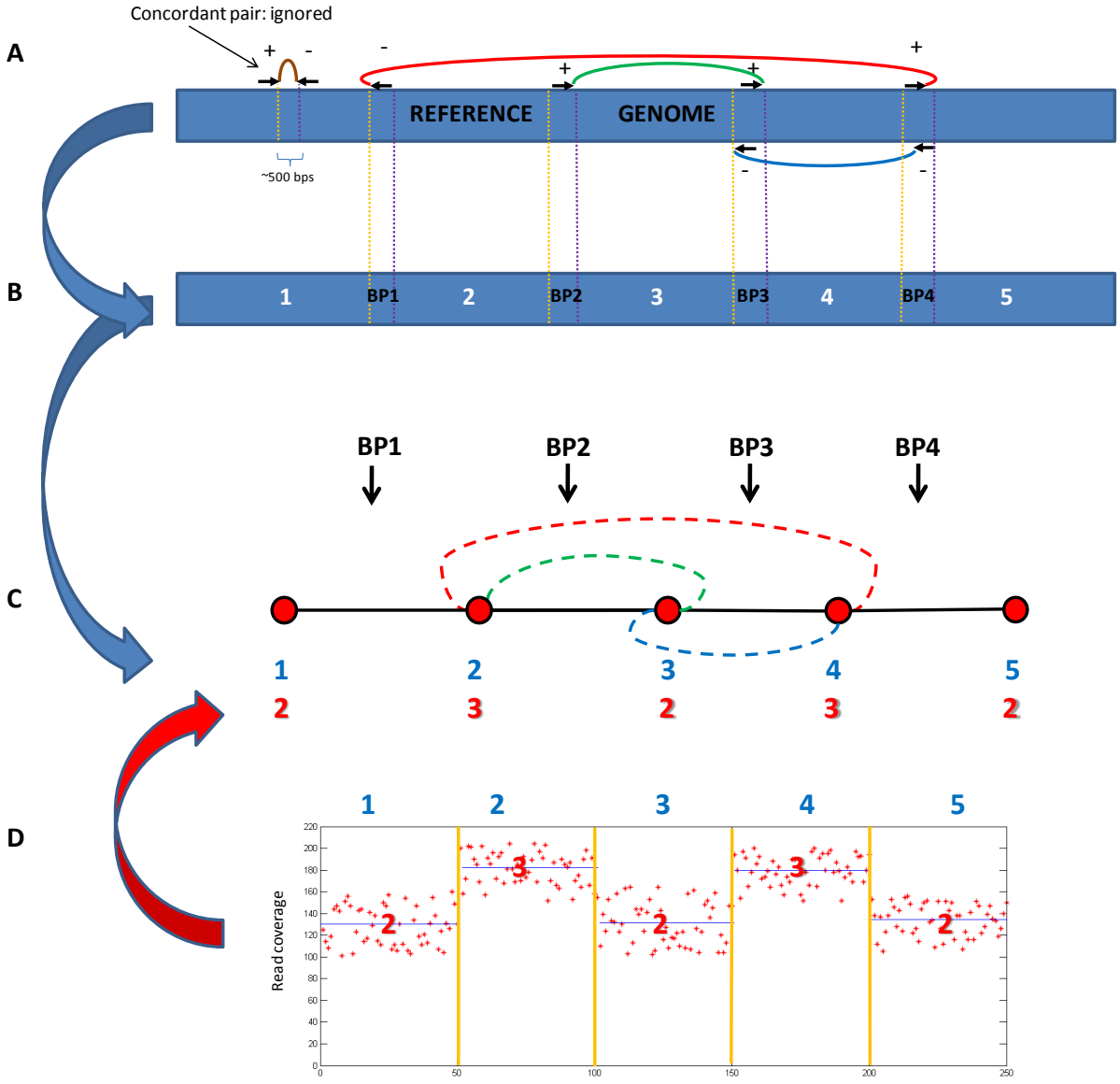

Figure 2.2: The construction of a bidirected graph. Discordant paired end reads (indicated by black arrows and coupled by coloured curves) is first used for the segmentation of the genomic sequence. If the mapping location of two reads overlap they identify a unique shared breakpoint (see breakpoints 3 and 4). The mapping orientation (forward,+ or backward,-) is then used to infer the orientation of the edges in the bidirected graph. $N + 1$ nodes are defined based on the N breakpoints inferred from the discordant read pairs ($N + 4$ in our example) orientation.

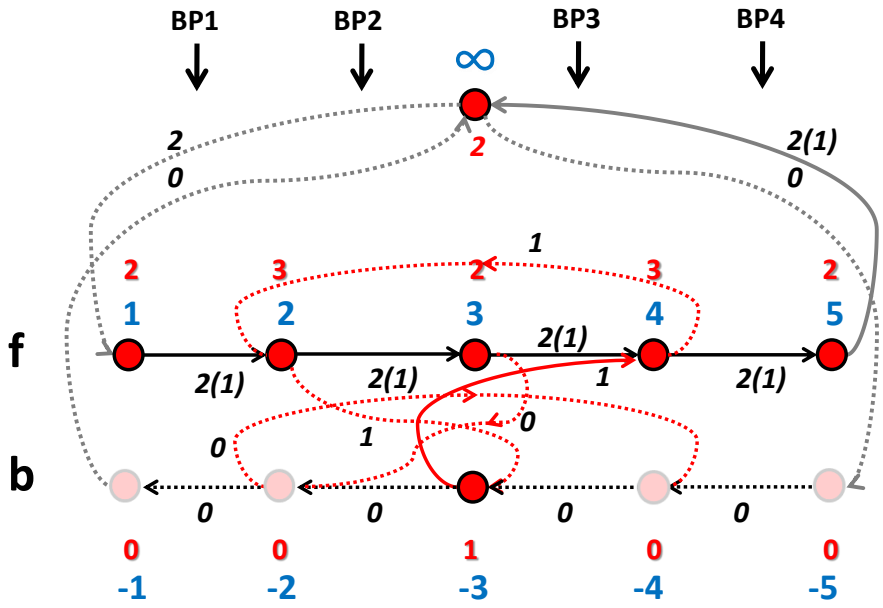

Figure 2.3: Schematic representation of a directed graph derived from the bidirected graph shown in Figure 2.1A. Top and bottom nodes represent copies of DNA segments with a forward and backward orientation, respectively; transparent nodes/edges have a null copy number. Fill edges represent one of the possible arborescences of the graph, and remaining edges are dotted. Black numbers indicate the number n of copies of each edge, and in round brackets the number $n-1$ of copies which do not belong the arborescence; blue numbers indicate segment labels; red numbers correspond to the copy number vector across nodes (identical for the two graphs).

2.2 Constructing Directed Graphs

Problem 2.2 is defined in such a way that the copy number vector across nodes (that is DNA segments) is part of the available input information. However, the reconstruction of a cancer genome sequence necessarily requires the estimation of the multiplicity of all edges in the original bidirected graph, allowing for the construction of all consistent directed graphs. For this step of our analyses, we have make use of Gröbner bases theory and the Buchberger's algorithm [56].

Let P be a bidirected graph with n nodes $\{1, 2, ...n\}$ (n is generally chosen as the infinity node) $C = [cp_1, cp_2...cp_n]$ be the associated copy number vector. It is crucial to remember that in such a graph no distinction is made between forward and backward orientation of a segment, so that a particular segment is represented by just one node. Then we construct a set of equalities based on the required conditions for a directed graph (where each segment is represented by two nodes, accounting for the two possible orientations). Let $E_{j,deg,dir}$ be the sum of all incoming ($deg{=}in$) or outcoming ($deg{=}out$) directed edges of node j, entering or exiting in a forward (exit on the right side or entrance on the left, $dir = f$) or backward (exit on the left side or entrance on the right, $dir{=}b$) direction. For each node j, we need five equations:

1. $E_{j,in,f} = cp_{j,f}$

2. $E_{j,out,f} = cp_{j,f}$

3. $E_{j,in,b} = cp_{j,b}$

4. $E_{j,out,b} = cp_{j,b}$

5. $cp_{j,f} + cp_{j,b} = cp_j$

These equations are equivalent to the conditions:

1. the sum of copy numbers over all forward directed incoming edges of node j is equal to the copy number of forward segment j, $cp_{j,f}$

2. the sum of copy numbers over all forward directed outgoing edges of node j is equal to the copy number of forward segment j, $cp_{j,f}$

3. the sum of copy numbers across all backward directed incoming edges of node j equals the copy number of backward segment j, $cp_{j,b}$

4. the sum of copy numbers across all backward directed outgoing edges of node j equals the copy number of backward segment j, $cp_{j,b}$

5. the sum $cp_{j,f} + cp_{j,b}$ equals the total copy number of segment j, cp_j.

In these equations, every $E_{j,deg,dir}$ represents a series of unknown monomials, every $cp_{j,dir}$ is a single unknown monomial, while all cp_j are known positive integer values. The output of the Buchberger's algorithm is a new set of equalities, which will be used to computationally construct all compatible sets of connection copy numbers.

Example 2.3. *Consider the bidirected graph of Figure 2.1A. The copy number vector* [23232] *has been derived from read coverage information, represented in the plot of Figure 2.2D. The inference of edge directionality then relies on the set of equations listed in Figure 2.1B. One of the directed graphs which can be inferred is represented in Figure 2.4Ai. We have* 2 *copies of segment* 1*, and if we check the equations for node* 1 *in Figure 2.1C, we find that* $i_b = a_f = cp_{1,f} = 2$*,* $i_f = a_b = cp_{1,b} = 0$*,* $cp_{1,f} + cp_{1,b} = 2 + 0 = 2$ *as required for*

any directed Eulerian graph.

Finding the solutions of these sets of equalities clearly implies the complete exploration of a wide space of possibilities, which cannot be limited to Z^+ but rather includes both positive and negative integer solutions (Z). Moreover, the number of equations is equal to $5n$, with $2m + (2n)$ unknown variables (where m is the number of edges in the starting bidirected graph). The computational effort of the Buchberger algorithm and the number of solutions found in Z therefore soon becomes a limiting factor with the increasing complexity of the graph.

2.3 Finding all Arborescences of a Directed Graph

Now, recall that genomes correspond to Eulerian cycles. For the construction of these cycles from a particular directed graph G, we will rely on the set of all arborescences rooted at the auxiliary node ∞. We have defined an arborescence (Definition 2.5) as a tree touching all nodes in the graph, and having edges pointing towards the root. Now, each arborescence can be used as a 'guide' for the construction of a particular subset of Eulerian cycles. Indeed, to construct an Eulerian cycle we walk through the graph, choosing every edge from the arborescence last. The path is completed when all copies of every edge has been used (and we are back at the starting node). We will take advantage of this observation later in the chapter, in order to define an algorithmic approach for the construction of Eulerian cycles.

For the challenge of finding all arborescences of a directed graph, we start from the Definition of *Laplacian Matrix*:

Definition 2.12. *Given a directed simple graph $G = (V, E)$ with m nodes, the corresponding* Laplacian matrix *representation is an m by m matrix such that:*

- *For every edge $e = (i, j)$ the off-diagonal entry in position (i, j) is -1*

- *The diagonal entry in position (i, i) is the absolute value of the sum of all entries across line i*

- *All other entries are zero*

Such definition proves very useful in the light of the following:
Theorem 2.1. *[22]* Matrix-tree theorem. *The number of non-identical spanning trees of a connected graph G is equal to any cofactor of its Laplacian matrix*

Thus the theorem tells us the number of spanning trees in G, which in the case of a directed graph, correspond to arborescences. We will now explain how to construct them with the help of an example. Let $G = (V, E)$ be a directed graph where $V = \{1, 2, 3, ...m\}$ is the set of vertices and E a set of edges of type (i, j) where $\{i, j\} \in V$. We first assign a monomial x_{ij} to each $e = (i, j) \in E$. We then construct an n by n symbolic Laplacian matrix $M_n(x)$. This is a modified Laplacian matrix for G, such that:

- The off-diagonal entry in position (i, j) is $-x_{ij}$

- The diagonal entry in position (i, i) is the sum of the variables $x_{i,j} \forall j \neq i$

- All other entries are equal to zero

$$\begin{bmatrix} (x_{1,2}+...x_{1,n}) & -x_{1,2} & -x_{1,3} & ... & -x_{1,n} \\ -x_{2,1}+... & (x_{21}+x_{23}+...x_{2n}) & -x_{2,3} & ... & -x_{2,n} \\ ... & & & & \\ -x_{n,1} & -x_{n,2} & ... & -x_{n,n-1} & (x_{n,1}+...+x_{n,n}) \end{bmatrix}$$

Now, since we have a symbolic matrix, any calculated non zero cofactor corresponds to a product of $x_{j,i}$'s, a term representing a set of edges. We are interested in arborescences rooted at the auxiliary node ∞, whose incoming and outcoming edges complete every Eulerian tour. Assume that index n is used to represent node ∞ (so entry $(n,1)$ corresponds to edge $(\infty,1)$, for example). Then the generating function $F_n(M)$ for the arborescences rooted at node ∞ is given by

$$F_n(M) = |M_{n,n}| \tag{1}$$

where $M_{n,n}$ is the square sub-matrix of $M_n(x)$ obtained by deleting the rows and columns with index n.

Example 2.4. *Let $G = (V, E)$ be the directed graph shown in Figure 2.4A. We have a set $V' = \{1, 2, 3, -3, 4, 5, \infty\}$ of nodes with non zero degree, and the set of edges $E = \{(1,2),(2,3),(3,4),(4,5),(5,\infty),(\infty,1),(2,-3),(-3,4),(4,2)\}$, to which we associate the monomials $\{a, b, c, d, e, f, g, h, i\}$, respectively.Then we construct the Laplacian matrix M:*

$$\begin{bmatrix} a & -a & 0 & 0 & 0 & 0 & 0 \\ 0 & (b+g) & -b & 0 & 0 & -g & 0 \\ 0 & 0 & c & -c & 0 & 0 & 0 \\ 0 & -i & 0 & (d+i) & -d & 0 & 0 \\ 0 & 0 & 0 & 0 & e & -e & 0 \\ 0 & 0 & 0 & -h & 0 & h & 0 \\ -f & 0 & 0 & 0 & 0 & 0 & f \end{bmatrix}$$

we have associated row and column indexes $1...5$ *to nodes* $1...5$, 6 *to node* -3 *and* 7 *to node* ∞. *Assuming that we want to find all arborescences rooted at vertex* ∞, *we shall then calculate:*
$det(M_{[7,7]}) = abcdeh + acdegh$
where each resulting term is the product of all ids assigned to the edges in the two arborescences of G.

2.4 Constructing Eulerian Cycles

Once all directed graphs and associated spanning trees are found, we can construct all possible Eulerian cycles. The *BEST theorem* [1] tells us how many such cycles exist for a given directed graph P:
Theorem 2.2. *[1, 2, 109]*

Let P=(V,E) be a directed graph in which, for every vertex $v \in V$, *the indegree and outdegree have the same value, d(v) and all of its vertices with non zero degree belong to a single, strongly connected component. Then G has at least one directed Euler tour, and the number of such tours is given by*

$$\varepsilon(P) = N_A \prod_{v \in V} (d(v) - 1)! \tag{2}$$

where A is the set of arborescences rooted at a fixed vertex in G, and N_A *is the number of such arborescences in A.*

Now, in the case of a genomic connection being duplicated we find that we traverse the

same edge more than once; thus we require multigraphs for the reconstruction of genomic sequences. This rises the problem of determining how many Eulerian cycles with distinct order of vertices exist. For example, the graph of Figure 2.4Ai contains two copies of edge $(\infty, 1)$ which is not part of the arborescence (represented by filled edges). Thus, when we walk through the graph and reach the auxiliary node ∞, we are free to choose either of the two copies, which inevitably results in the construction of distinct Eulerian cycles with identical sequence of nodes. We thus formulate the following modified version of 2, adapted for a multigraph. For the theorem presented we consider only the specific case where the root of all arborescences (node ∞) has single ingoing edge (n, ∞) and single outgoing edge $((\infty, 1))$, as in Figure 2.4.

Lemma 2.2. *Let G=(V,E) be a directed graph in which, for every vertex $v \in V$, the indegree and outdegree have the same value, d(v) and all of its vertices with non zero degree belong to a single, strongly connected component. Let A be the set of arborescences for G rooted at a node x having a single outgoing edge (x, s) with multiplicity $deg(x)$; let also E' denote an arborescence from A. Then G has at least one directed Euler tour, and the number of paths with distinct order of vertices is given by*

$$\varepsilon(P) = \sum_{r=1}^{R} \prod_{v \in V} \frac{(n_v - 1)!}{\prod_{e_{v,i} \in E'_r, e_{v,i} = (x,s)} (n_{e_{v,i}} - 1)! \prod_{e_{v,j} \notin E'_r, e_{v,i} \neq (x,s)} n_{e_{v,j}}!} \tag{3}$$

where n_v is the indegree of node n_v; R is the number of arborescences rooted at a fixed node x in G; E_r is an arborescence rooted at x; $e_{v,i}$ denotes an outgoing edge from v such that exactly one copy belongs to E_r; $e_{v,j}$ denotes an outgoing edge from v such that no copy belongs to E_r; $n_{e_{v,i/j}}$ is the number of copies of edge $e_{v,i/j}$; n_v is the outdegree of node v.

Proof. We know from Theorem 2.2 how to calculate the total number of Euler tours. Now, if G is a multigraph then there will be some edge e present in more than one copy in each Eulerian cycle. Select an arborescence E_r of G. Then construct a circuit starting from node

$s \in V$ (that is the ending node of edge (x, s)). The lemma assumes that node s such that $(x, s) \in E$ is unique in V. Now, for every edge e found in E_r, exactly one copy of e belongs to E_r and must be chosen as the last one traversed along any Eulerian circuit. Then we have exactly $(N_e - 1)!$ ways of ordering all other copies, where N_e is the multiplicity of edge e. For any edge e which is not found in E_r, we find that if $e \neq e' = (x, s)$ the number of orders is exactly $(N_i)!$; however, one copy of e' must be chosen as the last one which completes the circuit. This avoids the repetition of identical orders of nodes. Therefore we have $(N_{e'} - 1)!$ ways of ordering the copies of e'. Also, we note that a similar reasoning applies to the last edge of every cycle rooted at x, which we denote as (x, s). Thus we use the same term $(N_{e'} - 1)!$ for such edge. Lastly, we note that the number of ways of ordering all n_v outgoing edges from a node v, subject to the above described constraints for each distinct edge, is $\frac{(n_v - 1)!}{\prod_{e_{v,i} \in E_r \vee e_{v,i} = (x,s)} (n_{e'_{v,i}} - 1)! \prod_{e_{v,j} \notin E_r} n_{e_{v,j}}!}$. Such term must be calculated across all nodes, for each distinct arborescence. Thus we require a sum over arborescences of the product $\prod_{v \in V}$ across all nodes. □

The construction of each Eulerian circuit relies on an algorithm which takes a single arborescence, its root and the set of remaining edges E' as the required input information. It then constructs a path based on all possible edge choice orders for 'complex' nodes (that is, nodes with more than one possible outgoing edge), ignoring at the same time all the connections belonging to the arborescence, as long as other edges are available.

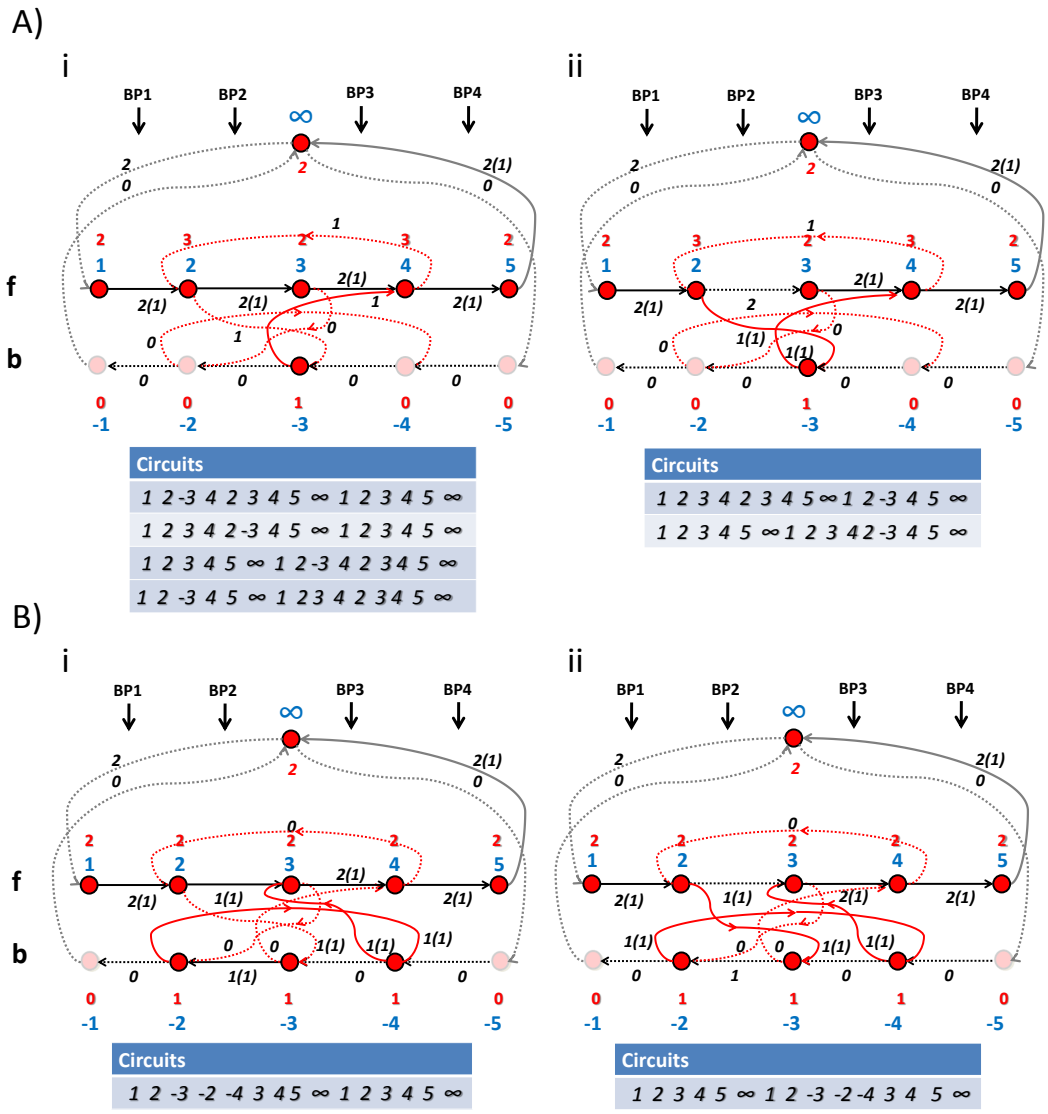

Figure 2.4: A) Schematic representation of two directed graphs (A-B) derived from the bidirected graph shown in Figure 2.1A, with associated arborescences (i-ii) and Eulerian cycles. Top and bottom nodes represent copies of DNA segments with a forward and backward orientation, respectively; transparent nodes/edges have a null copy number. Arborescences correspond to fill edges, and remaining edges are dotted. Black numbers indicate the number n of copies of each edge, and in round brackets the number $n-1$ of copies which do not belong the arborescence; blue numbers indicate segment labels; red numbers correspond to the copy number vector across nodes (identical for the two graphs). Tables below show all Eulerian cycles constructed with the represented arborescence, with each row corresponding to a distinct solution.

Example 2.5. *Let G be the graph such that*
$E = \{(1,2),(1,2),(2,3),(2,3),(3,4),(3,4),(4,5),(4,5),(5,\infty),(5,\infty),(\infty,1),(\infty,1)(2,-3),(-3,4),(4,2)\}$ *and* $V = \{1,2,3,-3,4,5,\infty\}$. *We have seen from Example 2.4 that graph G has two arborescences rooted at ∞: $S = \{(1,2),(2,3),(3,4),(4,5),(-3,4),(5,\infty)\}$ and $S' = \{(1,2),(2,-3),(-3,4),(3,4),(4,5),(5,\infty)\}$, corresponding to filled edges in Figure 2.4Ai and ii, respectively. According to the BEST theorem (2.2), the total number of cycles for graph G (that is, considering both arborescences) is $2\cdot(1!2!1!2!1!0!1!) = 8$. However, the number of cycles with distinct node sequences is $\frac{1!2!1!2!1!0!1!}{(1!1!1!1!1!1!1!)(1!1!)} + \frac{1!2!1!2!1!0!1!}{(1!1!1!1!1!1!1!)(2!1!)} = 4+2 = 6$ (equation 3). This is due to the presence of 2 couples of identical cycles associated with one of the arborescences (shown in 2.4Aii), that is only two distinct node sequences. In each fraction, the numerator is the same as for Equation 2. The denominator includes two terms: the former accounts for edges which do not belong to the arborescence, plus the special edge $(x,s) = (\infty,1)$ which concludes every cycle. The latter term accounts for all remaining edges. Consider arborescence S. Now, every cycle starts at node 1 and uses the edges from S as long as they are available; then the other edges are chosen. All resulting cycles, organised according to the corresponding arborescence, are shown in Figure 2.4Ai-ii. For example, looking at Figure 2.4Ai, we can start at node 1 and choose edge $(1,2)$; we are forced to select the copy of this edge which is not in the arborescence. From node 2, we choose edge $(2,-3)$ and then the only outgoing edge available, $(-3,4)$; next we might select $(4,2)$ which is not in the arborescence, and then $(2,3)(3,4)(4,5)(5,\infty)$, all having one copy in the arborescence but not the latter one. From ∞ we go bacj to node 1. Now, the only subpath we can proceed on is $(1,2)(2,3)(3,4)(4,5)(5,\infty)$,which is made up of edges all belonging to the arborescence. Lastly, we choose edge $(\infty,1)$ which completes our cycle: $1,2,-3,4,2,3,4,5,\infty,1,2,3,4,5,\infty(,1)$. We have thus shown how to efficiently construct an Eulerian cycle by choosing any edge from the arborescence last.*

2.5 Computational Analyses

We provide now the pseudocode for two main Matlab [56] functions which were developed for the automatic reconstruction of Eulerian cycles.

Arborescences:
We have seen how to identify arborescences using the Matrix-Tree Theorem. If we use a symbolic matrix, the non zero products obtained indicate the sets of edges corresponding to the arborescences, for example *abcd*. However, in the case of a numeric matrix (which we want to use for our computations) arborescences will be a product of numerical entries, which do not contain any information about the corresponding set of edges. Now, if we have a graph G containing n distinct edges (that is with different source and/or sink nodes) we can generate all possible numeric Laplacian matrices representing sets of $n-1$ edges (which is the number of edges in any arborescence). The number of possible matrices might however be very high. In order to improve performance, we can look at which nodes (different from the root) have only one type of outgoing edge (that is, only one possible following node). Since such edges are necessarily part of the arborescence, we can then ignore any Laplacian matrix which lacks one or more of such edges. Any generated matrix having a non zero cofactor represents an arborescence. By keeping track of which set of edges is considered each time (using a Boolean matrix) we can then determine the set of edges corresponding to that matrix. The pseudocode for the implemented Matlab function follows.

Arborescence algorithm:

Data: the set E of edges from a directed graph $G = (E, V)$
Result: Construct arborescences for each directed graph
Initialization;
Arborescences $A = \{\}$;
set of remaining edges $R = \{\}$;
Determine which node have a single outgoing edge in E'; collect these edges in E'';
Construct all possible sets S_i of n_a edges from E', such that $E'' \subset S_i$.;
for each S_i **do**
    construct Laplacian matrix representation of S_i, M_i, and calculate its determinant $d(M_i)$;
    if $d(M_i) == 1$ **then**
        add S_i to A ;
        add $E \setminus S_i$ to R ;
    end
end

Eulerian algorithm:

Data: the sets A and R derived from function *Arborescences*
Result: the set E of corresponding Eulerian cycles
Initialization: $E = \epsilon$;
for each *arborescence* $A[i]$ *and edge set* $R[i]$ **do**
  $C = \epsilon$;
  Determine the set V_c of nodes with more than one different outgoing edge Ri;
  Construct all possible combinations of single orders across all nodes $p(v_1), p(v_2)...p(|V_c|)$;
  Write combinations in C;
  for $i = 1 : |C|$ **do**
    starting node is n=1;
    while $A[i]$ *not empty OR* $R[i]$ *not empty* **do**
      if $R[i]$ *has an outgoing edge from* n **then**
        if n *is in* V_c*(: it has at least two distinct outgoing edges)* **then**
          Choose outgoing edge from n appearing earliest in order $C[i]$;
          Remove the edge from $R[i]$
        else
          Choose single outgoing edge and remove it from $R[i]$
        end
      else
        Choose single outgoing edge from and remove it from $A[i]$;
        Update n to the node the edge is directed to
      end
    end
  end
  Add resulting cycle to E
end
return E

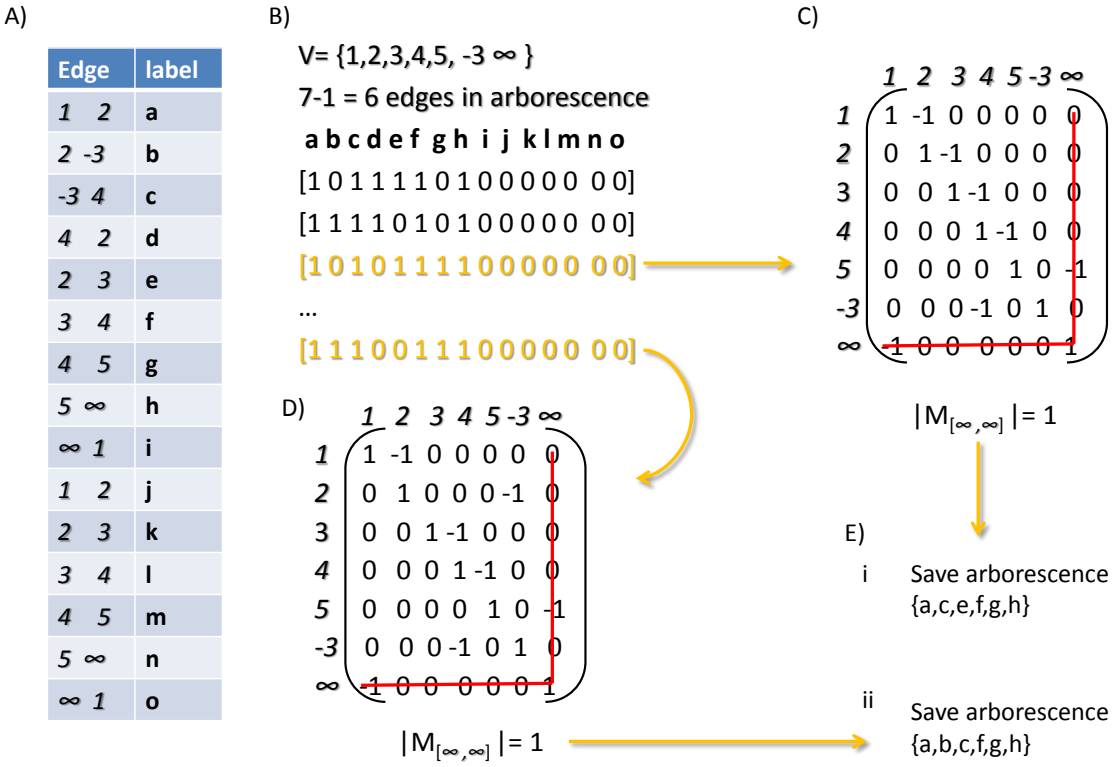

Figure 2.5: The *Arborescence* algorithm. A) set E of edges from the directed graph of Figure 2.4A, with every edge being labelled with a letter. This represents the input data. B) 1 by 15 (the number of edges in E) boolean vectors are constructed which may correspond to an arborescence. C-D) A Laplacian matrix is derived from each Boolean vector from B). If the determinant of submatrix $M_{[\infty,\infty]}$ equals 1, the set of edges with Boolean value 1 is saved as an arborescence (see Ei-ii).

2.6 Conclusions

We have succeeded in developing an Eulerian cycle approach for the reconstruction of rearranged genomes consistent with paired end and copy number input information. We have however observed a series of limitations intrinsic to this approach; in particular, both the

Gröbner bases for the construction of directed graphs explore a (possibly very vast) space full of CN profiles/graphs, trying to detect a limited number of useful solutions. Moreover, all Eulerian cycles obtained from directed graphs represent equally likely explanations of the data. Some limits related to our code become also clear when dealing with complex graphs, with high numbers of nodes and/or somatic connections. In such cases, both approaches for the generation of directed graphs become computationally demanding, if not impossible to perform. Even the construction of spanning trees might represent a limiting factor.

An additional unresolved matter is the probability of each chronological order of somatic connections. For copy-neutral rearrangements, parsimony models have been developed [46] and applied to cancer genome analysis [93]. Order information have also been inferred by looking at cluster of rearrangements with duplications [92] or by combining duplications and single-nucleotide mutations data [44, 30]. Our method gives little insight on the rearrangement classes underlying the complex structural and copy number changes; such area of research will be considered in chapter 3 through an algorithmic approach.

Additional limitations are represented by the available input data. Mapping of discordant paired reads is typically difficult for structural variants in repetitive regions of the human genome, possibly leading to missing or incorrect somatic connections in the data. Similarly, estimates of read depth are difficult to obtain in repetitive regions. Moreover, the generated directed graphs do not contain any allelic information (ratio between the intensity of alternative SNP alleles across the genome, provided by microarray experiments) of the cancer genome. Lastly, our method also assumes the presence of a single genome in the cancer sample, while in reality a cancer cell population will rather represent an heterogeneous mix of genetically different populations of cells.

3 An algorithmic approach for the inference of a cancer genome evolution

In chapter 2 we addressed the challenge of reconstructing cancer aberrant haplotypes from paired end data through the Eulerian path approach. However, such method gives little insight on the molecular mechanisms underlying the complex structural and copy number changes, as well as the transitional states separating the wild-type and final cancer sequences. It is then natural to ask the following question: how can we use the copy number and paired end information to reconstruct the intermediate stages of a rearrangement history, from the reference to the final cancer genome?

We want to start introducing our problem with the help of a simple example. Consider the bidirected graph represented in Figure 3.1A. This has two coloured edges, corresponding to two somatic connections absent in the reference. Assume we want to start an evolution from a reference chromosome consisting of segments $1, .., 4$, corresponding to the nodes in the graph and labelled according to their physical position inside the chromosome. Let $\{[1, 2, 3, 4]\}$ be the word of integers representing such a chromosome. Then if we add one somatic connection at a time in either of the two possible orders, we can construct the following evolutions:

$\{[1, 2, 3, 4]\} \rightarrow \{[1, 2, 3 - 3, -2, -1], [4]\} \rightarrow$
$\{[1, 2, 3, -3, -2, -2, -1], [4]\} \rightarrow \{[1, 2, 3, -3, -2, -2, -1]\}$

$\{[1, 2, 3, 4]\} \rightarrow \{[1], [\overline{2}], [3, 4]\} \rightarrow \{[1], [\overline{2}], [3, -3], [4]\} \rightarrow \{[\overline{2}]\}$

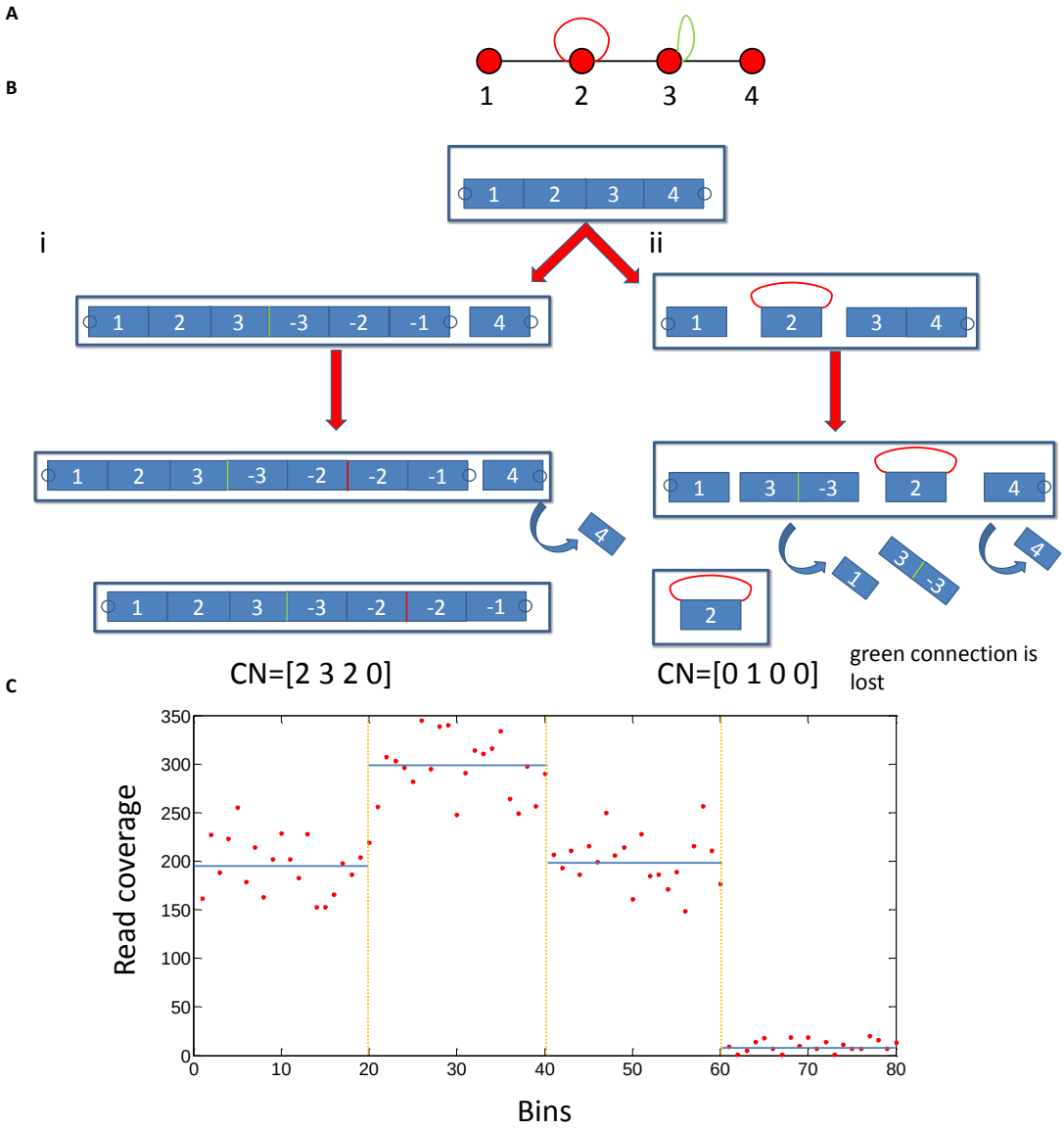

Figure 3.1: A) Example of a bidirected graph constructed from paired end data: black edges are connections observed in the reference; coloured edges are somatic connections, exclusively observed in the cancer sample. B) schematic representation of two evolutions consistent with the graph in A). Telomeres are indicated by circles. Somatic connections, indicated by coloured lines between segments (rectangles), are added once at a time, starting from an initial chromosome $[1, 2, 3, 4]$. In the last step, sequences lacking one or both telomeric ends are eliminated. In C) an example of read coverage values (y) across 80 adjacent windows (x) is shown.

here every word delimited by square brackets represents a distinct sequence of segments linked together into single chromosomes, and braces collect a set of these chromosome words together, defining a *genome*. Overlined words indicate a circular chromosome (in Figure 3.1B, we have a circular segment 2). Now, the red edge has been interpreted differently in the two evolutions: either as a tandem duplication of segment 2, or its circularization, leading to chromosome fragmentation. On the other hand, we have interpreted the green connection as a Breakage Fusion Bridge event, producing a palindromic chromosome $[1, 2, 3, -3, -2, -1]$. The last modification, common to both evolutions, is the elimination of chromosome which don't carry a telomere at both ends: that is, unrepaired fragments. The result is a modified set of chromosome words: a *rearranged genome*. We have thus shown that, by modelling each somatic connection as a *rearrangement event* (i.e. a modification of some chromosome words) we are able to construct multiple steps of an evolution which introduces a set of observed connections to a reference genome. Some of these connections might be lost in the last step (Figure 3.1Bii); however, in other cases the transformations might lead to a set of repaired chromosomes carrying all such connections (Figure 3.1Bi). In the latter case, we can say that the evolution is consistent with paired end information.

Moreover, each evolution provides us with a copy number vector (see Definition 2.1) across segments for every single transformation step. Then the vector associated with the final stage of an evolution must be consistent with read coverage values from the cancer sample. In Figure 3.1C, a putative read coverage plot is shown. Each segment corresponds to 20 consecutive bins, that is 20 coverage values. The mean values across segments $1, 2, 3, 4$ (indicated by blue lines) are respectively $196, 296, 200$ and 10. We then define $C = [196, 296, 200, 10]$. Now, evolution ii returns vector $[0, 1, 0, 0]$, which compare badly to the read coverage plot, or its vector of means C. Conversely, it is easy to observe that the copy number vector for evolution i, $[2, 3, 2, 0]$, is a much better copy number prediction for our example.

In this chapter, we implemented an algorithmic approach for a multi-step construction of cancer digital karyotypes, generalizing different classes of DNA break and repair mechanisms through the use of a long menu of operations. We will describe our approach with the help of *genome graphs* containing 2 different classes of connections: *wild-type adjacen-*

cies, linking segments which are adjacent in the reference genome sequence and *somatic adjacencies* corresponding to genomic connections acquired during cancer development. The main idea of our approach is to build up a rearrangement evolution as a sequence of steps from a reference genome G_0 to a modified set of chromosomes G_n, and finally to G'_n, collecting all repaired chromosomes in G_n but free from any DNA fragment, as discussed further in the text.

We now clarify the crucial aspects of our model, with the help of the following definitions.

Definition 3.1. *A* genome graph *is a bidirected graph $P = (V, E)$ where each node in V represents a segment, labelled with an integer number from 1 to n according to the position along the reference genome. The set E is a collection of edges, each connecting one side of a node i to one side of a node j, with possibly $i = j$. Any edge in P is represented as $e = (\pm i, \pm j)$ with $i \leq j$. Each of i and j has a positive (resp. negative) sign if edge e touches the right (resp. left) side of the corresponding node. If e is a loop touching both sides of the same segment i (a connection often associated with tandem duplication), we write: $e = (-i, i)$. Every edge of the form $(i, -(i + 1))$ is called a* wild type edge*, and simply represents the adjacencies between consecutive DNA segments in the reference genome. All other edges are acquired during cancer development and are called* somatic edges *(or* somatic connections*).*

The definition of telomere will prove crucial later on, in order to distinguish between DNA sequences which still carry one or two broken ends, or have been untouched/repaired (see Definition 3.9). Such concept is related to the graph theory definitions of chromosome and genomes, which we present next.

Definition 3.2. *We call a* linear chromosome word *any word of integers taking the form $C = [\pm t_1, \pm t_2, ..., \pm t_n]$. We also call a* circular chromosome word*, any word of integers taking the form $\overline{C} = \overline{[\pm t_1, \pm t_2, ..., \pm t_n]}$. Every letter t_1 in C is called a* forward oriented segment*, and every letter $-t_1$ in C is called a* backward oriented segment*.*

Definition 3.3. *Let $e = [\pm i, \pm j]$ be an edge from a genome graph P. Then the chromosome word representation of e corresponds to the single letter $\overline{i}$ if and only if $i = j$ and segment i is circularised. Otherwise, it corresponds to a couple of adjacent letters $[\pm i, \pm j]$, where*

the signs of i and j are chosen based on Definition 3.1.

Example 3.1. *Consider $e = [-2, 2]$ in the genome graph of Figure 3.1A, and the starting genome represented in the Figure, $G = \{[1, 2, 3, 4]\}$. If we add edge e to chromosome G, we are connecting the left side of segment 2 to the right side of the same segment. In the evolution shown in Figure 3.1Bi, this leads to a tandem duplication of segment -2, $[... - 2...] \rightarrow [... - 2, -2...]$ and e connects two copies of inverted segment -2. However, in the evolution shown in Figure 3.1ii, adding the same edge leads to the circularization of segment 2, corresponding to the single letter $\overline{2}$.*
The other somatic edge in the graph is $e' = (3, 3)$, touching only the left side of segment 3. When a palindromic chromosome is obtained (Figure 3.1i), we find that the correspondent chromosome word is $[1, 2, 3, -3, -2, -1]\}$, where subword $[3, -3]$ corresponds to e'.

Definition 3.4. *A word C is called a* valid chromosome *for a genome graph $P = (V, E)$ if and only if C is a chromosome such that all consecutive symbols x, y and all single symbols $\overline{x}$ in C correspond to an edge in P (i.e. they are a word representation of an edge in P by Definition 3.3).*

Definition 3.5. *Let $P = (V, E)$ be a genome graph and G be a genome. Then we say that G is a* valid genome *for P if and only if all chromosomes in G are valid chromosomes for P.*

Definition 3.6. *Let $P = (V, E)$ be a genome graph with N nodes. Define the k_{th} chromosome, with $k = \{1, 2, ..., K\}$, as $C_k = [1^{(k)}, 2^{(k)}..., n_k^{(k)}]$. Let also the sum of nodes across chromosomes be equal to the number of nodes N: $\sum_{1}^{K} n_k^{(k)} = N$. Then the* valid haploid reference genome *for P and the C_k's takes the form $G_0 = \{C_1, C_2, ..., C_K\}$, while the* valid diploid reference genome *is defined as $G_{0,0} = \{C_1, C_1, C_2, C_2, ..., C_K, C_K\}$.*

Definition 3.7. *Let $P = (V, E)$ be a genome graph, and G a reference haploid genome for P. For every chromosome $C_k = [1^{(k)}, 2^{(k)}..., n_k^{(k)}]$ from G we call the left side of $1^{(k)}$ a* left telomere *and the right side of $n_k^{(k)}$ a* right telomere*. We also define all $1^{(k)}$'s and $n_k^{(k)}$'s as* telomere nodes*.*

Now, with the help of the previous definitions, we present additional concepts regarding specific types of chromosomes and genomes.

Definition 3.8. *Given a reference haploid genome G with highest integer value N, we define the corresponding set of breakpoints as the set: $B = \{1, 2, ...N - 1\}$, where every i in B represents both the left end side of segment letter i and the right end side of segment letter $i + 1$.*

Definition 3.9. *Let $P = (V, E)$ be a genome graph with N nodes, and T be the set of telomeres. Then any linear chromosome such that one or both ends are not telomeres from T is called a* fragment *for P and T .*

Example 3.2. *Consider evolution of Figure 3.1Bi. The first rearrangement produces genome $\{[1, 2, 3, -3, -2, -1], [4]\}$. In the former, palindromic chromosome the leftmost end is the left side of segment 1, which is defined as a left telomere (indicated by a circle). On the opposite (right) end, we have an inverted segment $1, -1$, where the same telomere actually lies on the right side. Thus we find that both ends of the chromosome are telomeric. As for chromosome $[4]$, its right end corresponds to a telomere, but not the left one. Thus $[4]$ is a fragment.*

Definition 3.10. *Given a genome graph $P = (V, E)$, a genome sequence G is called a* valid final genome *for P if and only if all nodes in G are also in P, and every edge in E_r corresponds to at least one couple of consecutive symbols x, y in G.*

Definition 3.11. *Let s, s' be two letters in a genome word $G = \{...s...s'...\}$ or $G = \{...s = s'...\}$. If we add a somatic connection* e *linking s_1 and s', then such connection forms in exactly one of the following ways:*

- Tandem Duplication(TD) *case:* e *connects the left of s to the right of s' (Figure 3.2A)*

- Deletion(DEL) *case: $p(s) < p(s')$ and* e *connects the right of s to the left of s'(Figure 3.2B)*

- left Inversion (l-INV) *case:* e *connects the left of* s *to the left of* s' *(Figure 3.2C)*

- *it assumes a* right Inversion(r-INV) *case:* e *connects the right of* s *to the right of* s' *(Figure 3.2D)*

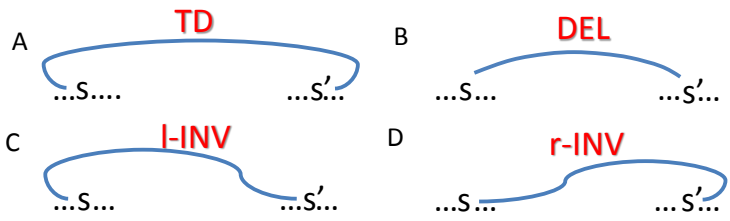

Figure 3.2: Schematic representation of the four possible orientations ('shapes') of a somatic connection. A) Tandem Duplication; B) Deletion; C) left Inversion; D) right Inversion.

The importance of concepts like valid chromosomes/genomes and fragments becomes clear when we attempt to select final genomes which compare best with our input data. We specifically want to select evolutions such that the final valid genome contains all somatic edges from a graph P.

We now introduce the precise definitions of all the different types of operations.

Definition 3.12. *A* Double Strand Break*(DSB) operation on a genome* G *is an operation splitting a chromosome word* C *into two non empty subwords* C_1, C_2.

Definition 3.13. *A* Duplication*(D) operation on a genome* G *containing* K *chromosomes is the following operation on one or more chromosomes* C_i *from* G*:* $C_i \to \{C_i, C_i'\}$*, where* $C_i = C_i'$.

Definition 3.14. *We call* fragment elimination*(FR) the operation on a genome* G *which removes all fragment chromosomes.*

Definition 3.15. *A* connection-free operation *on a genome* G *is any operation of class* Double Strand Break*(DSB),* Duplication*(D) or* Fragment elimination (FR).

Definition 3.16. *An intrachromosomal Double Cut and Join (DCJ) operation is one of the following operations on a chromosome word* $C = XYZ$*:*

1. $XYZ \rightarrow \{X, \overline{Y}, Z\}$, $Y \neq \epsilon$ *(Figure 3.3A)*

2. $XYZ \rightarrow \{XZ, Y\}$, $X, Y, Z \neq \epsilon$ *(Figure 3.3B)*

3. $XYZ \rightarrow \{X, -YZ\}$, $Y, Z \neq \epsilon$ *(Figure 3.3C)*

4. $XYZ \rightarrow \{X - Y, Z\}$, $X, Y \neq \epsilon$ *(Figure 3.3D)*

Definition 3.17. *An interchromosomal Double Cut and Join (DCJ) operation is one of the following operations on two chromosome words* $C = XY, C' = X'Y'$*:*

1. $\{XY, X'Y'\} \rightarrow \{X, X'Y, Y'\}$, $X', Y \neq \epsilon$ *(Figure 3.3E)*

2. $\{XY, X'Y'\} \rightarrow \{XY', Y, X'\}$, $X, Y' \neq \epsilon$ *(Figure 3.3F)*

3. $\{XY, X'Y'\} \rightarrow \{X, -YY', X'\}$, $Y, Y' \neq \epsilon$ *(Figure 3.3G)*

4. $\{XY, X'Y'\} \rightarrow \{X - X', Y, Y'\}$, $X, X' \neq \epsilon$ *(Figure 3.3H)*

Example 3.3. *Consider Figure 3.6A. In the represented evolution, two DCJ events lead to the formation of a repaired chromosome where segment* 2 *has been inverted.*

Definition 3.18. *An intrachromosomal Break-Induced Replication (BIR) operation is one*

of the following operations on a chromosome word $C = XYZ$:

1. $XYZ \to \{XYYZ\}$ $Y \neq \epsilon$ *(Figure 3.4A)*

2. $XYZ \to \{XZ, YZ\}$ $X, Y, Z \neq \epsilon$ *(Figure 3.4B)*

3. $XYZ \to \{XZ, XY\}$ $X, Y, Z \neq \epsilon$ *(Figure 3.4B)*

4. $XYZ \to \{X - Y, YZ\}$ $X, Y \neq \epsilon$ *(Figure 3.4C)*

5. $XYZ \to \{XY - X, Z\}$ $X \neq \epsilon$ *(Figure 3.4C)*

6. $XYZ \to \{XY, -YZ\}$ $Y \neq \epsilon$ *(Figure 3.4D)*

7. $XYZ \to \{X, -Z - YZ\}$ $Z \neq \epsilon$ *(Figure 3.4D)*

8. $XY \to \{-XX, Y\}$ $X \neq \epsilon$ *(Figure 3.6C)*

9. $XY \to \{X, Y - Y\}$ $Y \neq \epsilon$ *(Figure 3.6D)*

Next we introduce additional operations, allowing for the representation of copy-number-increasing rearrangements:

Definition 3.19. *An interchromosomal Break-Induced Replication (BIR) operation is one of the following operations on two chromosome words $C = XY, C' = X'Y'$:*

1. $\{XY, X'Y'\} \to \{X, X'Y, X'Y'\}$, $X' \neq \epsilon$ *(Figure 3.4E)*

2. $\{XY, X'Y'\} \rightarrow \{XY, X'Y, Y'\}$, $Y \neq \epsilon$ *(Figure 3.4E)*

3. $\{XY, X'Y'\} \rightarrow \{XY', Y, X'Y'\}$, $Y' \neq \epsilon$ *(Figure 3.4F)*

4. $\{XY, X'Y'\} \rightarrow \{XY, XY', Y, Y'\}$, $Y \neq \epsilon$ *(Figure 3.4F)*

5. $\{XY, X'Y'\} \rightarrow \{X - X', Y, X'Y'\}$, $X, X' \neq \epsilon$ *(Figure 3.4G)*

6. $\{XY, X'Y'\} \rightarrow \{XY, X - X', Y'\}$,$Y \neq \epsilon$ *(Figure 3.4G)*

7. $\{XY, X'Y'\} \rightarrow \{X, -YY', X'Y'\}$, $Y \neq \epsilon$ *(Figure 3.4H)*

8. $\{XY, X'Y'\} \rightarrow \{XY, X', -YY'\}$, $Y \neq \epsilon$ *(Figure 3.4H)*

Example 3.4. *Consider Figure 3.4A: we have a somatic connection touching the two sides of segment* 2*. The result of a intrachromosomal BIR operation on chromosome* $[1^{(1)}, 2^{(1)}, 3^{(1)}]$ *using that connection is a tandem duplication of segment* 2 *(*$[1, 2, 3] \rightarrow [1^{(1)}, 2^{(1)}, 2^{(1)}, 3^{(1)}]$ *in the chromosome word).*

Example 3.5. *Consider Figure 3.6C-D. The evolutions represented lead to palindromic chromosome words* $[1^{(1)}, -1^{(1)}]$ *and* $[-2^{(1)}, 2^{(1)}]$*, equivalent to what might happen after a single breakage fusion bridge cycle (see Figure 1.2).*

Definition 3.20. *Let* $P = (V, E)$ *be a genome graph, and let* G *be a valid genome for* P*. A* connection operation *on a genome* G *is an operation of class* Double Cut and Join *or* Break-Induced Replication (BIR) *utilizing one and exactly one somatic edge* e *from* E*. The operation is called* intrachromosomal *if* e *operates on a single chromosome word (see Figures 3.3A-D and 3.4A-D), and* interchromosomal *otherwise (see Figures 3.3E-H and 3.4E-H).*

Definition 3.21. *Given a genome graph $P = (V, E)$ and a valid evolution for P, $\mathcal{E}(P)$, we define* breakpoint reuse constraint *the following set of conditions:*

1. *for all wild type edges $e = (i, -(i+1))$, one and exactly one operation splits apart two adjacent symbols connected by e;*

2. *no operation splits apart two letters corresponding to a somatic connection.*

Definition 3.22. *Let $P = (V, E)$ be a genome graph. A* valid evolution *for P is a series of connection and connection free operations, $\mathcal{E}(P)$, such that*

1. *every genome in the evolution is valid for graph P*

2. *every somatic edge e_i in P is used by one and exactly one connection operation;*

3. *every operation in P is performed subject to the breakpoint reuse constraint;*

4. *the final genome word contains every somatic edge from E after performing fragment elimination.*

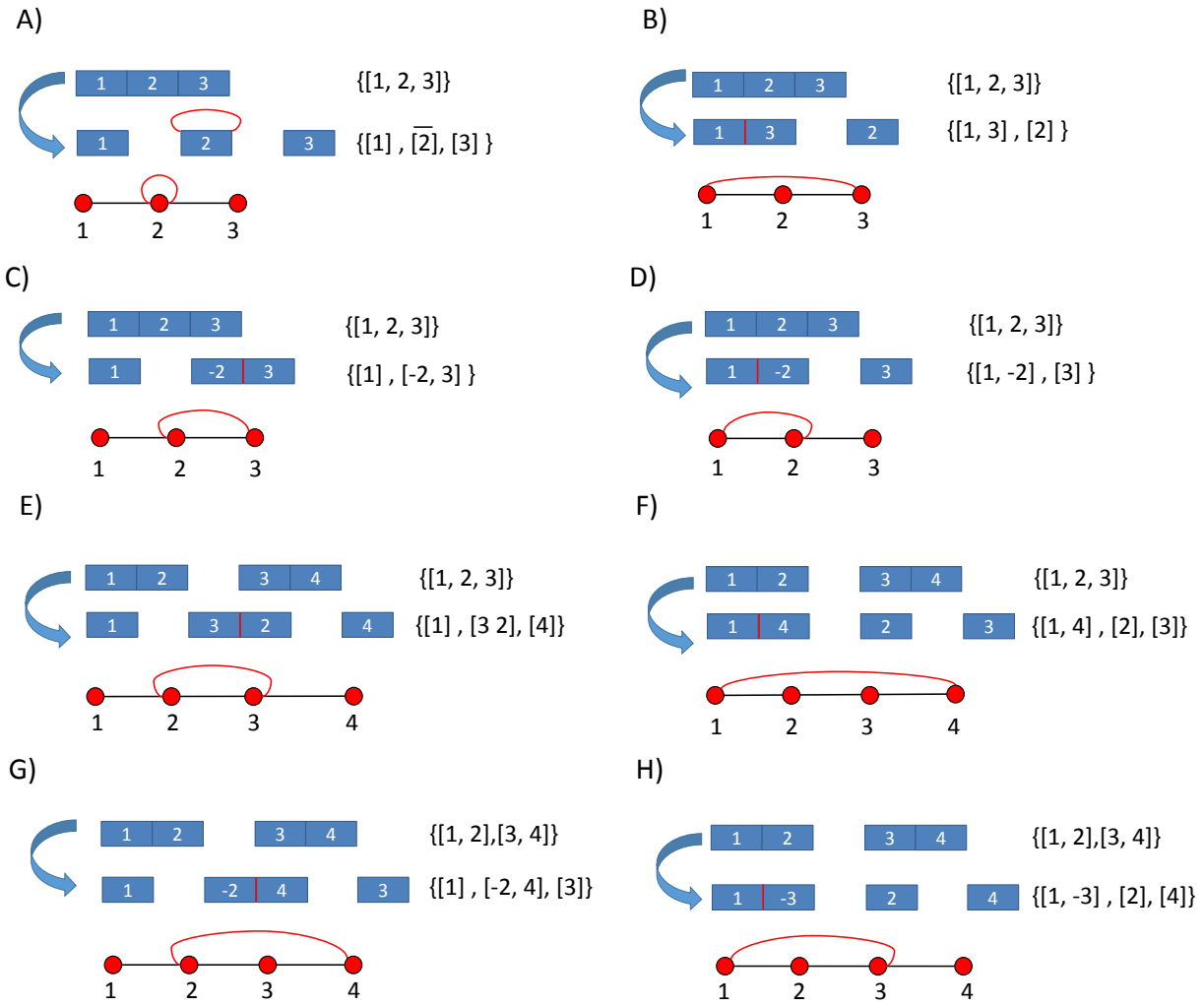

Figure 3.3: Full list of DCJ operations, described using a schematic representation where each rectangle corresponds to a different segment. Word representation and the genome graph are also shown. A-D) Intra-chromosomal DCJ operations for the 4 possible edge shapes. A) TD shape: adding edge $[-2\ 2]$, results in a circular chromosome. B) Deletion shape, edge $[1\ -3]$. C) Left inversion shape, edge $[-2\ -3]$. D) Right inversion shape, edge $[1\ 2]$. E-H) Inter-chromosomal DCJ operations for the 4 possible connection shapes. E) TD shape, edge $[-2\ 3]$. F) Deletion shape, edge $[1\ -4]$. G) Left inversion shape, edge $[-2\ -4]$. H) Right inversion shape, edge $[1\ 3]$.

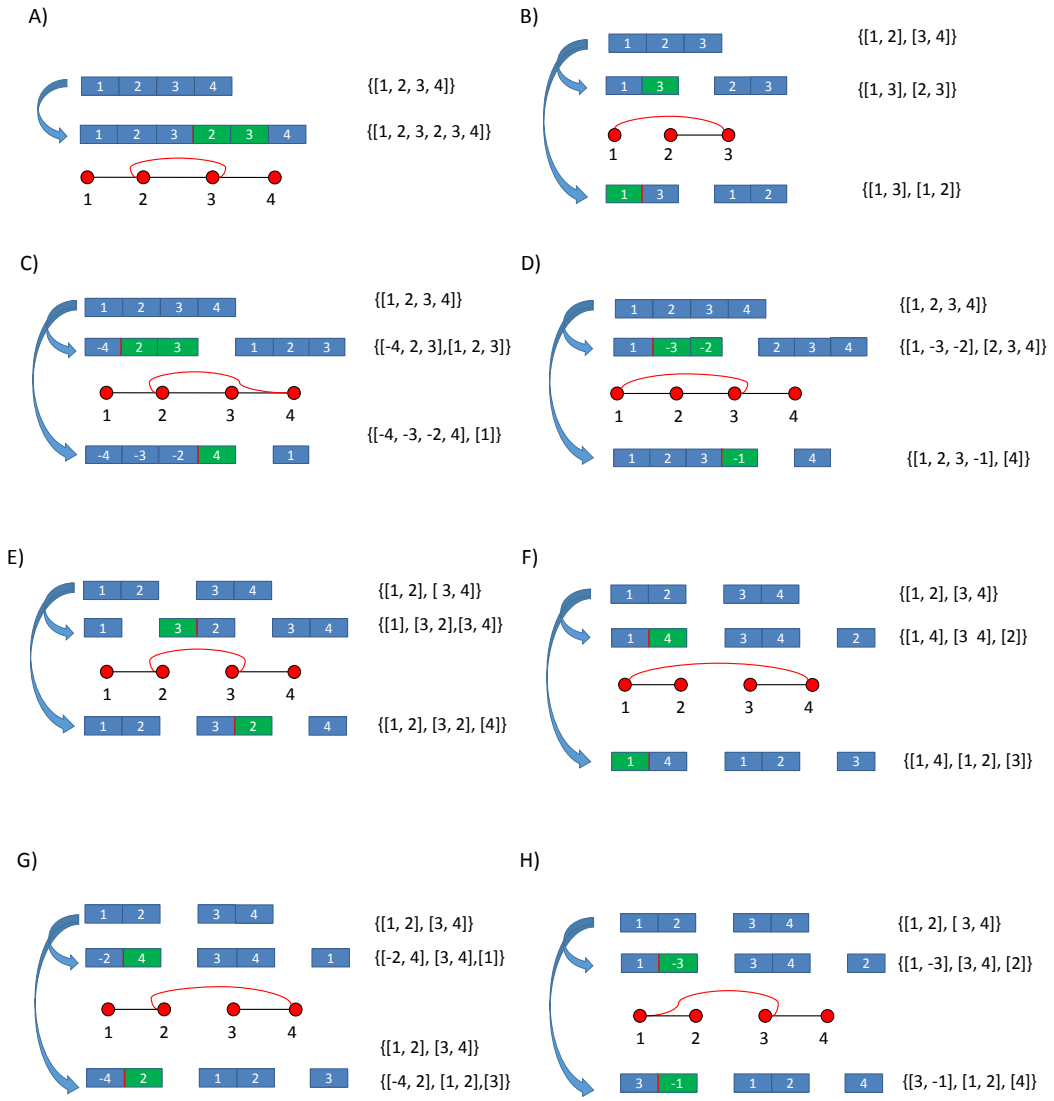

Figure 3.4: Full list of BIR operations for all 4 possible edge shapes, described using a schematic representation where each rectangle corresponds to a different segment. Green boxes correspond to the copied segment sequence. Word representation and the genome graph are also shown. A-D) Intra-chromosomal BIR operations. A) TD shape, edge $[-2\ 3]$. B) Deletion shape, edge $[1\ -3]$. C) Left inversion shape, edge $[-2\ -4]$. D) Right inversion shape, edge $[1\ 3]$. E-H) Inter-chromosomal BIR operations. E) TD shape, edge $[-2\ 3]$. F) Deletion shape, somatic connection $[1\ -4]$. G) Left inversion shape, edge $[-2\ -4]$. H) Right inversion shape, edge $[1\ 3]$.

Example 3.6. *Consider the evolutions shown in Figure 3.1. In both evolutions, we observe only connections present in the genome graph of Figure 3.1A (first condition of Definition 3.22) and each somatic edge is used exactly once (second condition). Moreover, both evolutions do not break the same wild type edge more than once, nor they split apart segments which are linked by somatic connections: thus they follow the breakpoint reuse constraints of Definition 3.21, which is the third condition for a valid evolution. However, evolution Bi creates a final genome with all somatic connections from the graph (fourth and last condition of validity), while in evolution Bii connection* $(3, 3)$ *is lost after fragment elimination. We conclude that evolution Bi is valid, while Bii is not.*

With the help of all provided definitions, we now define the main problem we want to solve:
Problem 3.1. *Given the paired end and read coverage input information from a cancer sample, determine a set of consistent evolutions, subject to the breakpoint reuse constraint, such that the final genomes after fragment elimination show a segment copy number profile consistent with the observed read coverage.*

3.1 Implications of the Breakpoint Reuse Constraint

Many models of genome evolution assume that more than one operation on a DNA sequence can introduce a break exactly in the same site: we will call this a *breakpoint reuse*. Assuming that the series of rearrangement occurring in a cancer genome represents a random process, the probability of two or even more chromosome breaks occurring in exactly the same site is extremely low. The breakpoint reuse constraint (Definition 3.21) assures that once a breakpoint has been introduced during an evolution, no other break on that position can ever occur. It also makes sure that every somatic connection in P corresponds to a single operation, i.e. is used only once across the evolution. It must be noticed that differential recombination frequency due to mutation hotspots is not taken into account in our model. It is known that fragile sites associated with 3 nucleotides repeats might form

hairpin structures during DNA replication, favouring recombination events [4]. In the case of a breakpoint being implicated in more than one rearrangement operation and located inside a known fragile site, it can be argued that the higher than expected recombination rate could cause the same breakpoint to be introduced more than once. In the other cases, accounting for recombinational hotspots has not a great impact on our evolutionary reconstruction. For example, assume we observe a connection touches two breakpoints a, b and b is located at a fragile site. If no other somatic connections implicate those breakpoints, we find that the only two ways we can reintroduce breakpoint b is by 1) performing a DSB break, or 2) reintroducing the same somatic connection once again (assuming this is possible without reintroducing breakpoint a as well). In case 1, we are creating a new fragment, which will hardly produce a different, valid evolution. Indeed, such series of operations is likely to produce unrepaired fragments (lost by fragment elimination) carrying somatic connections (see Figure3.5). In general, this modification is very unlikely to determine any improvement of our model. While case 2 is less problematic, we must notice that reusing the same somatic connection represents an unnecessary overcomplication of our evolutionary reconstruction. What we want is a parsimonious explanation of how new connections are acquired during cancer development. Associating several rearrangements to the same discordant pair of reads can significantly increase the complexity of our reconstructions, without providing any clear advantage. These are the reasons why we have excluded this possibility from our model, associating each connection to exactly one operation. A similar reasoning is valid when breakpoints a, b map to a fragile site. Even if we allow for the reintroduction of two breaks at positions a and b, we are restricted to the choice of either perform two DSB operations or reusing the same somatic connection.

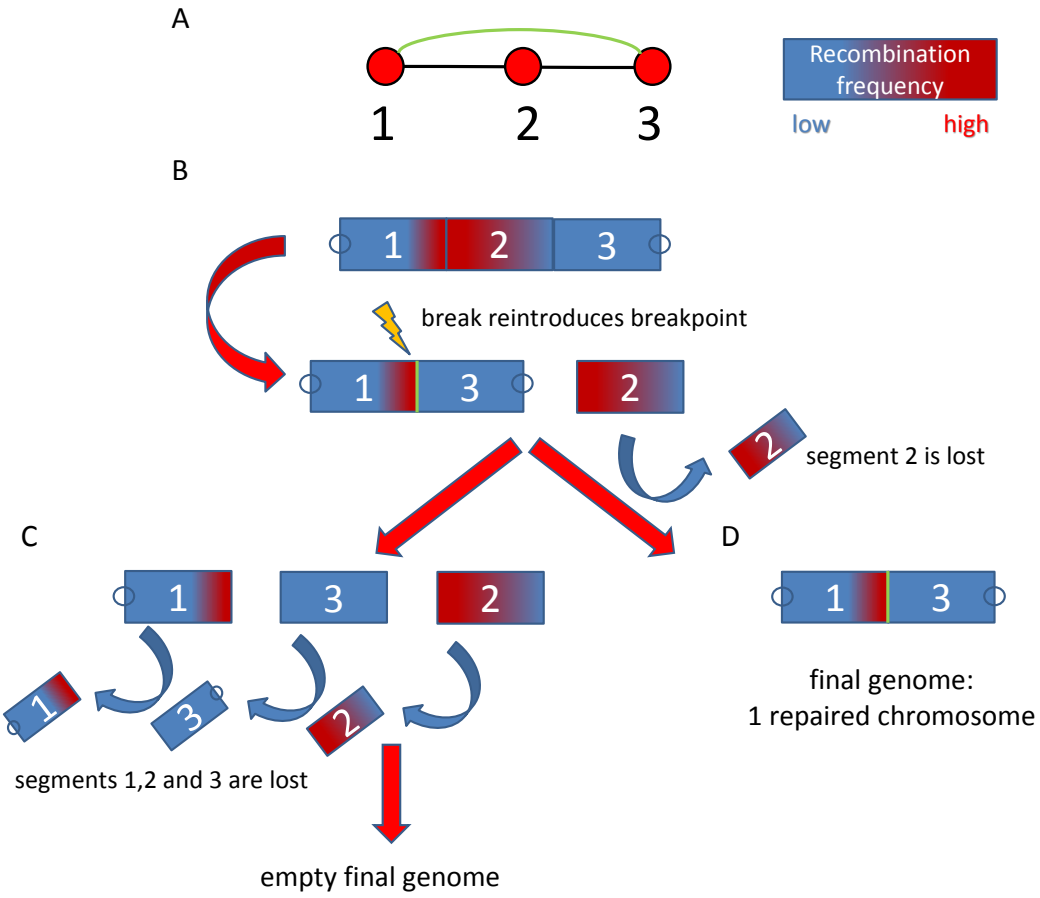

Figure 3.5: Implications of an alternative model of rearragement evolution, where breakpoints lying on a recombination hotspot (red regions) can be reintroduced. The genome graph (A) contains a single somatic connection, $(1, -3)$, interpreted in our example of evolution as the deletion of segment 2 (B). By re-introducing breakpoint 2 (between segment 1 and 2) through a DSB operation, a genome with 3 fragments (circles indicate the presence of a telomere) and no repaired chromosomes is generated (C), leading to an empty final genome (E). Note that this evolution is not valid according to Definition 3.22. In the original model, the DSB operation is not performed, leading to a final genome (part of a valid evolution) consisting of a single repaired chromosome (D)

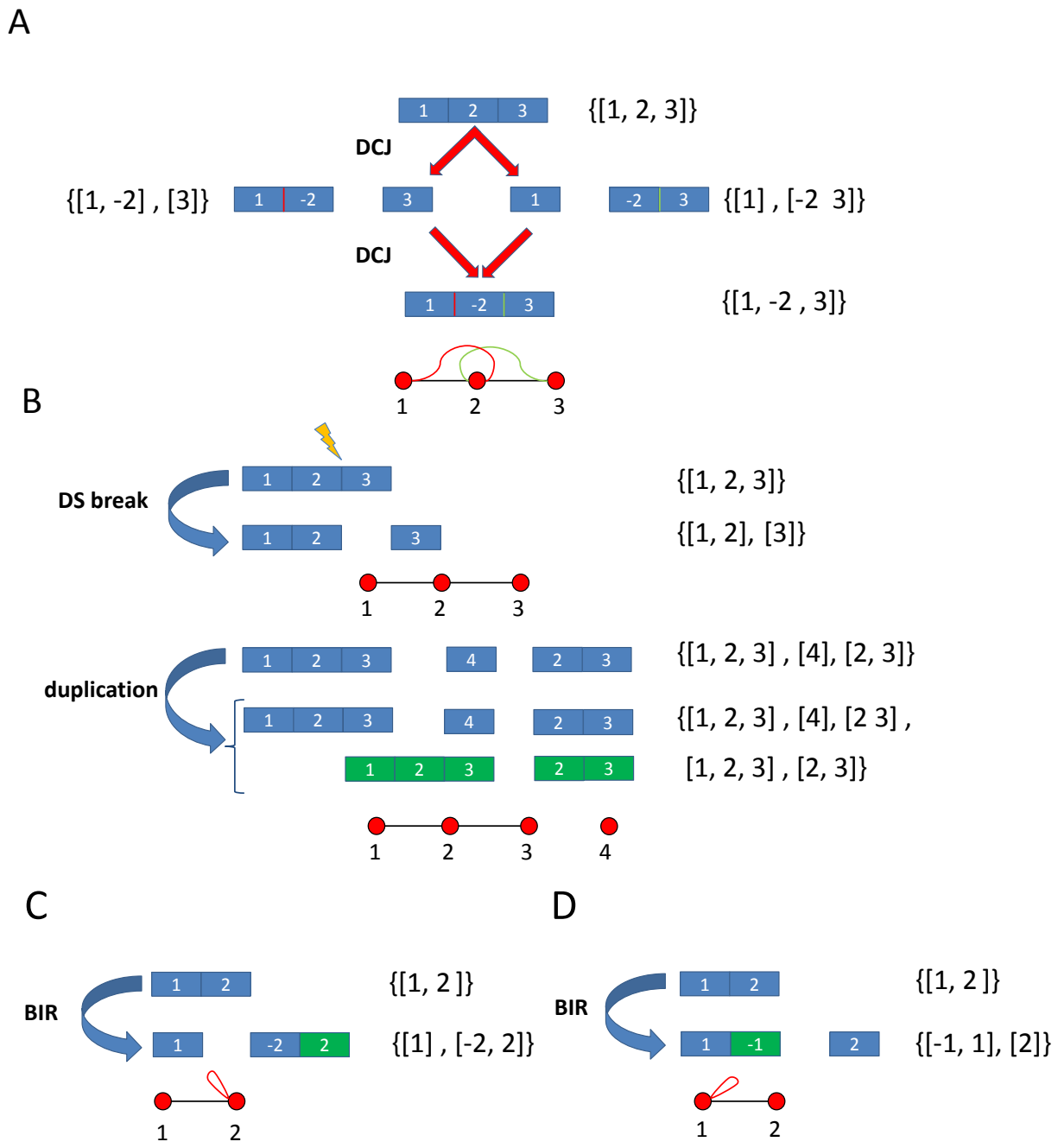

Figure 3.6: Schematic representation of some special series of operations and the genome graphs for the resulting genomes. The two somatic connections $(1, 2)$ and $(-2, -3)$ perform the inversion of segment 2. B) DS break operation on the right side of segment 2 C) Random chromosome duplication operation: light green boxes represent duplicated DNA segments. C) Left Breakage-Fusion Bridge: a BIR operation involving a unique breakpoint on the left side of segment 2 results in a break and an inverted duplication of that segment. D) Right Breakage-Fusion Bridge: a BIR operation involving a unique breakpoint on the right side of segment 1 results in a break and an inverted duplication of that segment.

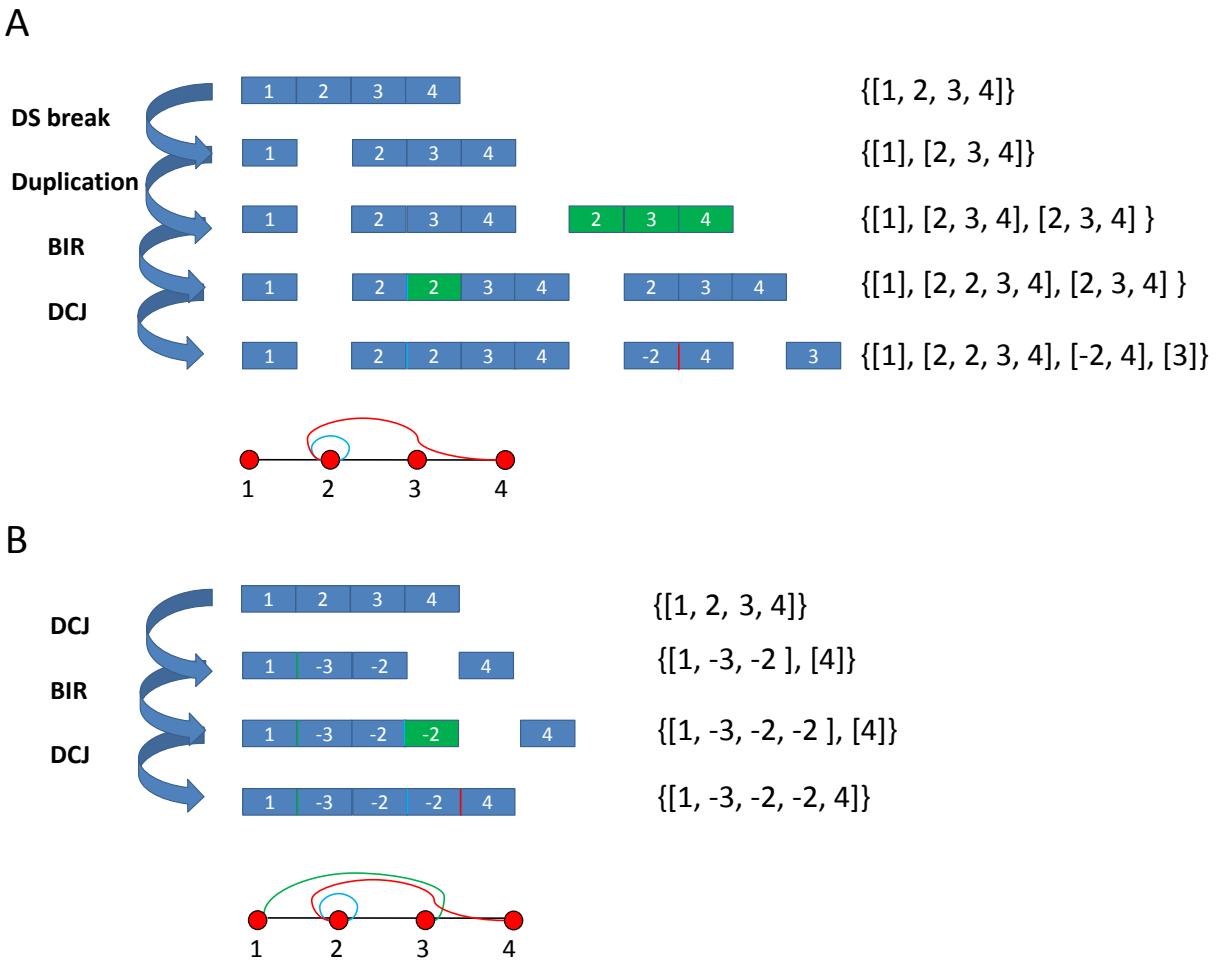

Figure 3.7: Schematic representation of two possible ways of performing connection operations with edges $(-2, 2), (2, 4)$, both touching node 2 (an example of segment side reuse). in A), a DSB operation followed by a chromosome duplication create two copies of the breakpoint between segments 1 and 2. This allows for both connection operations (coloured edges in the graph) to be performed, without reintroducing the break. In B) , a combination of three connection operations allows for the breakpoint to be available at multiple stages of the evolution, allowing for segment side reuse by connections $(-2, 2)$ and $(2, 4)$.

Figure 3.7 provides two examples of an evolution where the left side of node 2 (and therefore

the same breakpoint) is implicated in two different operations. This is a particular case of breakpoint reuse, which we will call *segment side reuse*. Now, the breakpoint separating segments 1 and 2, as all others, can be introduced only once, as the result of a single operation. in A), a break followed by duplication makes two copies of the breakpoint available, allowing for the two connection operations to take place under the breakpoint reuse constraint. In B), on the other hand, an earlier DCJ operation first introduces the breakpoint, which remains available throughout the next two evolutionary steps. Once again, we find that a complete evolution can be constructed under the constraints of Definition 3.21.

3.2 Unique word representation

We have already seen how to represent chromosomes as words of integers. However, we must note that such representation is not unique. Consider the following examples where two chromosome words are compared:

$[1, 2, 3, -1]$
$[1, -3, -2, -1]$

These correspond to the same set of segments carrying the same single somatic connection $(1, 3)$.
We use lexicographic ordering to choose among these two alternatives. We define our ordering as follows.
Definition 3.23. *Given the set of symbols L from a chromosome or a genome word, we define the following increasing coefficients for lexicographic ordering:*
$\{-\min(L), \min(L), -(\min(L)+1), \min(L)+1, ..., -(\max(L)-1), \max(L)-1, -\max(L), \max(L)\}$

where $\min(L)$ *and* $\max(L)$ *are the lowest and highest letters (segment labels), respectively*

If we look now at the example presented above, we find that both chromosomes start with letter 1, which cannot be used to define their relative order. When we look at the second letters, 2 and -3, we note that $2 < -3$ based on Definition 3.23. Such procedure is superfluous in the special case of palindromic words (which might arise by a BIR operation). For example, chromosome word $[1, 2, -2, -1]$ is identical to its reverse sequence.

3.3 Producing a set of evolutions from observed data

Following is a summary of the approach we have used in order to reconstruct a rearrangement history. Six distinct steps can be distinguished:

1. Construction of a genome graph based on available paired end data

2. Definition of segment boundaries and association of each segment with a mean read-depth value

3. Performing rearrangement operations

4. Eliminate fragments from each resulting genome

5. Discarding all final genomes lacking one or more somatic connections (selection of consistent genomes)

6. Comparison of segment read depth values with the segment copy number profile of each consistent genome and calculate a score for each comparison

Graph, segment boundaries and read coverage values. Paired end sequencing data gives us the information about the precise mapping position (a window of about 500 bps) and orientation of a pair of reads. We represent this information with the genome graph. Paired end sequences are used to identify rearrangements and define DNA segments along chromosomes, as described in Figures 1.9 and 2.2A-C. Recall that, in a typical paired end experiment for the detection of structural variants, a library of $\sim 100-500$ bps fragments is generated. The two ends of each fragment are then sequenced (pairs of reads) and mapped to the reference genome. The combination of fragment size and length of the reads will then determine the precision in locating breakpoints. Assume, for instance, that an insert size of 500 bps is used, and only the first 100 bps at each end are sequenced. This means that any pair of reads is separated, in the cancer genome, by about $500-(100\cdot 2) = 300$ bps. Now, this type of data is guaranteed to detect a rearrangement, provided the breakpoints are separated by at least 300 bps. However, special cases where two or more breakpoints map very close together might be impossible to detect; for example, assume a small (≤ 300 bps) inversion occurred, and the inverted region is included in a fragment of our sequencing library. The two paired end reads of that fragment might not be affected by the inversion, and will map to the reference genome exactly like a concordant read pair. A small deletion will be easier to identify, as the mapping distance of the two reads will be greater in the reference genome. In general, we can confidently use this method for the detection of larger scale rearrangements, the type of genomic variation we are interested in.

Performing operations. The reconstruction of a genome evolution is carried out by implementing each couple of paired end reads as a rearrangement operation. Since no information about the chronological order of these paired end reads/operations is available, for a graph with n somatic connections all $n!$ orders of operations need to be considered. For each connection we have two choices for the operation type (DCJ or BIR) and possibly multiple choices when the segments involved are present in more than one copy.

Eliminating fragments. We assume in our model that chromosomes showing at least one aberrant telomere will be lost. Therefore, from every generated final genome we remove all chromosomes showing at least one end different from the left side of segment 1 or the right side of the last segment in the corresponding wild-type chromosome.

Selection of fragment-free valid genomes. After fragment elimination, it is crucial to check which of the new resulting final genomes show all somatic connections present in the genome graph. Any other genome that has lost all copies of one or more connections is clearly the product of an evolution inconsistent with observed paired end data, and can be then filtered out.

Calculating a score for each final genome. We expect the following linear relation to hold:

$$y = \beta_0 + x\beta_1 + \epsilon \tag{4}$$

where y indicates the read depth; β_0 is a constant; β_1 is the read depth value corresponding to one DNA segment copy; x is the observed copy number of that particular segment and ϵ represents the error. Here we assume that $\epsilon \sim N(0, \sigma^2)$ (where σ represents the variance) has a normal distribution, which is a valid assumption when dealing with samples sequenced at high coverage. We need to consider how well the copy number of a fragment-free, valid final genome fits the available read depth data from the cancer sample. Given the paired end data and read coverage from a cancer sample C, assume we have a genome graph $P = (V, E)$ and the estimated breakpoint positions $[bp_1, bp_2...bp_{n-1}]$ where $n = \max(E)$. Our tumor read coverage information is represented by mean values for each 500 bp window (bin) across the whole genome.
Let $X_j = [x_1^{(j)}, x_2^{(j)}, ...x_n^{(j)}]$ be the set of consecutive coverage values such that $bp_j \leq x_1^{(j)} < x_j^{(j)} \leq bp_{j+1}$ and $1 \leq j < n-1$. Since bp_j and bp_{j+1} define the boundaries of DNA segment j, the values in X_j are exactly the observed coverage values across segment j. Note that

for special segments 1 and n, respectively the left and the right end rather correspond to a telomere.

Now, suppose we want to compare the sets $X_1, X_2, ..., X_J$ to the copy number vector $CN = [cp_1, cp_2, ..., cp_J]$ of a final genome word. From Equation 4, we expect a linear relation between each read coverage value $x_i^{(j)}$ and the copy number value cp_j. However, we notice that while cp_j is a single number, we have a set of values X_j for segment j. We then perform a regression analysis where the $x_i^{(j)}$'s represent the dependent variable, expected to be a function of the explanatory variable cp_j. We have dependent variable values $X_j = [x_1^{(j)}, x_2^{(j)}, ...x_j^{(j)}]$, while the explanatory variable value cp_j is identical across all $x_i^{(j)}$. We then combine the values of all segments to get the complete dataset for our regression:

explanatory variable(x): $\{cp_1^{(1)}, cp_1^{(2)}, ..., cp_1^{(|X_1|)}, cp_2^{(1)}, cp_2^{(2)}, ..., cp_2^{(|X_2|)}, ..., cp_n^{(1)}, cp_n^{(2)}, ...cp_n^{(|X_J|)}\}$

dependent variable($f(x)$): $\{X_1, X_2, ...X_J\}$

From this analysis we derive the estimated coefficient m' and constant value q' which define the set of estimated coverage values, $x_i^{\prime(j)} = m' * cp_j + q'$. These are the values which minimize the *sum of square residuals* (i.e. the sum of square differences between the $(x_i^{(j)} - x_i^{\prime(j)})^2$). Now, since such a discrepancy indicates how well our linear model fits the data, we can utilize it in order to asses the goodness of a particular solution; more precisely, we want this sum to be as small as possible. We then define the *final genome score* as follows:

$$\sum_{j=1}^{J} \frac{\sum_{i=1}^{|X_j|} (x_i^{(j)} - x_i^{\prime(j)})^2}{|X_j|} \tag{5}$$

Thus we sum the square residuals across all coverage values of a particular segment, and divide the result by the number of such values (i.e. the number of $\sim$ 500 bp intervals in that segment). Weighting the sum of square residuals for each segment reflects the need for normalising the values based on the different DNA segment lengths. We finally obtain

our score by summing these results across all segments.

It is important to notice here that different noise signals are expected in distinct sequencing experiments, which makes the comparison of scores across cancer samples hard. This problem has not been addressed in our study, which rather focused on the challenge of selecting a set of optimal evolutions for a single rearrangement cluster. The final genome score we have defined must then be intended as a measure *relative* to a specific sequencing rearrangement example, rather than absolute.

Example 3.7. *Consider the evolutions in Figure 3.1. Evolution Bi leads to copy number vector* $[2, 3, 2, 0]$*. For the calculation of the final genome score, we utilize the values plotted in Figure 3.1C (shown in red) as the dependent variable values. Now, every segment in our example corresponds to* 20 *coverage values (for* 20 *adjacent bins) so we simply repeat each value in* $[2, 3, 2, 0]$ *twenty times. We then obtain the following set of values for the dependent variable*
$[2^1, 2^2, 2^3, ...2^{20}, 3^1, 3^2, 3^3, ...3^{20}, 2^1, 2^2, 2^3, ...2^{20}\ , 0^1, 0^2, 0^3, ...0^{20}]$

From the regression analysis we derive $m = 96.12, q = 8.88$ *and a score value of* 6.38*. Thus* $y = 96.12x+8.88$ *(where* x *is the copy number and* y *the predicted read coverage). Substituting* x *with the copy number value gives* $Y' = [201.12, 297.24, 201.12, 8.88]$*: such values are plotted in blue across the four segment regions in Figure 3.1, and well describe the pattern of read coverage values across all segments (blue line in Figure 3.8B). Applying the same procedure to evolution Bii returns* $m = 160.88, q = 135.86$ *and a much higher score value of* 251.13*. We derive the predicted read coverage values* $Y' = [135.86, 296.74, 135.86, 135.86]$*. Now, the value* 296.74 *for segment* 2 *fits well the observed data. However, we have identical value* 135.86 *for segments* $1, 3$ *and* 4*, despite a clear drop in the read coverage signal from segment* 3 *to* 4*. Such value badly predict the coverage by either underestimating or overestimating the coverage signal of the three segments (green line in Figure 3.8B).*

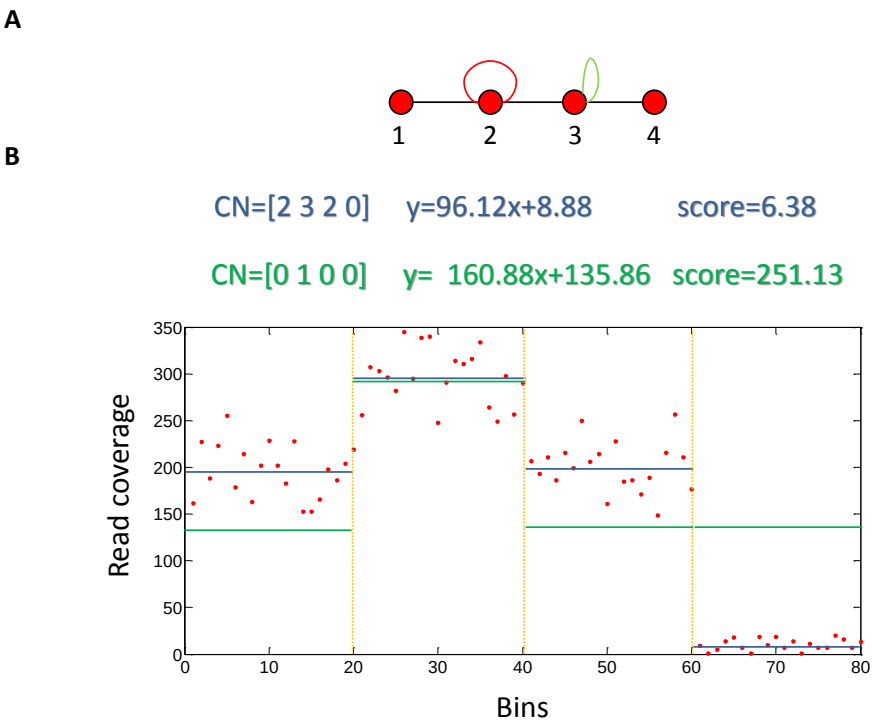

Figure 3.8: Example of calculation of the final genome score, using evolutions displayed in Figure 3.1. A) genome graph from Figure 3.1A. B) The read coverage values across 4 segments are plotted in red. Blue lines indicate the predicted coverage values for copy number vector $[2, 3, 2, 0]$ (from evolution Bi of Figure 3.1), while green lines indicate the predicted coverage values for copy number vector $[0, 1, 0, 0]$ (from evolution Bii of Figure 3.1). Estimated linear equations and the score values for each copy number vector are also shown.

3.4 Analysing real data

We explored the space of possible evolutions for cluster of rearrangements in different tumour samples. The simplest cluster we analysed comes from the breast cancer sample $PD4243$ and involves just chromosome 2. The genome graph for this cluster can be seen in Figure 3.9A. Recall that, in the genome graph representation, we have the correspondence one node:one segment, and each somatic connection might touch either the left or the right

side of a node (see Definition 3.1). We have been able to simulate all possible combinations of connection operations for this case (without DSB or chromosome duplications), as it involves a limited number of somatic connections. Consider the example shown in Figure 3.10. The evolution starts with the diploid reference genome $\{(1,2,3,4,5),(1,2,3,4,5)\}$ where chromosome 2 is divided into 5 DNA segments. Then, a series of 3 DCJ operations plus fragment elimination returns a final genome with copy number profile $[1,2,2,2,1]$. Regressing these copy number values against observed read coverage data gives the fitted coverage values (blue) shown in Figure 3.9B. The score of the final genome word (43.99) is the minimum possible value we found for the graph in Figure 3.9A.

Figure 3.11A shows the genome graph for a much more complex cluster, involving chromosomes 1 and 15 from the same breast cancer sample ($PD4243$). We shall notice two breakpoint sides (left of breakpoint 1, right of breakpoint 10) which are touched each one by two different somatic connections. Constructing all possible evolutions for a graph containing 12 somatic connections is computationally too demanding, and we have been forced to randomly explore the space of possibilities. For each randomly constructed evolution, each connection operation has been assigned either of the two possible classes DCJ and BIR, and a random chronological order for the 12 connection operations has been chosen as well.

Such approach has proven useful, allowing us to identify several evolutions associated with a low-scoring (365) final genome word. 44 evolutions with this exact score (therefore sharing the same copy number) were identified. If we look at Figure 3.12, we see a schematic representation of one optimal solution: two DS breaks and two chromosome duplications are performed, together with 12 connection operations utilising the somatic edges from the graph. For simplicity, the figure does not represent the operations using edges $(17,19)$ and $(-18,-20)$. Such rearrangements simply perform an inversion of subword $[...18,19...]$ (analogous to the inversion in Figure 3.6A) without affecting the final copy number vector. Our approach has thus proven useful in order to construct an hypothesis of evolution for a sample which would have been otherwise particularly hard to interpret.

3.5 Simulations

We tested our reconstruction approach through simulations of evolutions with $n = \{1, 2, 3...9\}$ rearrangement operations. We used simulations to answer the following questions:

1. Is it always possible to reconstruct a simulated evolution E_s?

2. Let G be the genome graph of a given evolution E_s. How many other evolutions have the same graph G? How many have the same final copy number as in E_s?

3. Does the restriction to graphs with segment side reuse affect the answers of questions 1 and 2?

Question 1 is crucial for testing the correctness of the code for many different graphs and all rearrangement operations. Answering question 2 gives us insight on how often the copy number information can be used to distinguish between two different evolutions, assigning a different score to each of them. Question 3 argues that the graph structure might affect the number of optimal solutions.

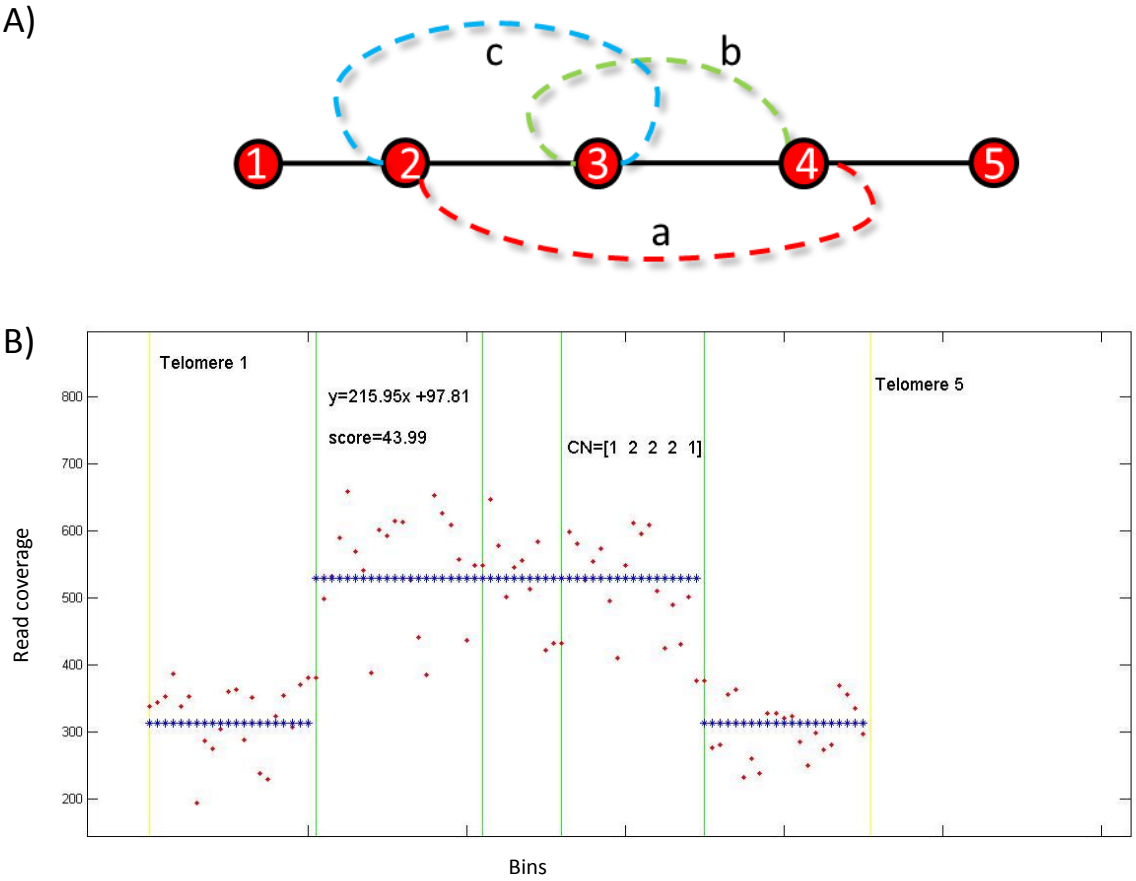

Figure 3.9: A) genome graph of a cluster of somatic connections found in chromosome 2, sample $PD4243$. B) Plot showing read coverage values per bin (red labelled) against the fitted values (blue labelled) of the regression for the evolution shown in Figure 3.10. The copy number vector for the final genome in Figure 3.10 is shown in square brackets: the final genome score is also displayed. Yellow and green vertical lines indicate the positions of telomeres and breakpoints, respectively.

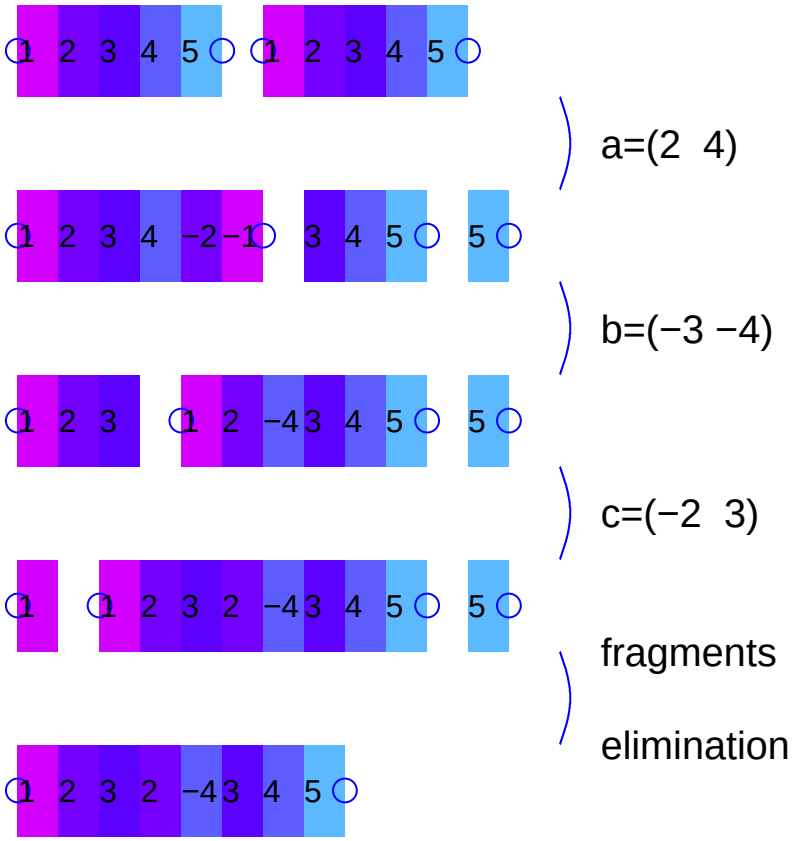

Figure 3.10: Example of a low-scoring evolution for the allelic and somatic graphs shown in Figure 3.9 (based on a cluster on chromosome 2, tumour sample $PD4243$). Numeric labels indicate DNA segments, ordered according to their position on the reference genome. Colours change together with increasing numeric labels, from purple (segment 1) to blue (segment 5). Lines represent the different steps of the evolution, ordered from top (starting with the reference diploid genome) to bottom (final genome with no fragments). Each line carries a description of the following event, indicating the newly introduced somatic edge, or the type of connection free operation (fragment elimination); also, the position of somatic connections is indicated by edge labels below each chromosome. Circles label telomeres. The sequence of segments from 1 to 5 corresponds to chromosome 2.

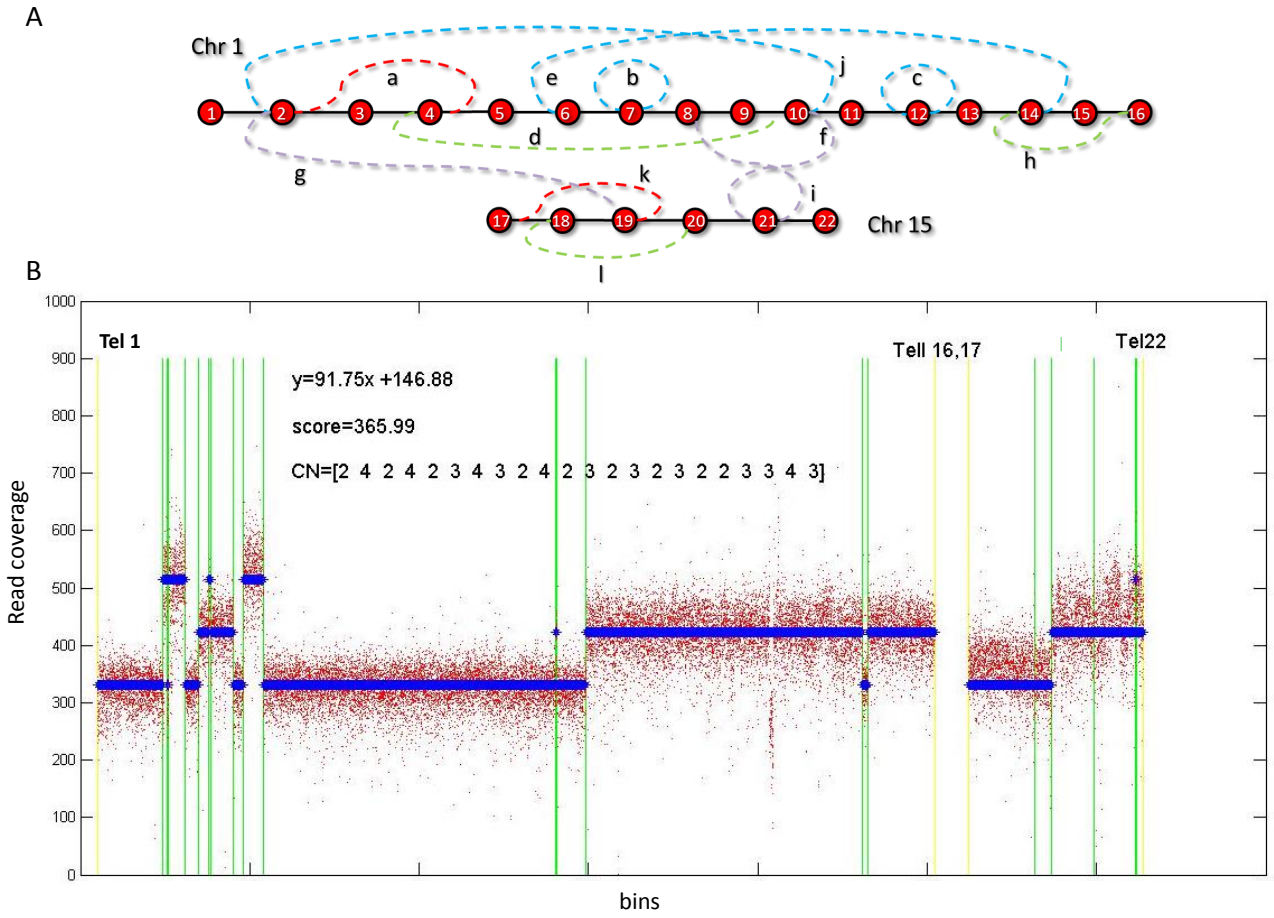

Figure 3.11: A) Genome graph for a cluster of somatic connections found in chromosomes 1 (nodes 1 to 16) and 15 (nodes 17 to 22), breast cancer sample *PD4243*. B) Plot showing read coverage values per bin (red labelled) against the fitted values (blue labelled) of the regression for the evolution shown in Figure 3.12. The copy number vector for the final genome in Figure 3.12 is shown in square brackets: the score of the regression is also displayed. Yellow and green vertical lines indicate the positions of telomeres and breakpoints, respectively.

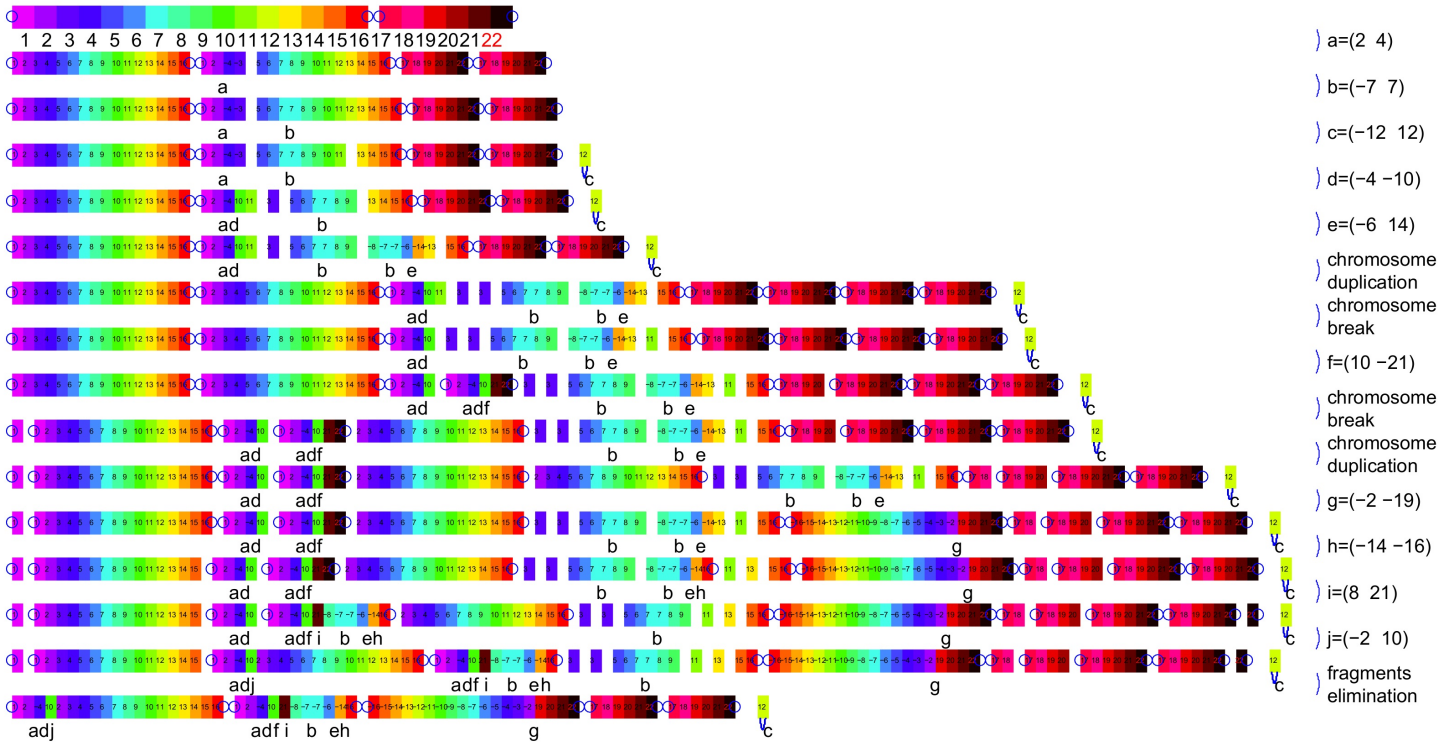

Figure 3.12: Representation of an evolution using 10 somatic edges from the genome graph shown in Figure 3.11A. Missing edges $k = (-17, -19)$ and $l = (18, 20)$ are used to perform an inversion (see section 3.1) and do not change the resulting final copy number; for clarity, the associated operations are not shown. Numeric labels indicate DNA segments, ordered according to their position on the reference genome. Colours change together with increasing numeric labels, from purple (segment 1) to black (segment 22). Lines represent the different steps of the evolution, ordered from top (starting with the reference diploid genome) to bottom (final genome with no fragments). Each line carries a description of the following event, indicating the newly introduced somatic edge, or the type of connection free operation (duplication or fragment elimination); also, the position of somatic connections is indicated by edge labels below each chromosome. Circles label telomeres. The sequence of segments from 1 to 16 corresponds to chromosome 1, while the sequence 17 to 22 represents chromosome 15.

3.5.1 Simulating unconstrained evolutions

We performed simulations using the following method: for all values $1 \leq n \leq 7$, the simulation algorithm randomly created 50 evolutions containing n connection operations followed by fragment elimination, each one with a specific associated graph; then for $n \leq 4$ connection operations, we generated the complete set of evolutions for P_s and counted the number of valid evolutions. We also counted how many evolutions had produced the same final (integer) copy number as in the simulated evolution.

For $n = 4, 5$ connection operations, the space of possibilities becomes very large. Therefore, we randomly generated 5000 valid evolutions for every genome graph (i.e. each of the 50 graphs) and calculated the same statistics. As the number n grows, we expect the average fraction of valid evolutions to decrease. The value 5000 was then chosen in order to minimise the cases where no valid evolution was constructed, while keeping the computational effort at acceptable levels. For a similar reasoning, the value was increased to 10000 for $n = 6, 7$. Although these evolutions were randomly constructed allowing for replacement, the number of *different* evolutions was counted.

Such simulations help answer questions 1 and 2. When the complete space of possibilities was produced, the original algorithm constructed the simulated evolutions in 100% of cases. As for the answer to question 2, it relies on the results shown on Table 3.1. Column 3 shows the mean number of valid evolutions across the 50 graphs of the 50 simulated evolutions, while column 4 gives the mean number of evolutions with the same copy number as the original simulated evolution. For example, constructing all possible evolutions for each of the 50 graphs with one somatic connection lead to an average only 2.2 valid evolutions, of which 1.8 on average showed the same copy number as the simulated evolution (*CN specific*). The copy number vector could identify a unique solution for 30 graphs out of 50 (indicated in column 6); in all other cases, we find that the simulated evolution shares the copy number with one or more other valid solutions. The copy number vector, however, proves less useful for the identification of optimal solutions when $n > 1$. With 2 somatic edges, the number of unique solutions dramatically drops down (4 out 50) and remains very low across all other n values. Thus, even with a limited number of rearrangements,

there is typically more than one valid solution for a specific copy number vector, and such number tends to increase as the number of operations grows (see column 4). This can be explained by various mechanisms. First, a set of operations might sometimes lead to the same result even if the chronological order of the operations is changed. In other cases, performing the same operations on different, but very similar chromosome words might lead to the same final copy number profile. Less often, we would also expect very different sets of operations creating distinct genomes, which nevertheless share the same final copy number. The fraction of CN specific to valid solutions (column 5) tends to decrease as the number of operations grows, suggesting that more copy number profiles are possible. The absolute number of valid evolutions drops from 5 to 6 connection operations. This is most probably due to the drop in the ratio between valid and inconsistent evolutions. For $n > 6$, these values grow relatively slowly, despite the increased sample size from $n = 5$ to $n = 6$. Once more this suggests that the fraction of valid evolutions to the total number of choices rapidly decreases with increasing n. Such pattern is likely due to the increasing difficulty in constructing evolutions where enough chromosomes are repaired, avoiding the loss of some somatic connections during fragment elimination. Thus, especially with a high number of connections, a significant fraction of the computational effort is used for the construction of inconsistent solutions. An improved algorithm enabling for an early discrimination between valid and inconsistent evolutions would thus prove significantly quicker.

3.5.2 Simulating evolutions with segment side reuse

Next a similar method was used, but with the specific constraint of having all graphs showing segment side reuse (see subsection 3.1): in these cases, genome graphs present one or more nodes touched by two distinct somatic connections on their left or right side (see Figure 3.11A, nodes 2 and 10). Based on such a graph structure, the reconstruction algorithm produces two distinct sets of evolutions: the normal set of connection evolutions, and additional solutions where the combination of early DS breaks followed by duplications guarantee the creation of at least two copies of the reused segment end (see Figure 3.7A).

n connections	n evols	valid evols	CN specific	CN spec/val evols	n unique sols
1	all($\overline{m} = 2.48$)	2.12	1.80	0.84	30
2	all($\overline{m} = 16.36$)	7.68	5.28	0.73	4
3	all($\overline{m} = 283.56$)	90.62	34.02	0.49	0
4	5000	134.88	48.64	0.59	1
5	5000	61.9787	19.00	0.36	4
6	10000	196.42	85.50	0.37	5
7	10000	132.48	52.76	0.29	4

Table 3.1: Statistics for unconstrained simulated evolutions. Rows correspond to the number of connection operations. Column 1 shows the number of connections. Columns 2 to 6 respectively indicate: mean total number of evolutions (for $n = 1, 2, 3$ connections) or the number of evolutions constructed for each of the 50 graphs; the mean number of valid evolutions and mean number of valid evolutions with correct copy number, across the 50 graphs; the ratio (mean valid CN specific / mean valid evolutions); the number of times (out of 50) a unique correct solution was found.

Simulated evolutions all consisted of n connection operations, with possibly a DS break and a duplication as additional operations ($n+2$ operations in total). $n = 1$ is missing for logical reasons (segment side reuse requires $n > 1$). The sampling approach was used for $n > 2$ connection operations: 5000 for $n = 3, 4, 5$, 10000 for $n = 6, 7$. Once more, 100% of the simulated evolutions was found in the set of constructed evolutions, which answers the first part of question 3. We now consider the data shown in Table 3.2 to complete our answer. As observed in Table 3.1, we find that the copy number information fails to identify a single, optimal solution in many cases. No unique solutions (column 6) were observed for $n = 2, 3$ connections, and the values for $n > 3$ are low and comparable to those in Table 3.1. However, we notice that the mean fraction of CN specific evolutions has a clearer and faster decreasing pattern than on Table 3.1, reaching its minimum, 0.12, at $n = 7$ (minimum value for the unconstrained evolutions: 0.29 at $n = 7$, see Table 3.1). Thus we conclude, once more, that the fraction of valid evolutions to the complete evolutionary space quickly decrease across n values. A parallel pattern is seen for the number of CN specific solutions: increasing across values $n = 2...4$, but getting smaller when more complex graphs ($n \geq 5$) are considered. We can now complete the answer to question 3: despite the general non-uniqueness of valid solutions, we observe more

CN specificity, and a lower number of valid solutions in graphs with segment side reuse. We interpret these results as follows: chromosome duplication creates additional material, which determines an increase in the number of rearrangement choices and presumably in the number of copy number vectors produced. The increase of copy number vectors determines a clear drop in the ratio values of Table 3.2 (column 5). Duplication events, on the other hand, often lead to more chromosome shattering and additional difficulty for the other operations to produce integer sequences. As a result, we observe more copy number heterogeneity, while the number of valid evolutions remains quite low and comparable to Table 3.1.

n connections	n evols	valid evols	CN specific	CN spec/val evols	n unique sols
2	all($\overline{m} = 40.7$)	16.36	11.98	0.77	0
3	5000	107.88	49.62	0.60	0
4	5000	141.84	51.62	0.36	4
5	5000	81.95	18.97	0.23	7
6	10000	56.86	7.76	0.20	5
7	10000	72.94	11.16	0.12	6

Table 3.2: Statistics for simulated evolutions showing segment side reuse. Column 1 shows the number of connections. Columns 2 to 6 respectively indicate: mean total number of evolutions (for $n = 2$ connections) or the number of evolutions constructed for each of the 50 graphs; the mean number of valid evolutions and mean number of valid evolutions with correct copy number, across the 50 graphs; the ratio (mean valid CN specific / mean valid evolutions); the number of times (out of 50) a unique correct solution was found.

3.6 Conclusions

We have seen from simulations that the copy number does not necessarily distinguish among all different evolutions, especially for simple graphs with 4 or less somatic edges. The ratio between the number of valid and copy number specific valid evolutions is generally high, indicating a considerable fraction of solutions sharing an identical final copy number vector. The situation slightly improves when we specifically consider genome graphs showing segment side reuse: here we observe smaller fractions of CN specific solu-

tions, which are however still not unique. From the biological point of view, this can be explained by a limited number of copy number profiles which can potentially arise from an evolution consistent with paired end data; the copy number would prove less useful exactly because it might represent a constraint for most, if not all, valid evolutions. Such problem is strictly related to the generally high number of valid evolutions obtained, with poor power of determining a single, optimal solution. Including allelic information, which was not available for our data, might represent a valid strategy for better distinguishing among the different generated explanations.

When constructing evolutions for clusters of 4 or more somatic connections, the space of possible evolutions typically becomes computationally very demanding. The main explanations are: 1) the rapid growth of the number $n!$ of possible ways of ordering n connection operations; 2) the tree-like structure arising from the multiple ways of performing each operation on each single genome obtained in the previous step; 3) the construction of a high number of inconsistent evolutions. One could assume that somatic connections involving the same breakpoint are consecutive, and this would often result in a significantly smaller space to explore. Also, any method allowing for the early discrimination between valid and inconsistent evolutions would significantly restrict the space of constructed evolutions, reducing the required computational effort.

Our approach has allowed us to always construct valid explanations of the observed data, no matter the complexity of the rearrangement clusters considered; however, the observance of typically non unique solutions, together with the rapid increase in the number of choices for high numbers of somatic edges currently represent a limit to the applicability of our method. Our algorithm approach includes an unedited modelling of Break Induced Replication, combined with a long menu of standard operations. The vast space of possibilities, despite representing a limiting factor, is a direct consequence of the high number of molecular mechanisms which have been modelled: the list includes simple double strand breaks, chromosome duplications, inversions, deletions, tandem and inverted duplications, (un)balanced translocations, breakage-fusion-bridge and chromosome circularization. Such

completeness allows our approach to highlight many potential mechanisms associated with tumor development, and can be applied to different research fields including comparative genomics, population genomics and molecular evolution.

4 The combinatorics of Tandem Duplication

Paired end and copy number information give us crucial insights on the evolution of a DNA sequence; they can be used to infer a rearrangement history, looking at both the structural changes and chronological orders of acquired somatic connections. However, we have seen in chapter 3 that a multiple-rearrangement model is typically associated with a combinatorial explosion of evolutionary choices; in these cases, the consistency with paired end and copy number information often leads to the identification of multiple optimal explanations of the data. Restricting a model of evolution to a single rearrangement class might allow for an improved power of restricting the space of optimal solutions. We have thus developed a tandem duplication model of evolution, a relatively simple process leading to DNA copy number variation. For our model, we have the constraint of unique breakpoint use. Given that paired end data can resolve breakpoints to the base-pair level, tandem duplications can be revealed and mapped with great precision in cancer data, as well as for other type of samples. When a tandem duplication occurs, it is reasonable to assume that the two associated breakpoints are generated in a random process; therefore, the chance for exactly the same nucleotide positions being implicated in another TD is likely to be small. Assuming unique breakpoint use is then reasonable in these circumstances. Looking at the space of chromosome structures that can originate from a tandem duplication (TD) process, we have noticed how these are associated with interesting combinatorial properties, which can lead to a complete mathematical description of the tandem duplication evolutionary space.

Consider the example shown in Figure 4.1A. We have a three step evolution starting with linear chromosome $\underline{12345}$. This is the representation of a series of five contiguous regions, labelled with numbers $\underline{1}$, $\underline{2}$, $\underline{3}$, $\underline{4}$ and $\underline{5}$. We will call such an initial structure the *reference*. The second step of the evolution corresponds to the new sequence $\underline{1234}\mathbf{1}\underline{2345}$(Figure 4.1Aii). That is the outcome of a Tandem Duplication copying $\underline{234}$ and inserting the new copy next to the original one. Note that we have introduced bold symbol **1**, indicating a

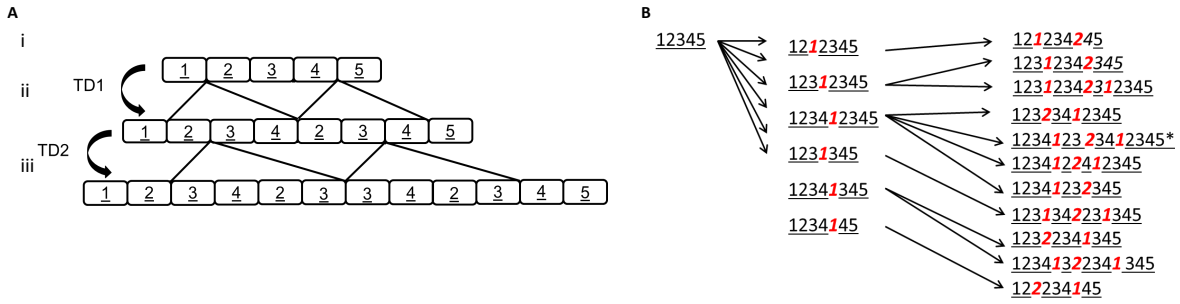

Figure 4.1: A Tandem duplication Process. A) Three structures i)-iii) arising from two tandem duplications on a reference of five regions; $1, 2, 3, 4, 5$. B) Eleven possible evolutions with two tandem duplications. The example in A is highlighted by *. Underlined numbers are segments. Red labelled, bold italicized numbers indicate connections between segments formed in the n^{th} tandem duplication.

newly introduced connection between the right side of segment $\underline{4}$ and the left side of segment $\underline{2}$. The third and last step of the evolution corresponds to structure $\underline{1234}\mathbf{1}\underline{23}\mathbf{2}\underline{34}\mathbf{1}\underline{2345}$, where a second bold symbol (connection) has been introduced. This event has duplicated region $\underline{3423}$, and the associated connection **2** links the right side of segment 3 to the left side of another copy of the same segment. Now, we notice that connection **1** maps between segments 1 and 2 on one end, and segment 4 and 5 on the other, while connection **2** maps respectively between segments $2, 3$ and $3, 4$. Thus every position has been implicated by exactly one TD event: an example of unique breakpoint use. In general, we see that N tandem duplications will implicate exactly $2N$ breakpoints and $2N + 1$ segments.

The example above is just one among 11 different evolutions arising from two tandem duplications, under the constraint of unique breakpoint reuse (Figure 4.1B).
We are interested in gaining understanding about the general space of evolutions for N events. We then informally define our first problem as follows.

Problem 4.1. *Count the number of different ways that an initial string of* $2N+1$ *segments can evolve under* N *tandem duplications, without reusing breakpoints.*

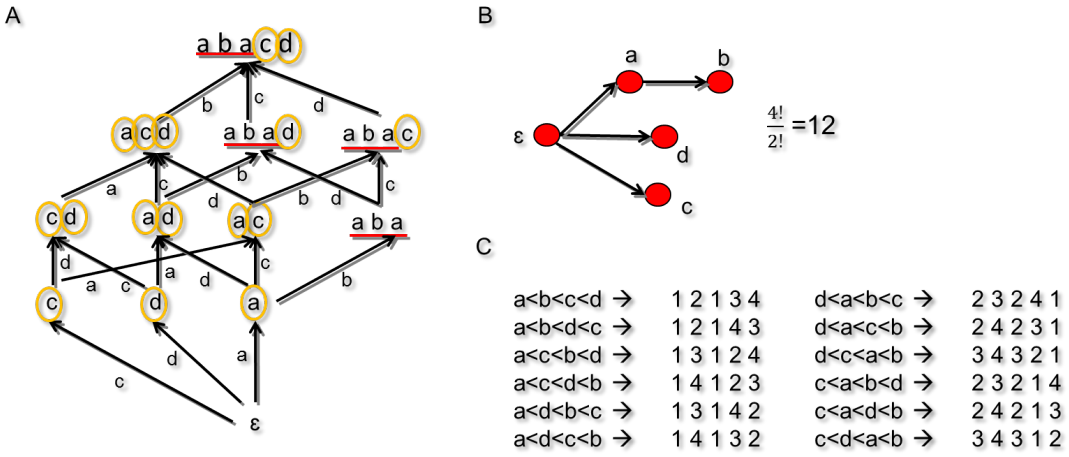

Figure 4.2: Evolution of a temporal word. A) Schematic representation of all different ways of deconstructing word $[a\ b\ a\ c\ d]$. Each letter indicates a specific TD connection. Nodes represent words. Edges indicates the direction of TD evolutions and are labelled with the newly added TD number. From each node, either a symmetry of the form $[Y\ x\ Y]$ (underlined letters) can be reduced to Y, or a single letter (which is not part of any $[Y\ x\ Y]$ subword) is removed, thus reversing the evolution. B) Hasse diagram of a partially ordered set, representing the order restrictions among TD events (nodes) for word $[a\ b\ a\ c\ d]$. Node ϵ is the source node and does not correspond to any TD event. Orientation of edges indicate the chronological order constraints among two nodes. The diagram is associated with 12 linear extensions. This corresponds to the number of TD evolutions represented in A). C) The 12 linear extensions compatible with the Hasse diagram in B), together with the corresponding connection words. These are obtained by substituting each symbol in $[a\ b\ a\ c\ d]$ with its positions $(1, 2$ or $3)$ in the linear extension.

Solving this problem has required a deep understanding of the labels we have just presented. First, we have tried to simplify the representation as much as possible, by ignoring all the labels representing segments. For example, the sequence $\underline{12345} \rightarrow \underline{1234}\mathbf{1}\underline{2345} \rightarrow \underline{1234}\mathbf{1}\underline{2}\mathbf{2}\underline{4}\mathbf{1}\underline{2345}$ can be converted into the simpler sequence $\epsilon \rightarrow \mathbf{1} \rightarrow \mathbf{121}$, where ϵ denotes the empty word. The use of bold symbols, which we will call *connection symbols*, will be the basis for our representation of a TD process. We will refer to sequences restricted to this type of symbols as *connection words*. Now, in a connection word the numerical value of each symbol clearly indicates the relative order in which TD events have occurred. Assume now we do not include this type of information, and use letters of the alphabet as our connection symbols: we thus get a different representation. For example, in evolution $\epsilon \rightarrow a \rightarrow ac \rightarrow$

$acd \to abacd$ (See Figure 4.2A) final word $abacd$ does not contain information about the relative order of symbols a, b, c and d. We call such a word a *temporal connection word* (or simply temporal word). Being left with no *a priori* knowledge of the relative order of TD events leads to a different type of question: can we gain some understanding on the possible chronological order of events by simply looking at the word structure? Let us consider word $abacd$ again. It is straightforward to notice a symmetry around letter b in the subword aba. This is the same structure observed in word 121, in evolution $\epsilon \to \mathbf{1} \to \mathbf{121}$, for example. Intuitively, we could then infer that, in the same way as connection 2 has duplicated connection 1, event b must have duplicated connection a and thus formed the observed structured aba. As for letters c and d, they are not intercalated between any repeated pattern, so they appear not to have duplicated any connection symbol. We also notice that, while symbol a is duplicated, b, c and d are present in a single copy in $abacd$. We then conclude that, whatever the evolution leading to that structure, a cannot possibly represent the latest TD event, since that symbol is duplicated.

This reasoning suggests that, by looking at the structure of a word, we can gain some insights on all TD evolutions it might have arisen from. Figure 4.2A, for example, shows all possible ways of *deconstructing* word $abacd$ (i.e. going back to ϵ through the recursive elimination of symmetries and remaining single letters). We will later use this process for a precise reconstruction, backward in time, of all TD evolutions which might lead to a specific temporal word. Clearly, such deconstruction approach also provides us with the number of evolutions associated with a specific word. However, the question arises whether it is possible to infer the number of TD evolutions without the time consuming construction of graphs such as in Figure 4.2A. This leads to the Hasse diagram of Figure 4.2B, where some order constraints (edges) among TD events (nodes) are inferred by just looking at the structure of word $abacd$: as already discussed, b clearly appears to have duplicated symbol a; thus we require b to have occurred later than a, and we add the edge $a \to b$ in the diagram. No other restriction can be inferred, and we find there are 12 linear extensions associated with our diagram: exactly the number of TD evolutions represented in Figure 4.2A. We will later present a detailed description of how to include the order information in a Hasse diagram. This leads to the definition of our second problem:

Problem 4.2. *Given a particular structure arising by n TDs with no breakpoint reuse,*

determine the number of deconstruction choices down to the (connection-free) reference structure.

The method presented for the solution of Problem 4.2 has a potential application in the fields of cancer and comparative genomics, for the aim of clarifying the evolution of tandem repeated regions; however, it is based on the strong assumption that both the reference and target genome sequences are available.

It is important to notice here that, both in the case of temporal words and connection words, we deal with a representation which does not always correspond to a single evolution. In Figure 4.2A we have several ways of deconstructing our word down to the empty word ϵ. We then find that, by choosing different paths from *abacd* to ϵ, we are building different series of temporal words, corresponding to distinct possible TD evolutions.

As for connection words considered in Problem 4.1, when we look at Figure 4.1B we find that three out of the eleven final structures share exactly the same connection evolution $\epsilon \rightarrow \mathbf{1} \rightarrow \mathbf{12}$ (restricting the words to bold symbols, highlighted in red). This non-uniqueness will be explored in the first sections of this chapter in connection words. First, the nature of connection words will be considered, observing how each word actually corresponds to many different TD-Evolutions (section *The Space of Connection Evolutions*). Next, a partially ordered set (*poset*[80]) based approach will be presented, allowing for the count all distinct TD-evolutions corresponding to a single connection word, (section *Using posets to count TD Evolutions*). Thirdly, we will see how these two pieces of information can be used to derive the number of possible evolutions for a given number of tandem duplications (that is Problem 4.1, see section *The Size of TD Space*). The last section of the chapter will focus on temporal words, exploring their properties and presenting a modified poset methodology to infer the number of deconstruction choices.

4.1 Representing a TD process with order information

We start by formalizing our representation of a TD process. As previously seen in the example of Figure 4.1, this is based on two different types of symbols: *segment* and *connection* symbols.

Definition 4.1. *A* segment *symbol* $\underline{i}$ *represents a copy of the* i^{th} *segment along the reference chromosome, where* $i \in \{1, ..., 2N+1\}$.

Definition 4.2. *A* connection *symbol* **i** *is the representation of a connection between two reference segments introduced during the* i^{th} *TD, where* $i \in \{1, ..., N\}$. *These symbols always have a segment symbol on either side, so that we have a subword of the form* $\underline{m}\mathbf{i}\underline{n}$. *Such representation indicates that* **i** *connects the right hand side of the DNA segment represented by* $\underline{m}$, *with the left hand side of the DNA segment represented by* $\underline{n}$.

We have seen from the examples in Figure 4.1B that a TD has the effect of copying a subsequence of contiguous segments, leading to a modified longer sequence. We formalize this process with the following definition:

Definition 4.3. *A* TD evolution U *is defined as any sequence of* TD words $[U_0 \to U_1 \to ... \to U_N]$ *such that*

- $U_0 = \underline{1} \cdot \underline{2} \cdot ... \cdot \underline{(2N+1)}$ *represents the initial reference sequence, divided into* $2N+1$ *segments*

- *Every TD on a word* $U_{n-1} = [X\ Y\ Z]$ *can be represented as a transformation of the form* $U_{n-1} = [X\ Y\ Z] \to [X\ Y\ \mathbf{n}\ Y\ Z] = U_n$ *for non empty subwords* X, Y, Z *all beginning and starting with a segment symbol*

We call n *the* TD number. *We also define* breakpoint numbers n_a *and* n_b *as the value of the rightmost symbol of* X *and the value of the rightmost symbol of* Y, *respectively.*

Any TD evolution is generated in such a way that n_a and n_b are $2N$ distinct values where $n \in \{1, ..., N\}$. Breakpoint numbers 0_a and 0_b are defined as 0 and $2N + 1$, respectively, representing the start and end of the original reference sequence. We let $\mathcal{U}_N$ denote the set of possible TD evolutions arising from N TDs.

It is important to note here 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 begins and ends with a segment symbol, so **n** is bordered by segment symbols on either side, as required by Definition 4.2.

Example 4.1. *Consider the evolution * of Figure 4.1B: $E = [\underline{12345} \rightarrow \underline{1234}\mathbf{1}\underline{2345} \rightarrow \underline{1234}\mathbf{1}\underline{23}\mathbf{2}\underline{34}\mathbf{1}\underline{2345}]$. The second TD duplicates segments $\underline{34}\mathbf{1}\underline{23}$ in the TD word $\underline{1234}\mathbf{1}\underline{2345}$. We then find that the subwords X, Y and Z correspond to $\underline{12}$, $\underline{34}\mathbf{1}\underline{23}$ and $\underline{45}$, respectively. 2_a, the value of the rightmost symbol in X, is 2, while 2_b, the value of the rightmost symbol in Y, is 3. Note that these two values demarcate breakpoints of the duplicated region; the 2^{nd} breakpoint is implicated by the left end of the duplicated region $\underline{34}\mathbf{1}\underline{23}$, between segments $\underline{2}$ and $\underline{3}$, and the 3^{rd} breakpoint is implicated by the right end of the duplicated region between segments $\underline{3}$ and $\underline{4}$.*

The values n_a and n_b defined above correspond to the leftmost and rightmost breakpoints implicated in the n^{th} tandem duplication.

As already mentioned, the representation which will be actually used to solve Problem 4.1 relies uniquely on connection symbols. We formally define it as follows.

Definition 4.4. *Given a valid TD evolution $U = [U_0 \rightarrow U_1 \rightarrow ... \rightarrow U_N]$, we define the corresponding* connection evolution *E to be the restriction of U to connection symbols: $[E_0 \rightarrow E_1 \rightarrow ... \rightarrow E_N]$. We refer to each word of this new sequence as a* connection word*. We also define $\mathcal{E}_N$ to be the set of all possible connection evolutions arising from N TDs*

Example 4.2. *In the evolution * of Figure 4.1B we have: $U = [\underline{12345} \rightarrow \underline{1234}\mathbf{1}\underline{2345} \rightarrow \underline{1234}\mathbf{1}\underline{23}\mathbf{2}\underline{34}\mathbf{1}\underline{2345}]$ which becomes $E = [\epsilon \rightarrow \mathbf{1} \rightarrow \mathbf{121}]$ when the segment symbols are*

removed.

We are now able to formalize Problem 4.1.
Problem 4.3. *Determine the size of the set* $\mathcal{U}_N$.

In order to solve Problem 4.3 we need to know first what different connection evolutions $E \in \mathcal{E}_N$ are possible, and secondly, for each such connection evolution, how may different orderings of breakpoint numbers $\{n_a, n_b\}_n$ for $n = 1, 2, ..., N$ are feasible. For example, the five structures represented in Figure 4.5C, despite being associated with five distinct breakpoint orderings, however, all correspond to the same connection evolution $\epsilon \to 1 \to 121$. 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 4.3. We consider these in turn.

4.2 The space of connection evolutions

We shall start by considering the space of connection evolutions arising from N TDs. If we take a two event evolution as an example, then we find that the first TD always produces word **1**, while the second TD can produce words **12**, **21** or **121**, giving 3 evolutions in total. Thus, we observe that with two TDs different choices are possible, leading to distinct final connection words: additionally, we observe that two of these final words have length 2 and one has length 3.
Theorem 4.1. *[89]*

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

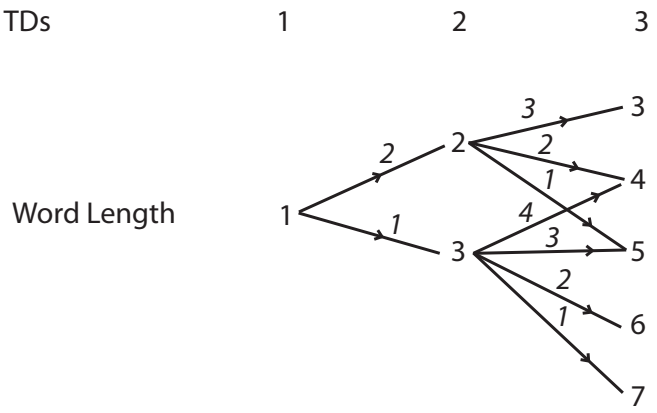

Figure 4.3: Schematic representation of the number of possible connection words up to the third TD. Nodes represent words. Numbers on edges indicate the number of choices [89].

where $w_{m,n}$ denotes the number of connection evolutions arising from n TDs such that the final connection word has length m, with initial value $w_{0,0} = 1$. The complete space of words for the first three TDs is shown in Figure 4.3.

Proof. Given a connection word E_n of k symbols, adding a TD leads to the duplication of a subword Y (see Definition 4.3) of length $r \in \{0, 1, .., k\}$. For any such values, we find that we can choose among $k - r + 1$ different sets of r consecutive symbols in E_n: that is, we have $k - r + 1$ ways of choosing a duplicated subword of length r. Given that a tandem duplication copies r connection symbols and introduces one new connection symbol, we obtain a new connection word of $m = k+r+1$ 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 $k \in \{\lfloor \frac{m}{2} \rfloor, ..., m-1\}$. Lastly, we note that the number of different ways of deriving such a word of length m is a function of k and m themselves. More precisely, it is equal to $k - (m - k - 1) + 1 = 2k - m + 2$, as stated in the theorem. □

Example 4.3. *Consider the representation of all possible choices shown in Figure 4.3. Here values $w_{m,n}$ correspond to the sum of products of the edge values along paths to the associated node, from the node labelled* 1*. For example, the node labelled* 5 *in the third*

column of nodes corresponds to $w_{5,3}$ *and has two paths, one with product* $2 \cdot 1$*, the other with* $1 \cdot 3$ *and we find* $w_{5,3} = 2+3 = 5$*, five connection words of length five;* **12312**, **21321**, **13121**, **12321** *and* **12131**.

Such finding was confirmed by computationally generating the complete space of connection evolutions. Also, we developed algorithms to determine the number of copy number profiles and TD evolutions as a function of the number of events (see Section *Computational Analyses*). The counts arising from the first few TDs can be seen in Table 4.1.

TDs	1	2	3	4	5	6
Words	1	3	22	377	15,315	1,539,281
CNVs	1	7	225	27,839	-	-
TD-Evolutions	1	11	627	154,869	156,882,297	640,550,418,651

Table 4.1: Counts of TDs, Words, CNVs and TD-Evolutions.

4.3 Using posets to count TD Evolutions

We have already seen, earlier in this chapter, that the connection evolutions representation is not unique. This naturally leads to problem of counting the number of TD evolutions corresponding to a single connection evolution. Every such TD evolution corresponds to a specific order of the $2N$ breakpoint numbers $\{n_a, n_b\}_n$ along the reference, for $n \in \{1, 2, ..., N\}$. For example, in Figure 4.1A we have $N = 2$ tandem duplications, $2N = 4$ breakpoints and the reference sequence is divided $2N + 1 = 5$ segments. We have faced this challenge with the help of three different representations:

1. a *zig-zag* plot representation, facilitating the visualization of breakpoint numbers along a chromosome;

2. a *2d-tree* representing the choices of the different orders of breakpoint numbers;

3. a *Hasse* diagram, whose associated number of linear extensions will provide the number of TD evolutions.

4.3.1 Zig-zag plots

For our model of TD evolution, any connection symbol **n** implicates two distinct breakpoints n_a, n_b, which define the boundaries of the duplicated region. The new copy is then inserted in our sequence next to the original one. We want to represent the resulting structure in the form of a plot of horizontal lines representing segments, and diagonal lines representing connections: we will call such a representation a *zig-zag plot*. Consider Figure 4.4A for an example. The breakpoint order $0_a < 1_a < 2_a < ... < 3_b < 0_b$ along the reference is indicated at the top. The plots correspond to the connection evolution given in Figure 4.4E. For each stage of the evolution, the sequence of segment symbols is represented from top to bottom as horizontal filled lines, while diagonal dotted lines represent the connection symbols introduced by the TD events. For example, in Figure 4.4Aiii we have the plot corresponding to connection word **121**. The four intervals $[0_a, 1_b]$, $[1_a, 2_b]$, $[2_a, 1_b]$ and $[1_a, 0_b]$ represent sequences of one or more adjacent segment symbols, connected together by a series of dashed lines (the connection symbols).

4.3.2 2d-trees

We now present a tree representation of a TD process, which will allow us to obtain the different possible orders of the breakpoint numbers. Precisely, we will use a 2-d tree representation, which can be defined as a graph such that all nodes (except root nodes) have exactly 2 parental nodes.

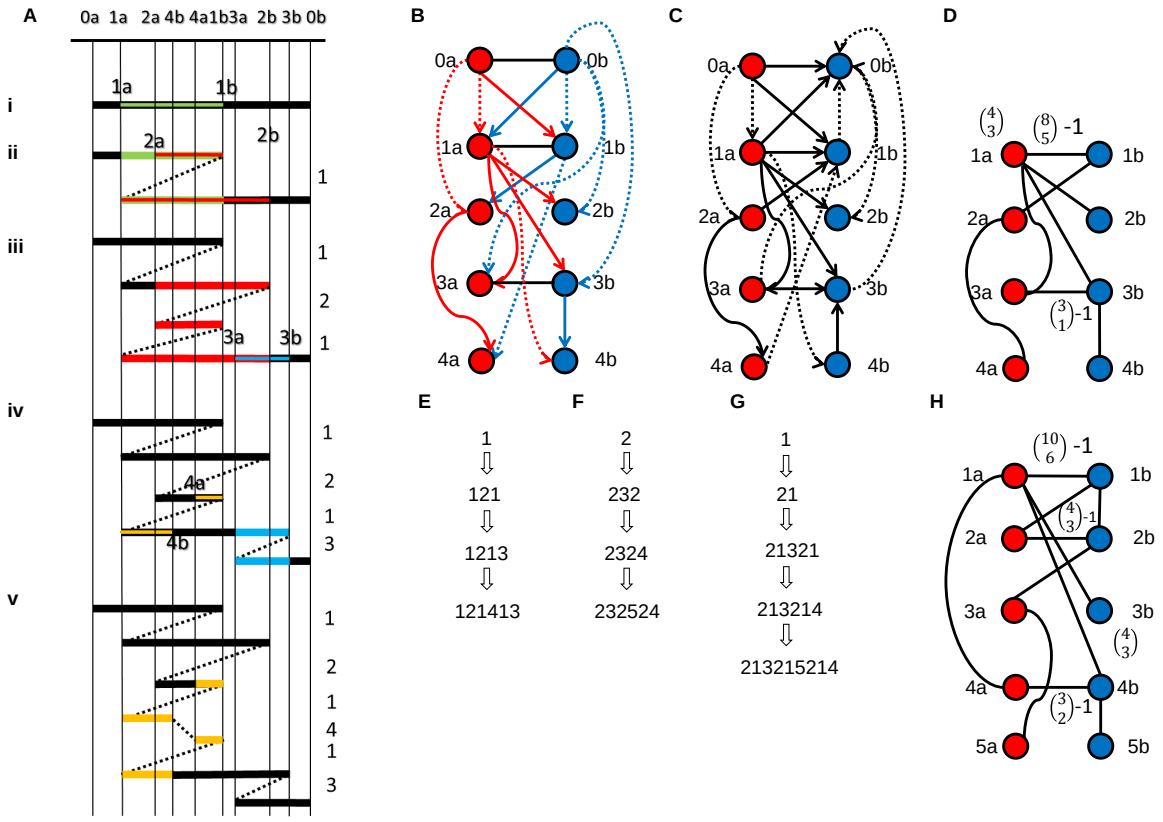

Figure 4.4: Representation of the TD Process. In A) we have *zig-zag* plots for a sequence of four TDs, resulting in five structures i)-v). The green regions indicates the region duplicated during each TD. Dashed lines indicate a connection between segments. Coordinates n_a and n_b indicate the end positions of the n^{th} duplicated region. B) Corresponding 2d-tree. Nodes correspond to breakpoints and edges demarcate an ordering. Red and blue colours indicate lower and upper bound breakpoints. Dashed and plain edges indicate minor and major edges. C) Corresponding Hasse diagram. D) The major graph corresponding to evolution E. F) Increases each symbol of E by 1. G) An induced evolution from F. H) The major graph corresponding to induced evolution G. The black nodes indicate the corresponding 1-nodeset.

Consider then how the zig-zag plots build up in the structures in Figures 4.4A i-v. The starting structure corresponds to the reference chromosome, the interval $[0_a, 0_b]$ represented in Figure 4.4Ai. In the 2-d tree of Figure 4.4B, we note the presence of a red node labelled

0_a and a blue node labelled 0_b. Red and blue respectively corresponds to node types a and b, indicating a left(a) and a right(b) end of an interval. The two nodes are bridged by an edge representing their ordering in the reference; $0_a < 0_b$. Four TDs will be introduced in this evolution, resulting in eight breakpoint numbers placed between $0_a = 0$ and $0_b = 9$.

The first TD event leads to the duplication of segment sequence highlighted in green (Figure 4.4Ai). Two breakpoints are implicated at this stage, 1_a and 1_b, resulting in the zig-zag diagram of Figure 4.4Aii. Such two positions both lie between the coordinates of 0_a and 0_b, so that we have order constraint $0_a < 1_a < 1_b < 0_b$. In the corresponding 2d-tree, we represent coordinates $1_a, 1_b$ with two nodes, both connected to parental nodes 0_a and 0_b. We must notice that parental node 0_a represents the lower bound for both 1_a and 1_b, while 0_b corresponds to their upper bound. We include this information in the tree by labelling edges from node 0_a as type a (red) and edges from node 0_b as type b (blue). Last, we observe the presence of a black edge, which we call a *fence* (similar to $(0_a, 0_b)$). This edge connects 1_a to 1_b, representing the restriction $1_a < 1_b$.

The second TD event duplicates the green region in Figure 4.4Aii, which also 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 4.4Aii 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.

Lastly, we notice that we have filled and dotted edges in the 2-d tree of Figure 4.4B. This allows us to include another type of information, which we call the *major* (solid) or *minor* (dashed) status of each pair of parental edges to a node: these terms 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.

We then proceed through the TDs building up the 2d-tree representation. We label each node as either type a (red colour) or type b (blue colour), depending whether the corresponding breakpoint represents the left or the right end of a segment. Then for any node n_a (resp. n_b, if the associated breakpoint lies on an interval $[u_a, v_b]$ in the zig-zag diagram, we add 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). The edge from $\max(u_a, v_b)$ is denoted *major* (solid), while the edge from $\min(u_a, v_b)$ is class *minor* (dashed). If n_a and n_b lie on the same segment (i.e. event n does not duplicate any connection) we have order constraint $n_a < n_b$ represented by an edge of class *fence* (black) between nodes representing breakpoint numbers n_a and n_b (this always includes 0_a and 0_b).

Now, based on the construction method described above the choice of major and minor is ambiguous for the first TD. Both 1_a and 1_b lie on the same interval $[0_a, 0_b]$ and thus have parental nodes 0_a and 0_b with equal TD-number 0. It has however proven consistent for the solution of Problem 4.3 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)[89]. In all other cases, the TD numbers of the two boundaries of any interval are different, so that we can unambiguously define the major and minor parental nodes.

Based on what we have discussed so far, we can make a summary of some important observations:

- The zig-zag representation is equivalent to the corresponding TD evolution
- The TD evolution is equivalent to the associated connection evolution and a specific ordering of breakpoint numbers

Now, what does the 2d-tree exactly correspond to? The answer to this question can be derived from a simple example like the following: if we take the connection evolution

$[\epsilon \rightarrow \mathbf{1} \rightarrow \mathbf{12}]$, we have intervals $[0_a, 1_b]$, $[1_a, 2_b]$ and $[2_a, 0_b]$ in the zig-zag plot. Now, if both breakpoints 3_a and 3_b from the next TD are placed in $[2_a, 0_b]$, we find that we could have both 3_a and 3_b before 2_b along the reference, both after, or 2_b in the middle position. For these three cases we will have different zig-zag plots and breakpoint orderings, but exactly the same 2-d tree. Thus the 2-d tree does not contain the same information as the other representations and is not equivalent.

We need a way to use these 2d-tree structures to count the number of TD evolutions corresponding to the same connection evolution. This will be discussed in the next section.

4.3.3 Linear Extensions

In order to infer the number of TD evolutions associated with a single connection evolution, the use of a partially ordered set [80] has proved extremely useful.

Definition 4.5. *A* poset *is a set of elements related to each other by some order constraints. Typically, it is represented in the form of 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 ordering of all elements in the poset which is compatible with all relations (constraints) is called a* linear extension.

We know that the 2-d tree includes some ordering information of the breakpoint orders. More precisely, it represents the order constraints between a breakpoint and the bounds of the interval it lies on (as discussed in the previous section). However, how can we convert this graph representation into a Hasse diagram representing a poset? Recall the intrinsic property of our 2-d trees: that is, any node x except for the root nodes has two parental nodes. The b parent is an upper bound for x, while the a parent is a lower bound. Given that we always have edge orientation from parental to daughter node, we

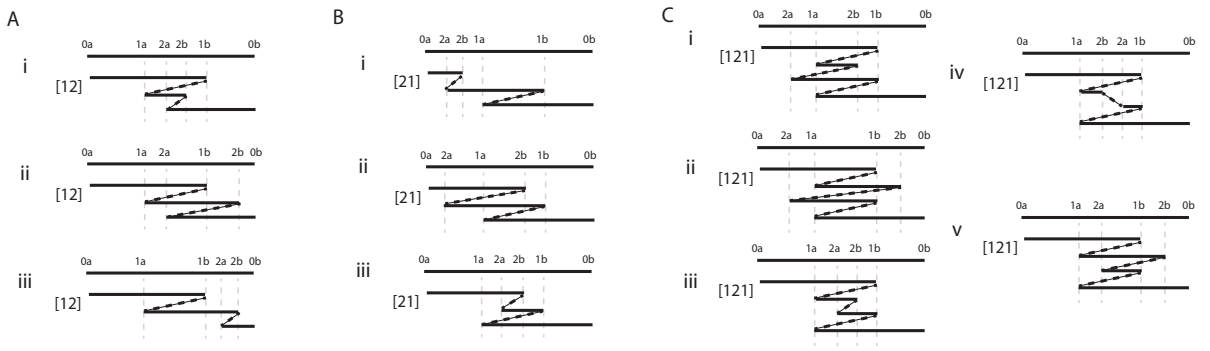

Figure 4.5: Zig-zag plots of structures arising from two TDs. A) Three structures associated with connection word 12. B) Three structures associated with connection word 21. C) Five structures associated with connection word 121 [89].

can conclude any edge $n_a \to x$ from the a parent correctly represents the order relation between the connected nodes, while any edge $n_b \to x$ from the b parent actually represent the constraint $x < n_b$. Consequentially, we find that by changing the orientation of all type b edges results in a correct representation of the order constraints between any couple of parental and daughter nodes. If we then direct all fence edges from n_a to n_b, what we obtain is precisely a Hasse diagram of all order relations associated with a connection evolution. Note also that this diagram has single source node 0_a and single sink node 0_b (representing the two bounds of the chromosome).

Example 4.4. *Consider Figure 4.4B-C. We have an edge from parental node* 0_a *to* 1_a *in the* 2*-d tree of Figure 4.4B, which correctly represents the order relation* $0_a < 1_a$*, found in the Hasse diagram (Figure 4.4C). However, looking at edge from* 0_b *to* 1_a*, we find that its orientation in the* 2*-d tree does not match the order constraint of the Hasse diagram (*$1_a < 0_b$*).*

Counting TD evolutions is then equivalent to counting the number of linear extensions associated with the poset. Now, the Hasse diagram obtained with this procedure, despite being a reliable representation of the order constraints, actually proves to be overly complicated. More precisely, we find that if we restrict the Hasse diagram to major edges and fence edges (that is, we remove the minor edges) we obtain a simplified graph, yet contain-

ing all the required ordering information [89]. We will refer to this type of graph obtained from any 2d-tree $T(E)$ of a connection evolution E as the *major graph*, denoted $T_{maj}(E)$. This new topology enables us to obtain a formula for the number of linear extensions, that is the number of TD evolutions. For the Theorem which follows, the Proof is beyond the scope of this thesis and can be found in [89].

Theorem 4.2. *[89] Consider a major graph where the nodes 0_a, 0_b and daughter edges have been removed. For each remaining node x let $x_1, ..., x_K$ be the number of nodes belonging to each of the K descending branches. If any pair of daughter nodes are connected by a fence, they contribute a factor $\binom{y_1+y_2}{y_1} - 1$, where y_1 and y_2 count the number of nodes descending down each branch connected by the fence. We then consider these two branches as a single branch with $y_1 + y_2$ daughter nodes, and associate the number $m(x) = \binom{x}{x_1,...,x_r}$ with node x. The number of distinct evolutions is then the product of these terms across nodes and fences.*

Example 4.5. *Consider the connection evolution $E = [\mathbf{1} \to \mathbf{121} \to \mathbf{1213} \to \mathbf{121413}]$ with 2d-tree in Figure 4.4B. After removing nodes 0_a and 0_b and restricting the graph to major and fence edges we obtain the graph of Figure 4.4D. Two fence edges are present in this major tree. The upper fence bridges nodes $1_a, 1_b$, which have respectively two and five descendants, corresponding to a count $\binom{8}{5} - 1 = 55$ ($1_a, 1_b$ are included). We note that node $1a$ has three branches descending; one fenceless branch with a single node 2_b, and two branches bridged by a fence: one branch with single node 3_a, the other containing nodes 3_b and 4_b. These 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 $\binom{4}{3} = 4$ ways of interlacing the position of 2_b from the other branch descending from 1_a and the amalgamated branch with three nodes. The total number of linear extensions, and so TD-evolutions, associated with connection evolution E is then $55 \cdot 2 \cdot 4 = 440$.*

4.4 The size of TD space

In the previous sections, we have presented an algebraic formalism for the representation of a TD evolution. Also, we have provided a method to calculate the number of TD evolutions corresponding to a single connection evolution, using its major graph. The elucidation of these properties, intrinsic to TDs, has been the basis for the solution of a more challenging problem: determining the total number of TD evolutions. This corresponds, for example, to the 11 evolutions arising from two TDs, represented in Figure 4.5; we clearly see that three of such evolutions correspond to word 12, three to 21 and five to 121. Such evolutionary space has the following size:

Theorem 4.3. *[89] The number $\mathcal{N}_n$ of distinct evolutions arising from n TDs is given by:*

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

This can be exemplified by calculating $\mathcal{N}_2 = (4^1 - (2(1)+1)) \cdot (4^2 - (2(2)+1)) = 11$, for example, which is in agreement with Figure 4.5. The first few terms in this series can be seen in the bottom row of Table 4.1.

The full proof of this result can be found in [89]. It relies on a way of relating space of all n TD structures to that of $n+1$ TD structures via the *inducement of evolutions*, which is now presented.

Consider the following connection evolutions:

$$E = [\mathbf{1} \rightarrow \mathbf{12} \rightarrow \mathbf{12312} \rightarrow \mathbf{1231242}]$$

$$E^{+} = [\mathbf{2} \rightarrow \mathbf{23} \rightarrow \mathbf{23423} \rightarrow \mathbf{2342353}]$$

$$E' = [\mathbf{1} \to \mathbf{121} \to \mathbf{1213} \to \mathbf{2134213} \to \mathbf{213421353}]$$

We can easily note that the first two connection evolutions only differ by the labelling of TD events: precisely, if we increase each symbol in E by 1 we get E^+. On the other hand, evolution E' is made up of five TD events. Now, the important thing to note is that, by deleting all copies of symbol 1 in E' we obtain evolution E^+. Conversely, we find that E' is the result of introducing a new first TD event in E^+. We can then formulate the following definition:

Definition 4.6. *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.*

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 4.1. *Let $D(E)$ be the process where we eliminate 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 connection word **1**. The second TD in E results in connection evolution [**1** $\to$ **12**], [**1** $\to$ **21**] or [**1** $\to$ **121**]. In all these cases, removing connection symbol **1** from the evolution results in the new word **2**, which becomes **1** when the symbols are reduced in value by 1. We then find that the process results in the correct initial word for $D(E)$. Now any TD event corresponds to a mapping of the form $AXB \to AX(n+1)XB$, for possibly empty subwords A, X or B, for the $(n+1)^{th}$ TD.

By eliminating all copies of the symbol 1 from the subwords A, X and B, and reducing all symbols by 1 to give A', X' and B', respectively, we obtain a mapping of the form $A'X'B' \rightarrow A'X'nX'B'$ which is a valid step in the n^{th} step of a TD word evolution, 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 $E' = [\mathbf{1} \rightarrow \mathbf{1}X_1' \rightarrow \mathbf{1}X_2' \rightarrow \mathbf{1}X_3' \rightarrow ... \rightarrow \mathbf{1}X_n']$ 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' results in E, as stated in the lemma. □

Thus it is possible to obtain all word evolutions for $n+1$ TDs from the set of evolutions with n TDs by introducing a new first event. Also, we find that for any connection evolution E on n TDs, the sum across all induced (with a new first TD event) evolutions of the number of TD evolutions is equal to $\mathcal{N}(E)(4^{n+1} - (2(n+1)+1))$ (where $\mathcal{N}(E)$ is the number of TD evolutions associated with word evolution E). This is consistent with Theorem 4.3, stating that the size of the TD space increases by a factor $4^{n+1} - (2(n+1)+1)$ after adding the $(n+1)_{th}$ TD. Moreover, the major graphs of all the induced word evolutions are tightly related to the trees of the original evolution. A detailed description of the subtree operations required to construct an induced 2-d tree can be found in [89].

Such findings are the last important step towards the formulation and proof of Theorem 4.3.

4.5 Computational Analyses

Both Theorems 4.1 and 4.3 have been confirmed computationally, using Matlab [56]. A description of the code for the count of all TD evolutions is now presented.

Linear extensions count algorithm The algorithm utilizes a matrix representation of TD evolutions. For n TD events we can define an n by 2 matrix M, where the i_{th} entry along the first column indicates the interval (a filled line in the zig-zag plot, separated by two diagonal dotted lines) where breakpoint i_a is added, and the i_{th} entry along the second column indicates the interval where breakpoint i_b is positioned. We call this an *evolutionary matrix*. For example, matrix

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 4 & 4 \\ 3 & 4 \end{bmatrix}$$

represents the evolution of Figure 4.4A. Positions $1_a, 1_b$ lie on the first interval along the zig-zag plot, $[0_a, 0_b]$ (which is always true, given that the reference sequence corresponds to a single interval $[0_a, 0_b]$). Breakpoint 2_a is added on the first interval $[0_a, 1_b]$ of the new structure, while 2_b is added on the second interval $[1_b, 0_b]$. Both breakpoint 3_a and 3_b lie on the fourth interval $[2_a, 1_b]$, giving entries $[4, 4]$ in the third row. This corresponds to connection evolution $\epsilon \rightarrow 1 \rightarrow 121 \rightarrow 1231$.

Now, the increase in the number of intervals following the n_{th} TD is given by:

$$k' = k + 1 + (i_b - i_a) \tag{6}$$

where k is the number of intervals prior to the n_{th} TD, and k' is the new number of intervals obtained by adding n_b on the i_bth interval and n_a on the i_ath interval. For example, in Figure 4.4A we start with a single interval ($k = 1$), then we get new structure with $k' = 1 + 1 + (1 - 1) = 2$ intervals after the first TD, then $k' = 2 + 1 + (2 - 1) = 4$, $k' = 4 + 1 + (4 - 4) = 5$ and finally $k' = 5 + 1 + (4 - 3) = 7$.

Let $S = \{M_1, M_2, ...M_k\}$ be a collection of n by 2 evolutionary matrices. Let also $I = \{j_1, j_2, ..., j_k\}$ be the collection of values representing the number of intervals in the zig-zag plot for the evolutions represented by the M_i's in S. Then we counted the size of TD space with the following pseudocode:

Generate evolutionary matrices for n TD events

Data: the number of TD events, n
Result: the complete set of $nx2$ evolutionary matrices
Initialization;
Initialise with segments $a = 1, b = 1$, number of intervals for each TD word $I = \{1\}$, set of associated evolutionary matrices $S = M_1 = [1, 1]$;
if $n = 1$ **then**
  return M_1
else
  for $x = 2...n$ **do**
    $S' = \epsilon$;
    $I' = \epsilon$;
    for every M_i *in* S *and associated number of intervals* j_i *in* I **do**
      for every *couple of integers* a, b, $1 \leq a \leq b \leq j_i$ **do**
        add row $[a, b]$ to M_i as the last (x_{th}) row; and add the new matrix to S'; calculate the associated new number of intervals and add it to I';
      end
    end
    update S to the new set of evolutionary matrices: $S = S'$;
    update I to the new set of interval counts: $I = I'$
  end
end

count the number of TD evolutions:

Data: the set of matrices in S

Result: count of the number of TD evolutions

Initialization;

$count = 0$

for ***each*** *matrix* m *in* S **do**

> Translate m into order constraints, represented by numeric vectors;
>
> Use function *M2bp_orders* (described below) to generate all linear extensions for the set of constraints; determine their count x;
>
> Generate connection evolution corresponding to matrix m;
>
> Use linear extensions and connection evolution to count the copies of each segment;;
>
> Construct the segment copy number profile of each TD evolution;
>
> $count = count+x$

end

return *number of TD evolutions count*

Example 4.6. *function M2bp_orders: we now provide an example of how to derive breakpoint orders from an evolutionary matrix. Consider the evolution of Figure 4.6A. We want to assign an integer value to each breakpoint: let then* $1_a = 1, 1_b = 2, 2_a = 3, 2_b = 4$*; let also* $0_a = 5, 0_b = 6$*. Note these numbers are labels and do not indicate an order. Thus we have an integer representation of our breakpoint which can be used as an input for our Matlab function. We have evolutionary matrix* M *starting with row* $(1, 1)$*. This corresponds to linear order* $0_a < 1_a < 1_b < 0_b$ *after the first TD. We represent that with numeric vector* [5 1 2 6]*, made up of the integer values assigned to the four breakpoints and including their relative order:* 1_a *and* 1_b *lying between* 0_a *and* 0_b*. The second row of* M *is* $(1, 2)$*, representing the fact that* 2_a *lies on interval* $[0_a, 1_b]$ *and* 2_b *lies on interval* $[1_a, 0_b]$*. We similarly represent these constraints with vectors* [5 3 2] *and* [1 4 6]*. Now from* [5 1 2 6] *and* [5 3 2] *we find that* 1 *and* 3 *both lie between* 5 *and* 2*. We can then combine the two vectors and write:*

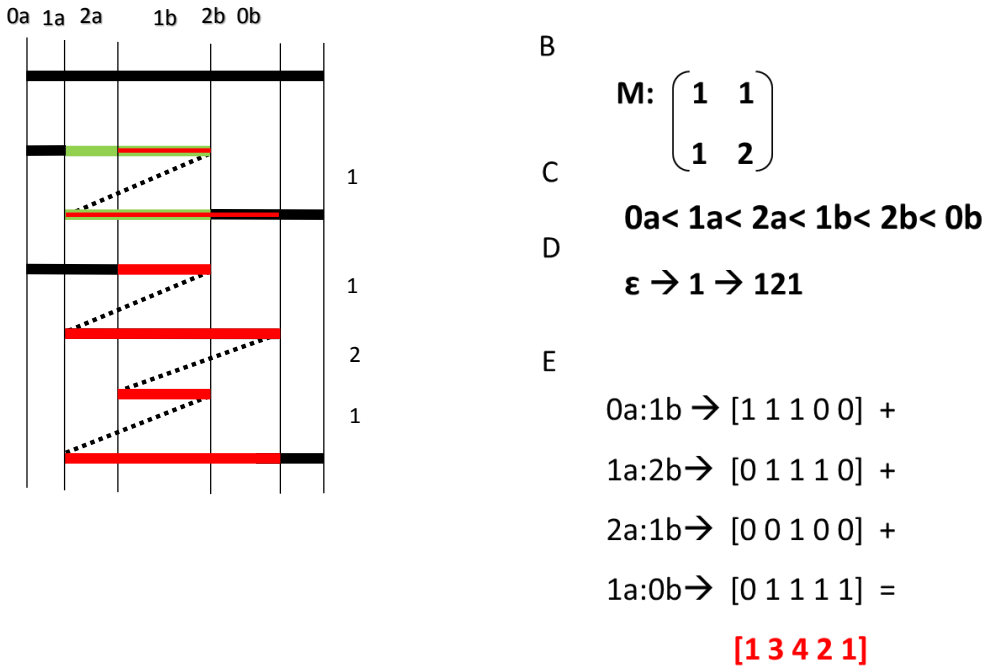

Figure 4.6: Inference of the copy number profile from an evolutionary matrix. A) Zig zag plot representation; B) corresponding evolutionary matrix C) a single linear extension for M, corresponding to the breakpoint order in A; D) connection evolution; E) counting the copies of each segment through the zig-zag structure: the counts can be inferred from the zig-zag plot, or computationally providing the final word in C and breakpoint ordering in D.

[5 3 1 2 6] *(A)*
[5 1 3 2 6] *(B)*
Now, remaining vector [1 4 6] *tells us where element* 4 *can be added. Thus we have to intercalate* 4 *in every position between* 1 *and* 6*, in both vector A and B. We then obtain the following vectors:*
5 3 1 4 2 6
5 3 1 2 4 6
5 1 4 3 2 6
5 1 3 4 2 6
5 1 3 2 4 6
representing all linear extensions associated with the evolution in Figure 4.6*A.*

Example 4.7. *inferring the copy number from a single linear extension: consider the evolution of Figure 4.6A. We want to infer the copy number profile of the represented evolution. Now, we notice from the zig-zag plot that we have breakpoint ordering* $1_a < 2_a < 1_b < 2_b$*, corresponding to vector* $v = [5\ 1\ 3\ 2\ 4\ 6]$ *from Example 4.6. Recall that such a vector also defines the boundaries of the five segments arising in the evolution:* $[5, 1], [1, 3], [3, 2]$*,* $[2, 4]$ *and* $[4, 6]$*. Now, it is possible to construct corresponding evolution* $[1 \rightarrow 121]$ *from evolutionary matrix M. We are then provided with two types of information: the complete linear extension and the final connection word* $w = [\mathbf{121}]$*. We then proceed with the inference of the copy number profile.*

The first symbol in w is **1***, and indicates that the first interval of the sequence is* $[0_a, 1_b]$*.* $1_b = 2$ *is in position* 4 *along v: this means the first interval covers segments* 1 *to* 4*:* $[5, 1], [1, 3], [3, 2]$ *and* $[2, 4]$*. Similarly, subword* [**12**] *in w indicates the second interval is* $[1_a, 2_b]$*: this corresponds to segments* 2 *to* 4*. Subword* [21] *indicates interval* $[2_a, 1_b]$*, that is single segment* 3*. Lastly, the third symbol in w corresponds to a somatic connection from* 1_b *to* 1_a*: thus the last interval spans from* 1_a *to telomere* 1_b*. Based on vector v, we find this interval is formed by segments* 2 *to* 5*. We are now able to count the number of copies of each segment through all intervals in the structure corresponding to w: our copy number vector is then* [1 3 4 2 1].

4.6 The combinatorics of TD temporal words

In the previous sections of this chapter we have seen how to represent a tandem duplication evolution with words of integers, using symbol n to represent the n^{th} event. Also, from Definition 4.3 we have learnt that any TD on a word $[XYZ]$ copies a possibly empty subword Y, leading to new structure $[XYnYZ]$.

We will now focus on Problem 4.2, where we no longer know the order of events and the symbols are thus unordered. We want to infer the number of connection evolutions consistent with a given word. The problem has already been introduced with the example of Figure 4.2. We start from the intuition, based on Definition 4.3, that reversing the TD process implies the elimination of a duplicated subword around a unique symbol n. However, can we always perform this kind of operations in a recursive way, and being confident about the validity of our deconstruction of the TD process? Can we include the order information in a Hasse diagram, and thus infer the number of TD evolutions for a specific word? In this section, we will show that these questions require a deeper understanding of the geometric properties of connection words.

We start with a formalisation of some key concepts and definitions.

Definition 4.7. *A word G of TD symbols is called a* TD word *if and only if it is equal to the empty vector ϵ or it is arises from a connection evolution $\epsilon \to G_1 \to ... \to G_n$ with no breakpoint reuse.*

Definition 4.8. *A* temporal word *is a TD word of connection symbols for which the chronological order of the corresponding TD events is unknown.*

Example 4.8. *Consider connection evolutions $E = [\epsilon \to 1 \to 12 \to 1312]$ and $E' = [\epsilon \to 1 \to 121 \to 1213]$. If we look at final connection words $[1312], [1213]$ and assume that the three TD symbols are unordered, we find that both words can be represented by temporal word $[a\ b\ a\ c]$. Thus we notice that the definition of temporal word is more general than the connection word of Definition 4.2.*

We next formalise the concept of symmetries in a word arising from a tandem duplication.

Definition 4.9. *Any subword B_e in a temporal word G is called a* proper block *if and only if it takes the form $[Y\ e\ Y]$, where e is unique in G and Y is the biggest repeated subword around e. Symbol e is called* centre *of the block B_e. B_e is defined* trivial *if and only if it contains only e.*

Definition 4.10. *Let $G = [Y\ X\ e\ X\ Z]$ be a temporal word resulting from n TDs, and let $B_e = [X\ e\ X]$ be the proper block centred at e. Then the deconstruction operation for event e is the transformation $[Y\ X\ e\ X\ Z] \rightarrow [Y\ X\ Z]$.*

Example 4.9. *Figure 4.2A shows the possible evolutions of temporal word $[a\ b\ a\ c\ d]$. Yellow circles indicates trivial blocks, while proper blocks are underlined in red. Edges connect nodes (words) separated by a single deconstruction step. For example, edge from $[a\ b\ a\ c\ d]$ to $[a\ c\ d]$ represent the reduction of block $[a\ b\ a]$ to a. The number of evolutions back to the empty word correspond to the number of paths in the graph, that is* 12*. Clearly some paths share some nodes, i.e. there are distinct evolutions generating the same temporal word at some stages.*

We must notice here that the definition of proper block does not fully describe the structural properties of connection (and thus temporal) words. Indeed, the structure of a proper block may be modified by one or more following TD events. We account for such situations in the following definition.

Definition 4.11. *A subword B in a TD word G is called a masked block for the proper block $B^0 = [W\ e'\ W]$ if and only if the structure of B^0 in G is modified by subsequent TD events.*

Example 4.10. *Consider the temporal word $[a\ b\ c\ a\ d\ a\ b]$. We have symbols a, b which are duplicated, while c and d are present in a single copy. Now subword $[a\ d\ a]$ is a proper block, suggesting that d has duplicated symbol a. We can then conclude that unique letter d is a candidate last event. Undoing d results in new word $[a\ b\ c\ a\ b]$ which is also a proper block. This describes the key concept of a masked block: a proper block whose structure has been disrupted by one or more internal TDs. Formally, we say that we have proper block $[We'W] = [a\ b\ c\ a\ b]$ turning into masked block $[V\ W'\ e_1\ W'\ Z] = [a\ b\ c\ a\ d\ a\ b]$ where $V = [a\ b\ c]$, $W' = a$, $e_1 = d$ (the event which modifies the original structure) and $Z = b$.*

Next we need to consider an important class of unique symbols, providing a better understanding of the relation between deconstruction operation and TD evolutions.

Definition 4.12. *a letter e in word G is called a candidate last event if and only if*

1. *it is the centre of a proper block $B_e = [W\ e\ W]$*

2. *there exists at least one way of recursively deconstructing one proper block at a time, starting with B_e, until all symbols are removed and we obtain trivial word ϵ.*

We must notice that the definition above does not allow for the precise identification of candidate last events from the final TD word alone. An improved definition of this class of symbols would require deeper investigation on the geometric structure of TD words, avoiding any ambiguity in the identification of unique symbols which cannot be eliminated.

Example 4.11. *Consider word $G = [a\ b\ a\ c\ a\ b\ a\ d\ a\ b\ a\ e\ c\ a\ b\ a]$. We have proper blocks $B_d = [a\ b\ a\ d\ a\ b\ a]$ and $B_e = [e]$. Subword $[a\ b\ a\ c\ a\ b\ a]$ is not a proper block since c is duplicated. Undoing symbol d leads to word $[a\ b\ a\ c\ a\ b\ a\ e\ c]$. The only unique symbol is indeed e, so the next deconstruction must eliminate e to give word $[a\ b\ a\ c\ a\ b\ a\ c]$. However, such a word has no proper blocks to deconstruct. We then find that such a structure cannot arise by a TD process, and therefore d is not a candidate last event in G. Moreover, we notice that, after undoing d and e, we have adjacent repetitions of subword $[a\ b\ a\ c]$. We will see further in the text that such structures cannot arise by a TD process (Theorem 4.4). Undoing symbol e first, however, leads to proper block $[a\ b\ a\ c\ a\ b\ a\ d\ a\ b\ a\ c\ a\ b\ a]$, and G can be deconstructed to the trivial word ϵ with temporal order of TD symbols $e < d < c < b < a$. Thus G corresponds to a masked block.*

In summary, there are two different types of structures within a temporal word: proper blocks and masked blocks. We will later show that any word arising from a TD process (i.e. a TD word) can be represented as a concatenation of subwords, each belonging to exactly one of these block categories.

Now, note that proper blocks include some important order information: event b in the example of Figure 4.2 has duplicated symbol a and thus must have occurred after a. Similarly, in Example 4.10, $[a\ b\ c\ a\ d\ a\ b]$ we find that c and d are candidate last events. We are thus getting a clearer idea on how the structure of TD words can help us in the inference of chronological orders.
We are now able to formalise Problem 4.2 as follows:
Problem 4.4. *Given a temporal word on n symbols arising from n tandem duplications, determine the order constraints between events, and use this information to count evolutions compatible with that word.*

The number of evolutions associated with a temporal word can be clearly derived by a graph like the one shown in Figure 4.2A. However, this might become very complex for words made up of many different symbols. Specifically, we need a better understanding of the geometric properties of words arising from a TD evolution, considering the way multiple events can affect the complexity of the final structure. Here we present the key concept of this section: a Hasse diagram adapted to the solution of Problem 4.4.
Definition 4.13. *A temporal poset graph for a TD word G on n symbols is a graph $I = (V, E)$ where each node in V represents a TD event in $\{1, 2, ...n\}$, and E is the set of edges representing some temporal order constraints between TD events.*

The construction of a temporal poset graph relies on some machinery we need to introduce. We now go through all these findings, which allow for the inference of order relations among TD symbols in a temporal word. For better clarity, we introduce notation $ind(e)$ to denote the set of positions of all copies of symbol e along the word it belongs to. Similarly, we denote $ind(W_j)$ the set of positions occupied by the j_{th} copy of a subword W along the word it is part of.
Theorem 4.4. *For any two copies of a duplicated subword X, the duplicating event e for X must be positioned between those two copies.*

Proof. Assume $...XX...$ is a possible subword for contradiction. For any two consecutive letters $a...a$ there must be a later duplicating letter between. In $...XX...$ pick the latest letter y. Then we have $y...y$ and all letters between the two copies of y are earlier. This is a contradiction which proves the statement of the Theorem. □

From the theorem we can then conclude the following:

Corollary 4.1. *For any two copies of a duplicated subword X the sequence $[X\ X]$ cannot be formed by any TD evolution.*

We now consider the more complex case of multiple symmetries around the same symbol.

Theorem 4.5. *Given the centre e of proper block B_e in a TD word, for any subword repetition $[W\ e\ W]$ around e different from B_e, we have general structure $B_e = [W\ Y\ W\ e\ W\ Y\ W]$, where Y is not empty.*

Proof. By Definition 4.9 we know that a proper block includes the biggest repeated subword around its centre. We then assume that $B'_e = [W\ e\ W] \subset B_e$. Let $B_e = [Z\ W\ e\ W\ V]$ where $[Z\ W] = [W\ V]$ since B_e is a proper block. Then we have $|Z| = |V|$. If $|Z| \leq |W|$ then $W = ZY$ and $W = Y'V$ implying $B_e = [Z\ Z\ Y\ e\ Y'\ V\ V]$. This contradicts Corollary 4.1. If $|Z| > |W|$ then $Z = WY$ and $V = Y'W$, implying $B_e = [W\ Y\ W\ e\ W'\ Y'\ W]$. Then $Y = Y'$ and $B_e = [W\ Y\ W\ e\ W\ Y\ W]$ as stated in the Theorem. □

We have already defined the deconstruction operation as the elimination of a proper block from a TD word. However, the presence of multiple symmetries around a symbol naturally leads to some ambiguities about the extent of the duplicated subword. We deal with this issue in the following theorem.

Theorem 4.6. *Let e be a unique letter in a temporal word. Assume e is the centre of a proper block $B_e = [W\ Y\ W\ e\ W\ Y\ W]$. Then the deconstruction $[...W\ e\ W...] \rightarrow [...W...]$ is not part of any TD evolution.*

Proof. Deconstructing B'_e leads to the new subword $D_e = [W\ Y\ W\ Y\ W]$ which contains two adjacent copies of subword $[W\ Y]$: such structure cannot arise by a TD process according to Corollary 4.1. We conclude that the deconstruction of B'_e is not part of a TD evolution. □

Example 4.12. *Consider temporal word* $[a\ b\ a\ c\ a\ b\ a]$ *corresponding to a proper block. We have candidate last event* c *(the only unique letter) and duplicated proper block* $[aba]$*. We then easily note the structure* $[W\ Y\ W\ e\ W\ Y\ W]$ *described in Theorem 4.5, with* $W = a$*,* $Y = b$ *and* $e = c$*. Undoing block* $[a\ c\ a]$ *leads to word* $[a\ b\ a\ b\ a]$ *where adjacent copies of* $[a\ b]$ *(and* $[b\ a]$ *), incompatible with Corollary 4.1, are observed.*

Now we are still lacking a discussion on particular situations, where two or more proper blocks share some duplicated symbols. For example, consider temporal word [1 2 3 1 2 4 2], where proper blocks [1 2 3 1 2] and [2 4 2] overlap. In these cases, we find that the following is true:

Theorem 4.7. *Let* B_y *and* B_x *be two distinct and possibly overlapping proper blocks in* G*, centred at candidate last events* x *and* y*. Then both deconstruction orders* $x < y,\ y < x$ *are possible.*

Proof. Assume first the two blocks do not overlap. Then we can represent G with the structure $[Z\ B_x\ W\ B_y\ V]$, for possibly empty subwords Z, W and V. Assume also, w.l.o.g., that B_x is on the left of B_y. Now any deconstruction order of the two blocks will lead to the same new word $G' = [Z\ X\ W\ Y\ V]$ where X and Y are the duplicated subwords of B_x and B_y, respectively. By Definition 4.12, G' is assumed to be part of a valid TD evolution, and we find that both deconstruction orders $x < y,\ y < x$ are possible.
If the two blocks overlap then G has structure $[Z\ B'_x\ W\ B'_y\ V]$, where W is the not-empty intersection between B_y and B_x. We notice that x (resp. y) cannot be part of the duplicated subword Y (resp. X), since the centre of symmetry of any proper block is unique by Definition 4.9. Consequentially we require $x, y \notin W$. Now we can rewrite the structure of G as $[Z\ X'\ W\ x\ X'\ W\ Y'\ y\ W\ Y'\ V]$ where $[X'\ W] = X$, $[W\ Y'] = Y$. Any

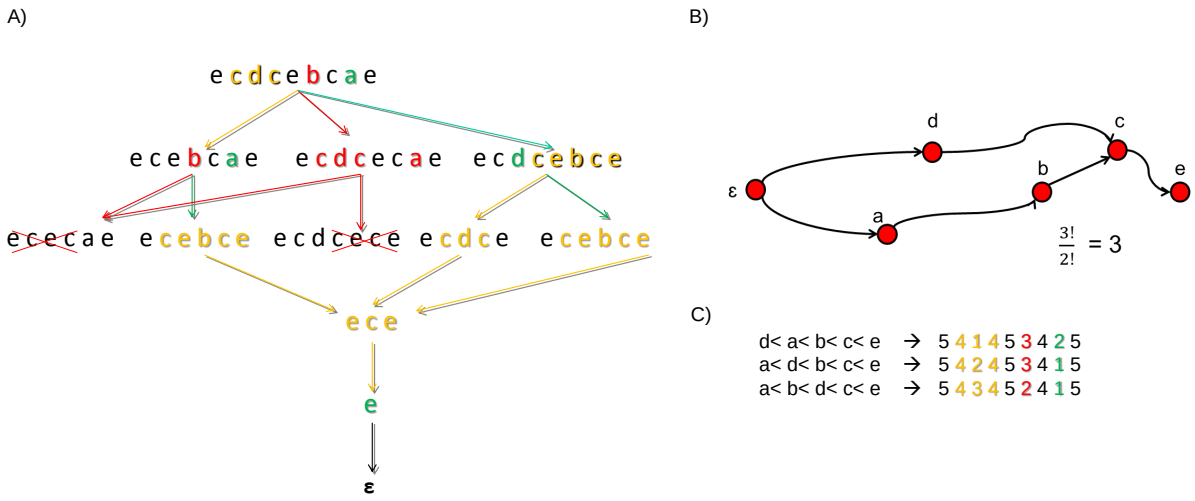

Figure 4.7: The deconstruction process. A) Representation of all deconstruction paths for temporal word $[e\ c\ d\ c\ e\ b\ c\ a\ e]$, as well as some incorrect deconstruction operations. Yellow labelled substrings indicate last event proper blocks, green-labelled letters represent last event trivial blocks, while red-labelled substrings represent other blocks. Note that deconstructing the red class of subwords does not lead to a new TD word (in words $[e\ c\ e\ c\ a\ e]$ $[e\ c\ d\ c\ e\ c\ e]$ we find adjacent repetitions of $[e\ c]$ and $[c\ e]$, resp., which are incompatible with Corollary 4.1). Edges represent single deconstruction steps, colour labelled according to the type of block involved in the operation. B) Hasse diagram for the temporal poset based on word $[e\ c\ d\ c\ e\ b\ c\ a\ e]$. Nodes represents TD events, while edges represent temporal order relations. The combinatorial term for the number of deconstruction paths is also shown. C) The three linear extensions compatible with the Hasse diagram in B), together with the corresponding connection words. These are obtained by substituting each symbol in $[e\ c\ d\ c\ e\ b\ c\ a\ e]$ with its positions $(1, 2, 3, 4$ or $5)$ in the linear extension.

deconstruction order of x and y then leads to new word $[Z\ X'\ W\ Y'\ V]$. □

Thus we have provided a series of rules allowing for the inference of order relations among TD symbols. The deconstruction of a proper block centred at a candidate last event represents an operation which is part of a complete TD evolution (Definition 4.9, Theorem 4.4), and the elimination of any smaller symmetry around a candidate last event leads to an invalid word (Theorems 4.5,4.6). Proper block structure tell us about existing order

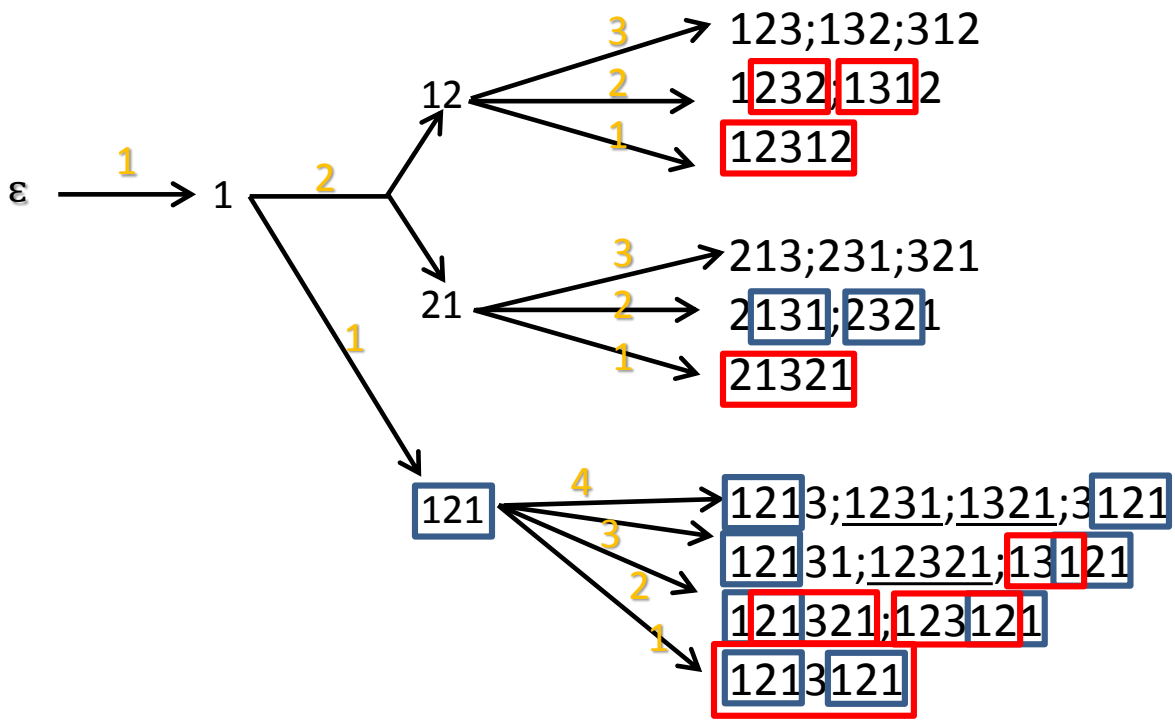

Figure 4.8: Schematic representation of all connection evolutions up to the third TD. Red/blue rectangles indicate proper blocks, while masked blocks are underlined. Remaining letters represent trivial blocks.

constraints between duplicated and duplicating symbols. Such relations among symbols can be represented as edges between nodes in a temporal poset graph, like in Figure 4.9B. Moreover, Theorem 4.7 guarantees that centres of symmetry are not restricted in order relative to each other. A series of problems have yet to be addressed though, which we discuss in the following section.

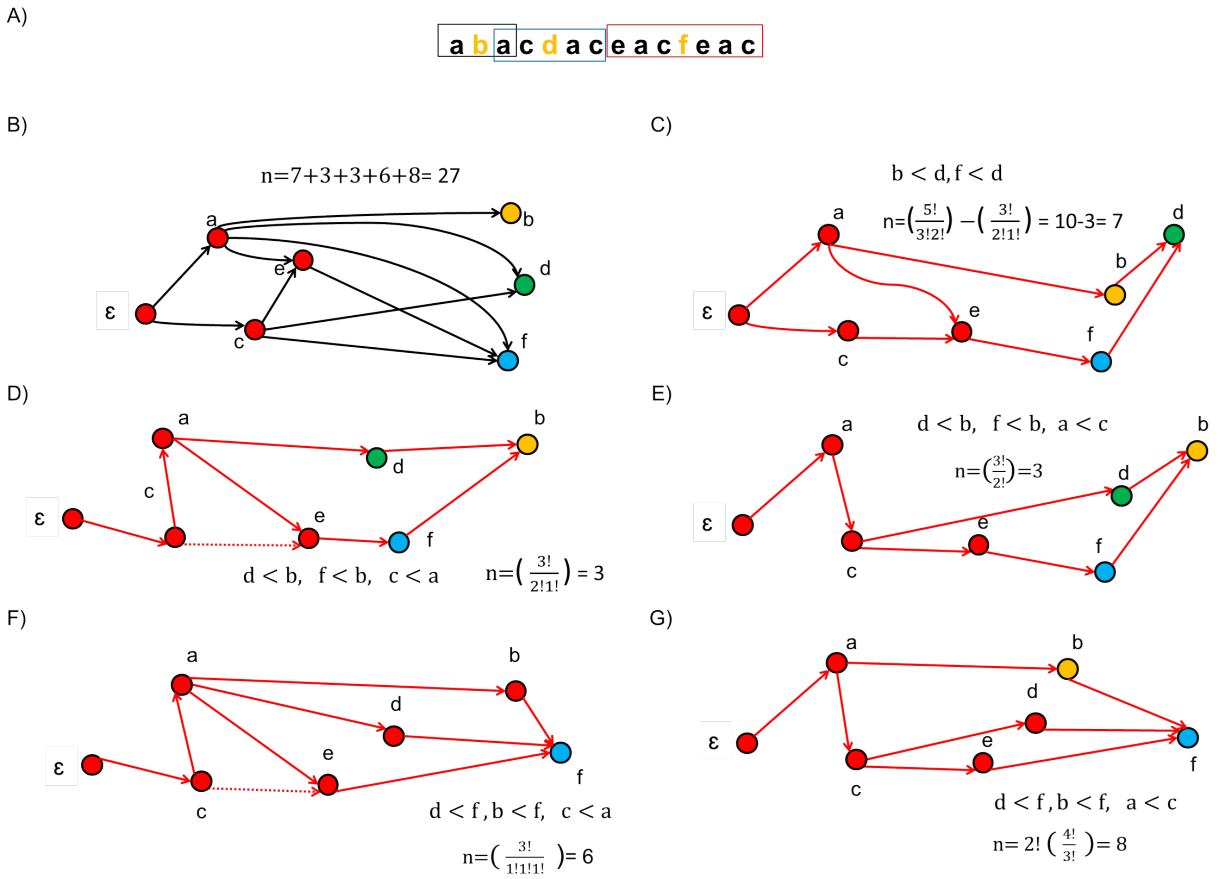

Figure 4.9: The deconstruction process. A) Representation of temporal word $[a\ b\ a\ c\ d\ a\ c\ e\ a\ c\ f\ e\ a\ c]$. B) Corresponding temporal poset graph, with the number of linear extensions n expressed as a sum across values for graphs in C-G. Node ϵ is the root; all other nodes represent TD events. Edges are inferred from the temporal word in A) and indicate order relations; dotted ones give redundant information. Any linear extension for this graph has exactly one of b, d or f (differentially colour-labelled) as the last element. C-G) reduction of the original poset graph to a set of simpler structures, each representing a subset of the solutions for the graph in B. The set of newly added order constraints is given for each graph, as well as the resulting number of linear extensions, n.

4.6.1 The construction of a temporal poset

Based on the Definition and Theorems presented, we now describe the procedure of constructing a temporal poset graph from a TD temporal word. Consider the word in Figure 4.9A: proper blocks are highlighted by coloured squares, while the correspondent centres are yellow labelled. We start the construction of the temporal poset by representing each symbol a, b, c, d, e, f in Figure 4.9A with a node. Then we proceed in the identification of order relations. From proper block $a\ b\ a$ we infer order constraint $a < b$ and we add an edge from a to b. From $a\ c\ d\ a\ c$ we derive restrictions $a < d, c < d$ and we add edges (a, d),(c, d) to our graph. Similarly, we infer order relations $e < f, a < f, c < f$ from the rightmost proper block $[e\ a\ c\ f\ e\ a\ c]$, and corresponding edges complete the construction of our temporal poset. Figure 4.9B shows the complete graph we have obtained. We must notice here that the restriction to exactly two parents, typical of 2-d trees, is lost in the case of temporal poset graphs: for example, we see that node f has three parents, while node c has single parent ϵ, representing the root. Thus we cannot construct a result analogous to Theorem 4.2 for the temporal poset case. The inference of all deconstruction paths for the word in Figure 4.9A leads to 27 different evolutions. Such number can be inferred from the corresponding poset graph (Figure 4.9B) by converting the structure into multiple, more tractable graphs: restricting the solutions to the case where d is the last event, for example, leads to the new graph of Figure 4.9C. This is equivalent to adding restrictions $b < d, f < d$ in the poset. Now if we ignore edge $a \to e$ we have a bubble structure where a, b are unrestricted to c, e, f. The number of ways of ordering these five elements is then $\binom{5}{3,2} = 10$. However, edge $a \to e$ excludes three of these ten solutions: specifically the cases $[c < e < a < b < f < d]$, $[c < e < a < f < b < d]$ and $[c < e < f < a < b < d]$. Thus we have $10 - 3 = 7$ linear extensions for this graph. In the graph of Figure 4.9G we have set f as the last event, and c later than a: that is additional restrictions $d < f, b < f, a < c$. Now, the bubble containing e and d has count $\binom{2}{1,1} = 2$, while c, e, d and b can be ordered in $\binom{4}{3,1} = 4$ different ways. The graph then has $2 \cdot 4 = 8$ different solutions. Summing the results across graphs in Figure 4.9B-G results in $n = 7 + 3 + 3 + 6 + 8 = 27$ linear extensions, as found with the deconstruction approach.

Moreover, the generalisation of the concept of masked block proved to be very challenging. Masked blocks provides with crucial additional information about the order relations among symbols. For example, consider temporal word $[e\ c\ d\ c\ e\ b\ c\ a\ e]$ represented in Figure 4.7A. Such a word contains two masked blocks: $[e\ c\ d\ c\ e]$ and $[c\ e\ b\ c\ a\ e]$. The former has unique symbol d as the centre of proper block $[c\ d\ c]$. Undoing d reveals a pre-existing proper block, $[e\ c\ e]$. Thus d is an event added to $[e\ c\ e]$ forming a masked block, and we have order relation $c < d$. As for the latter masked block, we notice that deconstructing a reveals proper block $[c\ e\ b\ c\ e]$, and we write $b < a$. Despite the relative ease in identifying such structures in these examples, the complexity of the TD word structure rapidly increases with the number of TD events, making the inference of all order relations harder. Thus, counting the linear extensions corresponding to all evolutions proved to be a non-trivial problem.

4.7 Conclusions

We have seen in Table 4.1 that the number of different evolutions increases hyper-exponentially. This means that beyond five or six tandem duplications it is at present unrealistic to attempt to computationally explore this space in its entirety. This makes it difficult to compare any observations to the entire set of possibilities. This is further compounded by the non-uniqueness of copy number vectors, which grow at a far slower rate than the number of evolutions. This means that even if the precise copy number vector is known, it will correspond to a multitude of evolutions, all of which explain the data equally well. One could attempt to apply the type of analyses of [72] to TDs; it is an open problem to determine the number of copy number vectors, or even an efficient algorithm to determine whether a copy number vector can arise from a process of tandem duplication under the assumption of unique breakpoint use.

A parallel investigation on the temporal properties of words arising by TD has also been

implemented. We showed how to include the order relations information in a graph, in order to determine the number of evolutions associated with a particular word structure. However, the complexity of these graphs, free from the two parents restriction typical of 2-d trees, represents the main limiting factor of this approach, for which a general formula for the count of linear extensions has not been obtained. While complex calculations and modifications of the graph can provide the solutions of the most challenging cases, the method suffers for a limited understanding of the space of possible poset graph structures. Moreover, the fast increasing complexity of word structures makes it non trivial to identify all order relations associated with TD evolutions.

The methods we have presented here are closely related to those used to study breakage fusion cycles [43]. This suggests it may be possible to describe a more general space of rearrangements using a largely equivalent approach. Since tandem duplication and breakage fusion cycles are known to play a major role in the formation of large scale copy number variation in cancer genomes, a generalization of these methods to the combined space of these rearrangement processes may help to better understand the evolution of such aberrant genomes.

5 The combinatorics of Inverted Duplication

Following the work on tandem duplication, we have focused on a similar evolutionary process occurring in cancer cells: inverted duplication (ID). The combinatorial questions arising from such a process are equivalent, but the structures obtained in such a modified model are more complex, making the challenge of describing the space of structures more difficult. By using modified versions of the bi-coloured 2d tree and zig-zag plot, we have described such structures, with associated similarities and differences compared to the corresponding findings for TDs. Inverted duplications can be responsible for aberrant neurodevelopmental phenotypes, as well as for gene amplifications in cancer genomes. The molecular mechanisms underlying inverted duplications are still not clear. Recently, a Fold-back model of inverted duplication was proposed [53]. This model predicts a variety of outcomes: inverted duplication with terminal deletions, adjacent to translocated sequences, or circular inverted duplicated chromosomes. Being mainly interested in comparing the combinatorial properties of TDs and IDs together, we will consider a simpler model where the new copy of the duplicated region is inserted adjacent to the original one, without any additional rearrangement or circularisation. This allows us for an easier comparison between the TD and ID evolutions, while avoiding our challenge to get overly hard by considering more than one type of rearrangements.

In the example of Figure 5.1A we start with a reference sequence divided into five contiguous regions. Using the same representation as for TDs, we label these regions with symbols $\underline{1}$, $\underline{2}$, $\underline{3}$, $\underline{4}$ and $\underline{5}$, respectively. The initial inverted duplication copies region $\underline{23}$ and inserts this new copy next to the original one with inverted orientation: this results in the new sequence $\underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{45}$. In this example we have used bold symbols $\mathbf{1}$ and $\overline{\mathbf{1}}$ to indicate the two newly introduced connections. The right side of segment $\underline{3}$ is connected to itself, indicating the beginning of the inverted sequence; the left side of segment $\underline{2}$ is connected to the left side of segment 4 (Figure 5.1A). Note also that the left hand end of the duplicated region $\underline{23}$ implicates the reference position between segments $\underline{1}$ and $\underline{2}$,

the right hand end implicates the reference position between segments $\underline{3}$ and $\underline{4}$. Here we introduce the concepts of *starting* and *ending* breakpoint of a connection: in the case of type $\overline{x}$ connections, we have identical starting and ending breakpoint, corresponding to the beginning of the inverted sequence; for a connection x, the start and the end breakpoint represent, respectively, the ending and the starting point of the inversion. In our example, 1 has single breakpoint 3, while $\overline{1}$ connects breakpoint 1 to breakpoint 3.

Next we have the second inverted duplication, copying region $\underline{2^{-1}4}$ to finally give $\underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{\mathbf{1}}\mathbf{2}\underline{\mathbf{2}4^{-1}}\overline{\mathbf{1}}^{\mathbf{-1}}\underline{\mathbf{2}}\overline{\mathbf{2}}\underline{5}$. We now have another pair of connections, labelled $\mathbf{2}, \overline{\mathbf{2}}$: the former connects the right side of segment $\underline{4}$ to itself, the latter links the right side of segment $\underline{2}$ to the left side of $\underline{5}$, as seen in Figure 5.1A. The leftmost end of the inverted duplicated region $\underline{2}\overline{\underline{1}}\underline{4}$ implicates the reference position between $\underline{2}$ and $\underline{3}$, the rightmost end implicates that between $\underline{4}$ and $\underline{5}$. We have thus implicated all four breakpoints between the five reference segments exactly once; a process with unique breakpoint use.

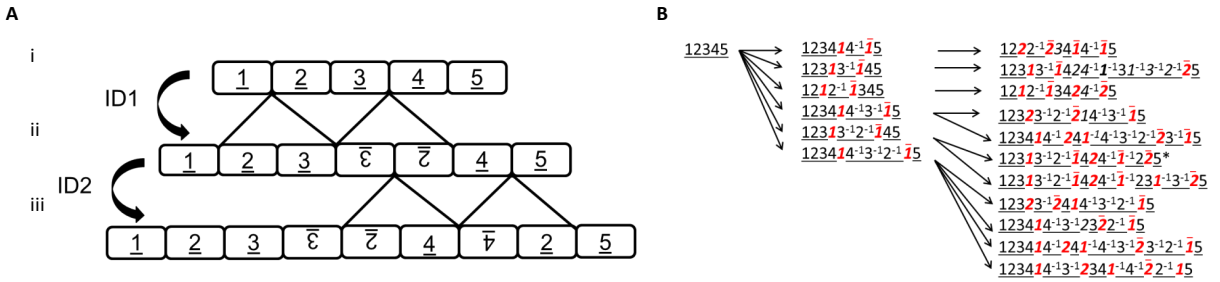

Figure 5.1: An Inverted duplication Process. A) Three structures i)-iii) arising from two tandem duplications on a reference of five regions; $1, 2, 3, 4, 5$. B) Eleven possible evolutions with two inverted duplications. The example in A is highlighted by *. Underlined numbers are segments. Red labelled, bold italicized numbers indicate connections between segments formed in the n^{th} tandem duplication.

Figure 5.1B shows all 11 different ways that two inverted duplications can act on five segments with unique breakpoint reuse. This is exactly the same count as for two tandem duplications, suggesting that the complete evolutionary space might be the same size.

We want now to formulate the following problem, equivalent to Problem 4.1:

Problem 5.1. *Count the number of different ways that an initial string of* $2N+1$ *segments*

can evolve under N inverted duplications, without reusing breakpoints.

5.1 Representation

We now formalize the representations described in the examples above. We first define N as the total number of IDs that take place. Similar to TDs, we have words made up of two classes of symbols: segment and connection symbols. However, we find that for inverted duplications we need two different connection symbols for each event n, which we denote n and $\overline{n}$.

Definition 5.1. *We define $\underline{i^{\pm 1}}$ to be a* segment *symbol, where sign indicates the orientation. This represents a copy of the DNA segment originally in the i^{th} reference position, where $i \in \{1, ..., 2N+1\}$.*

Definition 5.2. *We define any* $\mathbf{i}$ *and* $\bar{\mathbf{i}}$ *to be a* connection *symbol. These symbols always have a segment symbol on either side. In any subword of the form $\underline{m^{\pm 1}}$ $\mathbf{i}/\bar{\mathbf{i}}$ $\underline{n^{\pm 1}}$, $\mathbf{i}$ (or $\bar{\mathbf{i}}$) represents a connection between the end of the DNA segment represented by $\underline{m}^{\pm 1}$, and the start of the DNA segment represented by $\underline{n}^{\pm 1}$, formed during the i^{th} ID, where $i \in \{1, ..., N\}$.*

Then we can construct our ID evolutions as follows:

Definition 5.3. *An* ID evolution *U is defined as any sequence of* ID words *$[U_0 \to U_1 \to ... \to U_N]$ generated as follows. We initialize the sequence with ID word $U_0 = \underline{1} \cdot \underline{2} \cdot ... \cdot \underline{(2N+1)}$. We obtain an ID 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 $U_n = X \cdot Y \cdot \mathbf{n} \cdot Y^{-1}\overline{\mathbf{n}} \cdot Z$. We define n_A as the value of the rightmost symbol of X, x, if that symbol is not reversed, and $x^{-1} - 1$ otherwise. We also define n_B as the value of the rightmost symbol of Y,y, if such symbol is not reversed, and $y^{-1} - 1$ otherwise. We have an ID evolution provided n_A and n_B are $2N$ distinct values for $n \in \{1, ..., N\}$. The values n_A and n_B are referred to as* breakpoint numbers*, and the underlying value n is the* ID

number. *We also define breakpoint numbers* 0_A *and* 0_B *to be* 0 *and* $2N+1$*, respectively, representing the start and end of the set of original reference segments. We let* $\mathcal{U}_N$ *denote the set of possible ID evolutions arising from* N *IDs.*

Example 5.1. *In the evolution of Figure 5.2A we have* $E = [\underline{12345} \rightarrow \underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{45} \rightarrow \underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{4}\mathbf{2}\underline{4^{-1}}\overline{\mathbf{1}}^{\mathbf{-1}}\overline{\mathbf{2}}\underline{5}]$*. The second ID duplicates subword* $\underline{2^{-1}}\overline{\mathbf{1}}\underline{4}$ *in ID word* $\underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{45}$*. The subwords* X*,* Y *and* Z *are then* $\underline{123}\mathbf{1}\underline{3^{-1}}$*,* $\underline{2^{-1}}\overline{\mathbf{1}}\underline{4}$ *and* $\underline{5}$*, respectively. We then find that the rightmost symbol in* X 3^{-1} *is reversed, and thus we have* $2_A = 3-1 = 2$*; the value of the rightmost symbol in* Y *is* 4*, hence* $2_B = 4$*. Note that* $2_A, 2_B$ *demarcate breakpoints of the duplicated region; the* 2^{nd} *breakpoint is implicated by the left end of the duplicated region* $\underline{2^{-1}}\mathbf{1}\underline{4}$*, between inverted segments* $\underline{2}$ *and* $\underline{3}$*; the* 4^{th} *breakpoint is implicated by the right end of the duplicated region between segments* $\underline{4}$ *and* $\underline{5}$*. For a comparison with a TD evolution with 2 events, producing the same breakpoint ordering along the reference* $(1_A \leq 2_A \leq 1_B \leq 2_B)$*, refer to Figure 5.3A. Note the absence of inverted segment symbols in the TD case, as well as the presence of a unique connection symbol for each TD event.*

Now, we shall recall Definition 4.4 for Tandem Duplication, showing how any TD evolution can be restricted to connection symbols, leading to a simpler algebraic representation. We can then use the same concept for the ID process, as shown in the following example:

Example 5.2. *In the evolution * of Figure 5.1B we have:* $U = [\underline{12345} \rightarrow \underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{45} \rightarrow \underline{123}\mathbf{1}\underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{4}\mathbf{2}\underline{4^{-1}}\overline{\mathbf{1}}^{\mathbf{-1}}\overline{\mathbf{2}}\underline{5}]$ *which becomes* $E = [\epsilon \rightarrow \mathbf{1}\overline{\mathbf{1}} \rightarrow \mathbf{1}\overline{\mathbf{1}}\mathbf{2}\overline{\mathbf{1}}^{\mathbf{-1}}\overline{\mathbf{2}}]$ *when the segment symbols are removed.*

Definition 5.4. *The function* $\mathcal{C}$*, mapping any connection symbol* m *or* $\overline{m}$ *to the corresponding starting (s) or ending (e) breakpoint is defined as follows:*

$\mathcal{C}(m, s) = m_B$
$\mathcal{C}(m^{-1}, s) = m_B$
$\mathcal{C}(\overline{m}, s) = m_A$
$\mathcal{C}(\overline{m}^{-1}, s) = m_B$
$\mathcal{C}(m, e) = m_B$
$\mathcal{C}(m^{-1}, e) = m_B$

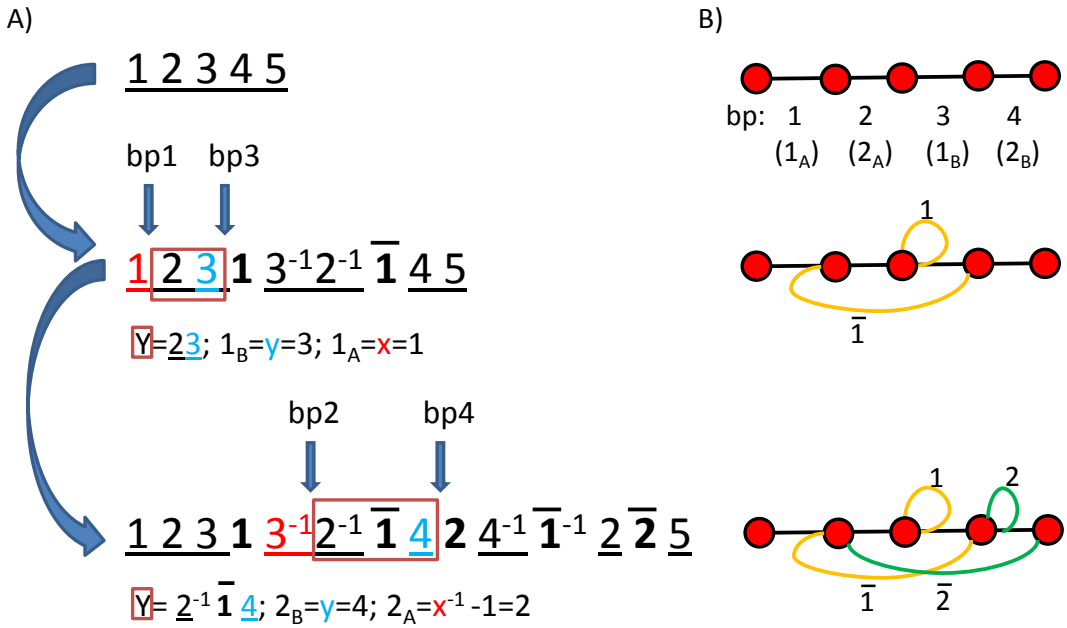

Figure 5.2: A) Evolution * of Figure 5.1B, with further explanation on the correspondence between breakpoint values N_A, N_B and adjacent letters $\underline{x}, \underline{y}$. B) inverted duplication graphs showing reference (black) and somatic (coloured) connections at each evolutionary step. Somatic connections labelled with the same colour belong to the same ID event.

$\mathcal{C}(\overline{m}, e) = m_B$
$\mathcal{C}(\overline{m}^{-1}, e) = m_A$

Example 5.3. *Consider Figure 5.2B. We note that for connections of type $m \in (1, 2)$, m_B is both the starting and the ending breakpoint. Thus we have, for example, $\mathcal{C}(1, s) = \mathcal{C}(1^{-1}, s) = \mathcal{C}(1, e) = \mathcal{C}(1^{-1}, e) = 1_B$. As for connection of type $\overline{m}$ $\overline{1}$, it connects 1_A and 1_B together. After two ID events, we get a word containing both letters $\overline{1}$ and $\overline{1}^{-1}$. From Figure 5.2B we see that connection $\overline{1}$ starts at breakpoint $1 = 1_A$ and ends at breakpoint $3 = 1_B$. So we have $\mathcal{C}(\overline{m}, s) = m_A, \mathcal{C}(\overline{m}, e) = m_B$. Conversely, letter $\overline{1}^{-1}$ is crossed in the opposite direction, from 1_B to 1_A; thus $\mathcal{C}(\overline{m}^{-1}, s) = m_B, \mathcal{C}(\overline{m}^{-1}, e) = m_A$.*

Lemma 5.1. *Let* **m** *and* **n** *be consecutive connection symbols in any connection word E_i,*

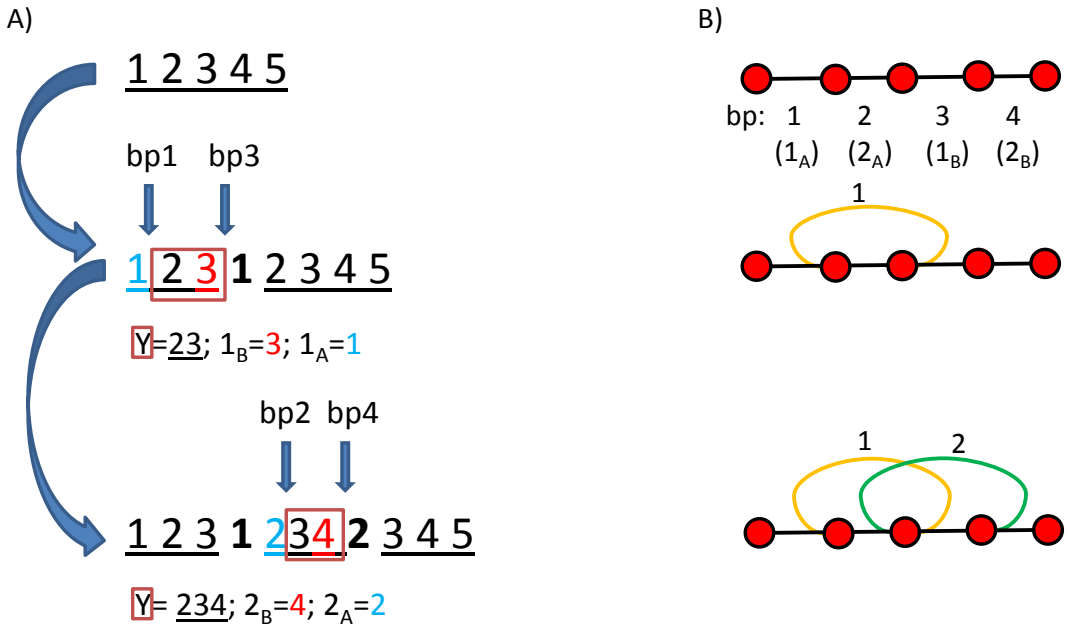

Figure 5.3: A) Example of tandem duplication evolution leading to the same breakpoint ordering as in Figure 5.2A, with further explanation on the correspondence between breakpoint values N_A, N_B and adjacent letters $\underline{x}, \underline{y}$. B) Tandem duplication graphs showing reference (black) and somatic (coloured) connections at each evolutionary step.

induced from ID word U_i. *Then* **m** *and* **n** *are either separated in* U_i *by segment symbols* $\underline{\mathcal{C}(m,e)+1} \cdot \ldots \cdot \underline{\mathcal{C}(n,s)}$ *if* $\underline{\mathcal{C}(m,e)} < \underline{\mathcal{C}(n,s)}$ *or* $\underline{(\mathcal{C}(m,e))^{-1}} \cdot \ldots \cdot \underline{(\mathcal{C}(n,s)+1)^{-1}}$ *if* $\underline{\mathcal{C}(m,e)} > \underline{\mathcal{C}(n,s)}$. *If* **m** *is the first connection symbol in* E_i, *in* U_i *it is preceded by segment symbols* $\underline{1} \cdot \ldots \cdot \underline{\mathcal{C}(m,s)}$. *If* **n** *is the last connection symbol in* W_i, *in* U_i *it is followed by segment symbols* $\underline{\mathcal{C}(n,e)+1} \cdot \ldots \cdot \underline{2N+1}$.

Proof. If we have consecutive symbols **mn** in a connection word, then in the corresponding ID word the two connections must be separated by a series of segments bounded by breakpoints $\mathcal{C}(m,e)$ and $\mathcal{C}(n,s)$. We have two possibilities: either $\mathcal{C}(m,e) < \mathcal{C}(n,s)$ along the reference sequence, or *vice versa*. In the former case, we find that the segment sequence corresponds to $\underline{\mathcal{C}(m,e)+1} \cdot \ldots \cdot \underline{\mathcal{C}(n,s)}$; in the latter, we have sequence $\underline{(\mathcal{C}(m,e))^{-1}} \cdot \ldots \cdot \underline{(\mathcal{C}(n,s)+1)^{-1}}$. □

Example 5.4. *Consider the final ID word of Figure 5.2A,* $\underline{123}\mathbf{1}\ \underline{3^{-1}2^{-1}}\overline{\mathbf{1}}\underline{\mathbf{4}}\mathbf{2}\underline{\mathbf{4^{-1}}}\ \mathbf{1^{-1}}\underline{\mathbf{2}}\overline{\underline{\mathbf{2}}}\underline{\mathbf{5}}$. *We have consecutive connection symbols* $\mathbf{1}$ *and* $\overline{\mathbf{1}}$ *separated by segments* $\underline{3^{-1}2^{-1}}$. *We have* $\mathcal{C}(1,e) = 1_B = 3$ *and* $\mathcal{C}(\overline{\mathbf{1}},s) = 1_A = 1$, *so* $\mathcal{C}(1,e) > \mathcal{C}(\overline{\mathbf{1}},s)$. *according to Lemma 5.1 these connection symbols must be separated by sequence of segment symbols* $\underline{(\mathcal{C}(\mathbf{1},\mathbf{e}))^{-\mathbf{1}}} \cdot \ldots \cdot \underline{(\mathcal{C}(n,s)+1)^{-1}}$ *From Definition 5.4 we have* $\mathcal{C}(\mathbf{1},\mathbf{e}) = \mathbf{1_B} = \mathbf{3}$ *(the third breakpoint along the chromosome, see Figure 5.2B) and we find that* $\mathcal{C}(\mathbf{1},\mathbf{e})^{-\mathbf{1}} = \underline{\mathbf{3^{-1}}}$ *is the first segment after connection symbol* $\mathbf{1}$. *Similarly, we have* $\mathcal{C}(\overline{\mathbf{1}},s) = 1_A = 1$ *and* $(\mathcal{C}(\overline{\mathbf{1}},s)+1)^{-1} = (1+1)^{-1} = 2^{-1}$ *is the second last segment symbol separating the connection symbols. If we consider consecutive connection symbols* $\overline{\mathbf{1}}$ *and* $\mathbf{2}$ *we find they are separated by segment symbol* $\underline{4}$. *Now,* $\mathcal{C}(\overline{\mathbf{1}},e) = 1_B = 3$ *and* $\mathcal{C}(\overline{\mathbf{2}},s) = 2_B = 4$, *so* $\mathcal{C}(\overline{\mathbf{1}},e) < \mathcal{C}(\overline{\mathbf{2}},s)$. *Thus, according to Lemma 5.1 these connection symbols must be separated by sequence of segment symbols* $\mathcal{C}(\overline{\mathbf{1}},e)+1 \cdot \ldots \cdot \mathcal{C}(\overline{\mathbf{2}},s)$. *From Definition 5.4, we find that* $\mathcal{C}(\overline{\mathbf{1}},e)+1 = 1_B+1 = 3+1 = 4$, *and* $\mathcal{C}(\overline{\mathbf{2}},s) = 2_B = 4$. *Thus the sequence of segments is the single symbol* $\underline{4}$.

It is then straightforward to formalize Problem 5.1:
Problem 5.2. *Determine the size of the set* $\mathcal{U}_N$ *for inverted duplications.*

We will next present a proof for the recursion on the number of ID connection evolutions. Also, we will provide an empirical evidence for an identical evolutionary space size between tandem and inverted duplication, based on the construction of all evolutions up to the third event.

5.2 The space of connection evolutions

We now consider how many connection evolutions can arise from N IDs, the size of the set $\mathcal{E}_N$. In Figure 5.4 we see a graph representation of the possibilities, where values $w_{m,n}$ (number of connection words of size m after n IDs) are equivalently obtained by

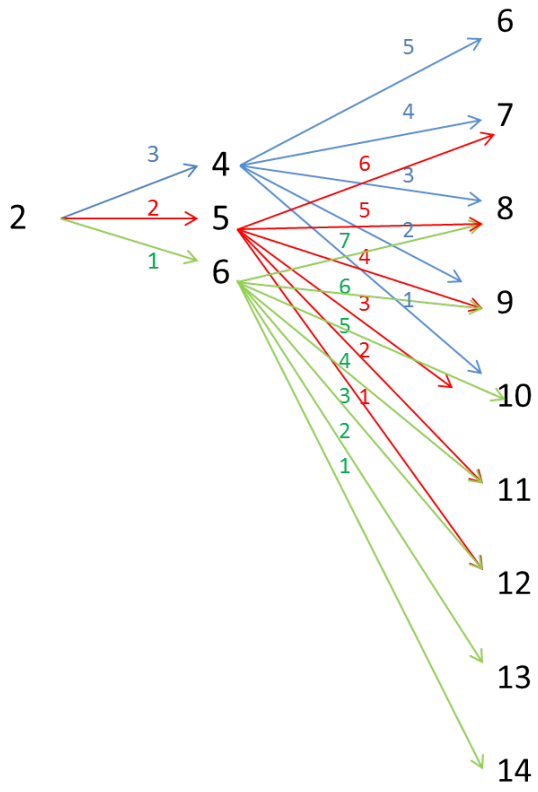

Figure 5.4: Schematic representation of the number of possible ID words. Numbers at nodes indicate the length of ID words. Numbers on edges indicate the number of choices.

taking products of the edge values along paths to the associated node, from the node labelled 1, and summing. For example, the node labelled 12 in the third column of nodes corresponds to $w_{12,3}$ and has two paths, one with product $2 \cdot 1$, the other with $1 \cdot 3$ and we find $w_{12,3} = 2 + 3 = 5$, five connection words of length 12; $\mathbf{1\bar{1}2\bar{1}^{-1}\bar{2}3\bar{2}^{-1}\bar{1}2^{-1}\bar{1}^{-1}1^{-1}\bar{3}}$, $\mathbf{121^{-1}\bar{2}1\bar{3}\bar{1}^{-1}\bar{2}^{-1}12^{-1}1^{-1}\bar{3}}$, $\mathbf{1\bar{1}2\bar{1}^{-1}1^{-1}\bar{2}3\bar{2}^{-1}1\bar{1}2^{-1}\bar{3}}$, $\mathbf{1\bar{1}2\bar{1}^{-1}1^{-1}31\bar{1}2^{-1}\bar{1}^{-1}\bar{3}\bar{2}}$ and $\mathbf{1\bar{1}2\bar{1}^{-1}3\bar{1}2^{-1}\bar{1}^{-1}1^{-1}\bar{3}1^{-1}\bar{2}}$.

We have obtained the following general result.

Theorem 5.1. *Initialise with* $w_{0,0} = 1$. *Then the number of words of size* m *after* n

inverted duplications (ID) is given by

$$w_{m,n} = \sum_{k=\lceil(\frac{m-2}{2})\rceil}^{m-2} (2k - m + 3) w_{k,n-1} \tag{7}$$

Proof. For a word of size k we have $k+1-i$ ways of performing an inverted duplication of an i letters long subword. Since we add 2 new letters at each event, the minimum increase of word size an ID can determine is 2(i.e. we have a new word of size $k+2$), while the maximum is equal to $k+2$ (i.e. we have a new word of size $k+k+2 = 2k+2$) Also, we have that an event transforming a word of size k to a word of size m ($k+2 \leq m$) has duplicated a subword of size $m-2-k$. Consequentially, we have $i = m-2-k$ and the number of ways of transforming a word of size k to a word of size m is $k+1-i = k+1-(m-2-k) = 2k-m+3$. Such equation is to be applied to the range of possible k values, i.e $\lceil(\frac{m-2}{2})\rceil : m-2$. □

The counts of connection evolutions arising from the first few IDs can be seen in Table 5.1.

The counts $|\mathcal{W}_n| = \sum_m w_{m,n}$ of words arising from n IDs can be seen in Table 5.1.

IDs	1	2	3	4
CNVs	1	6	169	-
Words	1	6	115	5887
ID-Evolutions	1	11	627	-

Table 5.1: Counts of copy number vectors, words and ID evolutions.

5.3 Using posets to count ID evolutions

5.3.1 Zig-zag plots

We will be using a modified version of the zig-zag plot representing a TD evolution. Horizontal (solid) lines still indicate DNA segments, diagonal dotted lines represent connection symbols of type $\overline{n}$, while dotted curved lines correspond to connection symbols of type n. An example of plot is shown in Figure 5.5A.

5.3.2 2d-trees

We now introduce the ID structure using the example in Figure 5.5. Similar to TDs, we start with a single segment $[0_a, 0_b]$, represented as an horizontal line in Figure 5.5Ai. Node 0_A is assigned a type *start* (coloured red), indicating it is the starting point of a segment. Node 0_B is assigned a type *end*, (coloured blue) indicating the end of a segment. As in TDs, these two nodes are bridged by a directed edge (fence) which represents their ordering in the reference; $0_A < 0_B$. Note that here we use lower cases to represent segment ends in the zig-zag plot, and upper cases to represent corresponding nodes in the 2-d tree; this distinction will prove crucial further down in the text.
Next comes the first ID event. This corresponds to the inverted duplication of a specific single region (coloured green in Figure 5.5Ai) and involves two positions; the left and right ends of the inverted-duplicated region. These correspond to a left breakpoint n_A and a right breakpoint n_B. Walking through the zig-zag plot in Figure 5.5Aii we find that the first inverted duplication starts at position 1_B, introduces somatic connection linking position 1_B to itself, a new following segment bounded by positions $1_A, 1_B$, and then terminates with a second, new somatic connection from 1_A to 1_B: in total, we encounter position 1_A only once, but position 1_B three times. Consequentially, we use a single label 1_a for the

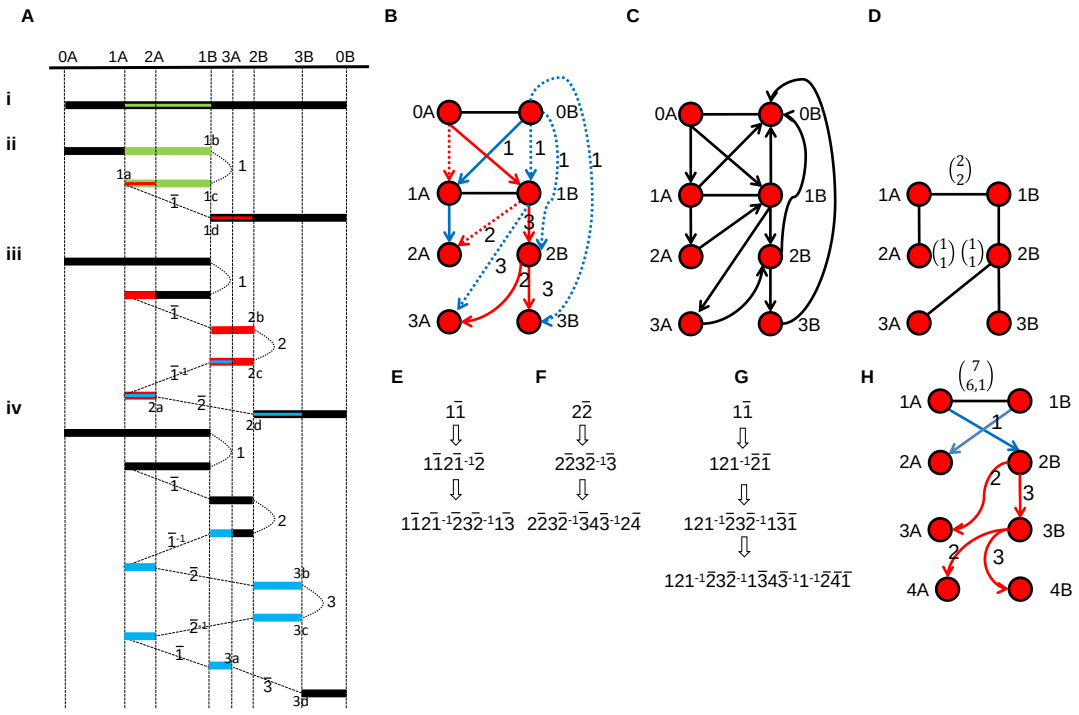

Figure 5.5: Representation of the ID Process. In A) we have *zig-zag* plots for a sequence of three IDs, resulting in four structures i)-iv). The green regions indicate the inverted-duplicated region during each ID. Dashed lines indicate a connection between segments. Coordinates n_A and n_B indicate the end positions of the n^{th} duplicated region. B) Corresponding 2d-tree. Nodes correspond to breakpoints and edges demarcate an ordering. Red and blue colours indicate the start and the end breakpoints. Dashed and plain edges indicate minor and major edges. C) Corresponding Hasse diagram. D) The major graph corresponding to evolution E. F) Increases each symbol of E by 1. G) An induced evolution from F. H) The major graph corresponding to induced evolution E.

unique copy of the left end position 1_A, and three labels $1_b, 1_c, 1_d$ for the three copies of 1_B, assigned according to the order they are found along the zig-zag plot. End 1_b is connected to 1_c in the duplication process, represented by the dashed line in the zig-zag diagram of Figure 5.5Aii. Similarly, 1_a is connected to 1_d, representing the end of the inverted duplication. These are the first couple of *somatic connections*, respectively labelled with numerical symbols 1 and $\overline{1}$. Therefore the first word of Figure 5.5E is $1\overline{1}$, different from the

initial TD word 1. Now segment ends 1_a and 1_b are both bound between the coordinates of 0_a and 0_b. 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 *start* (red) from node 0_A to 1_A and 1_B representing the fact that 0_a is the starting position in the segment where 1_A and 1_B are added. Similarly we have edges of type *end* (blue) from 0_B to 1_A and 1_B, representing the fact that 0_b is the ending position of that segment. The black edge is termed a *fence* after the equivalent class of edges used for TDs; it connects node 1_A to 1_B, representing the restriction $1_a < 1_b$.

Our second ID duplicates the green portion in Figure 5.5Aii, lying on the right side of 1_d, and forming two new breakpoints: 2_A and 2_B. Position 2_A is on segment $[1_d, 0_b]$ of Figure 5.5Aii and so must lie between positions 1_B and 0_B. These implies that $1_B, 0_B$ are its two parental nodes. We must note here that edge $(1_B, 2_A)$ is labelled with number 3: this indicates that 2_A lies on a segment bounded by 1_d on one end. Similarly, we will see that label numbers 1 or 2 are used for edges of type $(n_B, x_{A/B})$ when $x_{A/B}$ lies on a segment bounded by n_b or n_c, respectively.
The blue edge from node 1_B to 2_A indicates 1_d is the start bound of 2_a, since we walk left to right (we say *forward*) from 1_d to 0_b in the zig-zag plot. For the same reasoning, 0_b is the end bound of 2_a, indicated by the red edge from 0_B to 2_A. Note that the notion of *start* and *end* refers to the word structure represented by the zig zag plot, and does not necessarily correspond to leftmost and rightmost along the reference sequence. For instance, position 3_A in Figure 5.5Aiii lies on a segment $[2_b, 1_b]$ starting at position 2_b on the right side of the end position 1_b; in this case we walk right to left (*backward*) through the segment in the zig-zag plot, and we find that node 3_A has a right, start parental node 2_B, while 1_B is a left end parent.
The status of *major* (solid) and *minor* (dashed) is also assigned to each pair of parental edges to a node, in the same way as for TDs.

In the following section we will describe the use of 2d-tree structures to count the number of ID evolutions corresponding to a single connection evolution.

5.3.3 Linear Extensions

Looking at ID evolutions, it is easy to observe that distinct breakpoint orders might be compatible with the same connection evolution. We will see how to count all linear extensions associated with a 2-d tree, equivalent to the total number of ID evolutions. The nature of inverted duplications adds difficulties to the original TD problem. Here are the main differences between the two processes.

- Inverted duplications are characterised by A and B breakpoint types; while each A breakpoint implicates just one segment end, B breakpoints implicate three different ones. As for TDs, there is always a one to one breakpoint-segment end correspondence.

- An ID evolution creates inverted (backward) segments having a start breakpoint lying on the right side of the end breakpoint along the reference sequence; in TDs the start breakpoint(node) is always on the left of the end one, so that the orientation of all segments is preserved with respect to the reference.

All that follows will take into account such differences in order to formulate a theorem for the count of linear extensions from a 2d-tree.
If we look at Figure 5.6A, we notice there is a single order of breakpoints $1_A, 1_B, 2_A, 2_B$ along the reference which is compatible with word evolution $1\overline{1} \rightarrow 1\overline{1}2\overline{2}$. However, looking at Figure 5.6C we find three possible breakpoint orderings (i.e. evolutions) for word evolution $1\overline{1} \rightarrow 2\overline{2}1\overline{1}$. We will now have a closer look at how these different orderings arise.
The first two events result in four breakpoints that divide the original interval $[0_a, 0_b]$ into five regions A, B, C, D and E. For word evolution $1\overline{1} \rightarrow 2\overline{2}1\overline{1}$, we must consider the three choices that use two pairs of breakpoints uniquely such that the second somatic connection precedes the first. If we underline the inverted duplication and use '$|_i$' and '$|_{\bar{i}}$' for the two somatic connections of the i^{th} event, the three possible evolutions are:

$ABC\underline{D}E \to A\underline{B}CD|_1D^{-1}|_{\overline{1}}E \to AB|_2B^{-1}|_{\overline{2}}CD|_1D^{-1}|_{\overline{1}}E$, $AB\underline{CD}E \to A\underline{BC}D|_1D^{-1}C_{\overline{1}}^{-1}E$ $\to ABC|_2C^{-1}B^{-1}|_{\overline{2}}D|_1D^{-1}C^{-1}|_{\overline{1}}E$ and $A\underline{BCD}E \to AB\underline{C}D|_1D^{-1}C^{-1}B^{-1}|_{\overline{1}}E$ $\to ABC|_2C^{-1}|_{\overline{2}}D|_1D^{-1}C^{-1}B^{-1}|_{\overline{1}}E$. Each evolution correspond to a set of possible breakpoint positions that are subject to the restrictions $1_A < 1_B$ from the first event, and $2_A < 2_B < 1_B$ from the second.

We now present a generalization for inverted duplications, allowing for the construction of a Hasse diagram from the 2d-tree corresponding to a word evolution.

Lemma 5.2. *If the direction of the edges from the right parental nodes are reversed in the 2d-tree, and the orientation of fences is preserved, a Hasse diagram with single source node* 0_A *and single sink node* 0_B *is obtained.*

Proof. (*of Lemma 5.2*) When we add any node $x \in \{n_A, n_B\}$ to the 2d-tree, it has two parental nodes u_s and v_e. By construction, the node x represents a breakpoint that is placed on the segment $[u_s, v_e]$, where u_s is the start and v_e the end bound of the segment. In terms of reference position we have the ordering $u_s < x < v_e$ if $[u_s, v_e]$ is a forward segment, and $v_e < x < u_s$ otherwise. The edge directed from the leftmost breakpoint $n_l \in \{u_s, v_e\}$ to x represents the ordering $n_l < x$, while the edge from $n_r = \{u_s, v_e\} \setminus \{n_l\}$ corresponds to the ordering $x < n_r$. We note then that the direction of the edges in $T(E)$ does not always reflect the order of the breakpoints along the reference, and we must select the appropriate direction of the edges in order to represent increasing reference position in the Hasse diagram. More precisely, we keep the direction of edges from the left most parent n_l unchanged, while we select the opposite direction for edges connecting the right most parent n_r to its daughter. If we have a fence, we are adding two positions n_A and n_B to the same segment, and thus we have the additional ordering $n_l < n_r$, where $n_l = n_A, n_r = n_B$ (resp. $n_l = n_B, n_r = n_A$) if the two breakpoints are added on a forward (resp. backward) segment. This is represented in the 2-d tree by the direction from n_A to n_B (resp. n_B to n_A).

Consider now that for every ID evolution, we have necessarily $0_A < 1_A < 1_B < 0_B$ after the first ID event. The two breakpoints of the second ID event, $2_A, 2_B$ are subject each to some restriction of the form $j < 2_i < k$, $i \in \{A, B\}$, where j, k indicate the parental

nodes of 2_i; also, since $0_A, 0_B$ are by definition the first and the last positions along the reference, either $j = 0_A$ or $0_A < j < 2_i$, and either $k = 0_B$ or $2_i < k < 0_B$. Then if the direction of the edges in the 2d tree are changed according to the above described rules, we find that $0_A, 0_B$ are the source and sink nodes of the tree, respectively. For any breakpoint n_i from the n^{th} ID event, let j_J, k_K be the new parental nodes, $J, K \geq 1$. Each parental node is either a telomere (i.e. it has ID number 0), or is positioned in between their two parental nodes. The same reasoning applies to the parents of its parents, and so on until a telomere node is found. We then have restrictions of the form $0_A < j_1 < j_2 ... j_J < n_i$, and $n_i < k_1 < k_2 ... k_K < 0_B$. That is, any new breakpoint position is restricted in between two parental breakpoints, as part of a chain starting with 0_A and ending with 0_B. Such finding is valid regardless to the presence/absence of some fences in the 2d tree. □

rearrangement	TD	ID
inverted segments	no	yes
a-left,b-right correspondence	yes	no
no. letters per event	1	2
no. of copies of a B bp	1	3
no. of copies of an A bp	1	1

Table 5.2: Comparison of TD and ID features.

Definition 5.5. *Let T be a 2-d tree corresponding to an ID evolution E. The corresponding major tree $T(E)$ is obtained by selecting, for each node in T, the edge from the major parent.*

Lemma 5.3. *The restriction of the 2-d tree to the major edges results in two trees rooted to nodes 0_A and 0_B.*

Proof. In the construction of the 2-d tree, every node $x \neq \{0_A, 0_B\}$ has two parental nodes, labelled u_s and v_e, arising from the segment $[u_s, v_e]$ that breakpoint x is formed upon. These two nodes are connected to x by a major and minor parental edge, where

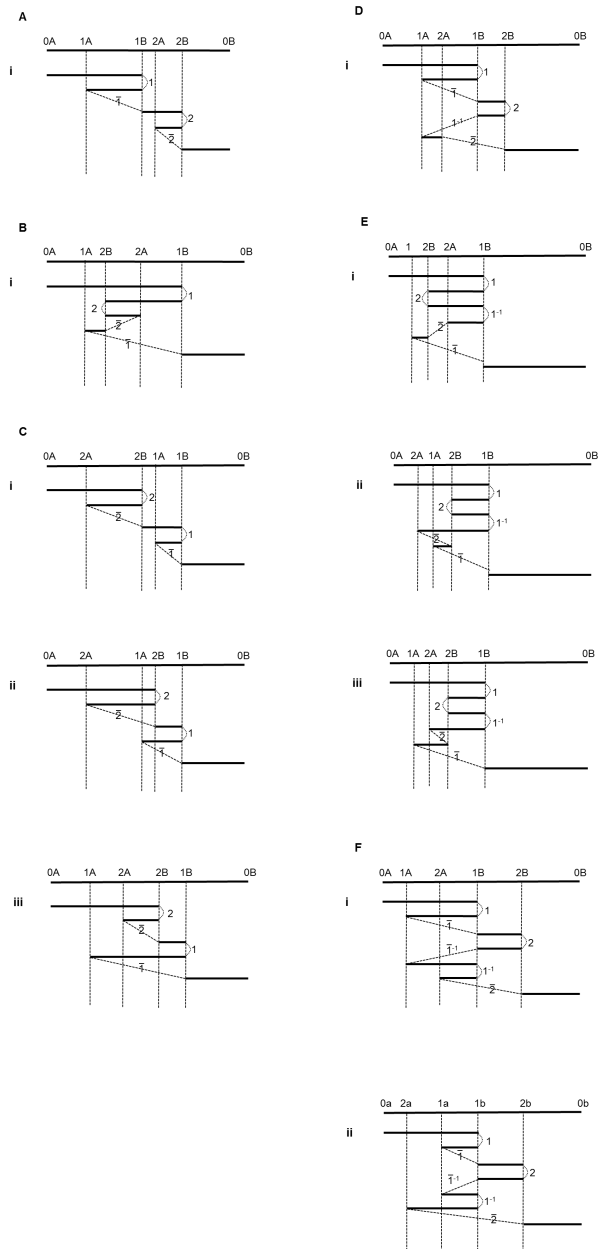

Figure 5.6: Evolutions arising from two IDs. A) Structure associated with word $1\overline{1}2\overline{2}$. B) Structure associated with word $12\overline{21}$. C) Three structures associated with word $2\overline{2}1\overline{1}$. D) Three structures associated with word $1\overline{1}2\overline{1}^{-1}\overline{2}$. E) Structure associated with word $121^{-1}\overline{21}$. F) Two structures associated with word $1\overline{1}2\overline{1}^{-1}1^{-1}\overline{2}$.

$max(u, v)$ and $min(u, v)$ are the major and minor ID numbers, respectively. Thus if we are restricted to the major edges, each node has one parental node, resulting in two trees attached to roots 0_A and 0_B. □

We now formulate a theorem for the count of linear extensions associated with a 2-d tree from an ID evolution. We will first go through all related Lemmas and Corollaries which are needed for the proof, then conclude the section with the main result, Theorem 5.2.

Definition 5.6. *Given a zig-zag plot Z, for any segment end n_i in Z, $i \in \{a, b, c, d\}$, the function*

$$\mu(n_i) : n_i \to n_J \tag{8}$$

with $J \in \{A, B\}$ maps n_i to one and exactly one corresponding node $n_J \in T(E)$, such that

- *$J = A$ if and only if $i = a$*
- *$J = B$ if $i = b, c, d$*

Example 5.5. *Consider the first ID event, common to any evolution. This results in the first zig-zag plot structure of Figure 5.5A, where segments ends $1_a, 1_b, 1_c, 1_d$ have been introduced. Every new end is positioned at one and exactly one of breakpoints $1_A, 1_B$; more precisely, we have 1_a mapping to 1_A and all other ends to 1_B. This is accurately described by the function μ, mapping each segment end of $\{1_a, 1_b, 1_c, 1_d\}$ to exactly one breakpoint in $\{1_A, 1_B\}$.*

Lemma 5.4. *Suppose that* $[u_s, v_e]$ *is any segment arising from an ID evolution* E*. Let* $T(E)$ *be the* 2*-d tree for evolution* E*. Then there exists an edge in* $T(E)$ *connecting nodes* $u_X = \mu(u_s)$ *and* $u_Y = \mu(v_e)$ *together. In addition, precisely one of the following cases holds:*

A) $u = v$*, nodes* u_X *and* v_Y *are of opposite* A/B *type and are connected by a fence (Figure 5.7C); we then have either of the order constraints* $u_X < v_Y$ *and* $v_Y < u_X$*.*

B) $u \neq v$*, nodes* u_X, v_Y *are connected by a single directed major edge (Figure 5.7D) and the positions satisfy either of the order constraints* $u_X < v_Y$ *and* $v_Y < u_X$*.*

C) $u \neq v$ *and nodes* u_X *and* v_Y *are connected by a minor directed edge; then there exist nodes with ID numbers in the order* $u_X \leq n_1 < n_2 < ... < n_I < v_Y$ *that are connected in a chain of major edges if* n_1 *has an ID number different from* $\min(u, v)$ *(Figure 5.7Eii); otherwise in a chain starting with a fence between nodes* n'_A *and* n'_B*, and continuing with a series of major edges (Figure 5.7Ei). These nodes are found in the same order such that:*

i) if $u_X < v_Y$ *along the reference sequence, all internal nodes* n_i *are right nodes and the positions satisfy either of the following single linear extensions:*

i1) $u_X < v_Y < (n_I) < ... < (n_2) < (n_1)$ *with* $n_1 \neq u_A$ *and nodes* u_X *and* v_Y *connected by a chain of major edges;*
i2) $u_X < v_Y < (n_I) < ... < (n_2) < (n_1)$ *with* $u_X = u_B, n_1 = u_A$ *and nodes* u_X *and* v_Y *connected by a chain starting with the fence connecting nodes* u_A, u_B *and continuing with* 1 *or more major edges.*

ii) if $u_X > v_Y$ *along the reference sequence, all internal nodes* n_i *are left nodes and the positions satisfy either of the following single linear extensions:*

ii1) $(n_1) < (n_2) < ... < (n_I) < v_Y < u_X$*, with* $n_1 \neq u_A$ *and nodes* u_X *and* v_Y *connected by a chain of major edges;*
ii2) $(n_1) < (n_2) < ... < (n_I) < v_Y < u_X$*, with* $u_X = u_B, n_1 = u_A$ *and nodes* u_X *and* v_Y *connected by a chain starting with the fence connecting nodes* u_A, u_B *and continuing with* 1 *or more major edges.*

Proof. We prove this by induction. Initially we start with a single segment $[0_a, 0_b]$ and the first ID results in three segments $[0_a, 1_b]$, $[1_c, 1_a]$ and $[1_d, 0_b]$. Now, in segment $[0_a, 1_b]$ we have that $\mu(0_a) = 0_A$ is the major parent of $\mu(1_b) = 1_B$ and the situation satisfies criterion B; in segment $[1_c, 1_a]$ we have that $\mu(1_c) = 1_B$ and $\mu(1_a) = 1_A$ have the same ID number and are connected by a fence (criterion A); in segment $[1_d, 0_b]$ we have that $1_d = u_s$ is the left end of a segment and there is a chain (consisting of a fence and a major edge) $\{(0_A, 0_B), (0_B, 1_B)\}$ corresponding to the linear extension $0_A < 1_B < 0_B$: that is $n_1 < u_X < v_Y$, consistent with criterion Cii2. All situations thus satisfy the conditions of the lemma.
For the induction we next assume that any segment $[u_s, v_e]$ satisfies the conditions of the lemma for all $u, v < n$. For each segment we thus have one of the following situations: 1) a single major edge connecting nodes u_X and v_Y, 2) a minor edge connecting them along with either a chain of major edges or a chain consisting of a fence followed by one or more major edges, or 3) a single fence edge. We then introduce the n^{th} ID duplicating a region with endpoints n_A and n_B. We need to check all resulting segments satisfy the Lemma. We have four cases to check.

Case I: The entire segment $[u_s, v_e]$ is duplicated or unmodified; then the poset graph is unchanged between nodes u_s and v_e and we have nothing to do.

Case II: breakpoint n_A, lies in $[u_c, u_a]$ and the two nodes $u_A = \mu(u_a), u_B = \mu(u_c)$ are connected by a fence. Then we have new segment $[u_a, n_a]$. Then node n_A is connected to u_A by a single major edge and the situation satisfies criterion B of the lemma.

Case III: Breakpoint n_A lies in $[u_s, v_e]$, $u \neq v$. We thus obtain a new segment $[v_e, n_a]$. Node n_A then has major and minor parents with ID number $\max(u, v)$ and $\min(u, v)$.

If $u < v$ we have a new major edge $v_Y \rightarrow n_A$, and segment $[n_a, v_e]$ satisfies criterion B of the Lemma. If $u > v$ we have a single minor edge $v_Y \rightarrow n_A$ and major edge $u_X \rightarrow n_A$ with some order $u_X < n_A < v_Y$ or $v_Y < n_A < u_X$. In the former case, we find that the new segment $[v_e, n_a]$ has a backward orientation, with left end n_a. Then the order $u_X < n_A < v_Y$ corresponds to $n_I < v_Y < u_X$ of case Cii1. In the latter case, we find that the new segment $[v_e, n_a]$ has a forward orientation, with right end n_a. The order $v_Y < n_A < u_X$ corresponds to $u_X < v_Y < n_I$ of case Ci1.
Now, by assumption of the induction hypothesis, we have two cases to consider:

Case IIIa: nodes u_X and v_Y are connected by a major edge. Then the new segment $[v_e, n_a]$ satisfies exactly one of situations B, Ci and Cii of the lemma;

Case IIIb: nodes u_X and v_Y are connected by a minor edge, along with a chain of major edges, or a fence followed by some major edges. As for case IIIa, the new segment $[v_e, n_a]$ satisfies exactly one of situations B, Ci and Cii of the lemma.

Case IV: If a breakpoint n_B is added to $[u_s, v_e]$ and $u \neq v$, we get three new segments: $[u_s, n_b]$, $[n_c, u_s]$ and $[n_d, v_e]$. Then we have 2 cases to consider:

Case IVa: nodes u_s and v_e are connected by a major edge, and the same reasoning as for case IIIa applies.

Case IVb: nodes u_s, v_e are connected by a minor edge, along with a chain of major edges or a fence followed by one or more major edges; then the same reasoning as for case IIIb applies.

Case V: If a breakpoint n_B is added to $[u_c, u_a]$, we get new segments $[u_c, n_b]$, $[n_c, u_c]$ and $[n_d, u_a]$. Node $n_B = \mu(n_b) = \mu(n_c)$ is connected to $u_B = \mu(u_c)$ by a single minor edge; moreover, there is a chain of two edges $\{(u_A, u_B), (u_A, n_B)\}$ starting with a fence which is followed by a major edge, where $u_A = \mu(u_a)$. This chain corresponds to either of the order constraints $u_A < n_B < u_B$ and $u_B < n_B < u_A$, which satisfy criteria Cii2 ($n_I < v_Y < u_X$) and Ci2 ($u_X < v_Y < n_I$) of the lemma, respectively.

Case VI: If both breakpoints n_A and n_B lie in $[u_s, v_e]$, with possibly $u = v$ we obtain segments $[u_s, n_b]$, $[n_c, n_a]$ and $[n_d, v_e]$. These are the same segments as in cases II-V and the same arguments are valid. □

Example 5.6. *Figure 5.7A shows a zig-zag plot compatible with the major graph in B (same as in Figure 5.5H), as well as different types of branches (C-E) which are part of the associated major graph. For example, in Figure 5.7D we have nodes* $1_B, 2_B$ *corresponding to the boundaries of segment* $[1_c, 2_b]$ *in the zig-zag plot (Figure 5.7Aiii). We then find that the two nodes are separated by a chain of edges starting with a fence, which is followed by major edge* $(1_A, 2_B)$*. In Figure 5.7Av we have segment* $[2_d, 4_a]$*, and correspondent nodes* $2_B, 4_A$ *are connected by a chain of major edges, as shown in Figure 5.7Eii.*

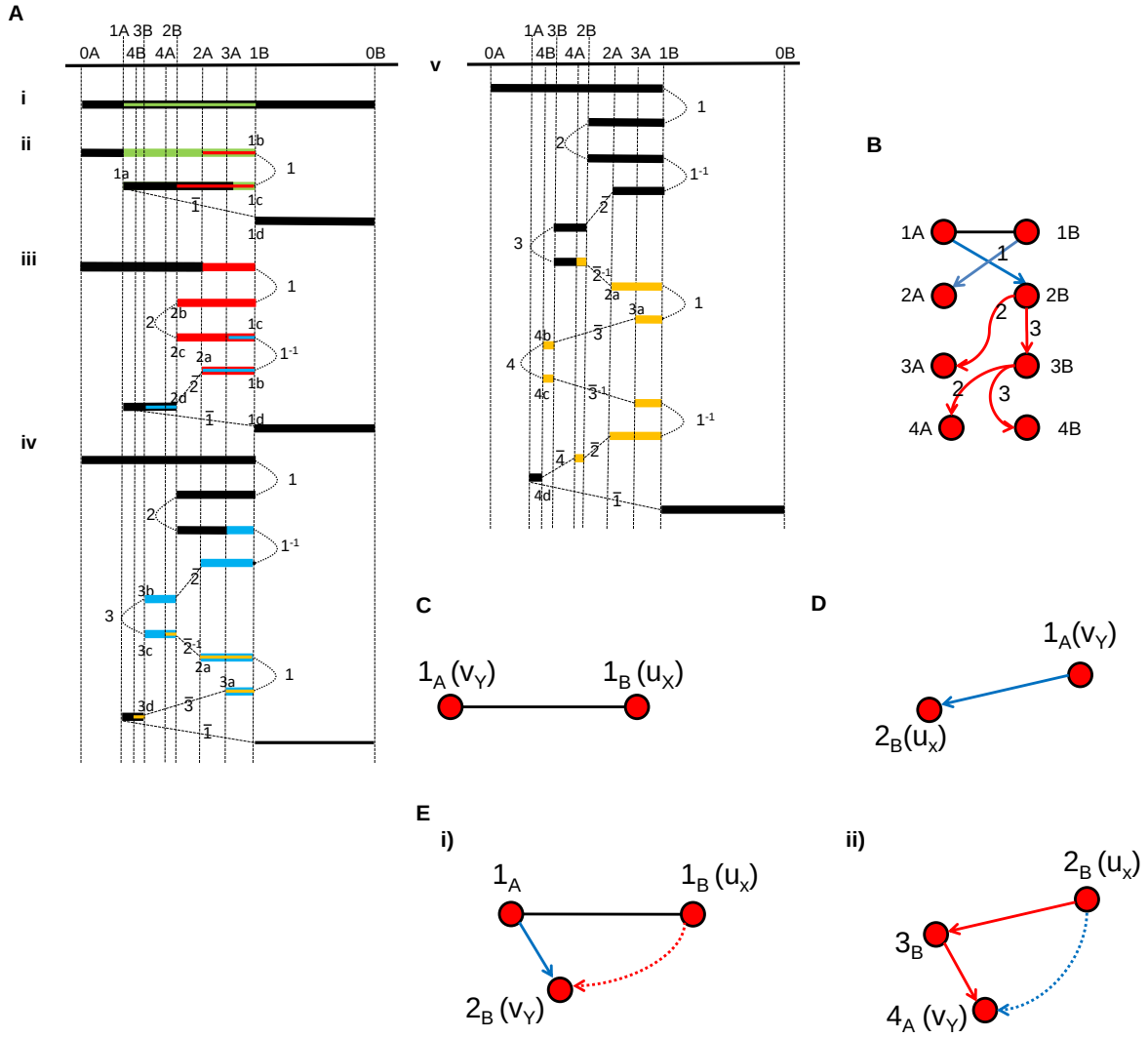

Figure 5.7: Zig zag plot (A) and the nesting structure of different types of branches from the major tree in B. The green regions in Ai-v indicate the inverted-duplicated region during each ID Each branch connects a node u_X (corresponding to the start bound u_s of a segment $[u_s, v_e]$) to a node v_Y (corresponding to the end bound v_e of the same segment). Red and blue colours indicate the start and the end parents, red-blue gradient indicate both colours are possible in the structure; fill and dotted lines connect nodes to its major and minor parents, respectively. B) Single fence edge ($u_X = 1_B, v_Y = 1_A$). C) Single major edge (where any of). Di) A chain starting with a fence and continuing with a series of major edges. The two gradient coloured edges must be of opposite start-end (red-blue) type. Dii) A chain of major edges.

Definition 5.7. *Given an ID evolution E, the corresponding zig-zag plot Z and a chain $n_1 \to n_2 \to ... \to n_N$ of major edges in the major tree $T(E)$, any type A node n_j, $j \leq N$ in C is called a* left node *if and only if n_a is a left end of a segment in Z, and* right node *otherwise. Also, any type B node n_j, $j \leq N$ in C is called a* left node *(resp.* right node*) if and only if $j < N$ and n_j is the left (resp. right) parent of n_{j+1}, or $j = N$ and n_b, n_c are formed as left (resp. right) ends of a segment.*

Lemma 5.5. *Let E be an ID evolution and $T(E), Z$ the corresponding major tree and zig-zag plot. Let also $n_A, n_B \in T(E)$ be two nodes bridged by a fence. Then n_A, n_B are of opposite type left-right.*

Proof. Consider adding two breakpoints n_A, n_B on a segment $[u_s, v_e]$. We get new segments $[u_s, n_b], [n_c, n_a]$ and $[n_d, v_e]$. First assume u_s is a left end; then v_e must be a right end. We then find that n_b, n_c are of the same type right as v_e, while n_a is of the same type left as u_s. The same correspondence is true if we assume u_s is a right end, with left-right swapped. Now we have the following two cases to consider:

i) u_s, n_a are left ends, v_e, n_c are right ends; then by Definition 5.7 $\mu(n_a) = n_A$ is a left node and $\mu(n_c) = n_B$ is a right node.

ii) u_s, n_a are right ends, v_e, n_c are left ends; then by Definition 5.7 $\mu(n_a) = n_A$ is a right node and $\mu(n_c) = n_B$ is a left node.

Thus we conclude that if n_A, n_B are connected by a fence they must be of opposite type left-right. □

Corollary 5.1. *If any node has a major parental left node (resp. right), its minor parent is the most recent common ancestor (in the major graph) of opposite type right (resp. left).*

Proof. Now by Lemma 5.4 any two nodes $\mu(u_s)$ and $\mu(v_e)$ corresponding to the boundaries of a segment $[u_s, v_e]$ are either linked by a major edge, a minor edge or a fence. If a new node x corresponding to a new breakpoint in this interval is formed, u_s and v_e are the major and minor parents, in some order. We will write L, R to indicate the left and right end of segment $[u_s, v_e]$, respectively. We have five cases to check:

Case I: (ID numbers $\mu(L) < \mu(R)$, major edge from $\mu(L)$ to $\mu(R)$). Then x has minor parent $\mu(L)$ and major parent $\mu(R)$. The minor parent $\mu(L)$ is then connected to x by the chain of major edges $\mu(L) \to \mu(R) \to x$. Node x has a major parent of type right and the minor parent $\mu(L)$ is the most recent ancestor of type left in the major graph.

Case II: (ID numbers $\mu(L) > \mu(R)$, major edge from $\mu(R)$ to $\mu(L)$. Analogous to Case I; swap L and R and left/right in argument.

Case III: (ID numbers $\mu(L) < \mu(R)$, minor edge from $\mu(L)$ to $\mu(R)$). Since $\mu(L)$ is the left parent of $\mu(R)$, we must have order constraint $\mu(L) < \mu(R)$; then we find by Lemma 5.4 that minor node $\mu(L)$ is connected to major $\mu(R)$ by a chain of major edges (or a fence followed by major edges) of the form $\mu(L) \to (n_1)_r \to (n_2)_r \to ... \to (n_I)_r \to \mu(R)$ for some internal nodes of type right. Now node x has major parental node $\mu(R)$ so there is also a major edge $\mu(R) \to x$. Together we have the chain of major edges $\mu(L) \to (n_1)_r \to (n_2)_r \to ... \to (n_I)_r \to \mu(R) \to x$. We then find x has a major of type right and the minor $\mu(L)$ is the most recent ancestor of type left in the major graph.

Case IV: (ID numbers $\mu(L) > \mu(R)$, minor edge from $\mu(R)$ to $\mu(L)$). Analogous to Case III; swap L, R and left/right in argument.

Case V: (ID numbers $u = v$, fence connecting $\mu(L)$ with $\mu(R)$). Then the A parent is the major one. We have two cases to consider:

Case Va: (major node $\mu(L)$ of type A). Then there is also a major edge $\mu(L) \to x$ and we have the chain consisting of a fence and a major edge $\mu(R) \to \mu(L) \to x$. We then find x has a major of type left and the minor $\mu(R)$ is the most recent ancestor of type right in the major graph.

Case Vb: (major node $\mu(R)$ of type A). Analogous to case Va, with L, R and left/right swapped in argument. □

We can now explain why minor edges can be removed from the Hasse diagram of an ID evolution. As it happens in TDs, any set of nodes connected by a directed chain of major edges is associated with a single ordering.

Corollary 5.2. *Consider any single directed chain of major edges connecting nodes* $\{l_i, r_j : i = 1, ..., I, j = 1, ..., J\}$ *where any* l_i *is a left parent or the last node (of type left by Definition 5.7) of the chain, and any* r_j *is a right parent or the last node (of type right) of the chain. Suppose furthermore that these nodes are in some order such that* l_i *is an ancestor of* l_{i+1} *for* $i = 1, 2, ..., I - 1$*, and* r_j *is an ancestor of* r_{j+1} *for* $j = 1, 2, ..., J - 1$*. These nodes have a single linear extension of the form:*

$$l_1 < l_2 < .... < l_I < r_J < .... < r_2 < r_1.$$

Proof. Now consider any sub-chain of nodes connected by major edges of the form $l_1 \to r_1 \to r_2 \to ... \to r_n$. Then r_{i+1} has right major parent r_i, so $r_{i+1} < r_i$. Also, r_1 has left major parent l_1 so $l_1 < r_1$. We also know that $r_2, ..., r_n$ all have l_1 as their minor parent by Corollary 5.1, so $l_1 < r_i$ for $i = 2, 3, ..., I$. Together we then have the single order $l_1 < r_n < ... < r_2 < r_1$. If the chain then continues as a chain of type l nodes

$r_n \to l'_1 \to l'_2 \to ... \to l'_m$, we similarly find that $l'_1 < l'_2 < ... < l'_m < r_n$. However, l'_1 has minor parent l_1 by Corollary 5.1 so $l_1 < l'_1$. We then find that these two orders combine into the single order $l_1 < l'_1 < l'_2 < ... < l'_m < r_n < ... < r_2 < r_1$. Thus we find that as we move down a chain of nodes connected by major edges, the l and r nodes lie in one single nested structure where the l nodes are increasing and the r nodes are decreasing in reference position as we move down the major graph; a single linear extension. □

Example 5.7. *Consider the example of Figure 5.8. We have node* 6_B *with left major parent* 5_A*, and we find that its minor parent is the most recent common ancestor of type right,* 2_B*. Similarly,* 2_B *has major right parent* 1_B*. Once more we find, by looking at the whole chain of edges starting with a fence, that its minor parent is the most recent common ancestor of opposite type left,* 1_A*.*

In order to formulate a theorem for the calculations of the number of linear extensions, we need to fully describe some important typical features of inverted duplications. As presented above, when dealing with IDs we have a particular type of B breakpoint, acting as a bound of three different segments at the same time. This has great implications on the structure of 2d-trees, as well as on the calculation of associated linear extensions. On the other hand, breakpoints of type A always bound a single specific segment, similar to what happens with TDs. Following are some crucial definitions about these two types of breakpoints, and the corresponding nodes in the 2d-tree.

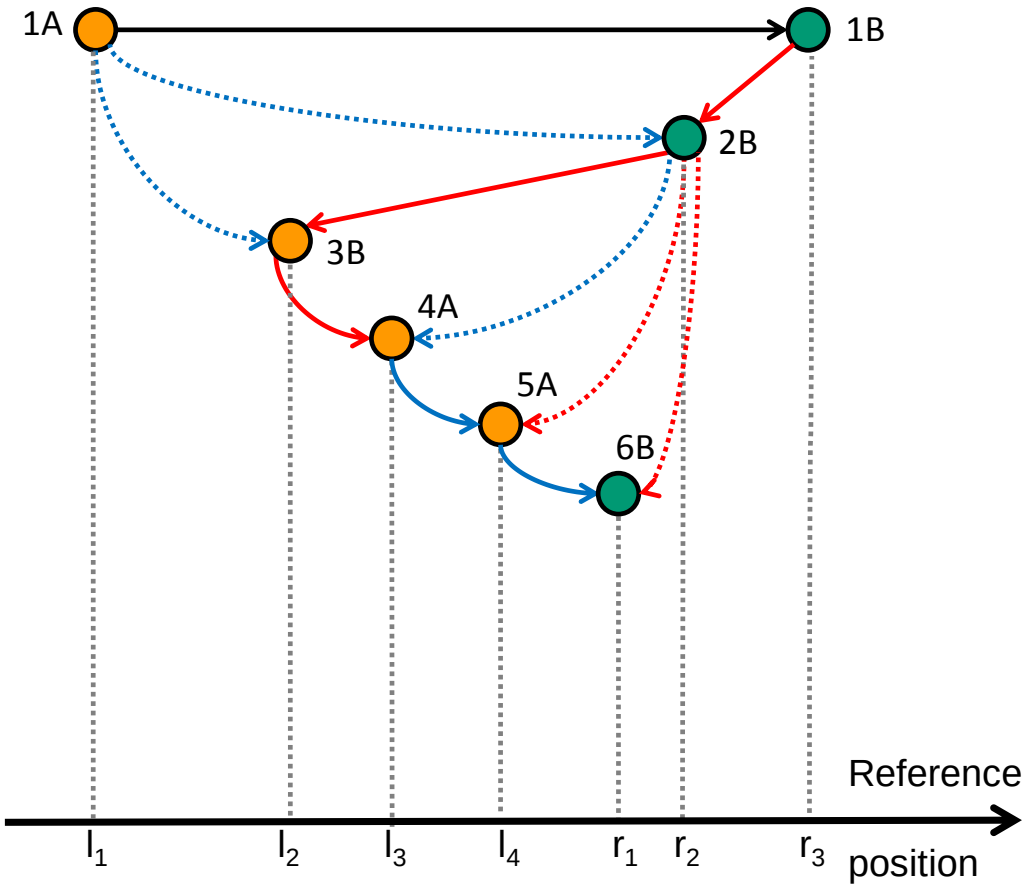

Figure 5.8: The nesting structure of a branch of a major tree. Orange and green colours indicate left and right parental nodes; red and blue colours indicate the start and the end parents; fill and dotted lines connect nodes to its major and minor parents, respectively.

Corollary 5.3. *Let u_A be a type A node in a major tree $T(E)$. Then we have exactly one of the following two cases:*

1. *we have order constraint $u_A < v$ for all descendants v of u, or*

2. we have order constraint $u_A > v$ for all descendants v of u.

Proof. Now, being node u of type A, we have unique corresponding segment end $\mu^{-1}(u_A) = u_a$. Assume we have a chain of $k \geq 1$ major edges down from u_A. Then for any node $x_{A/B}$ in such chain descending from u_A, one the following cases holds:

Case I: x is of type A. Remind that any left/right segment end of type a corresponds to a left/right A node by definition 5.7. Then we have 4 possible situations:
Case Ia: if u_a and x_a are both left ends of a segment, we find from Corollary 5.2 that the order constraint $u_A < x$ holds;
Case Ib: if u_a is a left end and x_a is a right end of a segment, we find from Corollary 5.2 that the order constraint $u_A < x_A$ holds;
Case Ic: if u_a is a right end and x_a is a left end of a segment, we find from Corollary 5.2 that the order constraint $u_A < x_A$ holds;
Case Id: if u_a and x_a are both right ends of a segment, we find from Corollary 5.2 that the order constraint $x_A > u_A$ holds;

Case II: x is of type B. Keeping in mind Definition definition 5.7 we look at the following cases:
Case IIa: if u_a and x_b are both left ends of a segment, then u_A, x_B are left nodes and we find from Corollary 5.2 that the order constraint $u_A < x_B$ holds;
Case IIb: if u_a is a left end and x_b is a right end of a segment, then u_A is a left node, x_B is a right node and we find from Corollary 5.2 that the order constraint $u_A < x_B$ holds;
Case IIc: if u_a is a right end and x_b is a left end of a segment, then u_A is a right node, x_B is a left node and we find from Corollary 5.2 that the order constraint $x_B < u_A$ holds;
Case IId: if u_a and x_b are both right ends of a segment, then u_A and x_B are right nodes and we find from Corollary 5.2 that the order constraint $x_B < u_A$ holds;

The above reasoning leads to the following conclusion: if u_A is a left node then $u_A < x_{A/B}$

for all descendants $x_{A/B}$, which corresponds to situation 1 of the lemma; if u_A if a right node then $u_A > x_{A/B}$ for all descendants $x_{A/B}$, which corresponds to situation 2. All possible situations therefore satisfy one and exactly one of the cases described by the lemma. □

Lemma 5.6. *i)Suppose $N = J + K + L$ branches descend from a single B node n_B in the major graph, such that*

- *J branches start with an edge labelled 1, and the x^{th} branch, $1 \leq x \leq J$, contains j_x descendant nodes*
- *K branches start with an edge labelled 2, and the y^{th} branch, $1 \leq y \leq K$, contains k_y descendant nodes*
- *L branches start with an edge labelled 3, and the z^{th} branch, $1 \leq z \leq L$, contains l_z descendant nodes*

Suppose also that none of the daughter nodes in any branch descending from n_B are connected by a fence. Let $j = \sum_{x=1}^{J} j_x$, $k = \sum_{y=1}^{K} k_y$, $l = \sum_{z=1}^{L} l_z$ and $m = j + k + l$. Then the number of linear extensions involving the associated m breakpoints is $C_B = \binom{j}{j_1, j_2 \ldots j_J} \binom{k}{k_1, k_2 \ldots k_K} \binom{l}{l_1, l_2 \ldots l_L} \binom{j+k}{j,k} \prod_{x=1}^{J} \phi_x \prod_{y=1}^{K} \phi_y \prod_{z=1}^{L} \phi_z$. Here $\binom{j+k}{j,k}$ is the number of ways of intercalating the j nodes from all j_x's branches with the k nodes from all k_y's branches; ϕ_x, ϕ_y, ϕ_z are the number of linear extensions associated with the x^{th} branch with label 1, the y^{th} branch with label 2 and the z^{th} branch with label 3, respectively.

ii) Suppose I branches descend from a single A node n_A in the major graph, such that the r^{th} branch, $1 \leq r \leq I$, contains i_r descendant nodes. Suppose also that none of the daughter nodes in any branch descending from n_A are connected by a fence. Then the number of linear extensions involving the associated m breakpoints is $C_A = \binom{i}{i_1, i_2 \ldots i_I} \prod_{r=1}^{I} \phi_r$, where ϕ_r

is the number of linear extensions associated with the r^{th} branch down node n_A.

iii)Suppose 2 of the branches descending from a single node z in the major graph contain $m1$ and $m2$ descendant nodes, respectively, and the two daughter nodes n_A, n_B of z in these branches are connected by a fence. Let i, j, k, l be the number of nodes descending from n_A, and from node n_B with labels 1, 2 and 3, respectively. We then associate the set of $I + J + K + L$ branches down from A and from B to the product $C_{n_A,n_B} = \binom{i}{i_1,i_2...i_I}\binom{j}{j_1,j_2...l_J}\binom{k}{k_1,k_2...k_K}\binom{l}{l_1,l_2...l_L}\binom{j+k}{j,k}\binom{i+j+k+1}{i+1,j+k}$. Here we multiply the terms associated to the branches down nodes n_A, n_B by the fence factor*: $\binom{i+j+k+1}{i+1,j+k}$. Such factor represents the number of ways of intercalating n_A and its descendants with the nodes in all j_y and k_z branches down node n_B (i.e. the branches with label numbers 1 and 2).*

Proof. i) We have ϕ_h linear extensions associated with any branch h down a node n_B. If we select one linear extension from each branch we have, by Corollary 5.2, $N = J + K + L$ orderings of the form:

$$(x_{i_1}^{(h)})_l < (x_{i_2}^{(h)})_l < ... < (x_{i_{m_h}}^{(h)})_r < (x_{i_{m_h+1}}^{(h)})_r$$

Here $(x_{i_j}^{(h)})_{l/r}$ are the breakpoints represented by the nodes in branch h, and m_h is the number of descendants from n_B in branch h. Now node n_B is the common ancestor of the N branches and so arises from the earliest ID. Then by Corollary 5.2 we find that n_B is restricted in position between all nodes $n_{1,2}$ from all branches of type 1 and 2, and all n_3's from type 3 branches. Indeed, we have $n_1/n_2 < n_B < n_3$ or $n_1/n_2 > n_B > n_3$ for any triplet n_1, n_2, n_3. Now node n_B is fixed in position and common to all N branches. Any pair of nodes within a branch h have one relative order from the linear extension selected from the ϕ_h possibilities of that branch. We then need to count the number of ways of intercalating j_1 nodes from the first branch of type 1, with j_2 nodes from the second branch of type 1, through to j_J nodes from the j^{th} of type 1. We need the same counts for the k_y's

and l_z's branches. There are respectively $\binom{j}{j_1,j_2,...,j_J}$, $\binom{k}{k_1,k_2,...,k_K}$ and $\binom{l}{l_1,l_2,...,l_l}$ ways to do this. Moreover, all j nodes from the branches of type 1 are unrestricted relative to all nodes from type 2 branches, and the number of ways of intercalating the former set of nodes with the latter is precisely $\binom{j+k}{j,k}$. Lastly, we notice that all l nodes from the l_z branches are restricted relative to the nodes from the other branches, thus we simply include the term for the branches of type 3 into the general product.

ii) Applying the same reasoning as for case i, we find that by picking up a single order for each branch h down a node n_A, we get I orderings of the form

$$(x_{i_1}^{(h)})_l < (x_{i_2}^{(h)})_l < ... < (x_{i_{m_h}}^{(h)})_r < (x_{i_{m_h+1}}^{(h)})_r$$

where n_A is either the leftmost or the rightmost node and can thus be ignored, while m_h is the number of descendants from n_A in branch h. We have ϕ_I linear extensions associated with any of such branches. Additionally, we need to count the number of ways of intercalating i_1 nodes from branch 1, with i_2 nodes from branch 2, through to the nodes from the last, I^{th} branch: there are $\binom{i}{i_1,i_2,...,i_I}$ ways to do that.

iii) We now consider the case of a fence between two daughter nodes n_A and n_B of a node z, which results in either of the extra conditions $n_A < n_B$ and $n_A > n_B$. Since n_A, n_B are connected by a fence, they must be of opposite type left-right by Lemma 5.5. Let n_l, n_r be the left and right nodes from $\{n_A, n_B\}$. By Corollary 5.2, if z is of type l the branches down nodes n_A, n_B will take the form:

$$(z)_l < n_l < (x_{i_2}^{(1)})_l < ... < (x_{i_{m_1}}^{(1)})_r$$
$$(z)_l < (x_{i_1}^{(2)})_l < ... < (x_{i_{m_2-1}}^{(2)})_r < n_r$$

Otherwise they will have the form

$$n_l < (x^{(1)}_{i_{m_2}})_l < ...(x^{(1)}_{i_{m_1}})_r < (z)_r$$
$$(x^{(2)}_{i_1})_l < ...(x^{(2)}_{i_{m_2-1}})_r < n_r < (z)_r$$

Here $(x^{(1)}_{i_j})_{l/r}$ and $(x^{(2)}_{i_j})_{l/r}$ are the breakpoints represented by the nodes descending from n_A and n_B, respectively. We need the combinatorial terms $\binom{i}{i_1,i_2...i_I}$ $\binom{j}{j_1,j_2...l_J}$ $\binom{k}{k_1,k_2...k_K}$ and $\binom{l}{l_1,l_2...l_L}$ for all distinct sets of branches down nodes n_A, n_B, as well as the term $\binom{j+k}{j,k}$ accounting for the intercalation between nodes from branches of type j_x and type k_y. Now, the presence of the fence implies that although we have the constraint $n_l < n_r$ (n_A, n_B in some order), nodes down n_A (inclusive) are not restricted with respect to nodes from branches of type j_x and type k_y. Thus we add the fence factor $\binom{i+j+k+1}{i+1,j+k}$, where 1 accounts for node n_A, representing the number of ways of intercalating these 3 sets of branches given the constraint $n_l < n_r$. □

Example 5.8. *Consider Figure 5.5H. We have nodes $1_A, 1_B$ bridged by a fence. We have five daughter nodes of 1_A, and single daughter node 2_A descending from 1_B with label number 1. We then have $i = 5, j = 1, k = 0$, and the count for the fence factor is $\binom{5+1+0+1}{5+1,1+0} = 7$. Consider now node 3_B: we have a branch of type 2 with single node 4_A, a branch of type 3 with single node 4_B, and no type 1 branches. We notice that 4_A and 4_B are restricted to each other, based on the branch types they belong to. Also, each branch is associated with a single linear extension: $\phi_y = \phi_z = 1$ We then find, using the formula of Theorem 5.2 for B nodes, that 3_B is associated with number $\binom{0}{0} \cdot \binom{1}{1} \cdot \binom{1}{1}\binom{1}{0,1} \cdot 1 \cdot 1 = 1$. Noticing that also all other nodes are associated with a count of 1, we conclude that we have 7 linear extensions for the 2-d tree of Figure 5.5H.*

We are now able to formulate a Proof for the following Theorem, allowing for the count of linear extensions of a 2-d tree:

Theorem 5.2. *Let the nodes $0_A, 0_B$ and daughter edges be removed from the 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, the fence contributes a factor $\binom{i+j+k+1}{i+1,j+k}$. We then associate the number $m(n_A) = \binom{i}{i_1,...,i_I}$ if n is of type A, and $m(n_B) = \binom{j}{x_{j,1},...,x_{j,J}} \cdot \binom{k}{x_{k,1},...,x_{k,K}} \cdot \binom{l}{x_{l,1},...,x_{l,L}} \cdot \binom{j+k}{j,k}$. The number of distinct evolutions is then the product of these terms across nodes and fences.

Proof. (Proof of Theorem 5.2) Any ID evolution starts with segment $[0_a, 0_b]$ being divided into 3 new segments $[0_a, 1_b]$, $[1_c, 1_a]$ and $[1_d, 0_b]$. All segments produced later in the evolution will always have at least one parental node with an ID 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 5.6 to any node with a single descending branch containing n nodes results in a combinatorial term of the form $\frac{n!}{n!} = 1$(lemma 5.2). The combinatorial factors from 0_A and 0_B can thus be ignored. For the remaining nodes we see from Lemma 5.6 that the orders ϕ_m associated with nodes in individual branches are multiplied into the combinatorial terms associated with the parental node. For A nodes, the single term $\binom{i}{i_1,i_2...i_I}$ is multiplied into the product of the ϕ_r's across every descending branch. For B nodes, we have a combinatorial term for each of the three sets of branches labelled with numbers $1, 2, 3$, multiplied into the products across the ϕ_j's, ϕ_k's and ϕ_l's; we also account for the possible intercalation among branches of type 1 and 2 by adding a factor $\binom{j+k}{j,k}$. Lastly, if a fence is present we add the factor $\binom{i+j+k+1}{i+1,j+k}$ to the general product. □

5.4 The size of ID space

We have seen how to represent an ID process as an automaton on words, as well as how to obtain the number of ID-Evolutions represented by a specific word evolution from the corresponding major graph. Following the same procedure adopted for TDs, we now look

at the problem of determining the total number of ID-Evolutions.
In Figure 5.6 we see all eleven evolutions that arise from two IDs; one single evolution corresponding to each of the words $1\overline{1}2\overline{2}$ and $12\overline{21}$, three corresponding to $1\overline{1}2\overline{2}$, one evolution corresponding to $1\overline{1}2\overline{1}^{-1}\overline{2}$, three corresponding to $121^{-1}\overline{21}$ and the last two evolutions corresponding to word $1\overline{1}2\overline{1}^{-1}1^{-1}\overline{2}$.
Looking at the complete space of structures arising from the first 3 inverted duplications, we found the following counts: $\mathcal{N}_1 = 1, \mathcal{N}_2 = 11, \mathcal{N}_3 = 627$. This is indeed what we found for Tandem Duplication (Table 4.1), suggesting that the two duplication processes, despite being clearly different, might share the same evolutionary space size. The results for 1 to 3 events have been obtained by manually calculating the sum over all major graphs of the number of liner extensions, as well as computationally.
Next, we discuss how the induction of evolutions works for IDs, and describe some important combinatorial properties we have been able to identify.

5.5 Induced evolutions

For the inducement of ID evolutions we will partly rely on the notation previously used for TDs, as well as on the concepts we have introduced early in the chapter.
We shall first prove that, similar to TDs, 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 5.7. *Let $D(E)$ be the process where we remove all copies of ID symbols 1 and $\overline{1}$ from word evolution E and reduce each symbol 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 ID 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 evolution E starts with trivial word $1\overline{1}$. The next ID in E results in 6 possible word evolutions: $[1\overline{1} \to 1\overline{1}2\overline{2}]$, $[1 \to 12\overline{21}]$, $[1 \to 2\overline{2}1\overline{1}]$, $[1 \to 1\overline{1}2\overline{1}^{-1}\overline{2}]$, $[1 \to 121^{-1}\overline{21}]$ or $[1 \to 1\overline{1}2\overline{1}^{-1}1^{-1}\overline{2}]$. For all these six choices, removing symbols $1, \overline{1}$ from the evolution leaves us with $2\overline{2}$, which become $1\overline{1}$ 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 ID word automaton as a mapping of the form $AXB \to AX(n+1)X^{-1}(\overline{n+1})B$, for possibly empty subwords A, X or B, for the $(n+1)^{th}$ ID. If we remove all copies of the symbols 1 and $\overline{1}$ 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 $A^*X^*B^* \to A^*X^*nX^{*-1}\overline{n}B^*$ which is a valid step in the n^{th} iteration of the ID word automaton, as required.

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

We now highlight a possible approach with an example. First we need some notation for the branches descending from the two roots $1_A, 1_B$ in any major graph $T_{maj}(E')$. We call any descendant from a B node a type $1, 2$ or 3 node after the label number of the edges in the branch it belongs to. We also let i, j, k be respectively: the number of descendants from node 1_A plus one; the number of descendants from node 1_B of type 1 and 2, plus one; the number of descendants from node 1_B of type 3, plus one. We will also use round normal brackets to represent triplets of counts (i, j, k), and square brackets to indicate the number of linear extensions associated with a particular graph.

A motivating example

Consider the original evolution

$1\overline{1} \to 1\overline{1}2\overline{1}^{-1}\overline{2}$

which becomes
$2\overline{2} \rightarrow 2\overline{2}3\overline{2}^{-1}\overline{3}$
after increasing ID numbers by 1. The original 2-d tree is shown in Figure 5.10i, and is associated with a number of orders $n(T) = 1$. For this example, we have a range of 15 possible induced evolutions, shown in Figure 5.9. The correspondent major trees are shown in Figure 5.10. If we look at the patterns of (i, j, k) counts across this set of major trees, we note the following:

- If we sum over induced trees with identical (i, j, k) triplets we get combinatorial term $\binom{i+j-1}{i,j-1} n(\tau)$
- The sum $i + j + k = N = 2n + 1 = 7$ has a fixed total
- Summing all entries for a fixed k value results in term $(2^{N-k-1} - 1) \cdot n(\tau)$

Moreover, we notice that the total number of induced evolutions is $1+3+7+15+31 = 57$, and $n(\tau) \cdot 2^{2n} - (2n-1) = 1 \cdot 2^6 - (6-1) = 57$, in agreement with Theorem 4.3 of Tandem Duplication. We now try to relate the counts across k values with the recursion of Theorem 4.3. Now, the sum of the first $2n$ elements of a geometric progression [41] with common ratio 2 is given by:

$$\sum_{m=0}^{2n-1} 2^m = 2^{2n} - 1 \tag{9}$$

We have a range of possible values $k = \{1, 2, ...2n - 1\}$ ($2n = 6$ in the example of Figure 5.10). Substituting these values into the count $(2^{N-k-1} - 1) \cdot n(\tau)$ presented above, we find

that the exponent $m = N - k - 1$ has possible values $\{N - (2n-1) - 1, N - (2n-2) - 1, ...N - 2\}$. Then replacing N with $2n+1$ we find $m \in \{1, 2, ...2n-1\}$. We can then write the sum of counts across m values as follows:

$$\sum_{m=0}^{2n-1}(2^m - 1) = 2^{2n} - 1 - 2n + 1 = \sum_{m=1}^{2n-1}(2^m - 1) = \\ 2^{2n} - 2 - 2n + 1 = 4^n - (2n+1) \tag{10}$$

Such term must be multiplied by the number $n(\tau)$ of linear extensions in the original major tree, thus obtaining exactly the same formula as in Theorem 4.3.

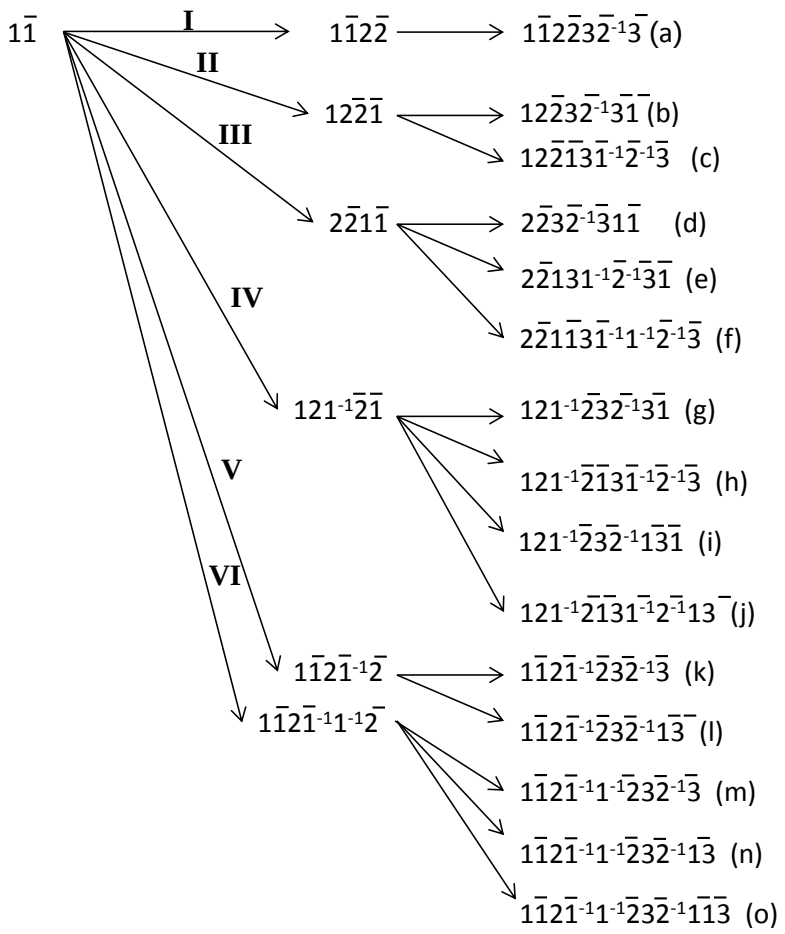

Figure 5.9: Full set of evolutions induced from $1\bar{1} \rightarrow 1\bar{1}2\bar{1}^{-1}\bar{2}$. Choices for the second event are labelled with Roman numbers.

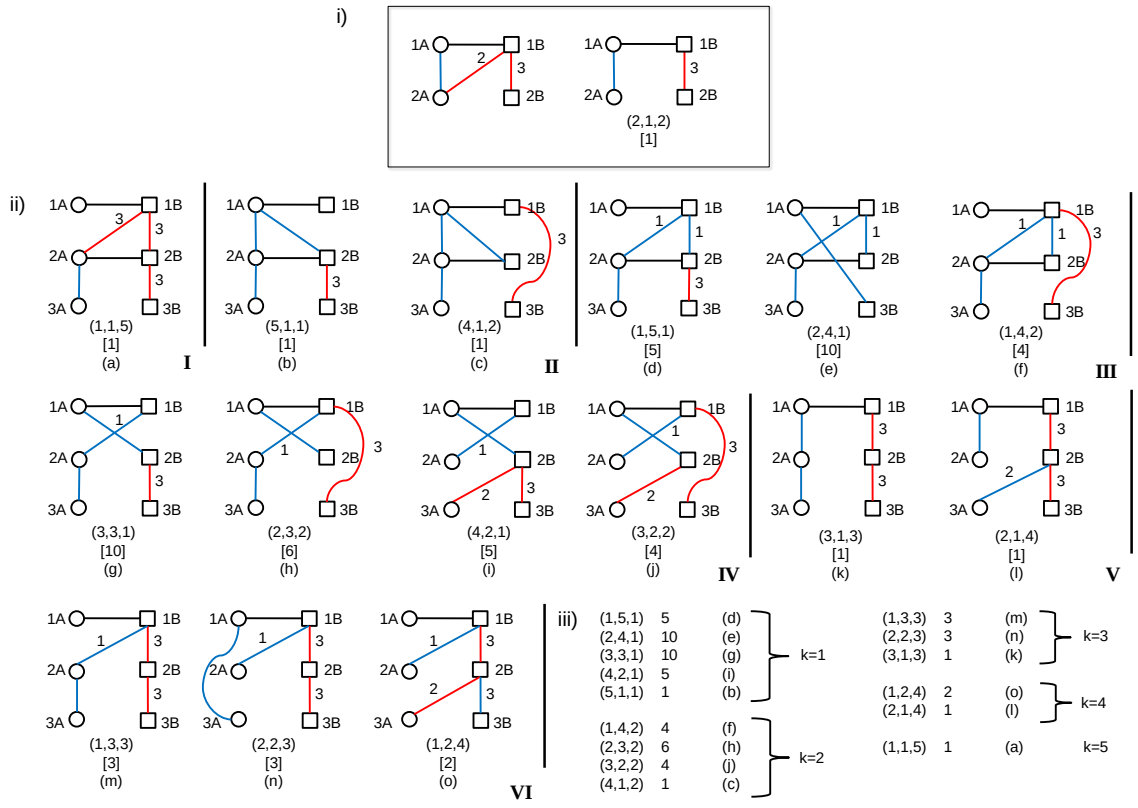

Figure 5.10: ii) Full set of major trees for the evolutions induced from $1\overline{1} \rightarrow 1\overline{1}2\overline{1}^{-1}\overline{2}$ (major graph shown in i). Roman numbers label subsets of evolutions corresponding to the same connection evolution, in a similar way to Figure 5.9. Red and blue edges connect each node respectively to its start and to its end parent (when present). At the bottom of each graph, the corresponding (i, j, k) triplet is shown, as well as the number of orders in square brackets. iii) Counts across (i, j, k) triplets are shown, grouped according to the k value.

5.6 Conclusions

The description given above must be the base for a generalisation of the β-trees used for tandem duplications [89]. Although we did not manage to generalise the TD results fully for IDs, we calculated the total size of the ID space up to the third event, using both a manual and a computational construction of evolutions. This was combined with additional trials for $n > 3$ step evolutions, giving support to the assumption of a number of evolutions identical to TDs. However, the differences in the two processes are evident when we look at the space of word evolutions, increasing at a higher rate in inverted duplications (see Table 5.1); if the evolutionary spaces have indeed the same size, this would imply a 'compensation' between the number of word evolutions and the number of linear extensions associated with each of them. What is still lacking is a clear definition of β-subtrees [89] for inverted duplications. Such step forward would allow for an attempt to prove the recursion on the total number of evolutions for inverted duplication, most probably using the same induction approach applied to TDs in [89].

6 Discussion

6.1 Biological implications of the project

This PhD project has combined the applied bioinformatics works of Chapters 2 and 3 with more mathematical and theoretical studies, presented in Chapters 4 and 5. The Eulerian path approach and the rearrangement evolution algorithm address the challenge of gaining a deeper understanding of cancer genome architecture, with the additional power of visualising a set of likely intermediate steps, separating the reference from the rearranged genome sequence. Such methods thus have the potential of elucidating the path taken by cancer cells towards the final karyotype, a series of transformations associated with increased tumour aggression (proliferative advantage and high survival rates). The comparison of early-mild versus late-aggressive cancer stages can potentially highlight the link between rearrangements and tumour aggression. An extremely interesting aspect is represented by the comparison of structural variation and genome evolution between different patients. One could possibly identify common rearrangements, acting as signatures of the pathology (similar to the Philadelphia translocation [67]). Analyses at multiple stages of cancer progression could even identify different early-stage karyotypes leading to an identical, late-stage genome. These cases are predicted by our model of rearrangement evolution, and potentially have serious implications in cancer prognosis. Analysis of the sequences flanking the breakpoints inferred from discordant paired end reads could identify signatures of extensive or short sequence homologies. The presence of microhomologies at junctions associated with copy number changes would support replication based mechanisms like MMBIR. The comparison of these observations with generated valid evolutions would be of great interest, giving additional indications towards some particular solutions proposed by the algorithm. Moreover, such analyses could be used in order to infer the relative contribution of different rearrangement types (including NAHR,MMBIR and NHEJ) to the aberrant cancer genome architectures.

Along with cancer genomics, the algorithmic approach of Chapter 2 has a clear appli-

cability to any pathological condition associated with structural variants. For instance, the Potocki-Lupsk syndrome is known to be caused by recurrent duplications, and replication based mechanisms of DNA repair are a possible explanation of these structural changes [118]. An other exciting application could be represented by the study of species evolution. Many organisms, especially Eukaryotes, have undergone extensive structural changes during their evolution, which can sometimes provide the necessary background for their adaptation to the environment. One could look at the extent of structural and copy number variation along different evolutionary time scales, performing pairwise species comparisons. The availability of genome assemblies for many Eukaryotic organisms would represent an important, additional information for the validation of constructed evolutions, and would imply the reformulation of Problem 3.1 into the challenge of reconstructing a known, target genome sequence (similar to the classical rearrangement distance problem [114, 47, 88, 46, 49, 50, 44]). The combinatoric studies of gene duplication provide a picture of the complete space of possibilities arising from a single rearrangement type. This picture also includes a set of possible copy number profiles. Comparing them to karyotypes from cancer patients would give insights on the possible evolutions leading to that genomic sequence, in a similar way of what could be done using the approach of chapter 2. The tightest link between our theoretical study and any applied work is represented by temporal words. Indeed, when studying a rearranged genomic sequence using paired end data, the chronological order in which somatic connections were introduced is unknown, differently from the case of connection evolutions used to infer the TD space. Thus, the study of temporal words we carried on has potential applications for the inference of chronological orders of TD events, provided the complete rearranged sequence is available.

6.2 General conclusions

In this PhD project we have touched on different research areas in combinatorics, linear algebra, informatics and genomics, keeping all the subject together thanks to the com-

mon link with cancer large-scale rearrangements. Our first work was the Eulerian cycles approach, generating cancer karyotypes without giving any insights on the intermediate evolutionary steps leading to the final sequence. An equivalent approach has been recently published [83], although showing slight differences in the graph representation: specifically, the adjacency graph (as the authors call the equivalent of our bidirected graphs) is an undirected graph where every node correspond to a segment end, similar to the breakpoint graphs described in [8, 47]; in our directed graphs, every node rather represents an entire DNA segment with a specific orientation, *forward* or *reverse*. The study presented in [83] explored some additional aspects of the general genome reconstruction problem: in the case where the copy number vector is unknown, the authors proposed a method of maximum likelihood assignment of edge multiplicities in the graph, based on observed read depth data. The possibility of telomere loss is also discussed, and a method is proposed to detect these events based on drops in read depth signal. Conversely, our work explored the problem of constructing directed graphs from the original bidirected graph and copy number vector in more detail, introducing a Gröbner basis approach for the inference of edge multiplicity.

We are aware of a series of limitations for our Eulerian cycle approach; as a consequence of exploring a space full of potential solutions, the construction of directed graphs soon becomes inefficient as the numbers of nodes and edges increase; an additional limiting factor relates to the computational effort in constructing the arborescences for each graph. As for the set of consistent Eulerian cycles obtained from directed graphs, the approach does not allow for the distinction of different order of events, or the nature of the underlying rearrangements.

In this research context, an unresolved matter is the calculation of a probability for each chronological order of somatic connections. In the case of copy-neutral rearrangements, parsimony models have been developed [46] and applied to cancer genome analysis [93]. Order information have also been inferred by looking at cluster of rearrangements with duplications [92] or by combining duplications and single-nucleotide mutations data [44, 30].

Further limitations concern the available input data. Mapping of discordant paired reads is typically difficult for structural variants in repetitive regions of the human genome, possibly causing missing or incorrect somatic connections in the data. These regions also represent a challenge for the estimation of read depth. Moreover, our method assumes the presence of a single genome in the cancer sample, while a cancer cell population will typically consist of an heterogeneous mix of genetically different sub-populations. Lastly, our method gives little insight on the molecular mechanisms underlying the complex structural and copy number changes.

The algorithmic reconstruction approach was designed to further improve our possibilities of highlighting the molecular basis of rearrangements, gaining insights about genomic structures at intermediate stages of an evolution, as well as solving some intrinsic limitations of the Eulerian cycle approach. Higher complexity graphs can be considered, and the effect of specific rearrangements operations and orders evaluated. Despite these advantages, the high number of valid solutions can sometimes represents a limitation. Typically, we observe that the available input information identifies a set of optimal solutions, rather than a unique explanation. This might well be the direct consequence of the limited input information used for our inferences. However, it is also a representation of the wide space these rearrangements operate on: a result of both the combinatorial explosion of possible chronological orders, and the high number of choices in the introduction of every single operation. Any attempt to limit the number of solutions should not reduce the range of operations included in the algorithm, crucial for a complete description of biologically meaningful mechanisms. The next step would rather be to focus on additional input information for the algorithm. Recent studies, for example, often provide information about the relative contribution, for each genomic region, of the maternal and paternal copies of the chromosomes. This information is typically derived from SNP (Single Nucleotide Polymorphism) array, and is widely used for the study of copy number variation, especially in cancer. Extending the current approach by including this type of input data would probably help limiting the number of optimal solutions. However, the identification of unique solutions without any associated information loss is likely to represent am utopia.

A wide range of algorithms for sorting a rearranged genome into a reference have been developed, including the following major contributions: [114, 47, 46, 49, 50, 40, 8, 7]. However, we addressed the problem of constructing an evolution *forward* in time with the crucial constraint of avoiding breakpoint reuse; moreover, our method includes an original modelling of microhomology based repair mechanisms leading to gene amplification, in an attempt to achieve a refined, more complete description of rearrangements associated with copy number variation.

Chapters 4 fully describes the structures arising from a TD process, providing a method to count the number of evolutions associated with a specific connection evolution. These findings lead to the main result that we have published in [89], where a formula for the complete space of TD evolutions is presented. Thus, we have observed the over-exponential growth of the number of different TD evolutions, which makes a full exploration of the space beyond 4 TD events computationally very demanding. Such space is a specific subset of evolutions for the reconstruction algorithm described in chapter 3, as TD can arise by an intrachromosomal BIR operation. We have also shown that the number of copy number vectors grows at a slower rate than the TD space, meaning that each of these vectors may correspond to a variety of evolutions. Such situation parallels the observations made for our algorithmic reconstruction approach (Chapter 3), indicating that our input information does not contain enough information to identify unique solutions.

A parallel challenge has been represented by the description of the geometric properties of TD connection words. Our findings has allowed for the development of a graph representation (temporal posets) of chronological order constraints associated with a word, useful when no *a priori* knowledge on the order of events is available. Such information can be used for a reliable inference of the number of TD evolutions associated with a specific word structure. However, as a result of the complex structure of these graphs, we were unable to derive a formula for an easy calculation of the number of solutions, as for the breakpoint ordering problems. A precise definition of the poset graph properties (which take the easier form of a 2-d tree for breakpoint-ordering problem) is still lacking. Moreover, we found

that the complexity of TD word structures sometimes makes it hard to identify all order relations associated with a specific word. Precisely, simple patterns of duplication can be highly modified by newly added symbols, and a complete generalisation of the resulting structures is still missing from this study.

Despite the similarities with TDs, the description of the ID space in chapter 5 has proven much more challenging. The presence of two different types of connections, as well as three different types of edges down from B nodes in a 2-d tree, makes the model more difficult to tract. Moreover, the presence of inverted segments implies that the correspondence a node-left segment ends, and b nodes-right segment ends is lost when dealing with IDs. Nevertheless, we have introduced a modified algebraic representation along with new types of 2-d trees and zig-zag plots, allowing for a complete description of the process. Similar to TDs, we have observed that the number of copy number profiles increases very slowly compared to the number of ID evolutions, leading to equivalent problems in the analysis of real data. As a consequence of the introduction of two new connections per event, different from the single connection of TDs, the number of words increases faster, as well as the maximum word size for k events.

Although manual and computational calculations strongly indicated that the number of solutions is identical to TDs, we were not able to formally prove this observation. Additional work is needed to elucidate the exact rules for inducing evolutions, with an approach parallel to what was done for TDs [89].

Inverted duplications do not correspond to any single connection operation described in chapter 3. They can rather result from a series of 2 operations, corresponding to a couple of connections $x, \overline{x}$ in the ID algebraic formalism.

The methods here used for TDs and IDs, together with those developed for the description of Breakage Fusion Bridge cycles [43] suggest there may be a more general space in which all different types of rearrangements operate and these methods apply; a generalization of these methods to the combined space of these rearrangement processes could lead to a deeper understanding of cancer genome evolution.

Highlighting the mechanisms responsible for cancer development, as well as confidently inferring the aberrant cancer genomic organisation, will require a strong effort from the scientific community; however, technological advances have dramatically changed the way this pathology is studied, shading light on its genomic features and associated mutation mechanisms; this description has reached a level of detail which would have been hard to imagine just a few decades ago. Knowledge from biology, mathematics, informatics, chemistry, sequencing technology and many other fields can now, as never before, create a powerful combination of expertises, whose potential has just started to be understood. This charming interdisciplinary research, tightly related with team working and international collaborations, is likely to be the key of our major steps towards better cancer treatments. And that is, indeed, what research is always about: gaining some knowledge, and seeking the next step forward.

References

[1] T. Aardenne-Ehrenfest and N.G. Bruijn. Circuits and trees in oriented linear graphs. *Simon Stevin: Wis-en Natuurkundig Tijdschrift*, 28:203, 1951.

[2] William W Adams and Philippe Loustaunau. *An introduction to Gröbner bases*, volume 3. American Mathematical Society Providence, 1994.

[3] Anjali S Advani and Ann Marie Pendergast. Bcr–abl variants: biological and clinical aspects. *Leukemia research*, 26(8):713–720, 2002.

[4] Andrés Aguilera and Belén Gómez-González. Genome instability: a mechanistic view of its causes and consequences. *Nature Reviews Genetics*, 9(3):204–217, 2008.

[5] David P. Clark and. *Molecular Biology*. Academic Cell, 2009.

[6] O.T. Avery, C.M. MacLeod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type iii. *The Journal of experimental medicine*, 79(2):137–158, 1944.

[7] Martin Bader and Enno Ohlebusch. Sorting by weighted reversals, transpositions, and inverted transpositions. In *Research in Computational Molecular Biology*, pages 563–577. Springer, 2006.

[8] Vineet Bafna and Pavel A Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[9] Gary Benson. Sequence alignment with tandem duplication. *Journal of Computational Biology*, 4(3):351–367, 1997.

[10] Gary Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic acids research*, 27(2):573, 1999.

[11] Gary Benson. Tandem cyclic alignment. In *Combinatorial Pattern Matching*, pages 118–130. Springer, 2001.

[12] Gary Benson and Lan Dong. Reconstructing the duplication history of a tandem repeat. In *ISMB*, pages 44–53, 1999.

[13] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[14] Denis Bertrand and Olivier Gascuel. Topological rearrangements and local search method for tandem duplication trees. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 2(1):15–28, 2005.

[15] Denis Bertrand, Mathieu Lajoie, and Nadia El-Mabrouk. Inferring ancestral gene orders for a family of tandemly arrayed genes. *Journal of Computational Biology*, 15(8):1063–1077, 2008.

[16] Denis Bertrand, Mathieu Lajoie, and Nadia El-Mabrouk. Inferring ancestral gene orders for a family of tandemly arrayed genes. *Journal of Computational Biology*, 15(8):1063–1077, 2008.

[17] Graham R Bignell, Thomas Santarius, Jessica CM Pole, Adam P Butler, Janet Perry, Erin Pleasance, Chris Greenman, Andrew Menzies, Sheila Taylor, Sarah Edkins, et al. Architectures of somatic genomic rearrangement in human cancer amplicons at sequence-level resolution. *Genome research*, 17(9):1296–1303, 2007.

[18] Mathilde Bouvel and Elisa Pergola. Posets and permutations in the duplication–loss model: Minimal permutations with d descents. *Theoretical Computer Science*, 411(26):2487–2501, 2010.

[19] Mathilde Bouvel and Dominique Rossin. A variant of the tandem duplicationrandom loss model of genome rearrangement. *Theoretical Computer Science*, 410(8):847–858, 2009.

[20] T. Boveri. *Zur frage der entstehung maligner tumoren*. Gustav Fischer, 1914.

[21] Graham Brightwell and Peter Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.

[22] F Buekenhout and M Parker. The number of nets of the regular convex polytopes in dimension 4. *Discrete mathematics*, 186(1):69–94, 1998.

[23] P.J. Campbell, P.J. Stephens, E.D. Pleasance, S. O'Meara, H. Li, T. Santarius, L.A. Stebbings, C. Leroy, S. Edkins, C. Hardy, et al. Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. *Nature genetics*, 40(6):722–729, 2008.

[24] J Marshall MR Stratton PJ Campbell CD Greenman, SL Cooke. Modelling breakage-fusion-bridge cycles as a stochastic paper folding process. *arXiv*, page 1211.2356,

2013.

[25] Pamela C Champe, Richard A Harvey, and Denise R Ferrier. *Biochemistry*. Lippincott Williams & Wilkins, 2005.

[26] J.K. Conner and D.L. Hartl. *A Primer of Ecological Genetics*. Sinauer Associates, 2004.

[27] John K Cowell. *Molecular genetics of cancer*. Gulf Professional Publishing, 2001.

[28] Elizabeth R Cregan. *All About Mitosis and Meiosis*. Capstone, 2010.

[29] UC San Diego. Covaris DNA shearing. `http://htg.ucsd.edu/services/covaris`, 2013.

[30] Steffen Durinck, Christine Ho, Nicholas J Wang, Wilson Liao, Lakshmi R Jakkula, Eric A Collisson, Jennifer Pons, Sai-Wing Chan, Ernest T Lam, Catherine Chu, et al. Temporal dissection of tumorigenesis in primary cancers. *Cancer discovery*, 1(2):137–143, 2011.

[31] Nature Education. Glossary. `http://www.nature.com/scitable/glossary`, April 2013.

[32] Dan TA Eisenberg. An evolutionary review of human telomere biology: the thrifty telomere hypothesis and notes on potential adaptive paternal effects. *American Journal of Human Biology*, 23(2):149–167, 2011.

[33] Greg Elgar, Tanya Vavouri, et al. Tuning in to the signals: noncoding sequence conservation in vertebrate genomes. *Trends in genetics: TIG*, 24(7):344, 2008.

[34] National Center for Biotechnology Information(NCBI). Human genome assembly data. `http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human/data`, 2015.

[35] Simon A Forbes, David Beare, Prasad Gunasekaran, Kenric Leung, Nidhi Bindal, Harry Boutselakis, Minjie Ding, Sally Bamford, Charlotte Cole, Sari Ward, et al. Cosmic: exploring the world's knowledge of somatic mutations in human cancer. *Nucleic acids research*, 43(D1):D805–D811, 2015.

[36] M.J. Fullwood, C.L. Wei, E.T. Liu, and Y. Ruan. Next-generation DNA sequencing of paired-end tags (pet) for transcriptome and genome analyses. *Genome research*, 19(4):521–532, 2009.

[37] D. Futuyma. *Evolution*. Sinauer Associates, 2009.

[38] Jean Gallier. *Discrete Mathematics*. 2011. Springer.

[39] Olivier Gascuel, Michael D Hendy, Alain Jean-Marie, and Robert McLachlan. The combinatorics of tandem duplication trees. *Systematic Biology*, 52(1):110–118, 2003.

[40] Simon Gog, Martin Bader, and Enno Ohlebusch. Genesis: genome evolution scenarios. *Bioinformatics*, 24(5):711–712, 2008.

[41] Eric Gosset. *Discrete mathematics with proof*. 2009. Wiley-Blackwell.

[42] C.D. Greenman, G. Bignell, A. Butler, S. Edkins, J. Hinton, D. Beare, S. Swamy, T. Santarius, L. Chen, S. Widaa, et al. Picnic: an algorithm to predict absolute allelic copy number variation with microarray cancer data. *Biostatistics*, 11(1):164–175, 2010.

[43] CD Greenman, SL Cooke, J Marshall, MR Stratton, and PJ Campbell. Modeling the evolution space of breakage fusion bridge cycles with a stochastic folding process. *Journal of mathematical biology*, pages 1–40, 2015.

[44] Chris D Greenman, Erin D Pleasance, Scott Newman, Fengtang Yang, Beiyuan Fu, Serena Nik-Zainal, David Jones, King Wai Lau, Nigel Carter, Paul AW Edwards, et al. Estimation of rearrangement phylogeny for cancer genomes. *Genome research*, 22(2):346–361, 2012.

[45] Robert C Griffiths and Paul Marjoram. Ancestral inference from samples of dna sequences with recombination. *Journal of Computational Biology*, 3(4):479–502, 1996.

[46] Sridhar Hannenhalli and Pavel A Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 581–592. IEEE, 1995.

[47] Sridhar Hannenhalli and Pavel A Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM (JACM)*, 46(1):1–27, 1999.

[48] D. Hansemann. Ueber asymmetrische zelltheilung in epithelkrebsen und deren biologische bedeutung. *Virchows Archiv*, 119(2):299–326, 1890.

[49] Tzvika Hartman and Ron Shamir. A simpler and faster 1.5-approximation algorithm for sorting by transpositions. *Information and Computation*, 204(2):275–290, 2006.

[50] Tzvika Hartman and Roded Sharan. A 1.5-approximation algorithm for sorting by transpositions and transreversals. *Journal of Computer and System Sciences*, 70(3):300–320, 2005.

[51] PJ Hastings, Grzegorz Ira, and James R Lupski. A microhomology-mediated break-induced replication model for the origin of human copy number variation. *PLoS genetics*, 5(1):e1000327, 2009.

[52] PJ Hastings, James R Lupski, Susan M Rosenberg, and Grzegorz Ira. Mechanisms of change in gene copy number. *Nature Reviews Genetics*, 10(8):551–564, 2009.

[53] Karen E Hermetz, Scott Newman, Karen N Conneely, Christa L Martin, Blake C Ballif, Lisa G Shaffer, Jannine D Cody, and M Katharine Rudd. Large inverted duplications in the human genome form via a fold-back mechanism. *PLoS genetics*, 10(1), 2014.

[54] J. Hicks, A. Krasnitz, B. Lakshmi, N.E. Navin, M. Riggs, E. Leibu, D. Esposito, J. Alexander, J. Troge, V. Grubor, et al. Novel patterns of genome rearrangement and their association with survival in breast cancer. *Genome research*, 16(12):1465–1479, 2006.

[55] Andrew J Holland and Don W Cleveland. Chromoanagenesis and cancer: mechanisms and consequences of localized, complex chromosomal rearrangements. *Nature medicine*, 18(11):1630–1638, 2012.

[56] The MathWorks Inc. *MATLAB 7.11.0(R2010b)*. 2010. Natick, Massachussets.

[57] Broad Institute. Coverage. `http://www.broadinstitute.org/crd/wiki/index.php/Read_coverage`, May 2013.

[58] A. Jauch, J. Wienberg, R. Stanyon, N. Arnold, S. Tofanelli, T. Ishida, and T. Cremer. Reconstruction of genomic rearrangements in great apes and gibbons by chromosome painting. *Proceedings of the National Academy of Sciences*, 89(18):8611–8615, 1992.

[59] A. Kallioniemi, O.P. Kallioniemi, D. Sudar, D. Rutovitz, J.W. Gray, F. Waldman, and D. Pinkel. Comparative genomic hybridization for molecular cytogenetic analysis of solid tumors. *Science*, 258(5083):818–821, 1992.

[60] Alexander Karzanov and Leonid Khachiyan. On the conductance of order markov chains. *Order*, 8(1):7–15, 1991.

[61] Robert C King, Pamela Mulligan, and William Stansfield. *A dictionary of genetics*. OUP USA, 2013.

[62] Bonnie Kirkpatrick, Yakir Reshef, Hilary Finucane, Haitao Jiang, Binhai Zhu, and Richard M Karp. Comparing pedigree graphs. *Journal of Computational Biology*, 19(9):998–1014, 2012.

[63] S.R. Knezevich, D.E. McFadden, W. Tao, J.F. Lim, and P.H.B. Sorensen. A novel etv6-ntrk3 gene fusion in congenital fibrosarcoma. *Nature genetics*, 18(2):184–187, 1998.

[64] Takashi Kohno and Jun Yokota. Molecular processes of chromosome 9p21 deletions causing inactivation of the p16 tumor suppressor gene in human cancer: Deduction from structural analysis of breakpoints for deletions. *DNA repair*, 5(9):1273–1281, 2006.

[65] Roman Kolpakov, Ghizlane Bana, and Gregory Kucherov. mreps: efficient and flexible detection of tandem repeats in DNA. *Nucleic acids research*, 31(13):3672–3678, 2003.

[66] T.G. Krontiris and G.M. Cooper. Transforming activity of human tumor DNAs. *Proceedings of the National Academy of Sciences*, 78(2):1181–1184, 1981.

[67] Razelle Kurzrock, Hagop M Kantarjian, Brian J Druker, and Moshe Talpaz. Philadelphia chromosome–positive leukemias: from basic mechanisms to molecular therapeutics. *Annals of internal medicine*, 138(10):819–830, 2003.

[68] P. Langerak, P. Russell, P. Langerak, and P. Russell. Regulatory networks integrating cell cycle control with DNA damage checkpoints and double-strand break repair. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 366(1584):3562–3571, 2011.

[69] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[70] R Lewis. *Human Genetics Concepts and Applications, 7th edition*. McGraw-Hill Higher Education, 2007.

[71] L.A. Loeb and C.C. Harris. Advances in chemical carcinogenesis: a historical review

and prospective. *Cancer research*, 68(17):6863–6872, 2008.

[72] Kinsella M and Bafna V. Modelling the breakage-fusion-bridge machanism: Combinatorics and cancer genomics. *Recomb 2012*, LNBI(7262):148–162, 2012.

[73] David J McBride, Dariush Etemadmoghadam, Susanna L Cooke, Kathryn Alsop, Joshy George, Adam Butler, Juok Cho, Danushka Galappaththige, Chris Greenman, Karen D Howarth, et al. Tandem duplication of chromosomal segments is common in ovarian and breast cancer genomes. *The Journal of pathology*, 227(4):446–455, 2012.

[74] Cooke SL Alsop K George J Butler A Cho J Galappaththige 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 McBride DJ, Etemadmoghadam D. Bouvel m and rossin d, a variant of the tandem duplication-random loss model of genome rearrangement, 2009, *Theoretical Computer Science*, **410**, 847-858. 2012.

[75] Cooke SL Alsop K George J Butler A Cho J Galappaththige 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 McBride DJ, Etemadmoghadam D. Bouvel m and pergola e, posets and permutations in the duplication-loss model: Minimal permutations with d descents, 2010, *Theoretical Computer Science*, **411**, 2487-2501. 2012.

[76] Cooke SL Alsop K George J Butler A Cho J Galappaththige 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 McBride DJ, Etemadmoghadam D. Allouche j. 2012. and Shallit J. 2003, *Automatic Sequences, Theory, Applications, Generalizations*, CUP.

[77] B. McClintock. The stability of broken ends of chromosomes in zea mays. *Genetics*, 26(2):234, 1941.

[78] Scott McGinnis and Thomas L Madden. Blast: at the core of a powerful and diverse set of sequence analysis tools. *Nucleic acids research*, 32(suppl 2):W20–W25, 2004.

[79] Felix Mitelman. Recurrent chromosome aberrations in cancer. *Mutation Research/Reviews in Mutation Research*, 462(2):247–253, 2000.

[80] Joseph Neggers and Hee Sik Kim. *Basic posets*. World Scientific, 1998.

[81] PC Nowell. A minute chromosome in human chronic granulocytic leukemia. *Science*, 132:1497, 1960.

[82] Tom MW Nye. Modelling the evolution of multi-gene families. *Statistical methods in medical research*, 18(5):487–504, 2009.

[83] Layla Oesper, Anna Ritz, Sarah J Aerni, Ryan Drebin, and Benjamin J Raphael. Reconstructing cancer genomes from paired-end sequencing data. *BMC bioinformatics*, 13(Suppl 6):S10, 2012.

[84] H Ogiwara, T Kohno, H Nakanishi, K Nagayama, M Sato, and J Yokota. Unbalanced translocation, a major chromosome alteration causing loss of heterozygosity in human lung cancer. *Oncogene*, 27(35), 2008.

[85] H Ogiwara, T Kohno, H Nakanishi, K Nagayama, M Sato, and J Yokota. Unbalanced translocation, a major chromosome alteration causing loss of heterozygosity in human

lung cancer. *Oncogene*, 27(35):4788–4797, 2008.

[86] Susumu Ohno et al. *Evolution by gene duplication.* London: George Alien & Unwin Ltd. Berlin, Heidelberg and New York: Springer-Verlag., 1970.

[87] M. Olszewski. Concepts of cancer from antiquity to the nineteenth century. *University of Toronto Medical Journal*, 87(3):181–186, 2010.

[88] Michal Ozery-Flato and Ron Shamir. Two notes on genome rearrangement. *Journal of Bioinformatics and Computational Biology*, 1(1):71–94, 2003.

[89] Luca Penso-Dolfin, Taoyang Wu, and Chris D Greenman. The combinatorics of tandem duplication. *Discrete Applied Mathematics*, 194:1–22, 2015.

[90] Benjamin A Pierce. *Genetics: A conceptual approach*. WH Freeman, 2010.

[91] D. Pinkel, J. Landegent, C. Collins, J. Fuscoe, R. Segraves, J. Lucas, and J. Gray. Fluorescence in situ hybridization with human chromosome-specific libraries: detection of trisomy 21 and translocations of chromosome 4. *Proceedings of the National Academy of Sciences*, 85(23):9138–9142, 1988.

[92] Benjamin J Raphael and Pavel A Pevzner. Reconstructing tumor amplisomes. *Bioinformatics*, 20(suppl 1):i265–i273, 2004.

[93] Benjamin J Raphael, Stanislav Volik, Colin Collins, and Pavel A Pevzner. Reconstructing tumor genome architectures. *Bioinformatics*, 19(suppl 2):ii162–ii171, 2003.

[94] E.P. Reddy, R.K. Reynolds, E. Santos, M. Barbacid, et al. A point mutation is responsible for the acquisition of transforming properties by the t 24 human bladder carcinoma oncogene. *Nature*, 300(5888):149–152, 1982.

[95] Eric Rivals. A survey on algorithmic aspects of tandem repeats evolution. *International Journal of Foundations of Computer Science*, 15(02):225–257, 2004.

[96] J.D. Rowley. A new consistent chromosomal adnormality in chronic myelogeneus leukaemia identified by quinacrine fluorescence and giemsa staining. *Landmarks in Medical Genetics: Classic Papers with Commentaries*, 243(51):104, 2004.

[97] E. Schröck, S. Du Manoir, T. Veldman, B. Schoell, J. Wienberg, MA Ferguson-Smith, Y. Ning, DH Ledbetter, I. Bar-Am, D. Soenksen, et al. Multicolor spectral karyotyping of human chromosomes. *Science*, 273(5274):494–497, 1996.

[98] Charles Semple and Mike Steel. Phylogenetics. 2003.

[99] C. Shih, LC Padhy, M. Murray, R.A. Weinberg, et al. Transforming genes of carcinomas and neuroblastomas introduced into mouse fibroblasts. *Nature*, 290(5803):261–264, 1981.

[100] Ram J Singh. *Plant cytogenetics*. Taylor and Francis Group, 2003.

[101] Catherine E Smith, Bertrand Llorente, and Lorraine S Symington. Template switching during break-induced replication. *Nature*, 447(7140):102–105, 2007.

[102] Philip J Stephens, Chris D Greenman, Beiyuan Fu, Fengtang Yang, Graham R

Bignell, Laura J Mudie, Erin D Pleasance, King Wai Lau, David Beare, Lucy A Stebbings, et al. Massive genomic rearrangement acquired in a single catastrophic event during cancer development. *Cell*, 144(1):27–40, 2011.

[103] Michael R Stratton, Peter J Campbell, and P Andrew Futreal. The cancer genome. *Nature*, 458(7239):719–724, 2009.

[104] C.J. Tabin, S.M. Bradley, C.I. Bargmann, R.A. Weinberg, A.G. Papageorge, E.M. Scolnick, R. Dhar, D.R. Lowy, E.H. Chang, et al. Mechanism of activation of a human oncogene. *Nature*, 300(5888):143, 1982.

[105] Y. Tan, R.A. Timakhov, M. Rao, D.A. Altomare, J. Xu, Z. Liu, Q. Gao, S.C. Jhanwar, A. Di Cristofano, D.L. Wiest, et al. A novel recurrent chromosomal inversion implicates the homeobox gene dlx5 in t-cell lymphomas from lck-akt2 transgenic mice. *Cancer research*, 68(5):1296–1302, 2008.

[106] H. Tanaka and M.C. Yao. Palindromic gene amplificationan evolutionarily conserved role for DNA inverted repeats in the genome. *Nature Reviews Cancer*, 9(3):216–224, 2009.

[107] C.T. Thompson and J.W. Gray. Cytogenetic profiling using fluorescence in situ hybridization (fish) and comparative genomic hybridization (cgh). *Journal of Cellular Biochemistry*, 53(S17G):139–143, 1993.

[108] Howard Turtle and W Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems (TOIS)*, 9(3):187–222, 1991.

[109] WT Tutte. *Graph theory*. Cambridge University Press, 2001.

[110] Alexandra Valera, Olga Balagué, Luis Colomo, Antonio Martínez, Jan Delabie, Lekidelu Taddesse-Heath, Elaine S Jaffe, and Elías Campo. Ig/myc rearrangements are the main cytogenetic alteration in plasmablastic lymphomas. *The American journal of surgical pathology*, 34(11):1686, 2010.

[111] Peter Van Loo, Silje H Nordgard, Ole Christian Lingjærde, Hege G Russnes, Inga H Rye, Wei Sun, Victor J Weigman, Peter Marynen, Anders Zetterberg, Bjørn Naume, et al. Allele-specific copy number analysis of tumors. *Proceedings of the National Academy of Sciences*, 107(39):16910–16915, 2010.

[112] James D Watson. Molecular biology of the gene/. 2008.

[113] J.D. Watson and FHC Crick. Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Annals of the New York Academy of Sciences-Paper Edition*, 758:12, 1995.

[114] Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

[115] Jialiang Yang and Louxin Zhang. On counting tandem duplication trees. *Molecular biology and evolution*, 21(6):1160–1163, 2004.

[116] Jiang Zeng. Multinomial convolution polynomials. *Discrete Mathematics*, 160(1):219–228, 1996.

[117] Cheng-Zhong Zhang, Mitchell L Leibowitz, and David Pellman. Chromothripsis and beyond: rapid genome evolution from complex chromosomal rearrangements. *Genes*

& development, 27(23):2513–2530, 2013.

[118] Feng Zhang, Lorraine Potocki, Jacinda B Sampson, Pengfei Liu, Amarilis Sanchez-Valle, Patricia Robbins-Furman, Alicia Delicado Navarro, Patricia G Wheeler, J Edward Spence, Campbell K Brasington, et al. Identification of uncommon recurrent potocki-lupski syndrome-associated duplications and the distribution of rearrangement types and mechanisms in ptls. *The American Journal of Human Genetics*, 86(3):462–470, 2010.

[119] Yanming Zhang, Pamela Strissel, Reiner Strick, Jianjun Chen, Giuseppina Nucifora, Michelle M Le Beau, Richard A Larson, and Janet D Rowley. Genomic DNA breakpoints in aml1/runx1 and eto cluster with topoisomerase ii DNA cleavage and dnase i hypersensitive sites in t (8; 21) leukemia. *Proceedings of the National Academy of Sciences*, 99(5):3070–3075, 2002.

[120] Yanming Zhang, Nancy Zeleznik-Le, Neelmini Emmanuel, Nimanthi Jayathilaka, Jianjun Chen, Pamela Strissel, Reiner Strick, Loretta Li, Mary Beth Neilly, Tomohiko Taki, et al. Characterization of genomic breakpoints in mll and cbp in leukemia patients with t (11; 16). *Genes, Chromosomes and Cancer*, 41(3):257–265, 2004.

7 Appendix

Following is the journal paper, based on the work presented on Chapter 4, which was accepted for publication in *Discrete Applied Mathematics* [89]. The work directly attributable to the candidate includes

- The development of the algebraic formalism, which forms the background for the definition of the main mathematical problem addressed in the paper

- The development of Matlab code which first allowed for the calculation of both the full TD space and the connection evolutions space

- An active involvement in the study of the inducement of evolutions, assessing the necessary graph operations required to transform the original tree into the induced one.

The Combinatorics of Tandem Duplication

Penso-Dolfin L[1], Wu T[1] and Greenman CD[1,2]*

[1]School of Computing Sciences, University of East Anglia, Norwich, UK, NR4 7TJ.
[2]The Genome Analysis Center, Norwich Research Park, Norwich, UK, NR4 7UH.

Abstract

Tandem duplication is an evolutionary process whereby a segment of DNA is replicated and proximally inserted. The different configurations that can arise from this process give rise to some interesting combinatorial questions. Firstly, we introduce an algebraic formalism to represent this process as a word producing automaton. The number of words arising from n tandem duplications can then be recursively derived. Secondly, each single word accounts for multiple evolutions. 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 corresponding to a given word. 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 $\prod_{k=1}^{n}(4^k-(2k+1))$. The space of structures arising from tandem duplication thus grows at a super-exponential rate with leading order term $\mathcal{O}(4^{\frac{1}{2}n^2})$.

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

7.1 Introduction

Tandem Duplications occur when a region of DNA is duplicated and inserted adjacent to the original segment, such as portrayed in Figure 7.1A.

This biological process has long been known to be implicated in the formation of gene clusters [86], [82] and more recently has been implicated in the formation of amplicons in cancer [73], [92], [93], [117]. 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 these 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 [72], [9], [11], [10], [65] and the construction of phylogenies describing their evolution [12], [15], [14]. In [15] 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. A survey of algorithmic approaches can be found in [95]. 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 [39] and [115]. The space of permutations arising from a tandem duplication-loss model is characterized in [74], [75].

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. In [16], the signed gene orders of several genomes are compared and explanatory phylogenetic evolutions derived. In [39] and [115] 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 [39] and [115] 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 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 [73]. 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 7.1A we start with five contiguous regions, labeled $\underline{1}$, $\underline{2}$, $\underline{3}$, $\underline{4}$ and $\underline{5}$. This is the original configuration and is termed the *reference*. We then have an initial tandem duplication, copying region $\underline{234}$ and inserting a new copy next to the first to give sequence $\underline{1234}\mathbf{1}\underline{2345}$. 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 $\underline{4}$ is connected to the left side of segment $\underline{2}$, as seen in Figure 7.1Aii. Note also that the left hand end of the duplicated region $\underline{234}$ implicates the reference position between segments $\underline{1}$ and $\underline{2}$, the right hand end implicates the reference position between segments $\underline{4}$ and $\underline{5}$. Next we have the second tandem duplication, copying region $\underline{42}$ to finally give $\underline{1234}\mathbf{1}\underline{2}\mathbf{2}\underline{4}\mathbf{1}\underline{2345}$. We now have another connection, labeled **2**, between the right side of segment $\underline{2}$ and the left of segment $\underline{4}$, as seen in Figure 7.1Aiii. Note that we now have two copies of the connection labeled **1**, which was also duplicated. The left hand end of the duplicated region $\underline{4}\mathbf{1}\underline{2}$ implicates the reference position between $\underline{3}$ and $\underline{4}$, the right hand end implicates that between $\underline{2}$ and $\underline{3}$. We have thus implicated all four breakpoints between the five reference

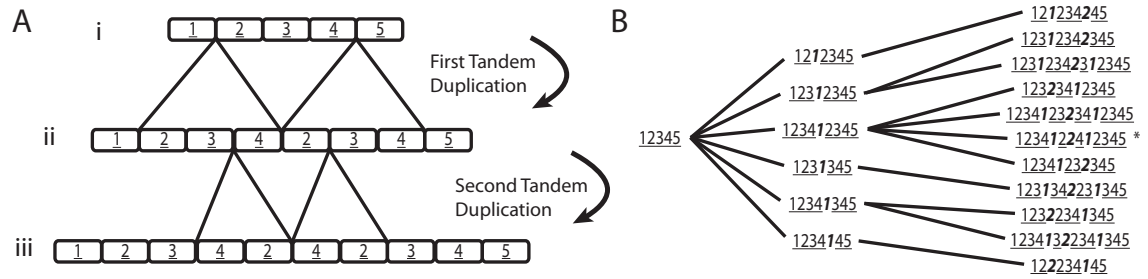

Figure 7.1: A Tandem Duplication Process. A) Three structures i)-iii) arising from two tandem duplications on a reference of five regions; $ABCDE$. B) Eleven possible evolutions with two tandem duplications. The example in A is highlighted by *. Underlined numbers are segments. Bold italicized numbers $\boldsymbol{n}$ indicate connections between segments formed in the n^{th} tandem duplication.

segments exactly once; unique breakpoint use.

In Figure 7.1B 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 7.1. *Count the number of different ways that an initial string of* $2N+1$ *segments can evolve under* N *tandem duplications, without reusing breakpoints.*

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 $\underline{12345} \to \underline{1234}\mathbf{1}\underline{2345} \to \underline{1234}\mathbf{1}\underline{2}\mathbf{2}\underline{4}\mathbf{1}\underline{2345}$ becomes the simpler sequence $\epsilon \to \mathbf{1} \to \mathbf{121}$, where ϵ denotes the empty word. Although this representation is simpler, it is not unique - five of the eleven cases in Figure 7.1B contain this sequence of connections. However, we will need to consider these sequences in more detail to solve Problem 7.1.

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 words involving connection symbols. Each such word will be seen to correspond to many different struc-

tures formed by tandem duplications. Thus, thirdly, we consider how to count the distinct cases that all correspond to a single word 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 7.1. Concluding remarks complete the paper.

7.2 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 7.1. *We define* $\underline{i}$ *to be a* segment *symbol. This represents a copy of the DNA segment originally in the* i^{th} *reference position, where* $i \in \{1, ..., 2N+1\}$.

Definition 7.2. *We define* **i** *to be a* connection *symbol. These symbols always have a segment symbol on either side. In any subword of the form* $\underline{m}\mathbf{i}\underline{n}$, **i** *represents a connection between the right hand side of the DNA segment represented by* $\overline{m}$*, and the left hand side of the DNA segment represented by* $\overline{n}$*, formed during the* i^{th} *TD, where* $i \in \{1, ..., N\}$.

From the examples in Figure 7.1B we see that each TD duplicates a subsequence of contiguous segments. We then construct words with the following automaton [76]:

Definition 7.3. *A* TD evolution U *is defined as any sequence of* TD words $[U_0 \to U_1 \to ... \to U_N]$ *generated as follows. We initialize the sequence with TD word* $U_0 = \underline{1} \cdot \underline{2} \cdot ... \cdot \underline{(2N+1)}$. *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* $U_n = X \cdot Y \cdot \mathbf{n} \cdot Y \cdot Z$. *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. *We have a TD evolution provided* n_a *and* n_b *are* $2N$ *distinct values for* $n \in \{1, ..., N\}$. *The values* n_a *and* n_b *are referred to as* breakpoint numbers*, and the underlying value* n *is the* TD number. *We also define*

breakpoint numbers 0_a *and* 0_b *to be* 0 *and* $2N+1$*, respectively, representing the start and end of the set of original reference segments. We let* $\mathcal{U}_N$ *denote the set of possible 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 begins and ends with a segment symbol, so **n** is bordered by segment symbols on either side, as required by Definition 4.2.

Example 7.1. *In the evolution * of Figure 7.1B we have:* $E = [\overline{12345} \to \overline{1234}\mathbf{1}\overline{12345} \to \overline{1234}\mathbf{1}\bar{2}\mathbf{2}\bar{4}\mathbf{1}\overline{12345}]$*. The second TD duplicates segments* $\underline{4}\mathbf{1}\underline{2}$ *in the TD word* $\overline{1234}\mathbf{1}\overline{12345}$*. The subwords* X*,* Y *and* Z *are then* $\overline{123}$*,* $\underline{4}\mathbf{1}\underline{2}$ *and* $\overline{345}$*, 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* $\underline{4}\mathbf{1}\underline{2}$*, between segments* $\underline{3}$ *and* $\underline{4}$*, and the* 2^{nd} *breakpoint is implicated by the right end of the duplicated region between segments* $\underline{2}$ *and* $\underline{3}$*.*

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

Lemma 7.1. *If the reference position between segments* $\underline{i}$ *and* $\underline{i+1}$ *be labelled as* i*, which we interpret as the* i^{th} breakpoint*, then the duplicated region in the iteration of Definition 7.3 corresponding to the* n^{th} *TD has a left breakpoint at reference position* n_a *and a right breakpoint at reference position* n_b*.*

Proof. We know from Definition 4.3 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. The rightmost segment in X is defined to be $\underline{n_a}$, and so the left most segment in Y is $\underline{n_a+1}$. 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 $\underline{n_b}$. The rightmost breakpoint between these segments is therefore the reference position between

segments $\underline{n_b}$ and $\underline{n_b+1}$, which has label n_b, as required. □

Definition 7.4. *Restricting a valid TD evolution* $U = [U_0 \to U_1 \to ... \to U_N]$ *to connection symbols induces a* connection evolution $E = [E_0 \to E_1 \to ... \to E_N]$ *and each member of the sequence is a* connection word. *We define* $\mathcal{E}_N$ *to be the set of possible connection evolutions that arises from N TDs.*

Example 7.2. *In the evolution * of Figure 7.1B we have:* $U = [\overline{12345} \to \overline{1234}\mathbf{1}\overline{2345} \to \overline{1234}\mathbf{1}\bar{2}\mathbf{2}\bar{4}\mathbf{1}\bar{2}\overline{345}]$ *which becomes* $E = [\epsilon \to \mathbf{1} \to \mathbf{121}]$ *when the segment symbols are removed.*

We now formalize Problem 7.1:

Problem 7.2. *Determine the size of the set* $\mathcal{E}_N$.

The construction of connection evolutions is required to solve this problem. Toward this end the following observation proves to be useful:

Lemma 7.2. *Let* $\mathbf{m}$ *and* $\mathbf{n}$ *be consecutive connection symbols in any connection word* E_i, *induced from TD word* U_i. *Then* $\mathbf{m}$ *and* $\mathbf{n}$ *are separated by segment symbols* $\underline{m_a+1} \cdot ... \cdot \underline{n_b}$ *in* U_i. *If* $\mathbf{m}$ *is the first connection symbol in* E_i, *in* U_i *it is preceded by segment symbols* $\underline{1} \cdot ... \cdot \underline{m_b}$. *If* $\mathbf{n}$ *is the last connection symbol in* W_i, *in* U_i *it is followed by segment symbols* $\underline{n_a+1} \cdot ... \cdot \underline{2N+1}$.

Proof. From the iteration in Definition 7.3, we firstly 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 $U_n = X \cdot Y \cdot \mathbf{n} \cdot Y \cdot Z$ is formed during the n^{th} TD from $U_{n-1} = X \cdot Y \cdot Z$. The symbol to the left of $\mathbf{n}$, the last symbol of Y, is segment $\underline{n_b}$ by definition. The symbol to the right of $\mathbf{n}$, is the first symbol of Y. Now the last symbol of X is $\underline{n_a}$ by definition. This is adjacent to the first symbol of Y, which is also a segment symbol, which being in consecutive reference order must be $\underline{n_a+1}$. We thus find that if we have consecutive symbols $\mathbf{mn}$ in a connection word, in the corresponding TD word, we have the segment symbol $\underline{m_a+1}$ to the right of $\mathbf{m}$ and $\underline{n_b}$ to the left of $\mathbf{n}$. These must run in consecutive reference order giving the

segments stated. Noting from Definition 7.3 that the first and last segments are always $\underline{1}$ and $\underline{2N+1}$ completes the Lemma. □

This has the following important consequence.

Corollary 7.1. *To each distinct TD evolution U there corresponds a unique connection evolution E and ordering of $2N$ breakpoint numbers $\{n_a, n_b\}_n$, where $n \in \{1, 2, ..., N\}$.*

Proof. If we have a given connection evolution E and ordering of breakpoint numbers $\{n_a, n_b\}_n$, where $n \in \{1, 2, ..., N\}$, then for any connection symbol $\mathbf{m}$ in a connection word we can use Lemma 7.2 to identify the segment symbols that lie on either side of $\mathbf{m}$ in the corresponding TD word. Repeating this for all the words in the connection evolution results in a uniquely specified TD evolution. □

Example 7.3. *If we take the last word of the connection evolution $[\epsilon \to \mathbf{1} \to \mathbf{121}]$ and the breakpoint number ordering $(1_a, 2_b, 2_a, 1_b) = (1, 2, 3, 4)$ then by Lemma 7.2, between connection symbols $\mathbf{1}$ and $\mathbf{2}$ we have segment symbols $\underline{1_a + 1} \cdot ... \cdot \underline{2_b} = \underline{2}$ in the corresponding TD evolution. Similarly between connection symbols $\mathbf{2}$ and $\mathbf{1}$ we have segment symbols $\underline{2_a + 1} \cdot ... \cdot \underline{1_b} = \underline{4}$ in the corresponding TD evolution. Before the first copy of connection symbol $\mathbf{1}$ we have segment symbols $\underline{1} \cdot ... \cdot \underline{1_b} = \underline{1234}$, and after the second copy of connection symbol $\mathbf{1}$ we have segment symbols $\underline{1_a + 1} \cdot ... \cdot \underline{2(2) + 1} = \underline{2345}$. For the full TD word we then obtain $\underline{1234}\mathbf{1}\underline{2}\mathbf{2}\underline{4}\mathbf{1}\underline{2345}$. Note that not all breakpoint number orderings are feasible. For example, if we attempt to use $(2_a, 2_b, 1_a, 1_b) = (1, 2, 3, 4)$, then between connection symbols $\mathbf{1}$ and $\mathbf{2}$ we try to place $\underline{1_a + 1} \cdot ... \cdot \underline{2_b} = \underline{3} \cdot ... \cdot \underline{2}$, which is not an increasing sequence of segment symbols.*

Thus to solve Problem 7.2 we need to know firstly what different connection evolutions $E \in \mathcal{E}_N$ are possible, and secondly, for each such connection evolution, we need to know how may different orderings of breakpoint numbers $\{n_a, n_b\}_n$ for $n = 1, 2, ..., N$ are feasible. This gives us three problems of increasing complexity; firstly, how to count the number

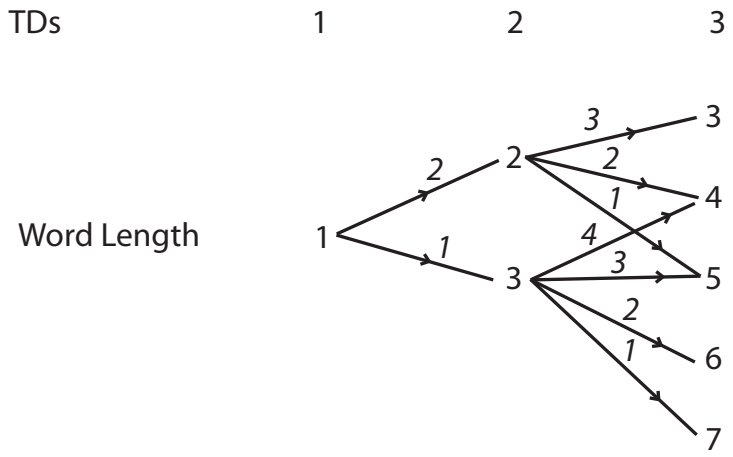

Figure 7.2: Schematic representation of the number of possible TD words. Numbers at nodes indicate the length of TD words. Numbers on edges indicate the number of choices.

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 7.2. We consider these in turn.

7.3 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 7.1. *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,*

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

where we have initial values $w_{i,0} = \{^{1,i=0}_{0,i\geq 1}$

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 in definition 7.3 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 $k \in \{\lfloor \frac{m-1}{2} \rfloor, ..., m-1\}$. Lastly, we note that there are $k-(m-k-1)+1 = 2k-m+2$ ways to do this. □

Example 7.4. *In Figure 7.2 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 labelled* 1*, and summing. For example, the node labelled* 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 7.1.

TDs	1	2	3	4	5	6
Connection Evolutions	1	3	22	377	15,315	1,539,281
TD Evolutions	1	11	627	154,869	156,882,297	640,550,418,651

Table 7.1: Counts of Connection and TD Evolutions.

7.4 Counting Evolutions with Posets

We are now interested in the number of TD evolutions that all correspond to a single connection evolution. From Corollary 7.1 we see that this is equivalent to counting the number of feasible orders of the $2N$ breakpoint numbers $\{n_a, n_b\}_n$, for $n \in \{1, 2, ..., N\}$, that arise from the TD process in Defintion 7.3. In order to do this, in section 7.4.1 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 7.4.2 we then use $2d$-*tree*s to encapsulate the choices of the different orders of the breakpoint numbers. Finally, in section 7.4.3 we will use this to construct Hasse diagrams, for which the number of linear extensions will provide the desired count.

7.4.1 Zig-zag plots

Now from Lemma 7.2 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 $\underline{m_a + 1}$ to $\underline{n_b}$. By Lemma 7.1, 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 segments 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 7.3A we

have such an example. The ordered breakpoint numbers are arranged along the top. The plots correspond to the connection evolution given in Figure 7.3E. For example, in Figure 7.3Aiii 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.

Definition 7.5. *If we have a connection word* $\mathbf{n}^{(1)} \cdot \mathbf{n}^{(2)} \cdot ... \cdot \mathbf{n}^{(K)}$*, along with corresponding breakpoint numbers* $n_a^{(k)}$ *and* $n_b^{(k)}$*, where* $k \in \{1, 2, ..., K\}$*, then we construct an initial* interval $[0_a, n_b^{(1)}]$*, internal intervals* $[n_a^{(1)}, n_b^{(2)}]$,$[n_a^{(2)}, n_b^{(3)}]$*, ...,* $[n_a^{(K-1)}, n_b^{(K)}]$ *and a final interval* $[n_a^{(K)}, 0_b]$*. Each interval* $[x, y]$ *is plotted horizontally (solid line) from reference coordinate* x *to* y*. Their vertical coordinates 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.*

Lemma 7.3. *The zig-zag representation is equivalent to the TD evolution it represents.*

Proof. Using Lemmas 7.1 we find that the intervals in the zig-zag representation of Definition 7.5 contain precisely those segments in Lemma 7.2. Furthermore, each dashed line connects intervals of the form $[n_a^{(k-1)}, n_b^{(k)}]$ and $[n_a^{(k)}, n_b^{(k+1)}]$. The TD number $n^{(k)}$ common to both intervals provides the value of the connection symbol required to complete the corresponding TD word. □

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

7.4.2 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

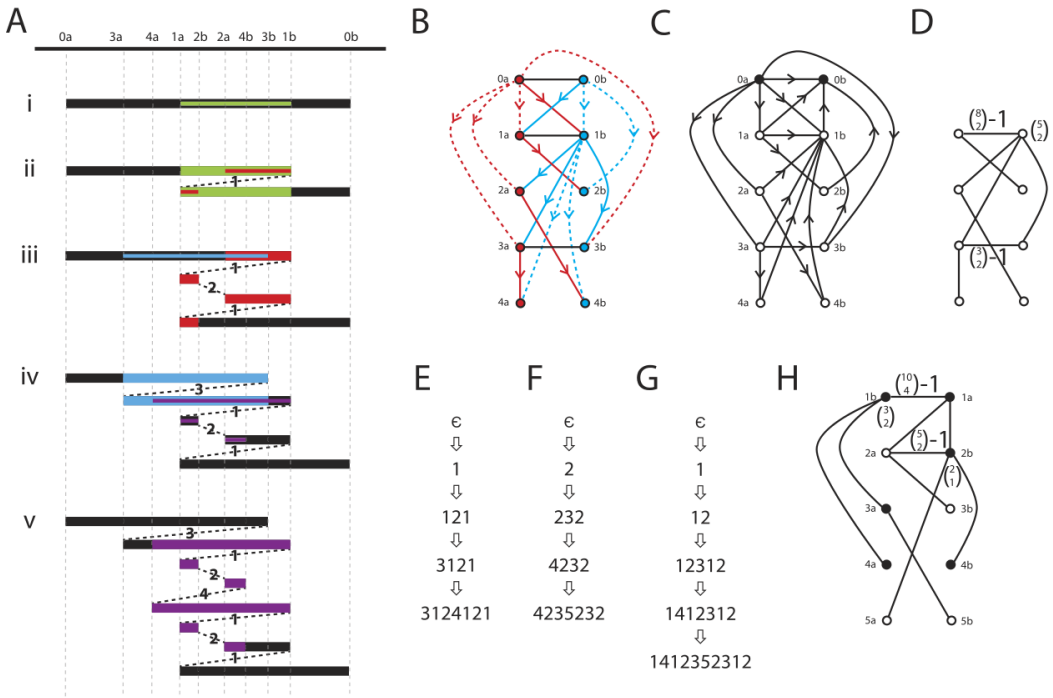

Figure 7.3: Representation of the TD Process. In A) we have *zig-zag* plots for a sequence of four TDs, resulting in five structures i)-v). The green regions indicates the region duplicated during each TD. Dashed lines indicate a connection between segments. Coordinates n_a and n_b indicate the end positions of the n^{th} duplicated region. B) Corresponding 2d-tree. Nodes correspond to breakpoints and edges demarcate an ordering. Red and blue colours indicate lower and upper bound breakpoints. Dashed and plain edges indicate minor and major edges. C) Corresponding Hasse diagram. D) The major graph corresponding to evolution E. F) Increases each symbol of E by 1. G) An induced evolution from F. H) The major graph corresponding to induced evolution G. The black nodes indicate the corresponding 1-nodeset.

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 [13], [108] and have seen other applications in genetics as recombination graphs [45], and pedigree graphs [62], for example.

Consider then how the zig-zag plots build up in the structures in Figures 4.4A i-v. We start with a single set of reference segments; the single interval $[0_a, 0_b]$ in Figure 7.3Ai. 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 7.3B. 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 4.4Ai), and implicates breakpoint numbers 1_a and 1_b, by Lemma 7.1, resulting in the zig-zag diagram of Figure 7.3Aii. 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_a to 1_a and 1_b representing the fact that 0_a 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 green portion in Figure 4.4Aii, 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 4.4Aii 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 2d-tree construction:

Definition 7.6. *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 the endpoints 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 4.4B 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; [$\mathbf{121} \rightarrow \mathbf{3121}$], for example.

We have seen that a zig-zag representation is equivalent to the TD evolution it is based upon, which is in turn equivalent to the induced connection evolution and corresponding 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 [$\epsilon \rightarrow \mathbf{1} \rightarrow \mathbf{12} \rightarrow \mathbf{12312}$], 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 evolution may differ.

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

7.4.3 Linear Extensions

Although we counted the number of connection evolutions with relative ease in section 4.2, there maybe several different TD evolutions that correspond to a single connection evolution. We see from Figure 7.4A, for example, that there are three TD evolutions that have a corresponding connection evolution [$\epsilon \rightarrow \mathbf{1} \rightarrow \mathbf{12}$], 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 7.3Aii, 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 7.4A i-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*) [80]. 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 7.4. *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 7.5. *For example, in Figure 7.3B we see the 2d-tree corresponding to the connection evolution given in Figure 7.3E. In Figure 7.3C 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 $0_a < 3_a < 4_a < 1_a < 2_a < 2_b < 2_b < 4_b < 3_b < 1_b < 0_b$ at the top of Figure 7.3A, is satisfied by this Hasse diagram.*

Proof. (*of Lemma 7.4*) When we add any node $x \in \{n_a, n_b\}$ to the 2d-tree, it has two

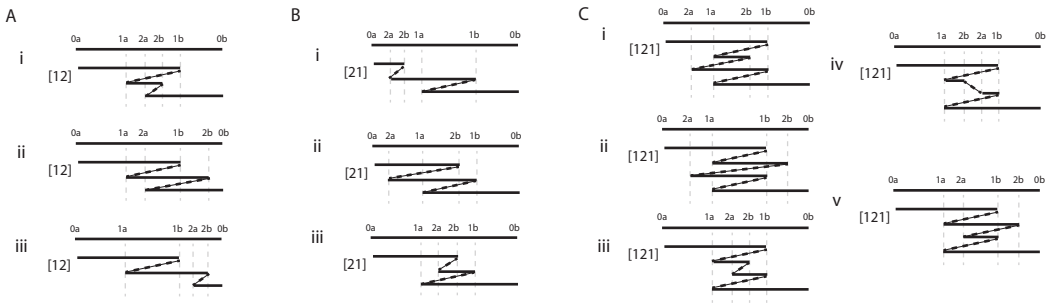

Figure 7.4: Zig-zag plots of structures arising from two TDs. A) Three structures associated with connection word 12. B) Three structures associated with connection word 21. C) Five structures associated with connection word 121.

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 correponding poset. Although finding any single linear extension from a poset can be achieved in polynomial time [60], counting the number of linear extensions is known to be #P-complete [21] and in general is slow to implement [80]. 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 next result tells us that this simplified structure contains two trees if we also ignore the fences.

Lemma 7.5. *The restriction of the Hasse diagram to the major edges results in two trees rooted to nodes* 0_a *and* 0_b*.*

Proof. In the construction of the poset graph, every node $x \notin \{0_a, 0_b\}$ has two parental nodes, labeled u_a and v_b, arising from the segment $[u_a, v_b]$ that breakpoint x is formed upon. These two nodes are connected to x by a major and minor parental edge, where $max(u, v)$ and $min(u, v)$ are the major and minor TD numbers, respectively. Thus if we are restricted to the major edges, each node has one parental node, resulting in two trees attached to roots 0_a and 0_b. □

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 removing the minor edges from the Hasse diagram does not lose any information.

Lemma 7.6. *If* $[u_a, v_b]$ *is any interval from a zig-zag plot arising in a connection evolution then either:*

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 $[0_a, 1_b]$ and $[1_a, 0_b]$ (such as in Figure 7.3Aii). Now node 1_b has major parental node 0_a and 1_a 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 7.2. *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).*

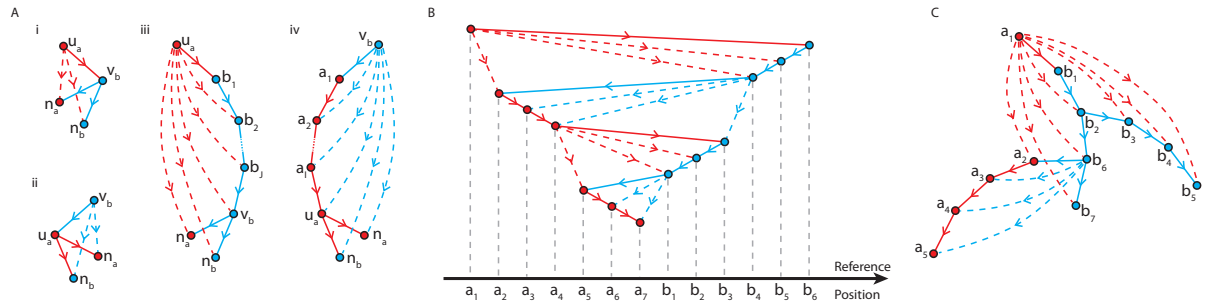

Figure 7.5: Major and minor edge structure. A) The addition of new nodes preserves major-minor structure. B) The nesting structure of a branch of a major tree. C) The general major-minor structure.

Example 7.6. *Consider the branch in Figure 7.5C. 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_5 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 7.2*) Now by Lemma 7.6 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 $x \in \{n_a, n_b\}$ 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 7.5Ai).

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 7.5Aii).

Case III: ($u < v$, minor edge from u_a to v_b). Then by Lemma 7.6 minor node u_a is connected to major v_b by a chain of major edges of the form $u_a \to (n_1)_b \to (n_2)_b \to ... \to (n_I)_b \to v_b$ 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 $u_a \to (n_1)_b \to (n_2)_b \to ... \to (n_I)_b \to v_b \to x$. 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 7.5Aiii).

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 7.5Aiv). □

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 7.3. *Consider any single directed chain of major edges connecting nodes* $\{a_i, b_j : i = 1, ..., I, j = 1, ..., J\}$ *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* $i = 1, 2, ..., I-1$, *and* b_j *is an ancestor of* b_{j+1} *for* $j = 1, 2, ..., J-1$. *These nodes have a single linear extension of the form:*

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

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 7.5B for an example).

Proof. Now consider any sub-chain of nodes connected by major edges of the form $a_1 \to b_1 \to b_2 \to ... \to b_n$. 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 $b_2, ..., b_n$ all have a_1 as their minor parent by Corollary 7.2, so $a_1 < b_i$ for $i = 2, 3, ..., I$. Together we then have the single order $a_1 < b_n < ... < b_2 < b_1$. If the chain then continues as a chain of type a nodes $b_n \to a'_1 \to a'_2 \to ... \to a'_m$, we similarly find that $a'_1 < a'_2 < ... < a'_m < b_n$. However, a'_1 has minor parent a_1 by Corollary 7.2 so $a_1 < a'_1$. We then find that these two orders combine into the single order $a_1 < a'_1 < a'_2 < ... < a'_m < b_n < ... < b_2 < b_1$. 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. □

We now explain how to count the linear extensions using the major graph. In all that follows $\binom{m}{r} = \frac{m!}{r!(m-r)!}$ represent binomial coefficients, and $\binom{m}{m_1,...,m_I} = \frac{m!}{m_1!...m_I!}$ represent multinomial coefficients. There are two situations we need to deal with.

Lemma 7.7. *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 $\binom{m}{m_1,...,m_K} \prod_{k=1}^{K} \phi_k$, where $m = \sum_{k=1}^{K} m_k$, and ϕ_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 ϕ_1 and ϕ_2 are the number of linear extensions associated with the m_1 and m_2 nodes in the respective branches.

Proof. i) We have ϕ_k linear extensions associated with branch k. If we select one linear extension from each branch, we have, by Corollary 7.3, K orderings of the form:

$$(x_{i_1}^{(k)})_a < (x_{i_2}^{(k)})_a < ... < (x_{i_{m_k}}^{(k)})_b < (x_{i_{m_k+1}}^{(k)})_b$$

Here $(x_{i_j}^{(k)})_{a/b}$ 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 7.3 either $z = (x_{i_1}^{(1)})_a = ... = (x_{i_1}^{(K)})_a$ is the left most node and is of type a, or $z = (x_{i_{m_1+1}}^{(1)})_b = ... = (x_{i_{m_K+1}}^{(K)})_b$ is the right most node and is of type b (in Figure 7.5B 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 ϕ_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 7.3, if z is of type a they will take the form:

$$(z)_a < n_a < (x_{i_1}^{(1)})_a < ... < (x_{i_{m_1-1}}^{(1)})_b$$

$$(z)_a < (x_{i_1}^{(2)})_a < ... < (x_{i_{m_2-1}}^{(2)})_b < n_b$$

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

$$(z)_a < (x_{i_1}^{(2)})_a < ... < (x_{i_{m_2-1}}^{(2)})_b < n_b < n_a < (x_{i_1}^{(1)})_a < ... < (x_{i_{m_1-1}}^{(1)})_b$$

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 7.2. *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 $\binom{y_1+y_2}{y_1} - 1$, 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 $m(x) = \binom{x}{x_1,...,x_r}$ 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 7.7 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 7.7 that the orders ϕ_m associated with nodes in individual branches are multiplied into the combinatorial terms (such as $\binom{m}{m_1,...,m_K}$) associated with the parental node. We thus multiply the terms of the form $\binom{m}{m_1,...,m_K}$ from nodes and $\binom{m}{m_1,...,m_K} - 1$ from fences. □

Example 7.7. *Consider the connection evolution $E = [\mathbf{1} \to \mathbf{121} \to \mathbf{3121} \to \mathbf{3124121}]$ with 2d-tree in Figure 7.3B. Once 0_a and 0_b 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 7.3D. The upper fence has two nodes attached to one side and six nodes to the other. This results in a count* $\binom{8}{2} - 1 = 27$. *We note that node* 1*b 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 $\frac{n!}{n!} = 1$. 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.

7.5 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 7.4 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 7.3. *The number $\mathcal{N}_n$ of distinct evolutions arising from n TDs is given by:*

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

Thus $\mathcal{N}_2 = (4^1 - (2(1)+1)) \cdot (4^2 - (2(2)+1)) = 11$, in agreement with Figure 7.4, for example. The first few terms in this series can be seen in the bottom row of Table 7.1.

7.5.1 A Motivating Example

Before constructing a proof of Theorem 7.3, we discuss a motivating example. Recall that $\mathcal{E}_n$ is the set of word evolutions on n TDs. Consider the following examples.

$$E = [\mathbf{1} \to \mathbf{121} \to \mathbf{3121} \to \mathbf{3124121}]$$

$$E^+ = [\mathbf{2} \to \mathbf{232} \to \mathbf{4232} \to [\mathbf{4235232}]$$

$$E' = [\mathbf{1} \to \mathbf{12} \to \mathbf{12312} \to \mathbf{1412312} \to \mathbf{1412352312}]$$

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 7.7. *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 4.4H. Although we can form this directly from the word evolution E' using the 2d-tree construction from the previous section, we note that Figure 7.3H is a subgraph of the 2d-tree from the original evolution E (Figure 7.3B). 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 $\mathcal{E}_{n-1}$ and $\mathcal{E}_n$, 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 $E = [\mathbf{1}]$, and increase the symbols by 1; $E^+ = [\mathbf{2}]$. We can introduce a new first TD in three ways; $E' = [\mathbf{1} \to \mathbf{12}]$, $E' = [\mathbf{1} \to \mathbf{21}]$ or $E' = [\mathbf{1} \to \mathbf{121}]$. 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 4.2. 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 4.3.

7.5.2 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 7.8. *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 $E = [X_1 \to X_2 \to X_3 \to ... \to X_n]$ from $\mathcal{E}_n$ we simply construct $E' = [\mathbf{1} \to \mathbf{1}X_1' \to \mathbf{1}X_2' \to \mathbf{1}X_3' \to ... \to \mathbf{1}X_n']$ 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 7.4. *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 4.1i that deletion of symbol 1 creates is a well defined mapping $D : \mathcal{E}_{n+1} \to \mathcal{E}_n$. 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 7.8ii 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 7.8i we know that $\bigcup_{E \in \mathcal{E}_n} \mathcal{E}(E) \supset \mathcal{E}_{n+1}$. □

Thus we can generate all of the word evolutions in $\mathcal{E}_{n+1}$ as a disjoint union of induced evolutions from $\mathcal{E}_n$.

We wish to construct the major graphs $T_{maj}(E')$ of all the induced word evolutions $E' \in \mathcal{E}(E)$ 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 7.8. *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 $X\mathbf{m}X$ *in word evolution* E^+ *becomes word* $X\mathbf{1m}X$ *in induced evolution* E'*, then* $m_b \in N$*.*

ii) If the word $X\mathbf{m}X$ *in word evolution* E^+ *becomes* $X\mathbf{m1}X$ *in induced evolution* E'*, then* $m_a \in N$*.*

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

Example 7.8. *In Figure 7.3F,G we have evolutions:*

$$E^+ = [\boldsymbol{2} \to \mathbf{2}\boldsymbol{3}\mathbf{2} \to \boldsymbol{4}\mathbf{232} \to \mathbf{423}\boldsymbol{5}\mathbf{232}]$$

$$E' = [\mathbf{1} \to \boldsymbol{12} \to \mathbf{12}\boldsymbol{3}\mathbf{12} \to \boldsymbol{14}\boldsymbol{1}\mathbf{2312} \to \mathbf{1412}\boldsymbol{3}\boldsymbol{5}\mathbf{2312}]$$

Now **2** *in* E^+ *becomes* **12** *in* E'*, so* $2_b \in N$*. Similarly,* $X\mathbf{3}X$ *becomes* $X\mathbf{31}X$ *so* $3_a \in N$ *(see italic symbols above). We see* $X\mathbf{4}X$ *becomes* $X\mathbf{141}X$*, 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* $N = \{1_a, 1_b, 2_b, 3_a, 4_a, 4_b\}$*.*

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 7.9. *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 $T(E)$ satisying 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; $N = \{1_a, 1_b, 2_b, 3_a, 4_a, 4_b\}$, these are the (solid) nodes in Figure 4.4H 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 7.9*) 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_b, and node n_b with parents u'_a and v'_b. We also have step $X\mathbf{u}\mathbf{v}X\mathbf{u}'\mathbf{v}'X \to X\mathbf{u}\mathbf{v}X\mathbf{u}'\mathbf{n}\mathbf{v}X\mathbf{u}'\mathbf{v}'X$ in the corresponding connection evolution E, where the connections from $\mathbf{v}$ to $\mathbf{u}'$ (inclusive) are duplicated. Note that subword $X\mathbf{u}\mathbf{v}X$ represents somatic connections across the region containing the breakpoint n_a, and $X\mathbf{u}'\mathbf{v}'X$ similarly covers breakpoint n_b.

We consider changes to the parts $X\mathbf{uv}X$ and $X\mathbf{u'v'}X$ of word $X\mathbf{uv}X\mathbf{u'v'}X$ when symbol **1** is introduced in the induced evolution separately.

Case Ia: If we have word $X\mathbf{uv}X$ after symbol **1** has been introduced into E', then by Definition 7.8, $u_a, v_b \notin N$. We then find we have evolution step $X\mathbf{uv}X \rightarrow X\mathbf{uv}X\mathbf{nv}X$ in E' and so symbol **n** is not adjacent and 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 $X\mathbf{u1v}X$ after symbol **1** has been introduced, then by Definition 7.8, $u_a, v_b \in N$. That is, the parents of n_a are in N. Now we have two possibilities. Firstly, we can have evolution step $X\mathbf{u1v}X \rightarrow X\mathbf{u1v}X\mathbf{nv}X$ 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 7.8, $n_a \notin N$. Secondly, we can have evolution step $X\mathbf{u1v}X \rightarrow X\mathbf{u1v}X\mathbf{n1v}X$, where the somatic connection **1** is duplicated. In this case we find symbol **n** is adjacent and left of symbol **1** so by Definition 7.8, $n_a \in N$. 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 $X\mathbf{u'v'}X$, 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 $X\mathbf{uv}X \rightarrow X\mathbf{unv}X$ in E. Then if we have corresponding step $X\mathbf{uv}X \rightarrow X\mathbf{unv}X$ in induced evolution E' we find that $u_a, v_b \notin N$ by Definition 7.8 and both $n_a, n_b \notin N$.

Alternatively we may find that we have a step of the form $X\mathbf{u1v}X \to X$ in E'. Then parental nodes $u_a, v_b \in N$ and we have three possibilities to consider.

We may have $X\mathbf{u1v}X \to X\mathbf{u1nv}X$. 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 $X\mathbf{u1v}X \to X\mathbf{un1v}X$. 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 $X\mathbf{u1v}X \to X\mathbf{u1n1v}X$. 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 7.10. *For any evolution E' induced from E, let N be the corresponding* 1*-nodeset obtained from Lemma 7.9. 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. type 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 7.9. *Consider again the original 2d-tree $T(E)$ in Figure 7.3B, where E is the evolution in Figure 7.3E. We wish to construct and the major graph $T_{maj}(E')$ (of induced evolution E') given in Figure 7.3H 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 7.3H. 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 b type by Lemma 7.10iii. This was 0_b, which is now mapped to 1_a, so we select edge $1_a \to 2_a$. Node $2_b \in N$, so we select the parental edge of same node type b. This was also 0_b, so we select the edge from mapped node $1_a \to 2_b$. By Lemma 7.10v, 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 $T(E)$; $2_a \to 3_b$. Now $4_a, 4_b \in N$ so we select the edges from parental nodes of same type; $1_b \to 4_a$ and $2_b \to 4_b$ parental node 2_b. Nodes $5_a, 5_b$ are adjacent to N so we select its parent edges of opposite type; $2_b \to 5_a$ and $3_a \to 5_b$.*

Observe that the differences between $T(E)$ and $T_{maj}(E')$ are a form of subtree prune and graft operations [98]; 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 7.10*)

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 $X\mathbf{uv}X \to X\mathbf{uv}X\mathbf{nv}X$ 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 $X\mathbf{u1v}X \to X\mathbf{u1v}X\mathbf{n1v}X$ in E'. Then the connection symbol $\mathbf{1}$ is duplicated and breakpoint n_a occurs between connections $\mathbf{u}$ and $\mathbf{1}$. 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; $u_a \to n_a$, 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 $X\mathbf{uv}X \to X\mathbf{uv}X\mathbf{nv}X$ in E that becomes $X\mathbf{u1v}X \to X\mathbf{u1v}X\mathbf{nv}X$ 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; $v_b \to n_a$, 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 $X\mathbf{uv}X \to X\mathbf{uv}X\mathbf{nv}X$ in E that becomes $X\mathbf{uv}X \to X\mathbf{uv}X\mathbf{nv}X$ 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 $X\mathbf{uv}X \to X\mathbf{unv}X$

in E. The corresponding step in the induced evolution E' takes one of four forms. Firstly, if $X\mathbf{u1v}X \to X\mathbf{u1n1v}X$ in E', then $\mathbf{n}$ is adjacent to $\mathbf{1}$ on both sides, so $n_a, n_b \in N$. Note that the n^{th} TD has duplicated symbol $\mathbf{1}$, so we do not have a fence in $T_{maj}(E')$. Secondly, if we have $X\mathbf{u1v}X \to X\mathbf{u1nv}X$ in E', then $n_a \notin N$ and $n_b \in N$. Note that $\mathbf{n}$ has not duplicated the symbol $\mathbf{1}$ and we still have a fence. Thirdly, the evolution $X\mathbf{u1v}X \to Xu\mathbf{n1v}X$ in E' similarly preserves the fence, with $n_a \in N$ and $n_b \notin N$. Finally, if we have $X\mathbf{uv}X \to X\mathbf{unv}X$ 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 $X \to \mathbf{n}X$ or $X \to X\mathbf{n}$ in connection evolution E. Consider first the step $X \to nX$. Note that n must represent a fence because there are no symbols to the left of $\mathbf{n}$ 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 can have corresponding step $\mathbf{1}X \to \mathbf{n1}X$ in E'. In this case, from Definition 7.8 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 $\mathbf{n}$ 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 $\mathbf{n}$ 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 $\mathbf{1}X \to \mathbf{1n}X$ is similar, resulting in a map from 0_b to 1_a.

For the third choice, the step becomes $\mathbf{1}X \rightarrow \mathbf{1n1}X$ in E'. We then find that $n_a, n_b \in N$ by Definition 7.8 as $\mathbf{n}$ is adjacent to $\mathbf{1}$ 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_a to 1_b for a consistent correspondence.

The argument using step $X \rightarrow X\mathbf{n}$ 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 7.10. We can then calculate the number of TD evolutions associated with each E' from Theorem 4.2. 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.

7.5.3 β-trees

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

Definition 7.9. *A* β-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 7.6A,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 $T_{maj}(T)$ of a β-tree T is the graph obtained when the minor edges are removed.

Secondly, we generalize the notion of 1-nodesets.
Definition 7.10. *A* β-subtree τ *of a* β*-tree* T *is a subset of nodes from* T *such that:*

i) The two root nodes are in τ*.*

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

iii) If a node is in τ*, both parental nodes are also in* τ*.*
Example 7.10. *Consider Figure 7.6A,B. Here the original* β*-trees are in Figures 7.6A,Bi.*

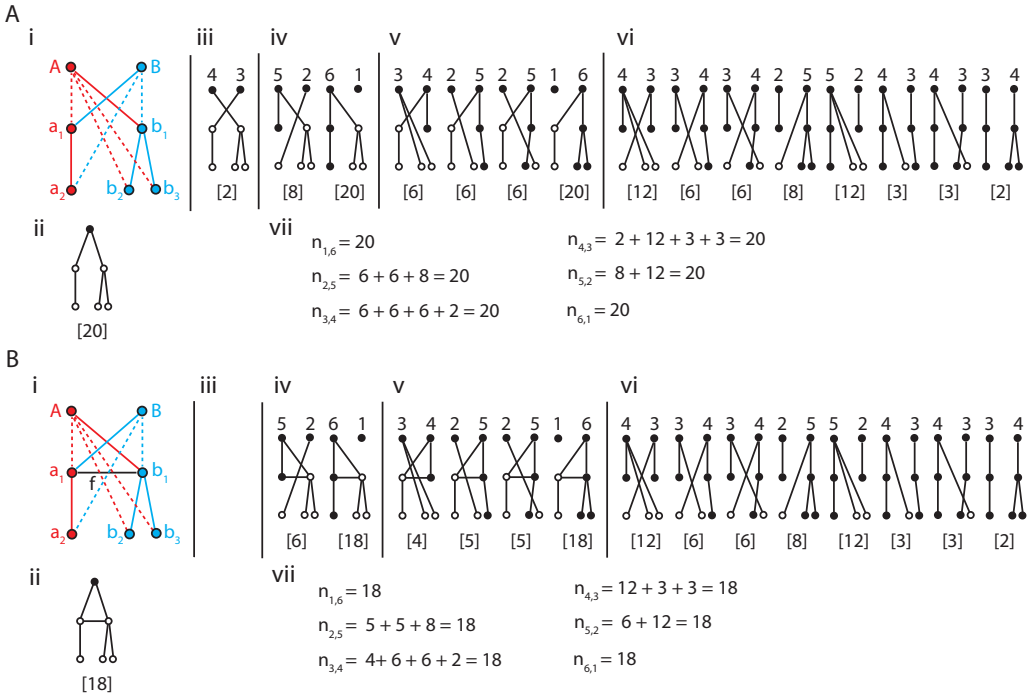

Figure 7.6: Full sets of tree operations. In A) we have a fenceless structure, in B) we have the same structure with a fence f. i) The full 2d-trees; blue are type b nodes or edges, red are type a nodes or edges. Solid lines are major edges, dashed lines are minor edges. Black edges are fences. ii) The major graph when nodes A and B are contracted. iii)-vi) Major graphs corresponding to β-subtrees indicated by blackened nodes. β-subtrees are partitioned into subgraphs τ_A connected to node A and τ_B connected to node B. iii) Major graphs when both $\tau_A, \tau_B = \phi$ are empty. iv) Major graphs when $\tau_A = \phi$ and $\tau_B \neq \phi$. v) The trees when $\tau_A \neq \phi$ and $\tau_B = \phi$. vi) Major graphs when $\tau_A, \tau_B \neq \phi$. vii) The total $n_{r,7-r}$ of combinatorial terms of trees with r nodes in component connected to node A.

The β-subtrees are indicated in Figure 7.6A,B iii-iv by the solid nodes. Note that the two roots are always in τ. These are the parents of the fence f in Figure 7.6B and so in agreement with Definition 7.10ii we find that at least one of the two nodes a_1 and b_1 bridged by f lies in τ.

The β-subtree τ of a β-tree T can be used to define a modified major graph, analogously to the construction of $T_{maj}(E')$ in Lemma 7.10, as follows:

Definition 7.11. *For a β-tree T and β-subtree τ, 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 τ 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) τ, select the parental edge of opposite type b (resp. type a).

iv) If any node is neither a member of, or immediately adjacent to, τ, 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 τ.

Note that this definition differs from Lemma 7.10 in one important way. By Definition 7.10, any β-subtree contains both root nodes. By Definition 7.11v, any fence between the two root nodes is not selected in $T(\tau)$. However, we find by Lemma 7.10ii 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 β-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 β-subtree τ:

  - $\overline{T}(\tau)$ 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 7.2.

- $\mathcal{S}$ denotes the set of valid β-subtrees according to Definition 7.10.

- ε denotes the trivial β-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 $T(\tau)$.

  - τ_x denotes the restriction of τ to x and its descendants.

  - $\mathcal{S}_x$ denotes the set of possible subsets τ_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 overline added, such as $\overline{c}_x(\tau)$, are the corresponding terms using $\overline{T}(\tau)$ 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 7.11. *In Figure 7.6Bii we have a β-tree. The graphs in each of Figures 7.6Biv-vi are the possible major graphs $T(\tau)$, where each β-subtree τ 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 β-tree in Figure 7.6Bii has two daughter branches with one node each, so we write $c_{b_1}(\varepsilon) = \binom{2}{1} = 2$. We trivially have $c_{b_2}(\varepsilon) = c_{b_3}(\varepsilon) = 1$ for leaf nodes b_2 and b_3; the descendants of b_1, so can also write $C_{b_1}(\varepsilon) = c_{b_1}(\varepsilon) \cdot c_{b_2}(\varepsilon) \cdot c_{b_3}(\varepsilon) = 2$.*

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

Theorem 7.4. *Let T be any β-tree with N nodes, and $\mathcal{S}$ the corresponding set of β-subtrees. Then for any $r \in \{1, 2, ..., N-1\}$ we have:*

$$\sum_{\{\tau \in \mathcal{S}: N_A(\tau) = r\}} C(\tau) = \overline{C}(\varepsilon) \tag{11}$$

Example 7.12. *We see in Figure 7.6Bii the major graph $\overline{T}(\varepsilon)$, 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 $\overline{c}_f(\varepsilon) = (\binom{5}{2} - 1) = 9$, by Theorem 7.2. 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 $\overline{c}_{b_1}(\varepsilon) = \binom{2}{1} = 2$. All other nodes give coefficient 1. Together we get the factor $\overline{C}(\varepsilon) = \overline{c}_f(\varepsilon) \cdot \overline{c}_{b_1}(\varepsilon) \cdot 1 = 18$, the number in square brackets 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 7.6Biv, which has combinatorial term $C(\tau_1) = 6$, and the fifth graph in Figure 7.6Bvi, 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 (11) for $r = 5$.*

Proof. (*of Theorem 7.4*) We prove this by induction on the number of nodes in the β-tree.

Induction Initial Case

Firstly consider β-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 β-trees depending on whether A and B are linked by a fence. Now any subtree from $\mathcal{S}$ must contain the two roots by Definition 7.10i, so there is only the one subtree $\varepsilon = \{A, B\}$ to consider. If there is a fence between A and B then by Definition 7.11v, the fence does not belong to $T(\varepsilon)$. Thus for both cases $T(\varepsilon)$ contains both root nodes and no edges. Now the only value that $r \in 1, ..., N - 1 = 1$ can take is 1 and $N_A(\tau) = r = 1$; the number of nodes attached to node A is 1. Now from Theorem 7.2, the combinatorial term associated with each node A and B is 1. Furthermore, the contracted tree $\overline{T}(\varepsilon)$ is a single node, which similarly has a combinatorial term of 1. We then find that:

$$\sum_{\{\tau \in \mathcal{S}: N_A(\tau) = r\}} C(\tau) = 1.1 = 1 = \overline{C}(\varepsilon)$$

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

Inductive Assumption

Next we make the inductive hypothesis that the theorem is true for all β-trees with $N \leq K - 1$ nodes, for some $K > 2$. We now consider a β-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 7.7Ai.

Note also that although the original β-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_a of type a descending from either root node, then when any β-subtree τ not including x_a (such as ε) is used, we find x_a is attached to B in $T(\tau)$ by Definition 7.11iii. Also, for any β-subtree τ including x_a (such as the entire nodeset from T), we find x_a is attached to A in $T(\tau)$ by Definition 7.11ii. When the two roots are contracted in $\overline{T}(\varepsilon)$ we find node x_a attached to the single root. Thus the choice of which root to use as the major parent of x_a has no affect on the validity of Equation (11) and we take the root B as stated. The argument for daughter node x_b is similar. This is equivalent to assuming $T = T(\varepsilon)$.

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 7.7Ai. The generalization is relatively straightforward (see comment at end of proof).

Now we require a sum over β-subtrees τ 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 7.7Ai) 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 7.7Aii. 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:

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

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

The left hand side of Equation (11) can then be split into sums across the four branches as follows:

$$\sum_{\substack{\{\tau \in \mathcal{S}: \\ n_A(\tau) = r\}}} C(\tau) = \sum_{\substack{\{r_a, r_b, r_f: \\ r_a + r_b + r_f \\ = r-1\}}} c_A(\tau) c_B(\tau) \cdot \sum_{\substack{\{\tau_a \in \mathcal{S}_a: \\ N_a(\tau_a) = r_a\}}} C_a(\tau_a) \cdot \sum_{\substack{\{\tau_b \in \mathcal{S}_b: \\ N_b(\tau_b) = r_b\}}} C_b(\tau_b) \cdot \sum_{\substack{\{\tau_{a'} \in \mathcal{S}_{a'}, \tau_{b'} \in \mathcal{S}_{b'}: \\ N_f(\{\tau_{a'}, \tau_{b'}\}) = r_f \\ \sim(\tau_{a'}, \tau_{b'} = \phi)\}}} C_f(\{\tau_{a'}, \tau_{b'}\}) \tag{12}$$

Now the sum is restricted to β-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 7.2. Thus we find that we associate node A with the multinomial coefficient:

$$c_A(\tau) = \binom{r-1}{r_a, r_b, r_f} \tag{13}$$

We similarly find we have:

$$c_B(\tau) = \binom{K-r-1}{n_a - r_a, n_b - r_b, n_f - r_f} \tag{14}$$

We thus have expressions for two terms in Equation (12). To calculate the remaining terms we show that each branch corresponds to a smaller β-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 β-tree T (represented in Figure 7.7Ai), consider the β-tree in Figure 7.7Bi which we obtain by removing all branches except the branch containing a, removing edge $B \to a$ and contracting nodes A to a together. We call the resulting β-tree T'. The corresponding major graphs for T and T' are represented in Figures 7.7A,Bii.

Now every β-subtree τ' of T' can be written as $\tau' = \{\tau_a, B\}$ for some $\tau_a \in \mathcal{S}_a$. 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 β-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 7.11ii, so $r_a > 0$. Thus the only case with $r_a = 0$ is $\tau_a = \phi$. For this case we note that $C_a(\tau_a) = \overline{C}_a(\varepsilon)$, and the following equation holds true.

$$\sum_{\substack{\{\tau_a \in \mathcal{S}_a: \\ N_a(\tau_a) = r_a\}}} C_a(\tau_a) = \overline{C}_a(\varepsilon) \tag{15}$$

Case Ib ($\tau_a \neq \phi$) We next verify Equation (15) for values $r_a \neq 0$.

Now for $\tau_a \neq \phi$ we have well defined β-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 (11) to T' and hence derive Equation (15) 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:

$$\sum_{\substack{\{\tau_a \in \mathcal{S}_b: \\ N_b(\tau_b) = r_b\}}} C_b(\tau_b) = \overline{C}_b(\varepsilon) \tag{16}$$

Case III: Dealing with Daughter Fences

We are interested in the remaining combinatorial term $C_f(\{\tau_{a'}, \tau_{b'}\})$ from Equation (12) 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 β-tree T' as the restriction of T to these two branches (see Figure 7.7Ci). If we also remove the fence f, we get the β-tree in Figure 7.7Di, 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 7.6A we see a β-tree with a fence and in 7.6B we see the same β-tree with the fence removed.

The combinatorial terms (in square brackets below each graph in Figures 7.6A,Biii-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 7.7Di 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 (11) is valid for T'' (Figure 7.7D). This gives us:

$$\sum_{\substack{\{\tau_{a'} \in \mathcal{S}_{a'} \\ \tau_{b'} \in \mathcal{S}_{b'} \\ N_f(\{\tau_{a'}, \tau_{b'}\}) = r_f\}}} C''(\tau_{a'}, \tau_{b'}) = \overline{C}''_{a'}(\varepsilon) \cdot \overline{C}''_{b'}(\varepsilon) \cdot \binom{n_{a'} + n_{b'}}{n_{a'}} = \overline{C}_{a'}(\varepsilon) \cdot \overline{C}_{b'}(\varepsilon) \cdot \binom{n_{a'} + n_{b'}}{n_{a'}} \quad (17)$$

Here we have used the fact that the components $\overline{C}''_{a'}(\varepsilon)$ and $\overline{C}''_{b'}(\varepsilon)$ derived from the β-tree T'' (corresponding to the two triangles in Figure 7.7Diii) are identical to the components $\overline{C}_{a'}(\varepsilon)$ and $\overline{C}_{b'}(\varepsilon)$ derived from the original β-tree T (see Figure 7.7Aiii). The combinatorial term $\binom{n_{a'}+n_{b'}}{n_{a'}}$ 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 7.7Diii), and we then apply Theorem 7.2.

Now we want the corresponding sum to Equation (17) 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 7.10v 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:

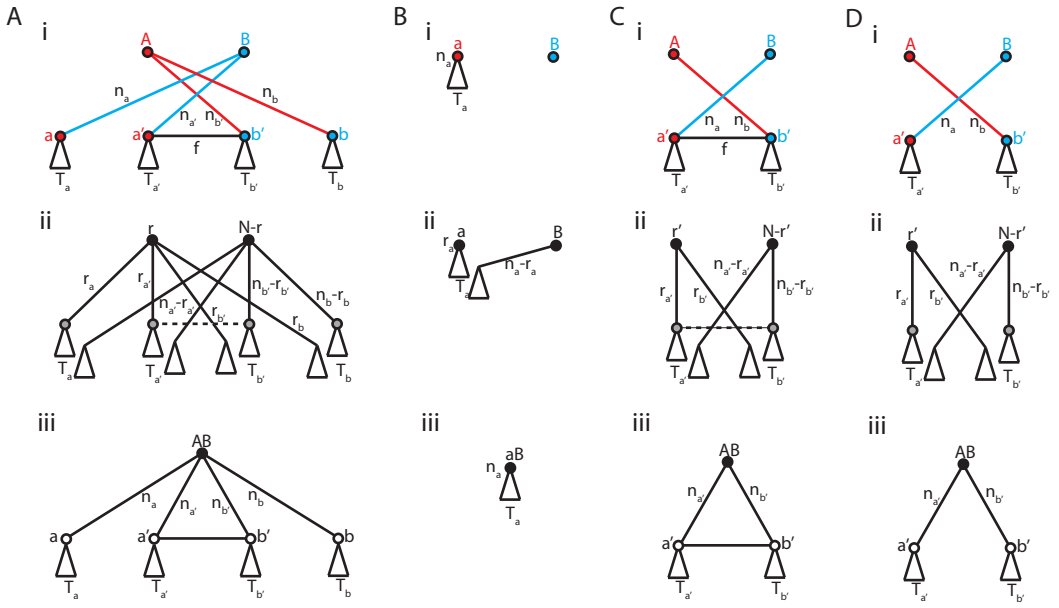

Figure 7.7: A) The general form of a 2d-tree. Triangles indicate a 2d-tree substructure. Dashed lines indicate possible presence of a fence. B) Reduction to a single branch. C) Reduction to a descending fence. D) The graphs of C with the fence removed. i) 2d-trees. ii) Trees $T(\tau)$. iii) Trees $\overline{T(\tau)}$ after root node contraction.

$$\sum_{\substack{\{\tau_{a'}\neq\phi\\ \tau_{b'}\neq\phi\\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C'(\{\tau_{a'},\tau_{b'}\}) = \sum_{\substack{\{\tau_{a'}\neq\phi\\ \tau_{b'}\neq\phi\\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C''(\{\tau_{a'},\tau_{b'}\}) \tag{18}$$

An example of this can be seen in Figure 7.6A,Bvi, where the combinatorial terms (in square brackets) are equal between the two groups.

Now from Equation (12) 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 $r_f = r_{a'} + r_{b'}$ 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 (15) corresponding to trees T' and T'' are equal for all values of r_f. For the

remaining three cases, where at lease 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 III ii ($\tau_{a'}, \tau_{b'} = \phi$ and $r_f = n_{b'}$) The β-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 β-subtree τ by Definition 7.10ii. We thus find that although $\varepsilon \in \mathcal{S}''$ in a valid β-subtree for T'', $\varepsilon \notin \mathcal{S}'$ is not a valid β-subtree for T' (Figure 7.6A,Biii 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 7.7Di), with major edges that match the original β-tree T, and we obtain combinatorial term $C''_f(\varepsilon) = \overline{C}_{a'}(\varepsilon) \cdot \overline{C}_{b'}(\varepsilon)$. We thus find:

$$\sum_{\substack{\{\tau_{a'} \neq \phi \\ \tau_{b'} \neq \phi \\ N_f(\{\tau_{a'}, \tau_{b'}\}) = r_f\}}} C'(\{\tau_{a'}, \tau_{b'}\}) = \sum_{\substack{\{\tau_{a'} \neq \phi \\ \tau_{b'} \neq \phi \\ N_f(\{\tau_{a'}, \tau_{b'}\}) = r_f\}}} C''(\tau_{a'}, \tau_{b'}) - \overline{C}_{a'}(\varepsilon) \cdot \overline{C}_{b'}(\varepsilon) \qquad (19)$$

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

Case II above, and we find that:

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

$$\overline{C}_{a'}(\varepsilon)\cdot \sum_{\substack{\{\tau_{b'}\neq\phi \\ N_{b'}(\tau_{b'})=r_f\}}} C''_{b'}(\tau_{b'})\cdot \binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}} = \tag{20}$$

$$\overline{C}_{a'}(\varepsilon)\cdot \overline{C}_{b'}(\varepsilon)\cdot \binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}}$$

Here we pick up a combinatorial factor $\binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}}$ 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; $\binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}}-1$ in $T'(\tau)$. That is:

$$\frac{C''(\{\tau_{a'},\tau_{b'}\})}{\binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}}} = \frac{C'(\{\tau_{a'},\tau_{b'}\})}{\binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}}-1}.$$

Substituting this into Equation (20) gives us:

$$\sum_{\substack{\{\tau_{a'}=\phi \\ \tau_{b'}\neq\phi \\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C'(\{\tau_{a'},\tau_{b'}\}) = \overline{C}_{a'}(\varepsilon)\cdot \overline{C}_{b'}(\varepsilon)\cdot \left(\binom{n_{b'}-r_{b'}+n_{a'}}{n_{a'}}-1\right) \tag{21}$$

Then subtracting Equation (21) from Equation (20) reveals the same constant difference

observed in the previous case:

$$\sum_{\substack{\{\tau_{a'}=\phi \\ \tau_{b'}\neq\phi \\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C'(\{\tau_{a'},\tau_{b'}\}) = \sum_{\substack{\{\tau_{a'}=\phi \\ \tau_{b'}\neq\phi \\ N_A=r_f+1\}}} C''(\{\tau_{a'},\tau_{b'}\}) - \overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon) \tag{22}$$

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:

$$\sum_{\substack{\{\tau_{a'}\neq\phi \\ \tau_{b'}=\phi \\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C'(\{\tau_{a'},\tau_{b'}\}) = \sum_{\substack{\{\tau_{a'}\neq\phi \\ \tau_{b'}=\phi \\ N_A=r_f+1\}}} C''(\{\tau_{a'},\tau_{b'}\}) - \overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon) \tag{23}$$

Thus in all three cases (III i-iii) the difference between the tree with and without the fence is $\overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon)$. Furthermore, for any single value of r_f, one of these three cases is applicable. We thus find, using Equations (19), (22) and (23) with (18) that:

$$\sum_{\substack{\{\tau_{a'}\in\mathcal{S}_{a'} \\ \tau_{b'}\in\mathcal{S}_{b'} \\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C'(\{\tau_{a'},\tau_{b'}\}) = \sum_{\substack{\{\tau_{a'}\in\mathcal{S}_{a'} \\ \tau_{b'}\in\mathcal{S}_{b'} \\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C''(\{\tau_{a'},\tau_{b'}\}) - \overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon)$$

Then substituting this into Equation (17) gives us:

$$\begin{aligned}\sum_{\substack{\{\tau_{a'}\in\mathcal{S}_{a'}\\ \tau_{b'}\in\mathcal{S}_{b'}\\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C'(\{\tau_{a'},\tau_{b'}\}) &= \overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon)\cdot\tbinom{n_{a'}+n_{b'}}{n_{a'}} - \overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon) \\ &= \overline{C}_{a'}(\varepsilon)\cdot\overline{C}_{b'}(\varepsilon)\cdot(\tbinom{n_{a'}+n_{b'}}{n_{a'}} - 1)\end{aligned} \tag{24}$$

Now $C'(\{\tau_{a'},\tau_{b'}\})$ matches the combinatorial term from the fence and its descendants, $C_f(\{\tau_{a'},\tau_{b'}\})$. Furthermore $\overline{C}_{a'}(\varepsilon)$, $\overline{C}_{b'}(\varepsilon)$ and $(\tbinom{n_{a'}+n_{b'}}{n_{a'}} - 1)$ match the terms in the graph $\overline{T}$ obtained from the branch containing node a', the branch containing node b', and fence f (by Theorem 7.2), and so equal $\overline{C}'(\varepsilon)$. Thus we find that:

$$\sum_{\substack{\{\tau_{a'}\in\mathcal{S}_{a}\\ \tau_{b'}\in\mathcal{S}_{b'}\\ N_f(\{\tau_{a'},\tau_{b'}\})=r_f\}}} C_f(\{\tau_{a'},\tau_{b'}\}) = \overline{C}_f(\varepsilon) \tag{25}$$

Completing the Induction

Thus finally substituting Equations (13), (14), (15), (16) and (25) into Equation (12) we find that we have:

$$\sum_{\{\tau\in\mathcal{S}:N_A(\tau)=r\}} C(\tau) = \overline{C}_a(\varepsilon)\cdot\overline{C}_b(\varepsilon)\cdot\overline{C}_f(\varepsilon)\cdot\sum_{\substack{\{r_a,r_b,r_f:\\ r_a+r_b+r_f\\ =r-1\}}}\tbinom{r-1}{r_a,r_b,r_f}\tbinom{K-r-1}{n_a-r_a,n_b-r_b,n_f-r_f}$$

Then applying the multinomial version of the Vandermonde identity [116] results in:

$$\sum_{\{\tau \in \mathcal{S}: N_A(\tau)=r\}} C(\tau) = \overline{C}_a(\varepsilon) \cdot \overline{C}_b(\varepsilon) \cdot \overline{C}_f(\varepsilon) \cdot \binom{K-2}{n_a, n_b, n_f}$$

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 7.7Aiii we see two branches containing nodes n_a and n_b, these have combinatorial terms equal to $\overline{C}_a(\varepsilon)$ and $\overline{C}_b(\varepsilon)$. We also have fence f bridging the two branches containing nodes $n_{a'}$ and $n_{b'}$. Application of Theorem 4.2 to f for the root contracted graph $\overline{T}(\varepsilon)$ (given in Figure 7.7Aiii) returns precisely the term $\binom{K-2}{n_a, n_b, n_f}$. We thus find that we have all the coefficients of $C(\overline{T})$ and Equation (11) 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. □

7.5.4 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 7.3) 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 τ such that the major graph $T_{maj}(E')$ corresponding to E' is obtained from Lemma 7.10. By Definition 7.10, τ is also a β-subtree, and we have an induced major graph $T(\tau)$. We also know from Theorem 7.4

that for any induced major graph $T(\tau)$ we can sum $C(\tau)$ over the β-subtrees τ to obtain $\overline{C}(\varepsilon)$.

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 7.10vi.

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 7.2, the number of TD evolutions associated with $T_{maj}(E')$ is given by $(\binom{2n}{r} - 1)C(\tau)$. Then, using Theorem 7.4, the total number of TD evolutions induced by E is given by:

$$\begin{aligned}
\mathcal{N}(\mathcal{E}(E)) = \sum_{r=1}^{2n-1} \sum_{\{\tau \in \mathcal{S}: N_A(\tau) = r\}} (\tbinom{2n}{r} - 1)C(\tau) &= \sum_{r=1}^{2n-1} (\tbinom{2n}{r} - 1) \cdot \sum_{\{\tau \in \mathcal{S}: N_A(\tau) = r\}} C(\tau) \\
&= \sum_{r=1}^{2n-1} (\tbinom{2n}{r} - 1) \cdot \overline{C}(\varepsilon) \\
&= \overline{C}(\varepsilon) \cdot ((\sum_{r=0}^{2n} \tbinom{2n}{r} - 2) - (2n-1)) \\
&= \overline{C}(\varepsilon).(2^{2n} - (2n+1))
\end{aligned}$$

Now by Theorem 7.2 $\overline{C}(\varepsilon)$ is the number of TD evolutions associated with connection evolution E. Thus we have:

$$\mathcal{N}(\mathcal{E}(E)) = \mathcal{N}(E) \cdot (4^n - (2n+1)) \tag{26}$$

Furthermore, by Corollary 7.4, the set of induced evolutions from $\mathcal{E}_{n-1}$ gives rise to a disjoint union of $\mathcal{E}_n$. Thus summing Equation (26) 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. □

7.6 Conclusions

We have seen in Table 7.1 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 attempt to computationally explore this space in its entirety. This will make it difficult to compare any observations, such as copy number information [72], [24], to the set of possible TD structures to determine evolutions that may explain the data.

The methods utilized in this work largely parallel those used to examine breakage fusion cycles [24]. These are a distinct form of rearrangement which suggest there may be a more general space in which rearrangements operate and these methods apply. Given that 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 [24].

The methods above and in [24] 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.