November 2020

Discrete Models and Algorithms for Analyzing DNA Rearrangements

Jasper Braun
University of South Florida

Discrete Models and Algorithms for Analyzing DNA Rearrangements

by

Jasper Braun

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Mathematics
with a concentration in Pure and Applied Mathematics
Department of Mathematics & Statistics
College of Arts and Sciences
University of South Florida

Date of Approval:
November 11, 2020

Keywords: Ordered graphs, Edge labellings, Ciliate genomics, Sequence alignment algorithms

Dedication

To my wife Stephanie and our unborn child.

May the force be with you.

Acknowledgements

First and foremost, I would like to thank my advisors Nataša Jonoska, Masahico Saito and our collaborator Laura Landweber. Drs. Jonoska and Saito have been with me for the entire journey, have given me perspective and guidance, where I needed it, and helped me find my own path. Dr. Landweber, although from afar, has been an invaluable beacon of support and advice throughout. Much help and friendship also came from the various past and present members of the Math-Bio research group at USF and Landweber lab at Columbia University, including but not limited to Lukas, Rafik, Jonathan, Denys, Daniel, Yi, and Rick. I also owe thanks to the dissertation committee members Margaret Park, Brendan Nagle, and Dmytro Savchuk for setting aside a significant amount of their time to accommodate my defense. Many professors in the USF math department earned my gratitude for providing me with the tools I needed to accomplish this work. There are too many to name them all, but some stand out. Milé Krajčevski, thank you for igniting my desire to pursue an advanced degree in math with your colorful teaching methods. Brian Curtin, thank you for your support when I had difficulties, and for sacrificing many a lunch break for answering my endless lists of questions. Brendan Nagle, thank you for an incredible year of Combinatorics; I will never forget the fun we had, and I truly believe you deserve an award for teaching.

I would never have gone through graduate school without the support and motivation my wife provided me with. My family, both in-laws and out-laws, also deserve much credit for making my studies possible. Last but not least, I want to thank the NIH and the NSF, whose financial support allowed me to focus on my research and contribute without distractions. This work was funded by NIH grant GM109459 and NSF grant 1800443.

Table of Contents

List of Tables

List of Figures

Abstract

In this work, language and tools are introduced, which model many-to-many mappings that comprise DNA rearrangements in nature. Existing theoretical models and data processing methods depend on the premise that DNA segments in the rearrangement precursor are in a clear one-to-one correspondence with their destinations in the recombined product. However, ambiguities in the rearrangement maps obtained from the ciliate species *Oxytricha trifallax* violate this assumption demonstrating a necessity for the adaptation of theory and practice.

In order to take into account the ambiguities in the rearrangement maps, generalizations of existing recombination models are proposed. Edges in an ordered graph model the relative positions of precursor DNA segments and their labels indicate the orientations and destinations in the product genome. Properties of these structures are introduced with the intention to narrow down the space of possible rearrangements that can be described by the model to include only the types of complexities that appear in nature. The various subspaces of rearrangements defined by these properties are explored via a series of combinatorial counting results. The new model is applied to sequencing data of *O. trifallax* to assess the extent to which these properties describe the rearrangements this organism undergoes.

To reduce the filtering of data, an algorithm which annotates the rearranging segments in precursor and product genomes without discarding ambiguities is presented. Furthermore, a generalization of the notion of scrambling that can be applied to such ambiguous rearrangement maps is defined. An algorithm that detects the generalized scrambling property in ambiguous rearrangement maps is also presented. Next, a computational tool implementing the two algorithms is introduced and tested. The annotation algorithm involves a step by which gapped sequence alignments are obtained from ungapped sequence alignments in an efficient and controlled manner. This method of combining ungapped alignments gives rise to another algorithm that can be applied to the more general problem of efficiently detecting gapped sequence alignments, which is implemented and tested in this work.

Chapter 1
Introduction

1.1 Ambiguous Rearrangements

The types of rearrangements considered in this work involve the correspondence of segments between two DNA sequences. Previous analyses of this kind of DNA rearrangements in the organism *Oxytricha trifallax*, assumed the appearance of a simple bijective correspondence in experimental sequencing data [11, 12, 16]. The analyzed data was then forced to conform to the assumption through processing steps resulting in the removal of sections of the data. Such systematic exclusion of information may introduce bias, and inevitably leads to a loss of characteristics encapsulated in the excluded data portion. Examination of the sequencing data used in these analyses reveals that the assumption of a bijective correspondence is not satisfied. Two regions in the precursor may correspond to the same region in the product and vice versa. Additionally, there are no obvious indicators that suggest precedence of some segments over others. In [16], sequence alignment quality measures are suggested as a means to disambiguate a mapping, but not in all ambiguous assignments of precursor to product segments, there is one with a higher sequence alignment quality. To enable comprehensive analyses of sequencing data pertaining to these types of rearrangements, this work investigates models and methods that work with the true many-to-many mappings that appear in DNA rearrangements in nature.

1.2 Biological Background of the Model Organism

Like other ciliated protozoa, *Oxytricha trifallax* has two nuclei. The **somatic nucleus** (also called **macronucleus**, or **mac**), transcribes the proteins carrying out most of the cell functions during its vegetative state. The **germline nucleus** (also called **micronucleus**, or **mic**), remains inactive for most of the life cycle of the cell. Usually, the organism reproduces asexually, but when starved, it undergoes sexual conjugation. During conjugation, the functional somatic nucleus disintegrates and reforms from the genetic material present in the otherwise inactive germline nucleus. Due to this relationship, the germline and somatic nuclei are often also named the **precursor** and **product**, respectively. The nucleotide sequences of the chromosomes of the somatic nucleus appear in the germline DNA often times broken up into multiple segments (called **Macronuclear Destined Sequences**, or **MDSs**), separated by segments of excess DNA (called **Internal Eliminated Sequences**, or **IESs**).

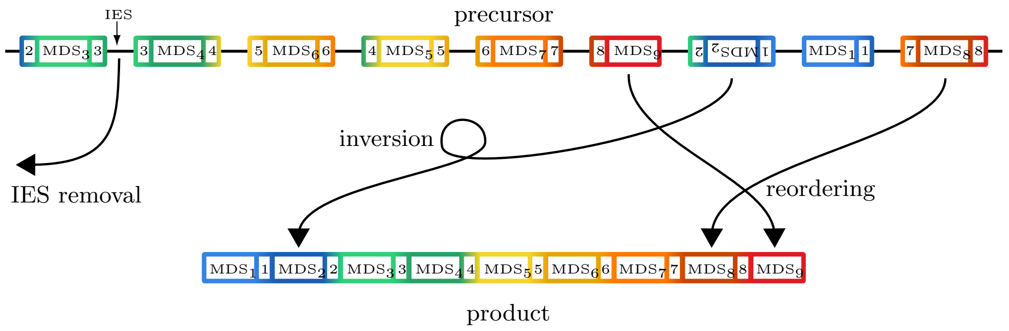


Figure 1.1: A schematic of the rearrangement of the Macronuclear Destined Sequences (MDSs) of the Actin I gene in *Oxytricha trifallax* from the precursor (top) to the product (bottom), as discovered in [38]. The regions between MDSs in the germline nucleus are called Internal Eliminated Sequences (IESs). During development of the somatic nucleus, IESs must be removed, and MDSs must be reordered and sometimes inverted. Sequences at the ends of each MDS match those at the ends of their successor in the product suggesting a guiding role of the matching repeats in the precursor during the recombination.

In [38], first evidence was presented that the MDSs of some genes appear in the germline nucleus out-of-order and in conflicting orientations when compared to their order and orientation in the somatic genome. At the top of Figure 1.1, the scrambled order of the MDSs of a gene in the germline nucleus is shown. In the article, a model of intramolecular recombination of the germline DNA to unscramble the disordered MDSs is proposed. This model is based on the alignment of short repeated sequences called **pointers** which appear in pairs in the germline DNA at the ends of MDSs that match flanking regions in the somatic nucleus. The model is consistent with the presence of circular DNA molecules that appear as a by-product of the recombination process as demonstrated in [43, 44]. Multiple papers suggest the involvement of RNA during the recombination of MDSs in *O. trifallax* as well as other ciliates [15, 20, 21, 25, 27, 31]. With the advent of next-generation sequencing technologies, the entire somatic and germline nuclei of *O. trifallax* were sequenced in [16, 28, 42] opening doors to a variety of downstream analyses. The existence of over 16,000 **nanochromosomes** in the somatic nucleus is established in [42], which is updated in [28] and extended by over 12,000 new isoforms.

Although under the simplifying assumption of the existence of a one-to-one correspondence of MDSs in rearrangments, several observations about various aspects of rearrangements in *O.* trifallax were made. High levels of scrambling of the MDSs in the germline genome were exhibited in [16], along with additional complexities, such as overlapping, or interweaving MDSs in the germline genome mapping to distinct somatic genes. Two simple patterns characterized by runs of odd, or even numbered MDSs followed by a run of even, or odd numbered MDSs, respectively, were discovered among scrambled MDS sequences by the authors of [11]. Repeated removal of these patterns was shown to resolve 96% of all scrambled sequences with no more than 4 iterations. Of the remaining scrambled cases, 22 could not be reduced to less than 3 MDSs through iterative pattern removal, but 21 of them had a third more intricate pattern in common [6].

In [6], mic loci are examined in which MDSs of some product sequences are nested between MDSs of other product sequences. Two numeric indices indicating the depth of such nesting are defined recursively. Viewing MDS and IES regions as integer intervals, the arrangement of MDSs and IESs of a mac locus on a mic locus can be expressed as the sequence $(M_1, I_1, M_2, I_2, \dots, M_N)$, where M_i are the MDS regions in their order of appearance in the mic and the I_j are the IES regions between them. Given two sequences of MDS and IES regions $A = (M_1, I_1, \dots, M_N)$, and $B = (M'_1, I'_1, \dots, M'_{N'})$ of two mac loci on the same mic locus, an IES I'_j of B is said to be **inserted** in an IES I_k of A, if $M'_j, I'_j, M'_{j+1} \subseteq I_k$. For an IES I of a mac locus on a mic locus, denote with $\mathcal{M}(I)$ the set of MDSs of other mac loci on the same mic locus which are contained in I and denote with $\mathcal{N}(I)$ the set of IESs of other mac loci on the same mic locus which are inserted in I.

Definition 1.2.1. The **insertion depth index** (**IDI**) of an IES I of a mac sequence in a mic sequence is defined by:

$$\mathrm{IDI}(I) = \begin{cases} 0, & \text{if } \mathcal{M}(I) = \emptyset \\ 1, & \text{if } \mathcal{M}(I) \neq \emptyset \text{ and } \mathcal{N}(I) = \emptyset \\ 1 + \max\limits_{I' \in \mathcal{N}(I)} \mathrm{IDI}(I'), & \text{if } \mathcal{N}(I) \neq \emptyset \end{cases}$$

The **IDI** of a set of IESs is the maximum IDI across all IESs in the set.

Figure 1.2 shows a sequence of MDSs of three different mac loci on the same mic locus. The IESs of the mac loci corresponding to the red, blue, and orange MDSs have IDI 2, 1, and 0, respectively.

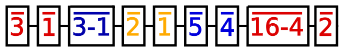


Figure 1.2: A representation of the MDSs of three mac contigs on mic sequence ctg7180000067077 as reported in [6]. The red, blue, and orange MDSs belong to mac contigs Contig6331.0, Contig9583.0, and Contig6683.0, respectively. The numbers indicate the final, linear order of the MDSs within the respective mac contigs. Multiple sequential MDSs are condensed.

In [6], IDI values as large as 4 were reported. Additionally, 8% of mac loci have a non-zero IDI and a positive correlation of the proportion of scrambled mac loci among loci with increasing IDI was reported. Finally, for over 80% of the mac loci which have at least one MDS contained in the IES of another, all the MDSs are located on the same IES.

1.3 Mathematical Models for DNA Rearrangements in Ciliates

A language theoretic approach to model the DNA recombinations was introduced by [22, 36, 39] in the form of so-called **splicing systems**. Splicing systems consist of sets of initial strings and sets of patterns which define operations that produce combinations of substrings of the original strings. These operations closely resemble the cutting and splicing processes that ciliates undergo when recombining germline MDSs to form functional somatic chromosomes. Inspired by the involvement of pointer sequences in the rearrangement, [37] propose three operations that can descramble MDS patterns observed in the genome of *O. trifallax*. With increasing evidence for the necessity of the participation of RNA molecules in the pointer-guided recombinations, a graph theoretical model was introduced by [3, 4]. In this new model, so-called (simple) **assembly graphs**, topologically model the rearrangement of MDSs in the germline DNA of *O. trifallax* as a simultaneous intramolecular recombination event. Graph vertices correspond to the alignments of matching pointer sequences and the recombination can be modelled by vertex-smoothing according to certain rules. The exact cuts and branch migrations required for the Biochemical equivalent of vertex-smoothings can be achieved with the help of RNA molecules [4]. After simultanous recombination (vertex-smoothing) of the aligned pointers (4-valent vertices), a single molecule (linear connected component) consists of the MDSs of a gene in orthodox order and orientation. The **genus range**, which is a topological structure related to the assembly graph model was investigated in [8, 10, 23]. Additionally, combinatorial concepts, such as **assembly polynomials** [3, 7, 9, 23] and **double-occurrence words (DOWs)** [5, 9, 13, 17, 24] were investigated. Chord diagrams relate in a natural way to DOWs and can be derived from assembly graphs.

Some unrelated language theoretical models were studied in [18, 26]. The latter, introduces two types of shuffling operations on a language consisting of words that resemble scrambled MDS sequences. The authors show that the vast majority of scrambling patterns appearing in the genome of *O. trifallax* can be resolved via 1 or 2 applications of these operations.

Assembly graphs, double occurrence words and chord diagrams rely heavily on a total order on the MDSs in the mic and the appearance of each pointer exactly twice. Frequently, ambiguities disrupt this assumption. Multiple MDSs in the mic may match the same region in the mac and vice versa. Repetitions of mic MDSs that match the same mac MDS imply occurrence of the pointers at the ends of the MDS more than twice. Micronuclear MDSs that intersect, or where one contains the other make defining a sensible total order on them difficult. Chapter 2 explores a mathematical model of rearrangement maps which generalizes existing models. The structure of overlaps and repeats is readily captured by the model at the cost of vastly increasing the number of theoretically possibilities which are unlikely to appear in nature. By introducing a variety of properties, and counting subsets with these properties, smaller more realistic spaces are explored.

1.4 Computational Tools for Analyzing DNA Rearrangements

Sequence alignment is a computationally intensive, but frequently necessary task in Bioinformatics. The main challenge for biological sequence alignment tools is the necessity for allowing imperfect matches. Searching large sequence databases for pairs of regions that resemble each other but are not required to be identical introduces a number of concerns. One needs to define how to rate sequence similarity, and then invent efficient algorithms that can find resembling regions in the data according to the defined similarity assessment. Decades ago, a dynamic programming algorithm by Needleman and Wunsch [34] provided an algorithm for finding the end-to-end alignment of two sequences that optimizes an alignment score which rewards identical matches and penalizes insertions, deletions and substitutions necessary to get from one sequence to the other. The sequence alignment scoring system is still used today in many applications. The Smith-Waterman algorithm is based on that of Needleman and Wunsch, but it finds sets of alignments of regions contained within two sequences with optimal alignment scores [41]. Many variants of the Needleman-Wunsch and Smith-Waterman algorithms were introduced, but their dynamic programming approach is expensive in both computational time and space. With the advent of high-throughput sequencing technologies, and the ever-increasing volumes of sequencing data, the demand for more efficient algorithms increased as well. The softwares LFASTA [35] which is based on a global sequence alignment tool [29] and the Basic Local Alignment Search Tool (BLAST) for nucleotide sequence alignments introduced in [1] and improved in [2, 14, 33, 45] are heuristic adaptations of the earlier dynamic programming algorithms. Both algorithms may sometimes miss optimal alignments but perform much faster and use less memory. In the study of DNA rearrangements in ciliates, these sequence alignment algorithms are used to find and establish the correspondence between the rearranging DNA segments in the precursor and product genomes and to detect telomeres at the end of product sequences. BLAST is the most popular tool used for the extraction of MDSs from the macronuclear and micronuclear genomes.

It was observed that further processing is required to extract the MDS rearrangements hidden among alignments detected by BLAST and make them accessible for downstream analyses. Customized scripts were used for the extraction of rearrangement maps of *O. trifallax* from alignments returned by BLAST in [11, 16]. These scripts were sufficient to obtain the presented results but are unavailable to the public, which prevents adaptation of the computation and reproduction of these results. Additionally, repeating and overlapping MDSs in the micronucleus were exluded from the analyses dismissing potentially significant information. The purpose of the software tool MIDAS [12] is to automate the process of annotating features that describe DNA rearrangement maps. While MIDAS annotates MDSs in the micronuclear and macronuclear genomes, it fails to describe the details of how the MDSs in the precursor relate to the MDSs in the product. The software provides a representation of the rearrangement maps in the form of a sequence of symbols corresponding to the MDSs in the micronucleus. Each symbol indicates the position of the corresponding macronuclear MDS relative to other macronuclear MDSs. It is unclear in which order the symbols are listed in cases where the MDSs in the mic overlap.

In Section 3.2, an algorithm is proposed which annotates features relevant to DNA rearrangements in ciliates. The current notion of scrambling is generalized and an algorithm that detects scrambling in rearrangements maps according to this new notion is discussed. A software tool called **Scrambled DNA Rearrangement Annotation Pipeline** (**SDRAP**) implementing these algorithms is presented and tested. Finally, the software is applied to annotate the DNA rearrangements of *O. trifallax* using the most recent macronuclear genome published in [28] and the results are discussed in Sections 3.2.6, and 3.2.7.

BLAST and the scoring system it uses were designed to detect and rate homology between sequences on an evolutionary scale. MDSs are regions of similarity within a precursor genome from a parent cell and a product genome in a daughter cell. Only few imperfections in their sequence similarity are expected and may arise, for example, from sequencing errors, assembly artifacts and allelic variations. For this reason BLAST was usually instructed to only return ungapped alignments (not allowing insertions and deletions to get from one sequence to the other) in previous annotation workflows for the ciliate *O. trifallax* [12, 16]. Sequence similarity can be further enforced by filtering the data by the percentage of identities across an alignment and by using different scoring parameter values (rewards for identities and penalties for mismatches). Another reason to instruct BLAST to only search for ungapped alignments is that non-scrambled consecutive MDSs that have little pointer overlap in the macronucleus and are located close to each other in the micronucleus may mistakenly combined to a single larger gapped alignment by the tool. On the other hand, the disadvantage of only considering ungapped alignments is that regions of similarity between the genomes may show up in fragments in the data inflating the numbers of MDSs considered in downstream analyses. A step that merges product segments was incorporated in the most recent software MIDAS which annotates ciliate rearrangements [11]. However, the method for merging alignments used by MIDAS combines alignments solely based on intersections of their product segments and irrespective of their locations in the micronucleus. In doing so, MIDAS combines alignments which may be distant in the micronucleus, possibly even from different micronuclear chromosomes.

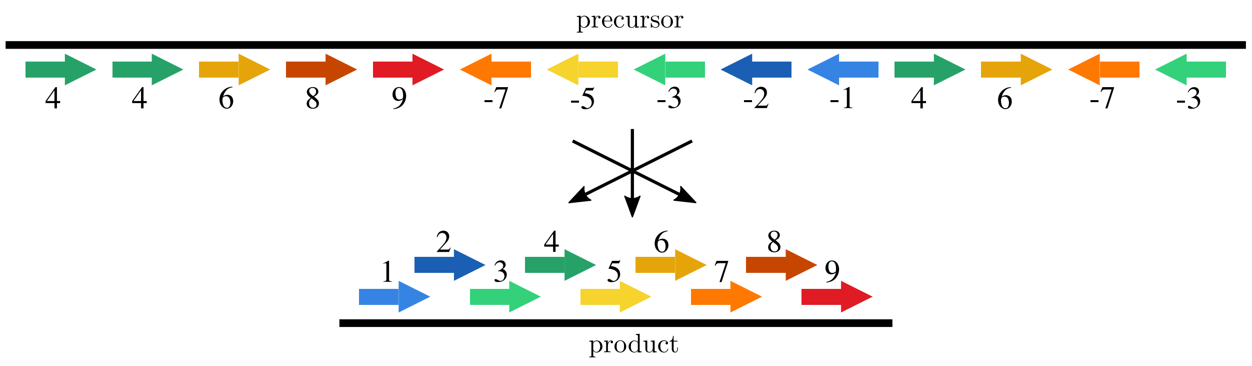
The software SDRAP introduced in Section 3.2 incorporates a technique that merges MDSs in a controlled way. Section 3.3 further develops the technique in the more general context of gapped sequence alignments. An algorithm is proposed which combines ungapped alignments to gapped alignments increasing the average alignment score and decreasing the number of alignments. The algorithm is implemented as the command-line tool named **PasteAlignments** which is compared to the gapped sequence alignment step in BLAST.

Chapter 2
Extended Model for DNA Rearrangements in Ciliates

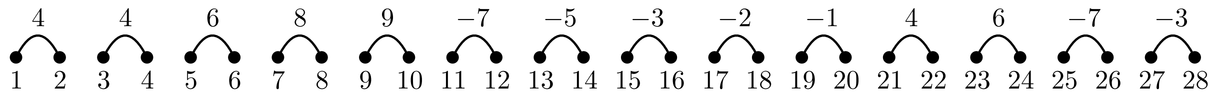
2.1 Introduction

Previous models heavily rely on each pointer occurring exactly twice in a rearrangement. This reliance on the number of pointers is based on the assumption that the data provides clear one-to-one mappings between precursor and product segments. However, independent examination of the data shows that this assumption is frequently violated. Figures 2.1(a) and 2.2(a) show two examples of rearrangement maps which are not one-to-one mappings. Both sets of alignments were obtained using BLAST+ 2.2.31 [45] with the command `blastn -task megablast -ungapped -lcase masking -word size 18 -dust no` on the micronuclear assembly from [16] (as database) and the macronuclear assembly from [42] (as query). In Figure 2.1(a), multiple segments in the precursor correspond to the same segment in the product. In Figure 2.2(a), the same region in the precursor matches multiple regions in the product. In both cases, the mac contig as telomeres (repetitive regions at the ends of chromosomes) at both ends, and the MDSs cover the region between telomeres by more than 99%. Additionally, the chromosome whose rearrangement map is depicted in Figure 2.1(a) encodes a hypothetical protein and the chromosome whose rearrangement map is depicted in Figure 2.2(a) encodes the Kelch motif, which indicates their importance. In Figure 2.2, all alignments are at least 128 base pairs long and have 100% percent identity. However, due to the ambiguities in the mappings which current models cannot handle, they were at least partially excluded from the analyses in [11, 16]. To avoid loss of information and an introduction of bias into the data by filtering out ambiguous data, more generally applicable models are required.

The model investigated in this chapter generalizes the double-occurrence word and chord diagram models and applies to arrangements where MDSs repeat and/or overlap. This model can be defined in various equivalent ways, but here it is broken down into two parts to decouple the different sources for complexity in the rearrangement. More specifically, sets of pairs of integers are considered, which mark the starting and end coordinates of MDSs in the precursor. Pairs are labelled according to the positions and orientations of the corresponding product segments in the product chromosome. For example, a pair labelled $3, -7$ matches the 3$^{\text{rd}}$ and the reverse complement of the 7$^{\text{th}}$ product segment in the product chromosome. The relative positions of the pairs corresponds to the relative positions of the MDSs in the precursor. The actual



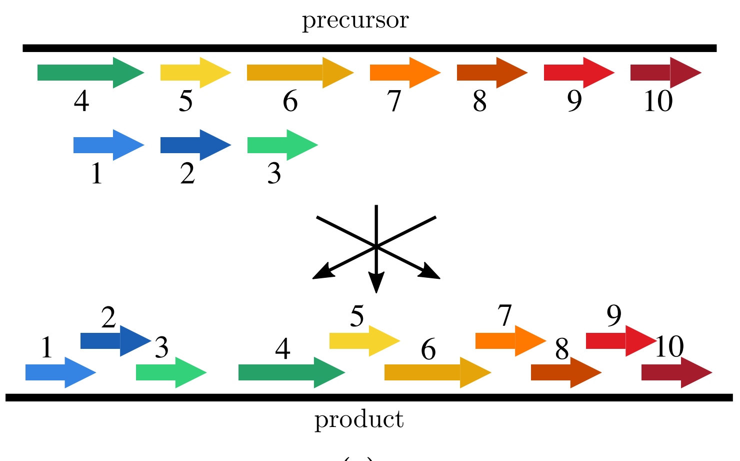
(a)



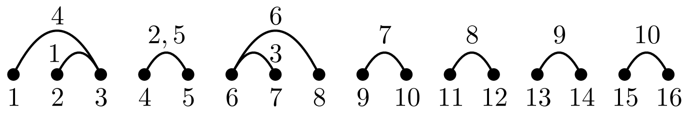
(b)

Figure 2.1: **(a):** Schematic of the rearrangement map for the MDSs of Contig14006.0 on ctg7180000069341 (not drawn to scale). Two alignments with poor sequence similarity ($< 90\%$ identity) were left out of this picture. **(b):** the set of pairs $\{\{1, 2\}, \{3, 4\}, \ldots, \{27, 28\}\}$ which represent the regions in the precursor genome which match the chromosome. Each pair has a label which indicates order and orientation of their matching counterpart in the product.

distances between and lengths of MDSs are neglected by the model. The 14 disjoint MDSs depicted in the MDS arrangement in Figure 2.1(a) corresponds to the set of 14 pairs of the integers 1 through 28 shown in Figure 2.1(b) where vertices mark the 28 numbers and edges define pairs on these numbers. Each of the vertices corresponds to an end coordinate of an MDS and each edge connects two numbers which mark end coordinates of the same MDS. In Figure 2.2(b), the set of pairs and the labels are formed the same way from the arrangement depicted in part (a). Here, the pairs $\{1, 3\}$ and $\{2, 3\}$ share an end point which translates to their corresponding MDSs 4 and 1 ending at the same coordinate. The same holds true for the pairs $\{6, 7\}$ and $\{6, 8\}$. The pair $\{4, 5\}$ is labelled by two integers 2 and 5 because it matches both the 2$^{\text{nd}}$ and the 5$^{\text{th}}$ product segment in the product chromosome. Figure 2.2(b) also exemplifies how separate consideration of sets of pairs and their labellings allows decoupling of different sources for complexity. Complexities arising from relative positions of MDSs in the precursor are generally modelled by the relative positions of the set of pairs. In this example, overlaps between MDSs are easily identified solely based on the set of pairs. Complexities arising from the relative positions of the corresponding product segments as well as the orientation in which they match these MDSs is modelled by the labels of the pairs. Note that the labelling of a single pair by multiple MDSs can be seen as a violation to this premise because it is an indication of identical positions of MDSs in the precursor. One could add a weight to each pair or consider double-edges,



(a)



(b)

Figure 2.2: **(a):** the rearrangement map for the MDSs of Contig12633.0 on ctg7180000068813 (not drawn to scale). MDSs 1 and 3 are contained in MDSs 4 and 6, respectively, in the mic. MDSs 2 and 5 occupy the same region. MDSs 3 and 4 are 42 bases apart in the product. No high-scoring alignment was detected for this region. **(b):** the set of pairs which represent the regions in the precursor genome which match the chromosome. Each pair has a label which indicates order and orientation of their matching counterparts in the product. The pair $\{4, 5\}$ has two labels as it represents the region occupied by both MDS 2 and 5. The 5' end of MDS 1 aligns with the 5' end of MDS 4 and the 3' end of MDS 6 aligns with the 3' end of MDS 3.

which was not done here.

In this chapter, pairings and labellings are formally defined. The space of possible rearrangement maps and their complexities is explored through a series of combinatorial counting results. Pairings and labellings are defined at high generality and properties are introduced whose requirement may narrow down the theoretical space of possible arrangements to a smaller set that is intended to describe the majority of arrangements which actually occur in the data. The structure of this chapter mimics the division of the model into two parts. Section 2.2 treats pairings in detail while Section 2.3 is concerned with labellings. Lastly, the relationship between this new model and older models is discussed in Section 2.4.

2.1.1 Preliminaries

For a positive integer N, denote with Z_N the set $\{1, 2, \ldots, N\}$. For two sets of integers $A, B \subseteq \mathbb{Z}$, define $A < B$ if $a < b$, for all $a \in A$ and $b \in B$. For any set A and positive integer k, denote with 2^A the family of all subsets of A and with $\binom{A}{k}$ the family of k-element subsets of A and denote with $\left\{ {A \atop k} \right\}$ the collection of all families of k-element subsets of A which partition A. For positive integers M, k, denote with $\left\{ {M \atop k} \right\}$ the number of ways to partition an M-element set by k-element subsets (i.e. $\left\{ {M \atop k} \right\} = \left| \left\{ {Z_M \atop k} \right\} \right|$).

The i^{th} coordinate of a tuple $\vec{a}$ is denoted a_i, or $\vec{a}_i$, unless stated otherwise. For any k-tuple of sets $\vec{A} = (A_1, \ldots, A_k)$, denote with $|\vec{A}|$ the k-tuple $(|A_1|, \ldots, |A_k|)$ and write $\nu(\vec{A}) = k$.

A **relation** R on a set A can be defined as a subset $R \subseteq A \times A$. An element $a \in A$ is said to **relate to** an element $b \in A$, denoted aRb, if $(a, b) \in R$. A relation R on A is called **symmetric** if $(a, b) \in R$ implies $(b, a) \in R$, for all $a, b \in A$ and R is called **anti-reflexive** if $(a, a) \notin R$, for all $a \in A$. A symmetric and anti-reflexive relation R on a set A can be viewed as a subset of $\binom{A}{2}$, since every member of R consists of two distinct elements of A, and because symmetry makes the order of coordinates in a pair $(a, b) \in A \times A$ redundant with respect to R. For the remainder of this chapter, a symmetric and anti-reflexive relation on any set A is viewed as a subset of $\binom{A}{2}$. To a symmetric and anti-reflexive relation R on a set A, the undirected **graph** $G(R)$ **induced by** R is the graph with vertex set $V(G(R)) = A$ and edge set $E(G(R)) = R$. For each $a \in A$, define the **neighborhood** $N_R(a)$ **of** a **with respect to** R as the neighborhood of a in $G(R)$. The **domain** of a relation is the set $\{a \in A : (a, b) \in R \text{ for some } b \in A\}$.

2.2 Pairings

The model discussed in this chapter is concerned only with the relative positions of MDSs and their corresponding counterparts in the product genome. It neglects information encoded in the undelying nucleotide sequences as well as distances between MDSs and lengths of MDSs. To a set of MDSs, we can associate a set of pairs, each of which contains the end coordinates of an MDS. Such a set of pairs can be viewed as a graph, where the end coordinates of MDSs are vertices and pairs (MDSs) are edges. See Figures 2.1 and 2.2 for two examples of rearrangements maps and graph representations of sets of pairs associated to them. Whenever the number of endpoints of members of two sets of pairs are the same, and one of the sets can be obtained from the other by applying the unique order isomorphism $\sigma^{\text{fwd}}_{A,B}$ between the two sets of endpoints A and B, we consider the two sets of pairs equivalent. Viewing sets of pairs as graphs, two sets are equivalent whenever the order isomorphism between the vertices is a graph isomorphism. Observe that this relationship between sets of pairs with the same number of end coordinates is an equivalence relation. We consider sets of pairs whose endpoints make up a consecutive set of integers $1, \ldots, N$ as the representatives of these equivalence classes and call them pairings.

Definition 2.2.1. A **pairing** of Z_N is a symmetric and anti-reflexive relation $\pi \subseteq \binom{Z_N}{2}$, which has domain Z_N. The family of all pairings of Z_N is denoted $\Pi(N)$.

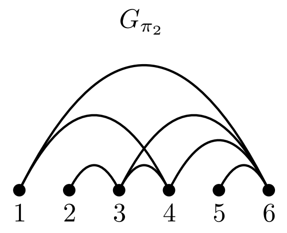
A non-example and an example of a pairing as well as the graph induced by the latter are depicted in Figure 2.3. The relation π_1 shown in part (a) of the figure is not a pairing for multiple reasons. First, the domain of π_1 does not include 2. Next, π_1 is not anti-reflexive because $5\pi_1 5$. Lastly, π_1 is not symmetric since $2\pi_1 1$ and $2\pi_1 4$, but $1,4 \not\pi_1 2$. The relation π_2, on the other hand is a pairing. As opposed to π_1, the domain of π_2 is Z_6. Additionally, no member of Z_6 is contained in its own neighborhood, and whenever $i\pi_2 j$, then $j\pi_2 i$. The graph G_{π_2} induced by π_2 is shown in part (c) of the same figure. The vertices of the graph are the elements of Z_6 and the graph has no isolated vertices since the domain of π_2 is Z_6. For every $\{i,j\} \in \binom{Z_6}{2}$, there is an edge between i and j if and only if $i\pi_2 j$. Observe that by symmetry of π_2, this graph is well-defined and anti-reflexivity of π_2 implies that the graph is loop-free.

Z_6	$N_{\pi_1}(Z_6)$
1	$\{2,3\}$
2	$\emptyset$
3	$\{1,5,6\}$
4	$\{2,6\}$
5	$\{3,5,6\}$
6	$\{3,4,5\}$

(a)

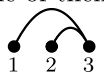
Z_6	$N_{\pi_2}(Z_6)$
1	$\{4,6\}$
2	$\{3\}$
3	$\{2,4,6\}$
4	$\{1,3,6\}$
5	$\{6\}$
6	$\{1,3,4,5\}$

(b)



(c)

Figure 2.3: A relation π_1 on Z_6 which is not a pairing in (a), a relation π_2 on Z_6, which is a pairing in (b), and the graph G_{π_2} induced by π_2 in (c). In (a) and (b), the members of Z_6 are listed together with their neighborhoods.

As opposed to existing models, pairings can model rearrangement maps where MDSs are contained in other MDSs in the precursor and where distinct MDSs share pointers at either end. Figure 2.2 exemplifies how pairings can model containment of MDS regions. In this figure, MDS 1 spans a region contained in the region covered by MDS 4. Since the two MDSs share one of their end coordinates, but not the other, their representation in the figure consists of the substructure 

, where MDS 4 corresponds to the top arc $\{1,3\}$ and MDS 1 corresponds to the bottom arc $\{2,3\}$. Frequently, rearrangement maps were translated into sequences of MDSs or sequences of pointers to derive models from them. However, the containment of MDS regions in the precursor in other MDS regions considered here make a meaningful definition of a total order on the MDSs difficult. Observe that a pairing only represents the relative positions of MDS regions on the precursor and reveals no information about the order and orientation of the corresponding regions in the product.

Remark 2.2.2. The function G which maps pairings $\pi \in Z_N$ to graphs $G(\pi)$ on the vertex set Z_N is a bijection between $\Pi(N)$ and the collection of all graphs without isolated vertices on Z_N.

Remark 2.2.2 suggests that instead of defining pairings as symmetric and non-reflexive relations on Z_N, pairings could have been defined as graphs on a totally ordered vertex set without isolated vertices. While defining pairings as graphs may appeal to a wider audience, Definition 2.2.1 is used here to stay semantically closer to their applications in DNA rearrangements in ciliate biology. In particular, the linear ordering of the elements in Z_N captures the configuration of the end coordinates of precursor intervals and a pairing of Z_N relates pairs of coordinates belonging to the same interval to each other. Due to their bijective correspondence the graph $G(\pi)$ induced by a pairing π may be used interchangeably with π.

Each member of π can be interpreted as the beginning and end coordinate of a precursor interval in an arrangement relative to the ends of other precursor intervals in the arrangement. Likewise, any arrangement produces a pairing when enumerating the set of distinct end coordinates of precursor intervals by Z_N and relating two of them whenever they are the end coordinates of some precursor interval. Thus, pairings can describe any configuration of relative positions of precursor intervals that could possibly occur. To narrow down the space of pairings being explored to a subset of $\Pi(N)$ which reflects more accurately the genome rearrangement data of *O. trifallax*, some relevant properties are introduced in this section, and subspaces of $\Pi(N)$ defined by these properties are counted.

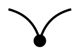
2.2.1 Unidirectional Pairings

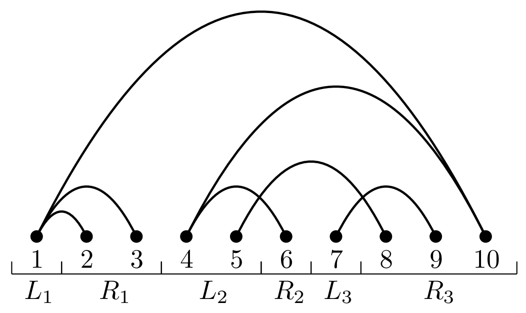
Overlaps between MDSs in the precursor genome of *O. trifallax* can be categorized into overlaps at the ends and containment of one interval in another. Frequently, when MDSs are contained in other MDSs, they align at either end, and so share the same end coordinate. When two precursor intervals overlap, but neither is a subset of the other, then either their interval ends are distinct, or the right end of one of the intervals is the same as the left end. The latter of the two cases has not been observed in the genome rearrangement data of the organism *O. trifallax* at a notable frequency, yet, so we introduce a property of pairings which reflects their absence. The main idea of this property is that if an end coordinate is shared by two or more MDSs in the precursor, it is either the left end for all of them, or the right end for all of them, but never the left end for some and the right end for others.

Definition 2.2.3. A pairing π of Z_N is **unidirectional** if for all $i \in Z_N$, either all members of $N_\pi(i)$ are greater than i, or all members of $N_\pi(i)$ are less than i. The **set of left ends** $L(\pi)$ with respect to a unidirectional pairing π of Z_N, is the set of all elements $i \in Z_N$, where all members of $N_\pi(i)$ are greater than i. Similarly, the **set of right ends** $R(\pi)$ with respect to π is the set of all elements $j \in Z_N$, where all members of $N_\pi(j)$ are less than j. The family of all unidirectional pairings of Z_N is denoted $\Pi_{\text{uni}}(N)$.

The graph induced by a unidirecitonal pairing π of Z_N is bipartite with parts $L(\pi)$ and $R(\pi)$. The sets $L(\pi)$ and $R(\pi)$ can be further partitioned into their maximal consecutive integer subsets. Denote with $\vec{L}(\pi), \vec{R}(\pi)$ the $\nu(\pi)$-tuples $L_1, \ldots, L_{\nu(\pi)} \subseteq L(\pi)$, and $R_1, \ldots, R_{\nu(\pi)} \subseteq R(\pi)$, where $L_1 < R_1 < L_2 < R_2 < \cdots < L_{\nu(\pi)} < R_{\nu(\pi)}$. For a unidirectional pairing $\pi \in \Pi_{\text{uni}}(N)$, set $\vec{l}(\pi) = |\vec{L}(\pi)|$, and $\vec{r}(\pi) = |\vec{R}(\pi)|$.

An example of a pairing which is not unidirectional is provided in Figure 2.3(b), where $N_{\pi_2}(3) = \{2, 4, 6\}$

contains numbers less than 3 and numbers greater than 3. An example of a unidirectional pairing is shown in Figure 2.4(a) along with the partitioning of Z_{10} by π into $(\nu(\pi) = 3)$-tuples $\vec{L}(\pi) = (L_1, L_2, L_3) = \big(\{1\}, \{4,5\}, \{7\}\big)$ and $\vec{R}(\pi) = (R_1, R_2, R_3) = \big(\{2,3\}, \{6\}, \{8,9,10\}\big)$. Here, $\vec{l}(\pi) = (1,2,1)$, and $\vec{r}(\pi) = (2,1,3)$. When the vertices of the graph induced by a pairing are sorted and arranged linearly in order from left to right, then the absence of the substructure  characterizes unidirectionality.



(a)

$$\begin{array}{ccc}
 & \begin{array}{cccccc} 2 & 3 & 6 & 8 & 9 & 10 \end{array} & \\
\begin{array}{c} 1 \\ 4 \\ 5 \\ 7 \end{array} & \left[\begin{array}{cc|c|ccc} 1 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] & \begin{array}{c} L_1 \\ L_2 \\ \\ L_3 \end{array} \\
 & R_1 \quad R_2 \quad\quad R_3 &
\end{array}$$

(b)

Figure 2.4: The graph induced by a unidirectional pairing π of Z_{10}, together with the partition of Z_{10} associated with π in (a), and the biadjacency matrix $M(\pi)$ associated with π in (b).

The task of counting unidirectional pairings can be accomplished using an inclusion-exclusion argument. This argument can be stated in the context of bipartite graphs, but requires some additional language. The adjacency matrix of a bipartite graph $G = (V, E)$ with bipartition $V = V_1 \cup V_2$ is of the form $\left[\begin{smallmatrix} 0 & B^T \\ B & 0 \end{smallmatrix}\right]$, if the vertices are arranged such that the first $|V_1|$ columns and rows correspond to the vertices in V_1 an the remaining $|V_2|$ columns and rows correspond to the vertices in V_2. In this case, the block B is called the **biadjacency matrix** of G. To a unidirectional pairing π we associate the biadjacency matrix $M(\pi) \in \{0,1\}^{|L(\pi)| \times |R(\pi)|}$, where each i^{th} row corresponds to the i^{th} member of $L(\pi)$ (in the integer ordering), and each j^{th} column corresponds to the j^{th} member of $R(\pi)$. Due to the ordering of the rows and columns of $M(\pi)$,the matrix $M(\pi)$ can be split into blocks $B_{p,q}$, for $(p,q) \in Z_{\nu(\pi)} \times Z_{\nu(\pi)}$, where each $B_{p,q}$ is the $|L_p| \times |R_q|$ biadjacency matrix of the subgraph of $G(\pi)$ induced by the vertices $L_p \cup R_q$. The blocks of a biadjacency matrix $M(\pi)$ of a unidirectional pairing π of Z_N are denoted $B_{p,q}$, for $(p,q) \in \nu(\pi) \times \nu(\pi)$. The biadjacency matrix $M(\pi)$ for the unidirectional pairing π from Figure 2.4(a) is shown in part (b) of the figure. Since each p^{th} coordinate L_p of $\vec{L}(\pi)$ consists of left ends only, and by the ordering of the coordinates in $\vec{L}(\pi)$ and $\vec{R}(\pi)$, the members of L_p can only be paired up by π with members of R_q for $q \geq p$. Similarly, members of the q^{th} coordinate R_q of $\vec{R}(\pi)$ can only be paired up by π with members of L_p, for $p \leq q$. Thus, the blocks $B_{p,q}$, where $p > q$, are 0-matrices. Furthermore, while two distinct unidirectional pairings π_1, π_2 of Z_N may have the same partitions, i.e. $\vec{L}(\pi_1) = \vec{L}(\pi_2)$ and $\vec{R}(\pi_1) = \vec{R}(\pi_2)$, the matrices $M(\pi_1)$ and

$M(\pi_2)$ may differ. These observations lead to Lemma 2.2.4 which asserts a bijective correspondence between unidirectional pairings and their corresponding biadjacency matrices.

Before stating Lemma 2.2.4, the set of biadjacency matrices corresponding to unidirectional pairings is described. Let $\nu \geq 1$ and let $\vec{l} = (l_1, \ldots, l_\nu), \vec{r} = (r_1, \ldots, r_\nu)$ be ν-tuples of positive integers adding up to N. The set of all $\pi \in \Pi_{\text{uni}}(N)$ with $\vec{l}(\pi) = \vec{l}$, and $\vec{r}(\pi) = \vec{r}$ is in one-to-one correspondence with a subset of the collection $\mathcal{T}(\vec{l}, \vec{r})$ of all matrices of the form:

$$\begin{bmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,\nu-1} & B_{1,\nu} \\ 0 & B_{2,2} & \cdots & B_{2,\nu-1} & B_{2,\nu} \\ 0 & 0 & \cdots & B_{3,\nu-1} & B_{3,\nu} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & B_{\nu,\nu} \end{bmatrix},$$

where $B_{p,q} \in \{0,1\}^{l_p \times r_q}$, for all $(p,q) \in Z_\nu \times Z_\nu$, and $B_{p,q} = [0]$ if $p > q$. More specifically, the subset of $\mathcal{T}(\vec{l}, \vec{r})$ in question consists of those matrices, which have no zero-rows, and no zero-columns.

Lemma 2.2.4. *Let $\mathcal{T}^*(\vec{l}, \vec{r})$ be the collection of those members of $\mathcal{T}(N)$, which have no zero rows and no zero columns. The mapping $M : \Pi_{\text{uni}}(N) \to \mathcal{T}^*(N)$ defined by $\pi \mapsto M(\pi)$, is a bijection.*

Proof. Let $\pi \in \Pi_{\text{uni}}(N)$. Then $M(\pi) \in \mathcal{T}(\vec{l}(\pi), \vec{r}(\pi)) \subseteq \mathcal{T}(N)$. Since the domain of π is Z_N, the matrix $M(\pi)$ does not have any zero rows, or columns. Hence, $M(\pi) \in \mathcal{T}^*(N)$, proving that M indeed maps into $\mathcal{T}^*(N)$.

Let $Q \in \mathcal{T}^*(N)$. Then for the pairing π of Z_N defined by $i\pi j$ whenever $Q_{i,j} = 1$, we have that $Q = M(\pi)$. Thus, M is surjective.

If $\pi_1 \neq \pi_2 \in \Pi_{\text{uni}}(N)$, then without loss of generality, there exist $\{i,j\} \in \binom{Z_N}{2}$, such that $i\pi_1 j$ and $i < j$, but $i \not\pi_2 j$. Then $M(\pi_1)_{i,j} = 1 \neq 0 = M(\pi_2)_{i,j}$, so that M is injective. □

There are many ways to partition Z_N into left ends and right ends. Lemma 2.2.5 counts the subset of $\mathcal{T}(\vec{l}, \vec{r})$ without zero rows for any choice of $\vec{l}$ and $\vec{r}$. Let $\mathcal{U}(N) = \{(\vec{l}, \vec{r}) : \nu(\vec{l}) = \nu(\vec{r}) \text{ and } \sum_{i=1}^{\nu(\vec{l})} (\vec{l}_i + \vec{r}_i) = N\}$, for $i \in Z_{\nu(\vec{l})}$. Next, let $\Gamma(N)$ be the the collection of pairs of tuples $(\vec{L} = (L_1, \ldots, L_\nu), \vec{R} = (R_1, \ldots, R_\nu))$, with $\nu \in \mathbb{N}$, of pairwise-disjoint sets which partition Z_N into $L = L_1 \cup \cdots \cup L_\nu$ and $R = R_1 \cup \cdots \cup R_\nu$, such that the coordinates L_p of $\vec{L}$, and R_q of $\vec{R}$ are consecutive integer subsets of L and R, respectively, and where $L_1 < R_1 < L_2 < R_2 < \cdots < L_\nu < R_\nu$. Denote with $\mathcal{T}(N)$ the union of the sets $\mathcal{T}(\vec{l}, \vec{r})$ taken over all $(\vec{l}, \vec{r}) \in \mathcal{U}(N)$. Any member $\varrho = (\vec{l}, \vec{r})$ of $\mathcal{U}(N)$ uniquely defines a member $\gamma(\varrho) = (\vec{L}, \vec{R}) \in \Gamma(N)$, such that $|\vec{L}| = \vec{l}$ and $|\vec{R}| = \vec{r}$. For each $\varrho = ((l_1, \ldots, l_\nu), (r_1, \ldots, r_\nu)) \in \mathcal{U}(N)$ and each $\gamma = ((L_1, \ldots, L_\nu), (R_1, \ldots, R_\nu)) \in \Gamma(N)$, write $\nu(\varrho) = \nu(\gamma) = \nu$. While each $\pi \in \Pi_{\text{uni}}(N)$ uniquely associates with a member $(\vec{L}(\pi), \vec{R}(\pi)) \in \Gamma(N)$, there may be multiple members $\pi \in \Pi_{\text{uni}}(N)$ for $(\vec{L}, \vec{R}) \in \Gamma(N)$ such that $\vec{L}(\pi) = \vec{L}$ and $\vec{R}(\pi) = \vec{R}$. For every $(\vec{L}, \vec{R}) \in \Gamma(N)$, define $\Pi_{\text{uni}}(\vec{L}, \vec{R}) = \{\pi \in \Pi_{\text{uni}}(N) : \vec{L}(\pi) = \vec{L} \text{ and } \vec{R}(\pi) = \vec{R}\}$. Similarly, for every $(\vec{l}, \vec{r}) \in \mathcal{U}(N)$, define $\Pi_{\text{uni}}(\vec{l}, \vec{r}) = \{\pi \in \Pi_{\text{uni}}(N) : \vec{l}(\pi) = \vec{l} \text{ and } \vec{r}(\pi) = \vec{r}\}$.

Lemma 2.2.5. *Let $(\vec{L}, \vec{R}) \in \Gamma(N)$, and $\varrho = (\vec{l}, \vec{r}) = (|\vec{L}|, |\vec{R}|)$, and $\nu = \nu(\varrho)$, let $l = \sum_{p=1}^{\nu} l_p$, $r = \sum_{q=1}^{\nu} r_q$, and let $J \subseteq R_1 \cup \cdots \cup R_\nu$. Denote with $\mathcal{T}_J^0(\vec{l}, \vec{r})$, the collection of all matrices M in $\mathcal{T}(\vec{l}, \vec{r})$ without zero rows, where $M_{i,j} = 0$, for all $i \in Z_l$, and $j \in J$. Then:*

$$|\mathcal{T}_J^0(\vec{l}, \vec{r})| = \prod_{p \in Z_\nu} \left(2^{\sum_{q=p}^{\nu} r_q - |R_q \cap J|} - 1\right)^{l_p},$$

where, $\vec{R}$ is the second coordinate of $(\vec{L}, \vec{R}) = \gamma(\varrho)$.

Proof. Each matrix $M = [B_{p,q}] \in \mathcal{T}_J^0(\vec{l}, \vec{r})$ can have non-zero entries only in the columns with indices in $Z_n \setminus J$. Since $p > q$ implies $B_{p,q} = [0]$, we have that for each $p \in Z_\nu$ and for each of row in the block row $[B_{p,1}, \ldots, B_{p,\nu}]$, there are $\sum_{q=p}^{\nu} r_q - |R_q \cap J|$ entries which may be non-zero. Since each pth block row has l_p rows, there are $(2^{\sum_{q=p}^{\nu} r_q - |R_q \cap J|} - 1)^{l_p}$ distinct configurations of non-zero entries in each pth block row for members of $\mathcal{T}_J^0(\vec{l}, \vec{r})$. Distinct configurations of non-zero entries in each row define distinct members of $\mathcal{T}_J^0(\vec{l}, \vec{r})$, so that the result follows by the Rule of Product. □

Corollary 2.2.6. *With the notation from Lemma 2.2.5, let $\mathcal{T}^*(\vec{l}, \vec{r}) \subset \mathcal{T}(\vec{l}, \vec{r})$ be the collection of all members of $\mathcal{T}(\vec{l}, \vec{r})$ which have no 0 columns or rows. Then:*

$$|\mathcal{T}^*(\vec{l}, \vec{r})| = \sum_{J \subseteq R_1 \cup \cdots \cup R_\nu} (-1)^{|J|} \prod_{p \in Z_\nu} \left(2^{\sum_{q=p}^{\nu} r_q - |R_q \cap J|} - 1\right)^{l_p}$$

Proof. The statement of this corollary follows from Lemma 2.2.5 by inclusion-exclusion from the fact that the set $\mathcal{T}^*(\vec{l}, \vec{r})$ is the set of all members of $\mathcal{T}(\vec{l}, \vec{r})$ which have no zero rows minus those members which have zero columns (i.e. $\mathcal{T}^*(\vec{l}, \vec{r}) = \mathcal{T}_\emptyset^0(\vec{l}, \vec{r}) \setminus \left(\cup_{j=1}^{r} \mathcal{T}_{\{j\}}^0(\vec{l}, \vec{r}) \right)$). □

Corollary 2.2.7. *The number of unidirectional pairings of Z_N is:*

$$|\Pi_{\text{uni}}(N)| = \sum_{(\vec{L}, \vec{R}) \in \Gamma(N)} \sum_{J \subseteq R_1 \cup \cdots \cup R_{\nu(\vec{R})}} (-1)^{|J|} \prod_{p \in Z_{\nu(\vec{L})}} \left(2^{\sum_{q=p}^{\nu(\vec{R})} |R_q| - |R_q \cap J|} - 1\right)^{|L_p|}.$$

Proposition 2.2.8. *1. The unique pairing $\pi_{\max}(\varrho)$ for each $\varrho \in \mathcal{U}(N)$ achieving $\max_{\pi \in \Pi_{\text{uni}}(\varrho)} |\pi|$, is defined by $i\pi j$, for all i, j, where $i \in \vec{L}(\pi)_p$ and $j \in \vec{R}(\pi)_q$ for some $p \leq q$.*

2. If $\pi \in \Pi_{\text{uni}}(N)$, then $|\pi| \leq \frac{N^2}{4}$.

Proof. Let $\pi \in \Pi_{\text{uni}}(\vec{l}, \vec{r})$. By definition of unidirectionality, $i\pi j$ implies $i \in \vec{L}(\pi)_p$ and $j \in \vec{R}(\pi)_q$ for some $p \leq q$. Thus, $\pi \subseteq \pi_{\max}(\vec{l}, \vec{r})$, so that $\pi_{\max}(\vec{l}, \vec{r})$ indeed uniquely achieves the maximum number of pairs over all unidirectional pairings in $\Pi_{\text{uni}}(\vec{l}, \vec{r})$. Write $\nu = \nu(\vec{l}, \vec{r})$. Then,

$$|\pi_{\max}(\vec{l}, \vec{r})| = \sum_{p=1}^{\nu} \sum_{q=p}^{\nu} l_p r_q \leq \sum_{p=1}^{\nu} \sum_{q=1}^{\nu} l_p r_q = \Big(\sum_{p=1}^{\nu} l_p \Big) \Big(\sum_{q=1}^{\nu} r_q \Big) = mn = m(N-m). \tag{2.1}$$

Equality holds in 2.1 if and only if $\nu = 1$ because each of the l_p and r_q are positive. Thus, $|\pi_{\max}(\vec{l}, \vec{r})|$ is at its maximum taken over all $(\vec{l}, \vec{r}) \in \mathcal{U}(N)$, when $\nu(\vec{l}, \vec{r}) = 1$ and when $m(N-m)$ is at its maximum. The function $f(m) = m(N-m)$ is at its absolute maximum for $m \in [0, N]$ when $m \in \{\lceil \frac{N}{2} \rceil, \lfloor \frac{N}{2} \rfloor\}$. Thus, the unidirectional pairing of Z_N with the most pairs is of size $|\pi_{\max}(N)| = \lceil \frac{N}{2} \rceil \cdot \lfloor \frac{N}{2} \rfloor \leq \frac{N^2}{4}$. □

2.2.2 Pairing Density

One element of Z_N can theoretically be paired up with all $N-1$ remaining elements of Z_N by a pairing $\pi \in \Pi(N)$. Such a pairing would describe a configuration of MDSs which all share a common end coordinate. In the genome rearrangement data of the organism *O. trifallax*, however, MDSs which share end coordinates do not appear frequently. Hence, rearrangements where all MDSs share an end coordinate are even less frequent, especially for larger numbers of MDSs. Nevertheless, shared precursor interval end coordinates do appear in the data and should not be completely disregarded. A notion of density is desirable, which describes the number of pairs that elements of Z_N belong to.

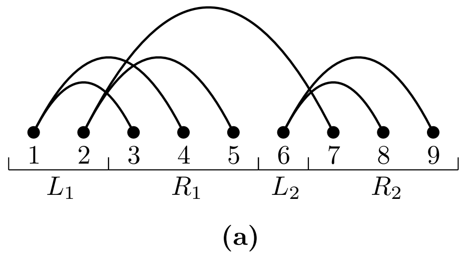
Definition 2.2.9. Let π be a pairing of Z_N. The **density $d_i(\pi)$ of π at $i \in Z_N$** is the number of elements in $N_\pi(i)$. A pairing $\pi \in \Pi(N)$ is said to have **density at most** d if for all $i \in Z_N$, $d_i(\pi) \leq d$. The families of all pairings of Z_N with density at most d, and all unidirectional pairings of Z_N with density at most d, are denoted $\Pi_d(N)$ and $\Pi_{\text{uni},d}(N)$, respectively.

Remark 2.2.10. The function $G(\pi)$ is a bijection between $\Pi_d(N)$ and the collection of the graphs on Z_N with maximum vertex degree less than or equal to d and without isolated vertices. The largest pairing of Z_N with density $d \in Z_{n-1}$ has size $N \cdot d$.

Let $\mathcal{U}_d(N)$ be the collection of those pairs $\varrho \in \mathcal{U}(N)$ for which a unidirectional pairing π with density at most d exists with partition $(|\vec{L}(\pi)|, |\vec{R}(\pi)|) = (\varrho) \neq \emptyset$.

Figure 2.5 shows two partitionings of Z_N, one of which admits a unidirectional pairing of density at most 2, while the other does not. Therefore, the latter corresponds to a member $((|\widetilde{L}_1|, |\widetilde{L}_2|), (|\widetilde{R}_1|, |\widetilde{R}_2|))$ of $\mathcal{U}(N) \setminus \mathcal{U}_d(N)$. The partitioning of Z_9 shown in Figure 2.5(b) does not admit a unidirectional pairing of Z_9

of density at most 2 with set of left ends $\widetilde{L} = \widetilde{L}_1 \cup \widetilde{L}_2$ and set of right ends $\widetilde{R} = \widetilde{R}_1 \cup \widetilde{R}_2$ since the 3 elements in $\widetilde{R}_1$ can only be paired with the single element in $\widetilde{L}_1$. The relation $\widetilde{\pi}$ shown in Figure 2.5(b) exemplifies an attempt to defining a unidirectional pairing of Z_9 of density at most 2 with set of left ends $\widetilde{L}$ and set of right ends $\widetilde{R}$. The domain of $\widetilde{\pi}$ is not Z_9 and addition of any more pairs to $\widetilde{\pi}$ results in elements in Z_9 with a neighborhood of size larger than 2. Observe that the sizes of $\widetilde{L}$ and $\widetilde{R}$ are the same as those of $L = L_1 \cup L_2$ and $R = R_1 \cup R_2$ in (a), respectively showing that not just the number of left and right ends determine the existence of a unidirectional pairing of Z_N of density at most d, but also the partitioning of Z_N.



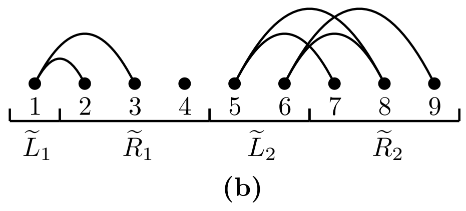


Figure 2.5: **(a):** the graph induced by a unidirectional pairing π of Z_9 of density at most 2 together with its associated partitioning of Z_9 into L_1, L_2, R_1, R_2. **(b):** the graph induced by an anti-reflexive and symmetric relation $\widetilde{\pi}$ on Z_9 together with a partitioning of Z_9 into sets $\widetilde{L}_1, \widetilde{L}_2, \widetilde{R}_1$, and $\widetilde{R}_2$.

For $d \in Z_{n-1}$, and $(\vec{l}, \vec{r}) \in \mathcal{U}_d(N)$, denote with $\mathcal{T}^*_d(N)$, and $\mathcal{T}^*_d(\vec{l}, \vec{r})$, the subsets of $\mathcal{T}^*(N)$, and $\mathcal{T}^*(\vec{l}, \vec{r})$, respectively, of those matrices whose columns and rows all have at most d non-zero entries. The restriction of the mapping M from Lemma 2.2.4 to the set $\Pi_{\text{uni},d}(\vec{l}, \vec{r})$, is a bijection onto $\mathcal{T}^*_d(\vec{l}, \vec{r})$. Hence, counting the sets $\mathcal{T}^*_d(\vec{l}, \vec{r})$ and $\mathcal{T}^*_d(N) = \cup_{\varrho \in \mathcal{U}_d(N)} \mathcal{T}^*_d(\varrho)$ amounts to counting the matrices in $\mathcal{T}^*_d(\vec{l}, \vec{r})$. To prove Corollary 2.2.6, inclusion-exclusion was used to count the collection $\mathcal{T}^*(\vec{l}, \vec{r}) = \mathcal{T}^0_{\emptyset}(\vec{l}, \vec{r}) \setminus \Big(\cup_{j=1}^r \mathcal{T}^0_{\{j\}}(\vec{l}, \vec{r}) \Big)$. The same approach leads to complications when attempting to apply it to $\mathcal{T}^*_d(\vec{l}, \vec{r})$. Expanding the count to the subsets of $\mathcal{T}^*(\vec{l}, \vec{r})$ of matrices with at most d non-zero entries in all rows, but not necessarily all columns provides an upper bound to the size of $\mathcal{T}^*_d(\vec{l}, \vec{r})$.

Lemma 2.2.11. *Let $(\vec{l}, \vec{r}) \in \mathcal{U}_d(N)$ and $\nu = \nu(\vec{l}, \vec{r})$. Let $\widetilde{\mathcal{T}}^*_d(\vec{l}, \vec{r})$ be the set of all matrices in $\mathcal{T}^*(\vec{l}, \vec{r})$, where every row has at most d non-zero entries. Then $\mathcal{T}^*_d(\vec{l}, \vec{r}) \subseteq \widetilde{\mathcal{T}}^*_d(\vec{l}, \vec{r})$ and*

$$|\widetilde{\mathcal{T}}^*_d(\vec{l}, \vec{r})| = \sum_{J \subseteq Z_r} (-1)^{|J|} \left(\prod_{p=1}^{\nu} \left[\sum_{u=1}^{d} \binom{\sum_{q=p}^{\nu} r_q - |R_q \cap J|}{u} \right]^{l_p} \right).$$

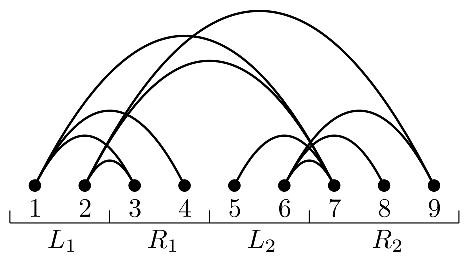
Proof. Let $(\vec{l}, \vec{r}) \in \mathcal{U}_d(N)$ and $J \subseteq Z_r$. Denote with $\mathcal{T}^0_{J,d}(\vec{l}, \vec{r})$ the collection of all matrices $M \in \mathcal{T}(\vec{l}, \vec{r})$ without zero rows, where $M_{i,j} = 0$, for all $(i, j) \in Z_l \times J$ and every row has at most d non-zero entries. The various configurations of up to d non-zero entries per row are in one-to-one correspondence with the collection $\mathcal{T}^0_{J,d}(\vec{l}, \vec{r})$. Each of the l_p rows in the pth block row have exactly $\binom{\sum_{q=p}^{\nu} r_q - |R_q \cap J|}{u}$ configurations of

$u \in Z_d$ non-zero entries. Then each such row has exactly $\sum_{u \in Z_d} \binom{\sum_{q=p}^{\nu} r_q - |R_q \cap J|}{u}$ different configurations of at most d non-zero entries. Applying the rule of product to count the number of configurations of non-zero entries in each block row gives

$$|\mathcal{T}^0_{J,d}(\vec{l},\vec{r})| = \prod_{p=1}^{\nu} \Bigg[\sum_{u \in Z_d} \binom{\sum_{q=p}^{\nu} r_q - |R_q \cap J|}{u} \Bigg]^{l_p}$$

The result follows by applying inclusion-exclusion and the fact that $\widetilde{\mathcal{T}}^*_d(\vec{l},\vec{r}) = \mathcal{T}^0_{\emptyset,d}(\vec{l},\vec{r}) \setminus \Big(\cup_{j=1}^{r} \mathcal{T}^0_{\{j\},d}(\vec{l},\vec{r}) \Big)$.

□



(a)

$$\begin{array}{cc}
 & \begin{array}{ccccc}3&4&7&8&9\end{array} \\
\begin{array}{c}1\\2\\5\\6\end{array} & \left[\begin{array}{cc|ccc}1&1&1&0&0\\1&0&1&0&1\\ \hline 0&0&1&0&0\\0&0&1&1&1\end{array}\right] \begin{array}{c}L_1\\ \\ \\ L_2\end{array} \\
 & \begin{array}{cc}R_1 & R_2\end{array}
\end{array}$$

(b)

Figure 2.6: The graph induced by a unidirectional pairing π of Z_9 and the partition $(\vec{L}(\pi) = (\{1,2\},\{5,6\}), \vec{R}(\pi) = (\{3,4\},\{7,8,9\}))$ of Z_9 associated with π in (a) and the biadjacency matrix $M(\pi)$ associated with π in (b). Since $N_\pi(7) = \{1,2,5,6\}$, the pairing is not of density 3. However, the matrix $M(\pi)$ is a member of $\widetilde{\mathcal{T}}^*_d((2,2),(2,3))$, which proves that $\widetilde{\mathcal{T}}^*_d((2,2),(2,3)) \setminus \mathcal{T}^*_d((2,2),(2,3)) \neq \emptyset$.

Corollary 2.2.12. *The number of unidirectional pairings of Z_N with density d is bounded above by:*

$$|\Pi_{\text{uni},d}(N)| \leq \sum_{\varrho=(\vec{l},\vec{r}) \in \mathcal{U}_d(N)} \sum_{J \subseteq Z_r} (-1)^{|J|} \left(\prod_{p=1}^{\nu(\vec{l},\vec{r})} \Bigg[\sum_{u=1}^{d} \binom{\sum_{q=p}^{\nu(\vec{l},\vec{r})} (r_q - |R_q \cap J|)}{u} \Bigg]^{l_p} \right).$$

Proof. The result follows from Lemma 2.2.11 since the restriction of the mapping M in Lemma 2.2.4 to the set $\Pi_{\text{uni},d}(N)$ is a bijection onto the set $\Big(\cup_{(\vec{l},\vec{r}) \in \mathcal{U}_d(N)} \mathcal{T}^*_d(\vec{l},\vec{r}) \Big)$ contained in $\Big(\cup_{(\vec{l},\vec{r}) \in \mathcal{U}_d(N)} \widetilde{\mathcal{T}}^*_d(\vec{l},\vec{r}) \Big)$. □

2.3 Labellings

Pairings describe configurations of MDSs in the precursor, but they give no information about which regions in the product they match. One way of conveying the relative positions of the product regions matching the MDSs described by a pairing is to label each pair according to the region it matches. Each MDS in the product can be identified by a number that indicates its position in the sequence of MDSs on the product.

Additionally, the parity of the label can indicate orientation of the alignment. A pair may be labelled with more than one MDS identifier and an MDS identifier may appear in multiple labels. Figure 2.1 exemplifies a labelling where some MDS identifiers $(4, -2, -3, -7)$ each appear in more than one label. Figure 2.2 depicts a label $\{2, 5\}$ of a pair $\{4, 5\}$ consisting of more than one MDS identifier, which means that this MDS matches both to the 2nd and 5th MDS in the product. Extending on the idea of double occurrence words, this labelling technique allows for the appearance of the same MDS identifier more than twice and for more than one MDS identifier at the same location. In this section, the necessary properties for such a labelling to model actual arrangements appearing in DNA rearrangement data sets similar to that of *O.* trifallax are established. The spaces of labellings with such properties is explored combinatorially via counting results.

For a set of integers $A \subseteq \mathbb{Z}$, denote with $\pm A$ the set $\bigcup_{a \in A}\{a, -a\}$.

Definition 2.3.1. Let $P \subseteq \binom{Z_N}{2}$ and let $K \in \mathbb{N}$. A ***K*-labelling** of P is a function $\lambda : P \to 2^{\pm Z_K} \setminus \emptyset$. The image $\lambda(P) = \{\lambda(x) : x \in P\}$ of λ is called the **set of labels of** λ and the set $\pm Z_K$ is called the **set of symbols of** λ. The collection of all K-labellings of P is denoted $\Lambda_K(P)$.

Three examples of 4-labellings of subsets of $\binom{Z_6}{2}$ are depicted in Figure 2.7. A K-labelling of a set of pairs $P \subseteq \binom{Z_N}{2}$ can be viewed as a labelling of the edges of the graph $G(P)$. Counting the sets $\Lambda_K(P)$, for arbitrary sets of pairs $P \subseteq \binom{Z_N}{2}$ amounts to counting the number of ways of assigning nonempty subsets of $\pm Z_K$ to the pairs in P.

Remark 2.3.2. Let $P \subseteq \binom{Z_N}{2}$. Then $|\Lambda_K(P)| = (2^{2K} - 1)^{|P|}$.

In general, K-labellings may not contain some of the symbols from Z_K, neither in the positive nor in the negative form of the symbols. Such labellings can often be viewed as K'-labellings for some $K' < K$, where the set of all symbols used in the labels is precisely $Z_{K'}$. Thus, to simplify this analysis, we may often assume all symbols in Z_K to be present in the labels of the K-labelling in consideration. A K-labelling corresponding to such an arrangement is **complete** in the sense that, each of the symbols $1, \ldots, K$ appears in one of the labels, regardless of its parity.

Define the **projection of a set** $A \subseteq \mathbb{Z}$ to be the set $\operatorname{proj}(A) = \{|a| : a \in A\}$. Let $\mathcal{F} \subseteq 2^{\mathbb{Z}}$ be a family of sets of integers. Call the set $\operatorname{proj}(\mathcal{F}) = \{\operatorname{proj}(A) : A \in \mathcal{F}\}$ of projections of members of $\mathcal{F}$ the **projection of** $\mathcal{F}$. Define the **support of** $\mathcal{F}$ to be the set:

$$\operatorname{supp}(\mathcal{F}) = \bigcup_{A \in \mathcal{F}} \operatorname{proj} A.$$

For a single set of integers $A \subseteq \mathbb{Z}$, or a singleton family of sets of integers $\mathcal{F} = \{A\}$, we may use the terms **support** and **projection** interchangeably referring to the same set $\operatorname{supp}(A) = \operatorname{proj}(\mathcal{F}) = \{|a| : a \in A\}$.

Example 1. If $\mathcal{F} = \{A_1 = \{1, -1, 3\}, A_2 = \{-2, -3\}\}$, then:

$$\operatorname{supp}(\mathcal{F}) = \operatorname{proj}(A_1) \cup \operatorname{proj}(A_2) = \{1, 3\} \cup \{2, 3\} = Z_3$$

Definition 2.3.3. Let $P \subseteq \binom{Z_N}{2}$ and $\lambda \in \Lambda_K(P)$. A symbol $a \in Z_K$ is said to be **covered by a label** $A \in \lambda(P)$ if $\{a, -a\} \cap A \neq \emptyset$. The set of symbols in Z_K covered by A is its projection $\operatorname{proj}(A)$. The symbol a is **covered by the** K**-labelling** λ if $a \in \operatorname{proj}(A)$, for some $A \in \lambda(P)$. The **set of symbols covered by** λ is the support of its labels $\operatorname{supp}(\lambda(P))$. Let $S \subseteq Z_K$. Denote with $\Lambda_{K,S}(P)$ the set of all K-labellings λ of P where $\operatorname{supp}(\lambda(P)) \subseteq S$. A K-labelling λ of P is **complete in** S if $\operatorname{supp}(\lambda(P)) = S$. Call a K-labelling simply **complete** if it is complete in Z_K. Denote with $\hat{\Lambda}_{K,S}(P)$ the set of K-labellings of P, which are complete in S and denote with $\hat{\Lambda}_K(P)$ the set of all complete K-labellings λ of P.

The set of elements of Z_4 covered by the 4-labelling λ_1 depicted in Figure 2.7(a) is the set $\operatorname{supp}(\lambda_1(P)) = \{1, 2, 4\}$. Hence, λ_1 is complete in $\{1, 2, 4\}$. Since 3 is not covered by λ_1, it is not complete in Z_4. The 4-labelling λ_2 in Figure 2.7(b), on the other hand, is complete since $\operatorname{supp}(\lambda_2(P)) = \{1, 2, 3, 4\}$. The 4-labelling λ_3 of the pairing P of Z_6 depicted in Figure 2.7(c) is a member of $\Lambda_{4,Z_4}(P)$, but it is not complete. However, when viewed as a 3-labelling, λ_3 is complete because $\operatorname{supp}(\lambda_3(P)) = Z_3$.

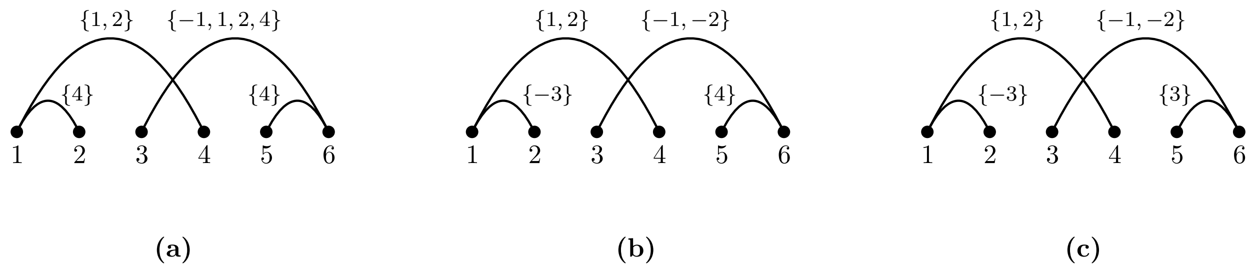


Figure 2.7: Three examples of 4-labellings of the set of pairs $P = \{\{1, 2\}, \{1, 4\}, \{3, 6\}, \{5, 6\}\} \subseteq \binom{Z_6}{2}$. The labelling in **(a)** has set of labels $\{\{4\}, \{1, 2\}, \{-1, 1, 2, 4\}\}$. The labelling in **(b)** has set of labels $\{\{-3\}, \{1, 2\}, \{-1, -2\}, \{4\}\}$. The labelling in **(c)** has set of labels $\{\{-3\}, \{1, 2\}, \{-1, -2\}, \{3\}\}$.

2.3.1 Coherent Labellings

The first property of labellings investigated in this section which is inherently satisfied by labellings derived from arrangements obtained from the genome of *O. trifallax*, is the property of **coherence**. When two labels A and B in the set of labels of a K-labelling λ intersect in an element, say $a \in \pm Z_K$, then the precursor region described by the pairs x and y, both match the $|a|^{\text{th}}$ product segment in the rearrangement map and thus consist of the same DNA sequence. Then the product regions matching the precursor region described by x also match the precursor region described by y and vice versa. Thus, we must have $A \cup B = \lambda(x) = \lambda(y)$. Analogously, when $A \cap -B \neq \emptyset$ then we must have $\lambda(x) = -\lambda(y)$ because the precursor DNA sequences corresponding to x and y are reverse complements of each other. It should be noted that the argument above uses the simplifying assumtption of identical sequence alignments between each precursor MDS and the product MDSs it maps to. In practice, however, commonly employed sequence similarity scoring functions are theoretically not transitive.

Define the **negation** of a set $A \subseteq \mathbb{Z}$ as $-A = \{-a : a \in A\}$.

Definition 2.3.4. Let $P \subseteq \binom{Z_N}{2}$. A K-labelling λ of P is called **coherent** if for all $x, y \in P$, the following two conditions are satisfied:

(i) If $\lambda(x) \cap \lambda(y) \neq \emptyset$, then $\lambda(x) = \lambda(y)$.

(ii) If $\lambda(x) \cap -\lambda(y) \neq \emptyset$, then $\lambda(x) = -\lambda(y)$.

Denote with $\Lambda_{K,\text{coh}}(P)$ the set of all coherent K-labellings of P and with $\Lambda_{K,S,\text{coh}}(P), \hat{\Lambda}_{K,S,\text{coh}}(P)$, and $\hat{\Lambda}_{K,\text{coh}}(P)$, the intersections of $\Lambda_{K,\text{coh}}(P)$ with $\Lambda_{K,S}(P)$, $\hat{\Lambda}_{K,S}(P)$, and $\hat{\Lambda}_K(P)$, respectively.

In Figure 2.7(a), the labels of pairs $\{1, 4\}$ and $\{3, 6\}$ have non-empty intersection $\{1, 2\}$. However, the two labels do not equal each other. Thus, the depicted labelling is incoherent. The 4-labellings in Figure 2.7(b) and (c), are coherent. Whenever two labels intersect in these K-labellings they equal each other and whenever one label intersect the negation of another label, each of the two labels equals the negation of the other. Three additional coherent 3-labellings of a different set of pairs are shown in Figure 2.8. In (b) and (c), the label $\{-2, 2, -3, 3\}$ intersects its own negation and thus is equal to its own negation.

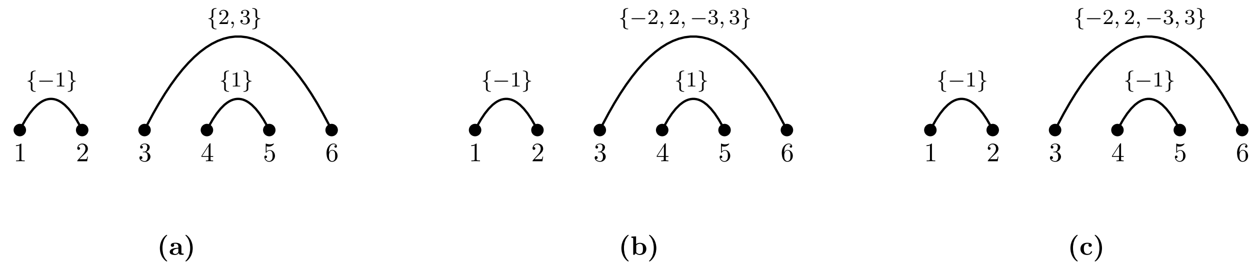


Figure 2.8: Three coherent 3-labellings of the set of pairs $P = \{\{1, 2\}, \{3, 6\}, \{4, 5\}\} \subset \binom{Z_6}{2}$.

Observe that both conditions in Definition 2.3.4 also apply when $x = y$. In that case, $\lambda(x) \cap -\lambda(x) \neq \emptyset$ implies that $\lambda(x) = \pm\lambda(x)$. Thus, if $\{a, -a\} \subseteq \lambda(x)$, for any symbol $a \in Z_K$, then $\lambda(x) = \pm\lambda(x)$. Sets of symbols $A \subseteq \pm Z_K$ where $A = \pm A$ are called **palindromic**. By contraposition, any anti-palindromic label

A from the set of labels of a coherent K-labelling must never contain a symbol a and its negation $-a$, at the same time. Sets of symbols $A \subseteq \pm Z_K$ where $A \cap -A = \emptyset$ are called **anti-palindromic**. Therefore, any label A in the set of labels of a coherent K-labelling must either be palindromic, or anti-palindromic. All but one of the labels in Figure 2.7 are anti-palindromic. The label $\{-1, 1, 2, 4\}$ from the set of labels of the incoherent K-labelling depicted in Figure 2.7(a) is neither palindromic, nor anti-palindromic. The label $\{-2, 2, -3, 3\}$ of the pair $\{3, 6\}$ in Figures 2.8(b) and (c) is palindromic. All other labels in Figure 2.8 are anti-palindromic.

Given a coherent K-labelling λ of a set of pairs $P \subseteq \binom{Z_N}{2}$, coherence implies that when a label $\lambda(x)$ of one pair $x \in P$ intersects the projection of a label $\lambda(y)$ of another pair $y \in P$, then the labels must either be equal, negations of each other, or both. Thus, the projections of two labels in $\lambda(P)$ intersect only if they are equal. Define the relation $\equiv_\lambda^{\mathrm{proj}}$ on the set of pairs by:

$$x \equiv_\lambda^{\mathrm{proj}} y \qquad \Longleftrightarrow \qquad \mathrm{proj}(x) = \mathrm{proj}(y).$$

Then $\equiv_\lambda^{\mathrm{proj}}$ is an equivalence class and coherence of λ implies that pairs from distinct equivalence classes have disjoint projections. Additionally, the set $P \big/ {\equiv_\lambda^{\mathrm{proj}}}$ of equivalence classes in P under the relation $\equiv_\lambda^{\mathrm{proj}}$ is in one-to-one correspondence with $\mathrm{proj}\,\lambda(P)$ via the mapping f, which maps the equivalence classes $[x]_{\equiv_\lambda^{\mathrm{proj}}}$ into $\mathrm{proj}(\lambda(x))$. Observe also that disjointness of the projections of the labels implies that $\mathrm{proj}(\lambda(P))$ partitions the support $\mathrm{supp}(\lambda(P))$. In Figure 2.7(c), for example, the projections of the labels of pairs $\{1, 4\}$ and $\{3, 6\}$ intersect in $\{1, 2\}$ and are negations of each other. Therefore, $\{1, 4\} \equiv_\lambda^{\mathrm{proj}} \{3, 6\}$, where λ is the depicted labelling. Similarly, $\{1, 2\} \equiv_\lambda^{\mathrm{proj}} \{5, 6\}$, so that the set of equivalence classes under $\equiv_\lambda^{\mathrm{proj}}$ is $P \big/ {\equiv_\lambda^{\mathrm{proj}}} = \Big\{\{\{1, 2\}, \{5, 6\}\}, \{\{1, 4\}, \{3, 6\}\}\Big\}$. The projections $\mathrm{proj}(\lambda(P)) = \big\{\{1, 2\}, \{3\}\big\}$ of the labels partition the support $\mathrm{supp}(\lambda(P)) = \{1, 2, 3\}$.

Given a projection $S \in \mathrm{proj}(\lambda(P))$ of some label in the set of labels of a coherent K-labelling λ, more than one label in $\lambda(P)$ may have S as its projection. Define the sets $\mathrm{pal}(S)$ and $\mathrm{apal}(S)$ to be the collections of palindromic and anti-palindromic sets of symbols, respectively, which have projection S. Furthermore, define $\mathrm{coh}(S) = \mathrm{pal}(S) \cup \mathrm{apal}(S)$. Then $\mathrm{pal}(S)$ is precisely the set of labels with projection S which could occur in the set of labels of a coherent K-labelling. However, not all of the members of $\mathrm{coh}(S)$ may occur in the set of labels of a single coherent K-labelling. The labels which have projection S are precisely the labels of the pairs in $\lambda^{-1}(\mathrm{proj}^{-1}(S))$, which constitute one of the equivalence classes $[x]_{\equiv_\lambda^{\mathrm{proj}}}$ in $P \big/ {\equiv_\lambda^{\mathrm{proj}}}$. For any two pairs x and y in $[x]_{\equiv_\lambda^{\mathrm{proj}}}$, coherence of λ implies that the labels of x and y must either be the same, negations of each other, or both. Therefore, fixing one member $x_0 \in [x]_{\equiv_\lambda^{\mathrm{proj}}}$, all other members y of $[x]_{\equiv_\lambda^{\mathrm{proj}}}$ must have label $\lambda(y) \in \{-\lambda(x), \lambda(x), \pm\lambda(x)\}$. If $\lambda(y) = \pm\lambda(x)$, then by coherence of λ, all members of $[x]_{\equiv^{\text{proj}}_{\lambda}}$, have the same label. Thus, there are at most two distinct labels in $\lambda(P)$ which have the same projection. In Figures 2.7(b) and (c), the set $S = \{1,2\}$ is a member of the set of projections of the labels. In this case, $\text{pal}(S) = \big\{\{1,2\},\{1,-2\},\{-1,2\},\{-1,-2\}\big\}$ and $\text{apal}(S) = \big\{\{-1,1,-2,2\}\big\}$. The pairs $\{1,4\}$ and $\{3,6\}$ are the two pairs whose labels have S as their projection. The labels of these two pairs are members of $\text{apal}(S)$ and are negations of each other. In Figure 2.8(c), the projection of the set of labels of the depicted 3-labelling λ of the set of pairs P is $\text{proj}(\lambda(P)) = \big\{S_1 = \{1\}, S_2 = \{2,3\}\big\}$. Here, the set of pairs whose labels have projection S_1 are $\lambda^{-1}(\text{proj}^{-1}(S_1)) = \big\{\{1,2\},\{4,5\}\big\}$, and both pairs have the same label $\{-1\} \in \text{apal}(S_1)$. The single pair whose label has projection S_2 is $\{3,6\}$ and its label is the only member $\{-2,2,-3,3\}$ of $\text{pal}(S_2)$.

Theorem 2.3.5. *Let $P \subseteq \binom{Z_N}{2}$. Then:*

$$|\hat{\Lambda}_{K,\text{coh}}(P)| = \sum_{h=1}^{\min\{K,|P|\}} \sum_{\mathcal{S}\in\left\{Z_K \atop h\right\}} \sum_{\mathcal{P}\in\left\{P \atop h\right\}} \sum_{f:\mathcal{P}\xrightarrow{bij}\mathcal{S}} \prod_{X\in\mathcal{P}} \big(1 + 2^{|X|}\cdot 2^{|f(X)|}\cdot \frac{1}{2}\big),$$

so that

$$2^K \cdot 2^{|P|} \sum_{h=1}^{\min\{K,|P|\}} 2^{-h}\cdot h! \cdot \left\{K \atop h\right\}\left\{|P| \atop h\right\} \leq |\hat{\Lambda}_{K,\text{coh}}(P)| < 2^K \cdot 2^{|P|} \sum_{h=1}^{\min\{K,|P|\}} h! \cdot \left\{K \atop h\right\}\left\{|P| \atop h\right\}$$

Proof. The proof proceeds by counting in how many ways P can be partitioned into sets which are the equivalence classes under $\equiv^{\text{proj}}_{\lambda}$ for coherent K-labellings λ. Given such a partitioning $\mathcal{P}$ of P, we then count in how many ways Z_K can be partitioned into the same number of classes and assigned to the members of $\mathcal{P}$ as the projections of the labels of these equivalence classes according to coherent K-labellings. With partitionings $\mathcal{P}$ and $\mathcal{S}$ of P and Z_K, respectively, and a fixed bijective correspondence $f : \mathcal{P} \longrightarrow \mathcal{S}$, we then only need to count the number of coherent K-labellings λ of P with $P \big/ _{\equiv^{\text{proj}}_{\lambda}} = \mathcal{P}$, $\text{proj}(\lambda(P)) = \mathcal{S}$, and $\text{proj} \circ \lambda = f$.

Observe that any number of distinct projections up to $\min\{K,|P|\}$ can be realized by some coherent K-labelling of P. To see this, let $m \in \min\{K,|P|\}$ and pick an arbitrary $m-1$-element subset P' of P. Put the pairs in P' in any order $x_1,\ldots,x_{m-1}$ and label each x_i with the label $\{i\}$. Then label the remaining pairs in $P\backslash P'$ with the label $Z_K\backslash Z_{m-1}$. The resulting labelling λ is complete since $\big(Z_K\backslash Z_{m-1}\big)\cup\big(\cup_{i\in Z_{m-1}}\{i\}\big) = Z_K$. Furthermore, $\big(\cup_{x\in P}\lambda(x)\big) \cap -Z_K = \emptyset$, so that $\lambda(x) \cap -\lambda(y) = \emptyset$ for all $x, y \in P$. The labels of the pairs $x_1,\ldots,x_{m-1}$ all have unique and disjoint supports, which are also disjoint from $Z_K \setminus Z_{m-1}$. Since all members of $P \setminus P'$ have the same label $Z_K \setminus Z_{m-1}$, so that λ is coherent. Of course Z_K and P cannot be partitioned into more than $\min\{K, |P|\}$ sets, so that:

$$\hat{\Lambda}_{K,\text{coh}}(P) = \bigcup_{m=1}^{\min\{K,|P|\}} \bigcup_{\mathcal{P}\in\left\{{P \atop m}\right\}} \bigcup_{\mathcal{S}\in\left\{{Z_K \atop m}\right\}} \bigcup_{f:\mathcal{P}\xrightarrow{bij}\mathcal{S}} \hat{\Lambda}_{K,\text{coh}}(\mathcal{P}, \mathcal{S}, f), \tag{2.2}$$

where $\hat{\Lambda}_{K,\text{coh}}(\mathcal{P}, \mathcal{S}, f)$ is the set of complete coherent K-labellings λ of P satisfying $\left(P \Big/ \equiv^{\text{proj}}_{\lambda}\right) = \mathcal{P}$, $\text{proj}(\lambda(P)) = \mathcal{S}$, and $\text{proj} \circ \lambda = f$. Observe that the sets $\hat{\Lambda}_{K,\text{coh}}(\mathcal{P}, \mathcal{S}, f)$ are disjoint for each distinct combination of parameters $\mathcal{P}$, $\mathcal{S}$, and f. Thus the expression in Equation 2.2 consists of disjoint unions.

Fix $m \in Z_{\min\{K,|P|\}}$, $\mathcal{P} \in \left\{{P \atop m}\right\}$, $\mathcal{S} \in \left\{{Z_K \atop m}\right\}$, and let $f : \mathcal{P} \longrightarrow \mathcal{S}$ be a bijection. For any choice of labels formed from the members S of $\mathcal{S}$ by choosing one member A_S of $\text{coh}(S)$, the K-labelling which assigns to all pairs in each of the members $X \in \mathcal{P}$ one of the labels $A_{f(X)}$ or $A_{-f(X)}$ is complete and coherent.

The collection $\text{apal}(S)$ has size $2^{|S|}$ since every member A of $\text{apal}(S)$ can be obtained by setting for each symbol a in S either $a \in A$, or $-a \in A$, but not both. Each of the outcomes of the $|S|$ binary decisions produces a unique member of $\text{apal}(S)$, so that $|\,\text{apal}(S)| = 2^{|S|}$. Since the only palindromic set of symbols with projection S is $\pm S$, we have that $|\,\text{pal}(S)| = 1$. Then for every $X \in \mathcal{P}$, there are precisely $1 + 2^{|f(X)|}$ choices for set $A_{f(X)}$. Once the sets A_S for $S \in \mathcal{S}$ are chosen, each pair in X must either have label $A_{f(X)}$, or $-A_{f(X)}$. When $A_{f(X)} = \pm A_{f(X)} \in \text{pal}(F(X))$, then all members of X have the same label $A_{f(X)}$. Otherwise, there are $2^{|X|}$ ways of assigning either $A_{f(X)}$, or $-A_{f(X)}$, to the pairs $x \in X$. The sum

$$\sum_{X\in\mathcal{P}} 2^{|X|} \cdot 2^{|f(X)|}$$

counts every member of $\hat{\Lambda}_{K,\text{coh}}(\mathcal{P}, \mathcal{S}, f)$ twice, since the same choice of labels $A_{f(X),x}$ for the pairs $x \in X$, and $X \in \mathcal{P}$ is counted a second time when each S is assigned the set $B_S = -A_S$ and each pair $x \in X$, for $X \in \mathcal{P}$ is assigned the label $-B_{f(X)}$. Thus,

$$|\hat{\Lambda}_{K,\text{coh}}(\mathcal{P}, \mathcal{S}, f)| = 1 + 2^{|X|} \cdot 2^{|f(X)|} \cdot \frac{1}{2}. \tag{2.3}$$

Combining Equations 2.2 and 2.3 gives the first part of the result.

For the lower bound, observe that the sets $X \in \mathcal{P}$ partition P so that their sizes add up to $|P|$, and that the sets $f(X)$ for $X \in \mathcal{P}$ are precisely the sets $S \in \mathcal{S}$ which partition Z_K, so that the sizes of the sets $f(X)$ add up to K. Then:

$$\prod_{X\in\mathcal{P}} 1 + 2^{|X|} \cdot 2^{|f(X)|} \cdot \frac{1}{2} \geq \prod_{X\in\mathcal{P}} 2^{|X|} \cdot 2^{|f(X)|} \cdot \frac{1}{2} = 2^K \cdot 2^{|P|} \cdot 2^{-|\mathcal{P}|}. \tag{2.4}$$

The only variable on the right-hand-side of the inequatlity in 2.4 which depends on $h, \mathcal{S}, \mathcal{P}$ or f is the size of $\mathcal{P}$, which is $|\mathcal{P}| = h$. Since there are $\left\{{K \atop h}\right\}$ and $\left\{{|P| \atop h}\right\}$ ways of partitioning Z_K and P, respectively, into h parts, and since there are $h!$ bijections between two h-element sets, rearranging the terms in the expression for $|\hat{\Lambda}_{K,\mathrm{coh}}(P)|$ and using the inequality in 2.4, gives the lower bound.

The upper bound is obtained the same way, except that instead of the inequality in 2.4, we use the fact that

$$1 + 2^{|X|} \cdot 2^{|f(X)|} \cdot \frac{1}{2} < 2^{|X|} \cdot 2^{|f(X)|},$$

for $|X|, |f(X)| > 0$. $\square$

As mentioned before, the pairs in a set of pairs P which is labelled by a K-labelling λ must be labelled either palindromically, or anti-palindromically. Denote with P_λ^{pal} the set of pairs in P which have palindromic labels and denote with $P_\lambda^{\mathrm{apal}}$ the set of pairs in P which have anti-palindromic labels. Then $P = P_\lambda^{\mathrm{pal}} \sqcup P_\lambda^{\mathrm{apal}}$. The difference in size of the sets $\mathrm{pal}(S)$ and $\mathrm{apal}(S)$ suggests that the size of $\hat{\Lambda}_{K,\mathrm{coh}}(P)$ varies greatly depending on the number of pairs which have palindromic labels. To confirm this observation, we now count the two sets

$$\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{pal}}(P) = \{\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(P) : \lambda(x) = \pm\lambda(x), \forall x \in P\}$$

and

$$\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{apal}}(P) = \{\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(P) : \lambda(x) \cap -\lambda(x) = \emptyset, \forall x \in P\}$$

of complete and coherent K-labellings λ of P where $P = P_\lambda^{\mathrm{pal}}$ and $P = P_\lambda^{\mathrm{apal}}$, respectively.

Proposition 2.3.6. *Let $P \subseteq \binom{Z_N}{2}$. Then:*

$$|\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{pal}}(P)| = \sum_{h=1}^{\min\{K,|P|\}} h! \cdot \left\{{K \atop h}\right\}\left\{{|P| \atop h}\right\}.$$

Proof. The proof works the same way as the proof of Theorem 2.3.5, except that for $h \in Z_{\min\{K,|P|\}}$, partitions $\mathcal{P} \in \left\{{P \atop h}\right\}$, $\mathcal{S} \in \left\{{Z_K \atop h}\right\}$, and bijection $f : \mathcal{P} \to \mathcal{S}$, the set $\hat{\Lambda}_{K,\mathrm{coh}}(\mathcal{P}, \mathcal{S}, f) \cap \hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{pal}}(P)$ contains only one K-labelling. The only K-labelling in this intersection is uniquely defined by $\lambda(x) = \pm f(X)$, for all $x \in X$, $X \in \mathcal{P}$. $\square$

Proposition 2.3.7. *Let $P \subseteq \binom{Z_N}{2}$. Then:*

$$|\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{apal}}(P)| = 2^K \cdot 2^{|P|} \sum_{h=1}^{\min\{K,|P|\}} 2^{-h} \cdot h! \cdot \left\{{K \atop h}\right\} \cdot \left\{{|P| \atop h}\right\}.$$

Proof. Again, the proof works the same way as the proof of Theorem 2.3.5, except that in this case, for $h \in Z_{\min\{K,|P|\}}$, partitions $\mathcal{P} \in \left\{ {P \atop h} \right\}$, $\mathcal{S} \in \left\{ {Z_K \atop h} \right\}$, and bijection $f : \mathcal{P} \to \mathcal{S}$, the set $\hat{\Lambda}_{K,\mathrm{coh}}(\mathcal{P}, \mathcal{S}, f) \cap \hat{\Lambda}^{\mathrm{apal}}_{K,\mathrm{coh}}(P)$ has size $2^K \cdot 2^{|P|} \cdot 2^{-h} = \prod_{X \in \mathcal{P}} 2^{|X|} \cdot 2^{|f(X)|} \cdot \frac{1}{2}$. The set $\hat{\Lambda}_{K,\mathrm{coh}}(\mathcal{P}, \mathcal{S}, f) \cap \hat{\Lambda}^{\mathrm{apal}}_{K,\mathrm{coh}}(P)$ is counted the same way as $\hat{\Lambda}_{K,\mathrm{coh}}(\mathcal{P}, \mathcal{S}, f)$ was counted in the proof of Theorem 2.3.5, except that the sets A_S, for $S \in \mathcal{S}$ are chosen from $\mathrm{apal}(S) = \mathrm{coh}(S) \setminus \{\pm S\}$ instead of $\mathrm{coh}(S)$. □

Observe that any set of pairs $P \subseteq \binom{Z_N}{2}$ can be completely and coherently labelled by the labelling which assigns $\pm ZK$ to all pairs in P. Then $\hat{\Lambda}^{\mathrm{pal}}_{K,\mathrm{coh}}(P) \neq \emptyset$, for any K, P. Furthermore, the size of $\hat{\Lambda}^{\mathrm{apal}}_{K,\mathrm{coh}}(P)$ given in Proposition 2.3.7 is the same as the lower bound for the size of $\hat{\Lambda}_{K,\mathrm{coh}}(P)$ given in Theorem 2.3.5. Thus, the lower bound in this theorem is never achieved. Observe that Propositions 2.3.6 and 2.3.7 assert that the sizes of $\hat{\Lambda}^{\mathrm{pal}}_{K,\mathrm{coh}}(P)$ and $\hat{\Lambda}^{\mathrm{apal}}_{K,\mathrm{coh}}(P)$ depend only on the sizes of P and Z_K. Thus, we may define for any integer:

$$\ell^{\mathrm{pal}}_{\mathrm{coh}}(K, p) = |\hat{\Lambda}^{\mathrm{pal}}_{K,\mathrm{coh}}(P)|,$$

and

$$\ell^{\mathrm{apal}}_{\mathrm{coh}}(K, p) = |\hat{\Lambda}^{\mathrm{apal}}_{K,\mathrm{coh}}(P)|,$$

where P is any set of pairs of size $|P| = p$. Similarly, the count in Theorem 2.3.5 allows us to define:

$$\ell_{\mathrm{coh}}(K, p) = |\hat{\Lambda}_{K,\mathrm{coh}}(P)|,$$

where P is any set of pairs of size $|P| = p$.

Propositions 2.3.6 and 2.3.7 suggest another way of counting the set $\hat{\Lambda}_{K,\mathrm{coh}}(P)$. The strategy in the following result (Theorem 2.3.8) is to count in how many ways each of P and Z_K can be split into two parts $P = P_1 \sqcup P_2$ and $Z_K = S_1 \sqcup S_2$ and adding for each of these choices the sizes of $\hat{\Lambda}^{\mathrm{pal}}_{|S_1|,\mathrm{coh}}(P_1)$ and $\hat{\Lambda}^{\mathrm{apal}}_{|S_2|,\mathrm{coh}}(P_2)$.

Corollary 2.3.8. *Let p be a positive integer. Then:*

$$\ell_{\mathrm{coh}}(K, p) = \sum_{K_{\mathrm{pal}}=0}^{K} \sum_{p_{\mathrm{pal}}=0}^{p} \binom{p}{p_{\mathrm{pal}}} \binom{K}{K_{\mathrm{pal}}} \ell^{\mathrm{pal}}_{\mathrm{coh}}(K_{\mathrm{pal}}, p_{\mathrm{pal}}) \cdot \ell^{\mathrm{apal}}_{\mathrm{coh}}(K - K_{\mathrm{pal}}, p - p_{\mathrm{pal}}),$$

where we use the convention that $\ell^{\mathrm{pal}}_{\mathrm{coh}}(K', p') = \ell^{\mathrm{apal}}_{\mathrm{coh}}(K', p') = 0$ whenever exactly one of K' or p' is 0 and $\ell^{\mathrm{pal}}_{\mathrm{coh}}(0, 0) = \ell^{\mathrm{apal}}_{\mathrm{coh}}(0, 0) = 1$.

Proof. Let $\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(P)$. Then λ uniquely defines the partitionings $P = P_{\mathrm{pal}} \sqcup P_{\mathrm{apal}}$ and $Z_K = S_{\mathrm{pal}} \sqcup S_{\mathrm{apal}}$, where $P_{\mathrm{pal}} = P^{\mathrm{pal}}_{\lambda}$, $P_{\mathrm{apal}} = P^{\mathrm{apal}}_{\lambda}$, $S_{\mathrm{pal}} = \mathrm{supp}(\lambda(P^{\mathrm{pal}}_{\lambda}))$ and $S_{\mathrm{apal}} = \mathrm{supp}(\lambda(P^{\mathrm{apal}}_{\lambda}))$. Additionally, λ uniquely defines the members $\lambda_{\text{pal}} \in \hat{\Lambda}_{K,S_{\text{pal}},\text{coh}}(P_\lambda^{\text{pal}})$ and $\lambda_{\text{apal}} \in \hat{\Lambda}_{K,S_{\text{apal}},\text{coh}}(P_\lambda^{\text{apal}})$, satisfying:

$$\begin{cases} \lambda_{\text{pal}}(x) = \pm x, & x \in P_{\text{pal}}, \text{ and} \\ \lambda_{\text{apal}}(y) \cap -\lambda_{\text{apal}}(y) = \emptyset, & y \in P_{\text{apal}}, \end{cases} \tag{2.5}$$

and

$$\lambda|_{P_\lambda^{\text{pal}}} = \lambda_{\text{pal}} \text{ and } \lambda|_{P_\lambda^{\text{apal}}} = \lambda_{\text{apal}}. \tag{2.6}$$

Conversely, all partitionings $P = P_{\text{pal}} \sqcup P_{\text{apal}}$ and $Z_K = S_{\text{pal}} \sqcup S_{\text{apal}}$ and every choice of $\lambda_{\text{pal}} \in \hat{\Lambda}_{K,S_{\text{pal}},\text{coh}}(P_{\text{pal}})$ and $\lambda_{\text{apal}} \in \hat{\Lambda}_{K,S_{\text{apal}},\text{coh}}(P_{\text{apal}})$, which satisfy 2.5, uniquely define a member $\lambda \in \hat{\Lambda}_{K,\text{coh}}(P)$, where $S_{\text{pal}} = \text{supp}(\lambda(P_\lambda^{\text{pal}}))$, and $S_{\text{apal}} = \text{supp}(\lambda(P_\lambda^{\text{apal}}))$, which satisfies 2.6. □

2.3.2 Consistent Labellings

The reasons the property of coherence was introduced is that a pair with multiple labels implies that the sequences in the product defined by the MDS identifiers in the label must all be the same sequence as in the precursor defined by the pair (or highly similar to it). As a consequence, any label containing one of the MDS identifiers of another label must contain all of them. This argument can be extended to pairs enclosed in other pairs. When a pair's coordinates are between the coordinates of another, the sequence described by the inner pair is a subsequence of the outer pair. Any appearances of MDS identifiers from the outer label in the label of another pair imply the presence of that subsequence inside the sequence defined by that pair, as well. Consequently, any pair labelled with one of the MDS identifiers from the outer pair must enclose a pair marking the same subsequence with the same label as in the original inner pair. The property of consistence reflects this behavior.

Define the relation $\sqsupseteq$ on the set $\binom{Z_N}{2}$ by:

$$\{i_1, j_1\} \sqsupseteq \{i_2, j_2\} \Longleftrightarrow i_1 \leq i_2 < j_2 \leq j_1.$$

Define the **closure** of a pair x in a set of pairs P by:

$$\bar{x} = \{y \in P : x \sqsupseteq y\}.$$

For a subset $X \subseteq P$ of a set of pairs P, define the **domain** of X by:

$$\text{dom}(X) = \bigcup_{x \in X} x.$$

Denote with $G_P[x]$ the subgraph of $G(P)$ induced by the vertices in $\mathrm{dom}(\bar{x})$. In other words, $G_P[x]$ is the subgraph of $G(P)$ with vertex set $\mathrm{dom}(\bar{x})$ and edge set $\bar{x}$. In Figure 2.7, we have that $\{1,4\} \sqsupseteq \{1,2\}$ and $\{3,6\} \sqsupseteq \{5,6\}$. Here, the closures of $\{1,4\}$ and $\{3,6\}$ are $\big\{\{1,2\},\{1,4\}\big\}$ and $\big\{\{3,6\},\{5,6\}\big\}$, respectively. Additionally the domain of the closure of $\{1,4\}$ is the set $\{1,2,4\}$ and the domain of the closure of $\{3,6\}$ is the set $\{3,5,6\}$. The subgraphs $G_P[\{1,4\}]$ and $G_P[\{3,6\}]$ of $G(P)$ are the graphs with vertex sets $\{1,2,4\}$ and $\{3,5,6\}$ and with edge sets $\big\{\{1,2\},\{1,4\}\big\}$, and $\big\{\{3,6\},\{5,6\}\big\}$, respectively.

Recall that $\sigma^{\mathrm{fwd}}_{A,B}$ denotes the unique order isomorphism between the totally ordered sets $(A,<)$ and $(B,<)$. Denote with $\sigma^{\mathrm{rev}}_{A,B}$, the unique order isomorphism between the totally ordered sets $(A,>)$ and $(B,<)$. To shorten notation, we denote with $\sigma^{\mathrm{fwd}}_{x,y}$ and $\sigma^{\mathrm{rev}}_{x,y}$ the functions $\sigma^{\mathrm{fwd}}_{\mathrm{dom}(\bar{x}),\mathrm{dom}(\bar{y})}$ and $\sigma^{\mathrm{rev}}_{\mathrm{dom}(\bar{x}),\mathrm{dom}(\bar{y})}$, respectively, when x and y are pairs from a set of pairs $P \subseteq \binom{Z_N}{2}$. For a K-labelling λ of P, define the relations $\cong^{\mathrm{fwd}}_{P,\lambda}$ and $\cong^{\mathrm{rev}}_{P,\lambda}$ on P by:

$$x \cong^{\mathrm{fwd}}_{P,\lambda} y \iff \begin{cases} |\bar{x}| = |\bar{y}|, \\ \sigma^{\mathrm{fwd}}_{x,y} \text{ is a graph isomorphism between } G_P[x] \text{ and } G_P[y], \text{ and} \\ \sigma^{\mathrm{fwd}}_{x,y} \text{ preserves edge labels,} \end{cases}$$

and

$$x \cong^{\mathrm{rev}}_{P,\lambda} y \iff \begin{cases} |\bar{x}| = |\bar{y}|, \\ \sigma^{\mathrm{rev}}_{x,y} \text{ is a graph isomorphism between } G_P[x] \text{ and } G_P[y], \text{ and} \\ \sigma^{\mathrm{rev}}_{x,y} \text{ negates edge labels.} \end{cases}$$

In Figure 2.7, the functions $\sigma^{\mathrm{fwd}}_{\{1,2\},\{5,6\}}$ and $\sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}$ are graph isomorphisms between the two graphs $G_P[\{1,2\}]$ and $G_P[\{5,6\}]$, and between the two graphs $G_P[\{1,4\}]$ and $G_P[\{3,6\}]$, respectively. For the 4-labelling λ_a in Figure 2.7(a), $\{1,2\} \cong^{\mathrm{fwd}}_{P,\lambda_a} \{5,6\}$. However $\lambda_a(\{1,4\}) = \{1,2\} \neq -\{-1,1,2,4\} = \lambda_a\big(\{\sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}(1), \sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}(4)\}\big)$, so that $\sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}$ does not negate labels and thus $\{1,4\} \ncong^{\mathrm{rev}}_{P,\lambda_a} \{3,6\}$. For the 4-labelling λ_b in Figure 2.7(b), the label of $\{1,4\}$ is the negation of the label of $\{3,6\} = \big\{\sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}(1), \sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}(4)\big\}$. However, $\{1,2\}$ is in the closure of $\{1,4\}$ (and therefore in the subgraph $G_P[\{1,4\}]$) and is mapped by $\sigma^{\mathrm{rev}}_{\{1,4\},\{3,6\}}$ into the pair $\{5,6\}$. Then since $\lambda_b(\{1,2\}) \neq -\lambda_b(\{5,6\})$, we have again that $\{1,4\} \ncong^{\mathrm{rev}}_{P,\lambda_b} \{3,6\}$. With the labelling λ_c from Figure 2.7(c), on the other hand, we do have $\{1,4\} \cong^{\mathrm{rev}}_{P,\lambda_c} \{3,6\}$.

Definition 2.3.9. Let $P \subseteq \binom{Z_N}{2}$. A K-labelling λ of P is called **consistent** if for all $x, y \in P$:

(i) If $\lambda(x) \cap \lambda(y) \neq \emptyset$, then $x \cong^{\mathrm{fwd}}_{P,\lambda} y$, and

(ii) If $\lambda(x) \cap -\lambda(y) \neq \emptyset$, then $x \cong^{\mathrm{rev}}_{P,\lambda} y$.

Denote with $\Lambda_{K,\text{con}}(P)$ the set of all consistent K-labellings of P and with $\Lambda_{K,S,\text{con}}(P)$, $\hat{\Lambda}_{K,S,\text{con}}(P)$, and $\hat{\Lambda}_{K,\text{con}}(P)$ the intersections of $\Lambda_{K,\text{con}}(P)$ with $\Lambda_{K,S}(P)$, $\hat{\Lambda}_{K,S}(P)$, and $\hat{\Lambda}_K(P)$, respectively.

Observe that the 4-labellings λ_a and λ_b of the pairing P in Figures 2.7(a) and (b), respectively, are inconsistent because $\lambda_a(\{1,4\}) \cap \lambda_a(\{3,6\}) \neq \emptyset$, but $\{1,4\} \ncong^{\text{fwd}}_{P,\lambda_a} \{3,6\}$, and because $\lambda_b(\{1,4\}) \cap -\lambda_b(\{3,6\}) \neq \emptyset$, but $\{1,4\} \ncong^{\text{rev}}_{P,\lambda_b} \{3,6\}$. Note that λ_b is coherent despite it being inconsistent. The labelling depicted in Figure 2.7(c), on the other hand, is consistent. The labelling λ depicted in Figure 2.9 is consistent. Observe that $\{13,18\} \cong^{\text{rev}}_{P,\lambda} \{13,18\}$.

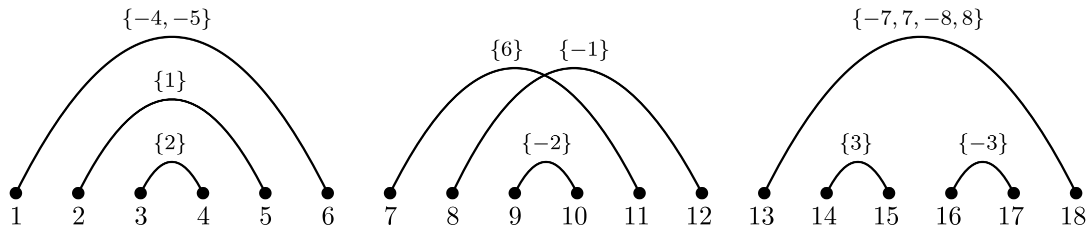


Figure 2.9: A set of pairs $P \subset \binom{Z_{18}}{2}$ together with a consistent 8-labelling.

The next two results (Proposition 2.3.10 and Theorem 2.3.11) establish that the property of consistency of K-labellings is relevant only when the underlying set of pairs has members which are contained in other members and that the requirement of coherence becomes the same as consistency when no pair contains another.

Proposition 2.3.10. *Let $P \subseteq \binom{Z_N}{2}$ and let λ be a K-labelling of P. If for two distinct $x, y \in P$, $x \sqsupseteq y$ and $\lambda(x) \cap \pm\lambda(y) \neq \emptyset$, then λ is not consistent.*

Proof. Suppose λ is consistent, $x \neq y$, $x \sqsupseteq y$ and $\lambda(x) \cap \pm\lambda(y) \neq \emptyset$. Then either $x \cong^{\text{fwd}}_{P,\lambda} y$, or $x \cong^{\text{rev}}_{P,\lambda} y$. In either case, $G_P[x]$ is isomorphic to $G_P[y]$. But $V(G_P[y]) = \bar{y} \subsetneq \bar{x} = V(G_P[x])$, so that $G_P[x]$ is isomorphic to its proper subgraph $G_P[y]$, which is a contradiction. □

Theorem 2.3.11. *Let $P \subseteq \binom{Z_N}{2}$. Then $\Lambda_{K,\text{con}}(P) \subseteq \Lambda_{K,\text{coh}}(P)$ and equality holds if and only if there is no $x \neq y \in P$ with $x \sqsupseteq y$.*

Proof. Let $\lambda \in \Lambda_{K,\text{con}}(P)$ and suppose $\lambda(x) \cap \lambda(y) \neq \emptyset$, for some $x, y \in P$. Then by consistency of λ, $x \cong^{\text{fwd}}_{P,\lambda} y$, so that $\sigma^{\text{fwd}}_{x,y}$ is a graph isomorphism between $G_P[x]$ and $G_P[y]$ which preserves labels. Then $\lambda(x) = \lambda(\sigma^{\text{fwd}}_{x,y}(x)) = \lambda(y)$. Similarly, $\lambda(x) \cap -\lambda(y) \neq \emptyset$ implies $\lambda(x) = -\lambda(\sigma^{\text{rev}}_{x,y}(x)) = -\lambda(y)$, so that λ is coherent.

Now, suppose there is no $x \neq y \in P$ with $x \sqsupseteq y$. Then $\bar{x} = \{x\}$ for every $x \in P$, so that $\sigma^{\text{fwd}}_{x,y}$ and $\sigma^{\text{rev}}_{x,y}$ are graph isomorphisms for any $x, y \in P$.

Let $\lambda \in \Lambda_{K,\mathrm{coh}}(P)$ and suppose $\lambda(x) \cap \lambda(y) \neq \emptyset$. Then by coherence of λ, $\lambda(x) = \lambda(y) = \lambda(\sigma^{\mathrm{fwd}}_{x,y}(x))$. Thus, $x \cong^{\mathrm{fwd}}_{P,\lambda} y$. If instead $\lambda(x) \cap -\lambda(y) \neq \emptyset$, then coherence of λ implies $\lambda(x) = -\lambda(y) = -\lambda(\sigma^{\mathrm{rev}}_{x,y}(x))$, so that $x \cong^{\mathrm{rev}}_{P,\lambda} y$. Thus, $\lambda \in \Lambda_{K,\mathrm{con}}(P)$.

Conversely, if there are $x \neq y \in P$ with $x \sqsupseteq y$, consider the labelling λ of P, where $\lambda(z) = \pm Z_K$, for all $z \in P$. Then λ is complete and coherent, but by Proposition 2.3.10 and since $\lambda(x) = \lambda(y)$, we have that λ is inconsistent. □

The property of consistency of a K-labelling of a set of pairs P closely relates to the structure of the set of pairs with respect to the relation $\sqsupseteq$. Define the **height** $h_P(x)$ of a pair $x \in P$ to be the maximum length h of sequences $y_1, \ldots, y_h$ of distinct pairs $y_j \in P \setminus \{x\}$, such that $x \sqsupseteq y_1 \sqsupseteq \cdots \sqsupseteq y_h$:

$$h_P(x) = \max\{h : \exists \text{ distinct } y_1, \ldots, y_h \in P \setminus \{x\} \text{ where } x \sqsupseteq y_1 \sqsupseteq \cdots \sqsupseteq y_h\}.$$

Denote the maximum height of all pairs in P by $h(P)$:

$$h(P) = \max\{h_P(x) : x \in P\}.$$

For $h \in \{0, 1, \ldots, h(P)\}$, define the h^{th} **level of** P to be the set $L_h(P)$ of all pairs in P with height h:

$$L_h(P) = \{x \in P : h_P(x) = h\}.$$

In Figure 2.9, $h(P) = 2$. The 0^{th} level of the depicted set of pairs consists of the pairs $\{3,4\}, \{9,10\}, \{14,15\}$ and $\{16,17\}$. The first level of pairs consists of the pairs $\{2,5\}, \{7,11\}, \{8,12\}$ and $\{13,18\}$, and the third level consist of the pair $\{1,6\}$.

Counting in how many ways a set of pairs P can be labelled consistently is complicated by the restrictions consistency imposes in addition to being coherent. Whether or not in a consistent labelling the labels of two pairs may intersect each other, or the negations of each other, depends on the structure and labels of the pairs in their closures. One recursive strategy to counting consistent K-labellings of a set of pairs P begins with counting in how many ways the 0^{th} level $L_0(P)$ can be labelled coherently. No pair in the 0^{th} level of P contains any other pair, so that by Theorem 2.3.11, coherence suffices to guarantee consistency of the labelling of the 0^{th} level. Then for each way of labelling the 0^{th} level, the number of ways of how labels can be added to the pairs of the first level $L_1(P)$, while ensuring that the overall labelling of $L_0(P) \cup L_1(P)$ is consistent, is counted. The recursion continues by counting in how many ways each h^{th} level can be labelled consistently for each of the consistent labellings of $L_0(P) \cup \cdots \cup L_{h-1}(P)$. With the labels of all pairs of level up to $h-1$ known, consistency of a labelling of the h^{th} level can be checked since each pair in the closure of a member of $L_h(P)$ has a label. The 8-labelling in Figure 2.9, for example, can be obtained using this strategy as follows:

1. Choose a subset $S_1 \subset Z_8$ from the set of symbols for the 0^{th} level of P. Here, the chosen subset S_1 of symbols is the set $S_1 = \{2, 3\}$. The pairs with height 0 are $\{3, 4\}, \{9, 10\}, \{14, 15\}$ and $\{16, 17\}$.

2. Label $L_0(P)$ coherently with labels formed from the set of symbols S_1. By Theorem 2.3.11, the resulting labelling of $L_0(P)$ is consistent. Here, we chose the labels $\{2\}, \{-2\}, \{3\}$, and $\{-3\}$, for the respective pairs with height 0.

3. Choose a subset $S_2 \subset Z_8 \setminus S_1$ from the set of remaining symbols for the first level of P. Here, the chosen subset S_2 of symbols is the set $S_2 = \{1, 6, 7, 8\}$. The pairs with height 1 are $\{2, 5\}, \{7, 11\}, \{8, 12\}$ and $\{13, 18\}$.

4. Label $L_1(P)$ consistently with labels formed from the set of symbols S_1. Consistency of any such labelling can be checked since any pair $x \in P$, which is not a member of $L_1(P)$ but which is in the closure of a member of $L_1(P)$ was already labelled in the previous step. This step is trickier than for the 0^{th} level, since coherence of the labelling of $L_1(P)$ may not suffice to guarantee consistency. For example, only the label of $\{13, 18\}$ may be palindromic, since this pair is the only pair where the reversal of the end points of other pairs contained in the pair is a graph isomorphism which negates labels. Additionally, the symbols used for labels of the pairs $\{\{2, 5\}, \{7, 11\}$, and $\{8, 12\}$ must differ from the symbols used for the label of $\{13, 18\}$ because the labels of the closure of the latter pair contain symbols which are not included in the closures of the other pairs in this level. This means that no label-preserving or label-negating graph isomorphism can exist between the closure of $\{13, 18\}$ and the closure of any of the other pairs with height 1. Lastly, if we decide to use the same symbol in the label of more than one pair x_1, x_2, we must ensure that at least one of $\sigma^{\text{fwd}}_{x_1,x_2}$, or $\sigma^{\text{rev}}_{x_1,x_2}$ is a graph isomorphism which preserves labels, or negates labels, respectively, which depends on the labels and structure of other pairs in the closures of x_1 and x_2. Here, we chose the labels $\{1\}, \{6\}, \{-1\}$, and $\{-7, 7, -8, 8\}$, for the respective pairs with height 1.

5. For the second and last level of pairs, which consists only of the pair $\{1, 6\}$ we used the remaining symbols $S_3 = Z_8 \setminus (S_1 \cup S_2)$. Similar considerations to the previous step are made. Here, we chose the anti-palindromic label $\{-4, -5\}$ and the resulting labelling of P is consistent.

Observe that a palindromic label can only be assigned to a pair $x \in P$ if $\sigma^{\text{rev}}_{\text{dom}(\bar{x})\setminus\{x\},\text{dom}(\bar{x})\setminus\{x\}}$ is an isomorphism of the subgraph of $G(P)$ induced by the vertex set $\text{dom}(\bar{x}) \setminus \{x\}$ with itself, which negates labels. This does not mean that each of the pairs in $\mathrm{dom}(\bar{x}) \setminus \{x\}$ must have a palindromic label. For example, the pair $\{13, 18\}$ in Figure 2.9 is labelled palindromically, but neither $\{14, 15\}$, nor $\{16, 17\}$ are labelled palindromically. Thus, when recursively counting the set of consistent labellings of a set of pairs P level by level, we must be able to distinguish between pairs which can be labelled palindromically and which cannot. Call a set of pairs P **reverse-complementary** if $\sigma^{\mathrm{rev}}_{Z_N,Z_N}$ induces a graph isomorphism of $G(P)$ with itself. Define a K-labelling λ of a set of pairs P to be **l-reverse-complementary** if P is reverse-complementary and, $\sigma^{\mathrm{rev}}_{Z_N,Z_N}$ negates edge labels. The pairing in Figure 2.7 is reverse-complementary. Additionally, the coloring depicted in Figure 2.7(c) is l-reverse-complementary, since reversal of the vertices induces a graph isomorphism of the graph with itself which negates labels. The pairing in Figure 2.9 is non-reverse-complementary. However, each of the restrictions of P to Z_6, to $Z_{12} \setminus Z_6$, and to $Z_{18} \setminus Z_{12}$ are reverse-complementary. Only the restriction of the depicted coloring to the pairs in $P|_{Z_{18} \setminus Z_{12}}$, however, is l-reverse-complementary. The pairing in Figure 2.10 is also reverse-complementary. If a K-labelling of the depicted pairing assigns L_1 to x_1, L_2 to x_2, $-L_2$ to x_3, and $-L_1$ to x_4, for some $L_1, L_2 \subseteq \pm Z_K$, then the labelling is l-reverse-complementary.

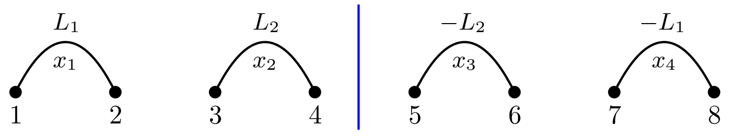


Figure 2.10: The flat pairing $F_4 = \{x_1, x_2, x_3, x_4\}$ together with an l-reverse-complementary labelling. Any l-reverse-complementary labelling of F_4, which assigns labels L_1 and L_2 to x_1 and x_2, respectively, must assign the label $-L_2$ to x_3 and $-L_1$ to x_4.

The recursive approach described above starts by counting the number of ways to label the subset of pairs of a set of pairs $P \subseteq \binom{Z_N}{2}$ which do not contain other pairs. By Theorem 2.3.11, this reduces to counting the number of ways of labelling these pairs coherently, which is done in Theorem 2.3.5. However, in order to distinguish between which of the pairs in the next level can be labelled palindromically, we must know for each of the vertices x in the that level, how many of the labellings of the previous lower levels $\mathrm{dom}(\bar{x}) \setminus \{x\}$ are l-reverse-complementary. For a set of pairs P and a set of symbols $S \subseteq Z_K$, define $\hat{\Lambda}^{\mathrm{rev}}_{K,S,\mathrm{con}}(P)$ to be the set of all l-reverse-complementary members of $\hat{\Lambda}_{K,S,\mathrm{con}}(P)$ and define $\hat{\Lambda}^{\mathrm{nrev}}_{K,S,\mathrm{con}}(P) = \hat{\Lambda}_{K,S,\mathrm{con}}(P) \setminus \hat{\Lambda}^{\mathrm{rev}}_{K,S,\mathrm{con}}(P)$ to be the set of all non-l-reverse-complementary members of $\hat{\Lambda}_{K,S,\mathrm{con}}(P)$. Observe that if P is non-reverse-complementary, then $\hat{\Lambda}^{\mathrm{rev}}_{K,S,\mathrm{con}}(P) = \emptyset$. On the other hand, if $\lambda \in \Lambda_K(P)$ is l-reverse-complementary, then P must be reverse-complementary. Observe that the sizes of $\hat{\Lambda}^{\mathrm{rev}}_{K,S,\mathrm{con}}(P)$ and $\hat{\Lambda}^{\mathrm{rev}}_{K,S',\mathrm{con}}(P)$ are the same as long as $|S| = |S'|$. Thus, we focus our study mainly on the sets $\hat{\Lambda}^{\mathrm{rev}}_{K,\mathrm{con}}(P) = \hat{\Lambda}^{\mathrm{rev}}_{K,Z_K,\mathrm{con}}(P) \subseteq \hat{\Lambda}_{K,\mathrm{con}}(P)$. For example, the coloring of the pairing P depicted in Figure 2.7(c) is a member of $\hat{\Lambda}^{\mathrm{rev}}_{3,\mathrm{con}}(P)$, since it is l-reverse-complementary. Reversal of the vertices 14, 15, 16, and 17 in Figure 2.9 induces a graph isomorphism of the subgraph induced by these vertices with itself, which negates labels. Thus the labelling of the two pairs $x_1 = \{14, 15\}$, and $x_2 = \{16, 17\}$ is a member of $\hat{\Lambda}^{\text{rev}}_{K,\{3\},\text{con}}(\{x_1, x_2\})$. Therefore, the pair $\{13, 18\}$ in can be labelled palindromically without violating consistency of the labelling.

To simplify the task, we restrict ourselves to sets of pairs which to not cross each other. Call a set of pairs P **non-crossing** if there exist no two distinct $(i_1, j_1), (i_2, j_2) \in P$ where $i_1 \leq i_2 \leq j_1 \leq j_2$. Observe that the only non-crossing pairing P of Z_N where no pair contains another is the reverse-complementary pairing $\{\{1, 2\}, \{3, 4\}, \ldots, \{N - 1, N\}\}$, and it exists only when N is even. We call the unique non-crossing pairing of Z_{2n} the **flat** pairing of Z_{2n} and denote it F_n. The flat pairing F_4 is shown in Figure 2.10. If n is even, then for a K-labelling of F_n to be l-reverse-complementary, the sequence of labels of the pairs $\{1, 2\}, \ldots, \{n - 1, n\}$ in the first half of the pairing according to the integer ordering, must be the reverse of the sequence of negations of the labels of the pairs $\{n + 1, n + 2\}, \ldots, \{2n - 1, 2n\}$ in the second half of the pairing. Using this strategy, any complete and coherent K-labelling of the first half of F_n can be extended to a member of $\text{pal}(Z_K, P)$. Conversely, by definition of l-reverse-complementary K-labellings, the sequence of labels of the first half of the pairing is the reverse of the sequence of negations of the labels of the second half of the pairing. In Figure 2.11, for example, an l-reverse-complementary 4-labelling of F_8 is shown. The labels of the pairs x_1, x_2, x_3 and x_4 determine the labels of the remaining pairs.

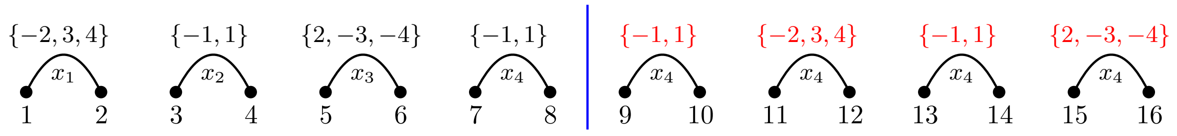


Figure 2.11: An l-reverse-complementary pairing of F_8. The labels of x_1, x_2, x_3, and x_4 on the left determine the remaining labels (in red).

Theorem 2.3.12. *Let K and n be positive integers and suppose n is even. Then:*

$$|\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n)| = \ell_{\text{coh}}(K, n/2)$$

Proof. Denote the pairs in F_n with $x_i = \{2i - 1, 2i\}$, for $i \in Z_n$. Define the function $f : \hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n) \longrightarrow \hat{\Lambda}_{K,\text{coh}}(F_{n/2})$ by:

$$f(\lambda) = \lambda|_{F_{n/2}}.$$

Since each member of $\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n)$ is consistent f does map into $f(\lambda)$. Suppose $\lambda_1 \neq \lambda_2 \in \hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n)$. Then $\lambda_1(x_i) \neq \lambda_2(x_i)$ for some $i \in Z_n$. If $i < n/2$, then of course $f(\lambda_1(x_i)) \neq f(\lambda_2(x_i))$. If, on the other hand, $i \in Z_n \setminus Z_{n/2}$, then since λ_1 and λ_2 are l-reverse-complementary, and since $\sigma^{\text{rev}}_{Z_{2n},Z_{2n}}(j) = 2n - j + 1$, we have that

$$\lambda_1(x_{n-i+1}) = -\lambda_1(x_i) = -\lambda_2(x_i) = \lambda_2(x_{n-i+1}).$$

Since $i \in Z_n \setminus Z_{n/2}$, we must have $n - i + 1 \in Z_{n/2}$, so that $f(\lambda_1) \neq f(\lambda_2)$. Thus, f is injective. Given a member λ of $\hat{\Lambda}_{K,\mathrm{coh}}(F_{n/2})$, we can define λ' on F_n by setting

$$\lambda'(x) = \begin{cases} \lambda(x_i) & i \in Z_{n/2} \\ -\lambda(x_{n-i+1}) & i \in Z_{2n} \end{cases}$$

Then $\lambda' \in \hat{\Lambda}^{\mathrm{rev}}_{K,\mathrm{con}}(F_n)$ and $f(\lambda') = \lambda$. Thus, f is surjective. $\square$

Let n be an odd positive integer. Denote the pairs in F_n by $x_i = \{2i-1, 2i\}$, for $i \in Z_n$. As with even n, for a K-labelling of F_n to be l-reverse-complementary, the sequence of the labels of pairs $x_1, \ldots, x_n$, in that order, must be the same as the sequence of the negations of the labels of the pairs in reverse order. Because n is odd, the label of the pair $x_{\lceil n/2 \rceil}$ in the middle of the sequence must be equal to its own negation. This fact is illustrated in Figure 2.12(a). Here an l-reverse-complementary K-labelling of F_7 is depicted. The sequence of labels of the pairs read from left to right must be the reverse of the negations of the sequence of labels of the pairs read from right to left. Since $x_{\lceil 7/2 \rceil} = x_4$ is in the middle, its label must be its own negation. A strategy for counting F_n for odd n is illustrated in Figure 2.12(b). First, for some integer $K' < K$, a K'-element subset of the set of symbols Z_K is chosen to become the projection of the label of the pair $x_{\lceil n/2 \rceil} = x_4$ in the middle. In this case, $K' = 2$ and the chosen 2-element subset of Z_K is $\{3, 4\}$. Since the label of x_4 must be its own negation, the label will be $\pm\{3, 4\}$. Labels of other pairs in F_n may have projection intersecting the projections of the label of $x_{\lceil n/2 \rceil}$. We choose a p-element subset of the pairs $x_1, \ldots, x_{\lfloor n/2 \rfloor}$ preceeding the middle pair. In this example the ($p = 1$)-element subset is $\{x_1\}$. The chosen subset of pairs together with their counter-parts in the other half (here, x_7 is the counterpart of x_1 and in general x_i is the counterpart of $x_{n-i+1} = x_{7-i+1}$) will be precisely the pairs in the labelling whose labels have projection intersecting the projection of the label of the middle pair x_4. Since x_4 will be assigned the palindromic label $\pm\{3, 4\}$ and because the resulting labelling must be consistent, we have that the labels of x_1 and its counterpart x_7 will also be $\pm\{3, 4\}$. The subgraph of $G(F_7)$ induced by the remaining pairs x_2, x_3, x_5 and x_6 is isomorphic to $F_{n-2p-1} = F_4$ via the mapping $f = \sigma^{\mathrm{fwd}}_{\{3,4,5,6,9,10,11,12\},Z_8}$. Any member of $\hat{\Lambda}^{\mathrm{rev}}_{2,\mathrm{con}}(F_4)$ can be mapped into $\hat{\Lambda}^{\mathrm{rev}}_{4,\mathrm{con}}(F_7)$ using f and using the label $\pm\{3, 4\}$ for the pairs x_1, x_4 and x_7. The general result of reducing the count of the set $\hat{\Lambda}^{\mathrm{rev}}_{K,\mathrm{con}}(F_n)$ to sets $\hat{\Lambda}^{\mathrm{rev}}_{K',\mathrm{con}}(F_{n'})$, where $K' < K$, and $n' < n$, and n' is even is stated in Theorem 2.3.13.

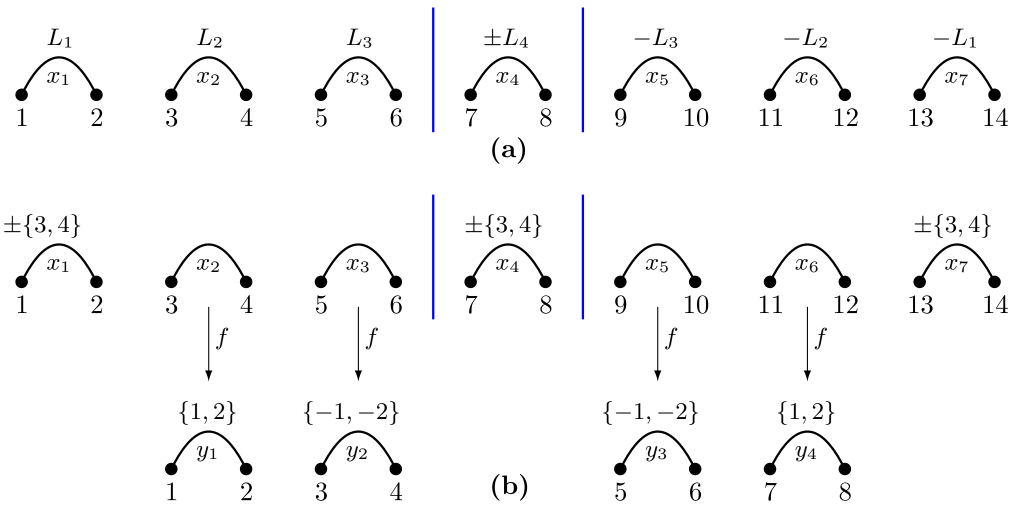


Figure 2.12: A generic l-reverse-complementary K-labelling of F_7 in (a) and an illustration of how to derive a member of $\hat{\Lambda}^{\text{rev}}_{4,\text{con}}(F_7)$ by combining a member of $\hat{\Lambda}^{\text{rev}}_{2,\text{con}}(F_4)$ with a labelling of $\{x_1, x_4, x_7\}$. (a) illustrates that the sequence of labels of the pairs $x_1, \dots, x_7$ of an l-reverse-complementary K-labelling of F_7 read from left to right must be the same as the sequence of negations of the labels of the pairs read from right to left. (b) illustrates how the expression on the right-hand side of Theorem 2.3.13 counts the set $\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n)$. Here, $K' = 2$, $p = 2$. The K'-elements subset of Z_K chosen to be the projection of the label of $x_{\lceil n/2 \rceil} = x_4$ is $\{3, 4\}$ and the set of pairs in the left half whose labels share the same projection is the set $\{x_1\}$.

Theorem 2.3.13. *Let K and n be positive integers and suppose n is odd. Then:*

$$|\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n)| = 1 + \sum_{K'=1}^{|K|-1} \sum_{p=0}^{\lfloor n/2 \rfloor - 1} \binom{K}{K'} \binom{\lfloor n/2 \rfloor}{p} \cdot \ell_{\text{coh}}(K - K', \frac{n - 2p - 1}{2}).$$

Proof. Denote the pairs in F_n with $x_i = \{2i-1, 2i\}$, for $i \in Z_n$. Denote with P_l the set $P_l = \{x_1, \dots, x_{\lfloor n/2 \rfloor}\}$. For partitions $P_l = P_1 \sqcup P_2$ and $Z_K = S_1 \sqcup S_2$ of P_l and Z_K, where P_2, S_1 and S_2 are all non-empty, define the set $\Omega(P_1, P_2, S_1, S_2)$ to be the set of all members of $\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n)$ which assign the label $\pm S_1$ to all pairs in P_1 and to $x_{\lceil n/2 \rceil}$ and whose restriction to P_2 is a coherent K-labelling of P_2, which is complete in S_2:

$$\Omega(P_1, P_2, S_1, S_2) = \{\lambda \in \hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n) : \lambda(P_1 \cup \{x_{\lceil x/2 \rceil}\}) = \{\pm S_1\} \text{ and } \lambda|_{P_2} \in \hat{\Lambda}_{K,S_2,\text{coh}}(P_2)\}.$$

Denote with λ_0 the K-labelling of F_n which assigns the label $\pm Z_K$ to every pair in F_n. In the remainder of the proof, we show that:

$$\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(F_n) \setminus \{\lambda_0\} = \bigsqcup_{P_1 \sqcup P_2 = P_l} \bigsqcup_{S_1 \sqcup S_2 = Z_K} \Omega(P_1, P_2, S_1, S_2),$$

where the partitions indexing the unions must satisfy that P_2, S_1 and S_2 are non-empty. Set $\mathcal{U} = \bigsqcup_{P_1 \sqcup P_2 = F_n} \bigsqcup_{S_1 \sqcup S_2 = P_l} \Omega(P_1, P_2, S_1, S_2)$. Recall that we only need to concern us with the labelling of the

the pairs $x_1, \ldots, x_{\lceil n/2 \rceil}$ for l-reverse-complementary labellings of F_n because their labels uniquely determine the labels of the remaining pairs.

By definition, $\mathcal{U} \subseteq \hat{\Lambda}^{\mathrm{rev}}_{K,\mathrm{con}}(F_n)$. Since S_1 is a proper subset of Z_K, no member of $\mathcal{U}$ labels $x_{\lceil n/2 \rceil}$ with $\pm Z_K$, so that $\lambda_0 \notin \mathcal{U}$.

Conversely, let $\lambda \in \hat{\Lambda}^{\mathrm{rev}}_{K,\mathrm{con}}(F_n) \setminus \{\lambda_0\}$. Let $P_1 = \{x \in P_l : \lambda(x) = \lambda(x_{\lceil n \rceil})\}$ and let $P_2 = P_l \setminus P_1$. Let $S_1 = \mathrm{proj}(\lambda(x_{\lceil n \rceil}))$ and let $S_2 = Z_K \setminus S_1$. Then P_1 and P_2 partition P_l and S_1 and S_2 partition Z_K. If P_2 is empty, then $P_1 = P_l$, so that all pairs must have label $\lambda(x) = \lambda(x_{\lceil n/2 \rceil})$, so that completeness implies $\lambda = \lambda_0$, which is a contradiction. Similarly, $S_2 = \emptyset$ implies that $\mathrm{proj}(x_{\lceil n/2 \rceil}) = Z_K$. Because n is odd and λ $l$reverse-complementary, we must then have $\lambda(x_{\lceil n/2 \rceil}) = \pm Z_K$, so that the label of any pair intersect the label of $x_{\lceil n/2 \rceil}$. But then consistency of λ implies that $\lambda = \lambda_0$, which is again a contradiction. Since $x_{\lceil n/2 \rceil}$ must have some label, S_1 is non-empty. Since λ is l-reverse-complementary and n is odd, we must have that $\lambda(x_{\lceil n/2 \rceil}) = \pm S_1$. By definition of P_1, $\lambda(P_1 \cup \{x_{\lceil n/2 \rceil}\}) = \{\pm S_1\}$. Consistency of λ implies that whenever $\lambda(x) \cap \lambda(y)$, then $\lambda(x) = \lambda(y)$, so that $\mathrm{supp}(\lambda(P_2)) \cap S_1 = \emptyset$. But then completeness of λ implies that $\mathrm{supp}(\lambda(P_2)) = Z_K \setminus \mathrm{supp}(\lambda(P_1)) = Z_K \setminus S_1 = S_2$. Consistency of λ implies coherence of $\lambda|_{P_2}$. Thus, $\lambda|_{P_2} \in \hat{\Lambda}_{K,S_2,\mathrm{coh}}(P_2)$, as desired. □

To count consistent K-labellings of a set of pairs by counting the ways parts of it can be labelled and in how many ways these partial labellings can be combined to form labellings of the entire set, we need to establish how labellings and sets of labellings are combined. For two K-labellings λ_1, λ_2 of two disjoint sets of pairs P_1, P_2, respectively, let their **union** be the K-labelling of $P_1 \cup P_2$ defined by:

$$\lambda_1 \cup \lambda_2(x) = \begin{cases} \lambda_1(x) & x \in P_1 \\ \lambda_2(x) & x \in P_2. \end{cases}$$

For two sets Λ_1, Λ_2 of K-labellings of two disjoint sets of pairs P_1, P_2, respectively, let their **concatenation** be the set of K-labellings of $P_1 \cup P_2$ defined by:

$$\Lambda_1 \circ \Lambda_2 = \{\lambda_1 \cup \lambda_2 : \lambda_1 \in \Lambda_1, \lambda_2 \in \Lambda_2\}.$$

In Figure 2.9, the union of the 8-labellings λ_1 of $P_1 = \{\{1,6\},\{2,5\},\{3,4\}\}$, λ_2 of $P_2 = \{\{7,11\},\{8,12\}\}$, and λ_3 of $P_3 = \{\{13,18\},\{14,15\},\{16,17\}\}$, which assign the labels depicted in the figure to the respective pairs, is the member $\lambda_1 \cup \lambda_2 \cup \lambda_3$ of $\Lambda_8(P_1) \circ \Lambda_8(P_2) \circ \Lambda_8(P_3)$.

In a set of pairs P some pairs may never be labelled with the same symbols or the negations of the symbols used in the label of other pairs. For example, in the set of pairs depicted in Figure 2.9, no two of the pairs

$\{1,6\},\{2,5\},\{3,4\}\}$ may have use the same symbol in their label, or negations of the same symbol since it would violate the condition in Proposition 2.3.10. Additionally, the label of the pair $\{13,18\}$ may not share a symbol or negation thereof with the label of any other pair in the pairing because the structure in its closure is unique preventing a graph isomorphism between the graph $G_P[\{13,18\}]$ and $G_P[x]$, for any other $x \in P$ other than $\{13,18\}$ itself. This leads to the conclusion that consistent K-labellings of a set of pairs P may not exist if K is too small depending on the structure of P. For a set of pairs $P \subseteq \binom{Z_N}{2}$ denote with $\mu(P)$ the minimum integer K, such that there exists a complete and consistent K-coloring of P. For example, the minimum number of symbols required to consistently label the set of pairs $P_1 = \{\{1,6\},\{2,5\},\{3,4\}\}$ in Figure 2.9, is $\mu(P_1) = 3$, since by Proposition 2.3.10, mutual containment of the pairs in P_1 requires pairwise disjoint projections of their labels and since any 3-labelling of P_1 using any permutation of the labels $\{1\}$, $\{2\}$, and $\{3\}$ to label the pairs in P_1 is consistent.

Using the aforementioned recursive approach to counting the number of consistent K-labellings of a set of pairs P, a central problem becomes to count the number of K-labellings of the set $\{x\}$ and the number of consistent K-labellings of $\overset{\circ}{P} = \bar{x} \setminus \{x\}$ and how often these labellings can be combined to a consistent K-labelling of P. Figure 2.13 illustrates this problem on the set of pairs $P = \{x, y_1, y_2, y_3\}$. Here, we seek to count the number of consistent K-labellings λ_x of $\{x\}$, the number of consistent K-labellings λ_y of $\overset{\circ}{P} = \{y_1, y_2, y_3\}$, and to count how often the two labellings can be combined to a consistent K-labelling $\lambda_x \cup \lambda_y$ of P. Proposition 2.3.10 implies that the projection of L_x must be disjoint from the support of $\{L_1, L_2, L_3\}$. Counting the set $\hat{\Lambda}_{K,\mathrm{con}}(P)$ then requires partitioning the set Z_K into a set $S \subseteq Z_K$ which will be the projection of L_x and its complement $Z_K \setminus S$ which will be the support of $\{L_1, L_2, L_3\}$. The complement of S must then contain at least $\mu(\overset{\circ}{P})$ elements, so that the size s of S is restricted to the set $Z_{K-\mu(\overset{\circ}{P})}$. Note that the label $\lambda_x(x) = L_x$ can only be palindromic when the labelling λ_y of $\bar{x} \setminus \{x\}$ is l-reverse-complementary. Then the union of any member of $\hat{\Lambda}^{\mathrm{rev}}_{K,Z_K\setminus S,\mathrm{con}}(\overset{\circ}{P})$ and any member of $\hat{\Lambda}_{K,S,\mathrm{con}}(\{x\})$ forms a consistent K-labelling of all of P, whereas the unions of members of $\hat{\Lambda}^{\mathrm{nrev}}_{K,Z_K\setminus S,\mathrm{con}}(\overset{\circ}{P})$ with members of $\hat{\Lambda}_{K,S,\mathrm{con}}(\{x\})$ are consistent only if the latter come from the set $\hat{\Lambda}^{\mathrm{nrev}}_{K,S,\mathrm{con}}(\{x\}) \subseteq \hat{\Lambda}_{K,S,\mathrm{con}}(\{x\})$.

Theorem 2.3.14. *Let P be a pairing of Z_N where $x = \{1, N\} \in P$. Denote with $\overset{\circ}{P}$ the set $P \setminus \{x\}$ Then:*

$$\hat{\Lambda}_{K,\mathrm{con}}(P) = \bigsqcup_{s=1}^{K-\mu(\overset{\circ}{P})} \bigsqcup_{S\in\binom{Z_K}{s}} \left[\left(\hat{\Lambda}_{K,S,\mathrm{con}}(\{x\}) \circ \hat{\Lambda}^{\mathrm{rev}}_{K,Z_K\setminus S,\mathrm{con}}(\overset{\circ}{P}) \right) \sqcup \left(\hat{\Lambda}^{\mathrm{nrev}}_{K,S,\mathrm{con}}(\{x\}) \circ \hat{\Lambda}^{\mathrm{nrev}}_{K,Z_K\setminus S,\mathrm{con}}(\overset{\circ}{P}) \right) \right],$$

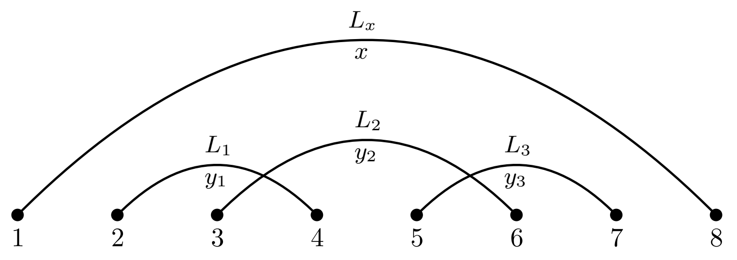


Figure 2.13: A pairing $P = \{x, y_1, y_2, y_3\}$ of Z_8 where $\bar{x} = P$. If the pairs x, y_1, y_2, y_3 have labels L_x, L_1, L_2, L_3, respectively, then the labelling is consistent only if $\operatorname{proj}(L_x) \cap \operatorname{supp}(\{L_1, L_2, L_3\}) = \emptyset$. Additionally, any member of $\hat{\Lambda}_{K,S,\text{con}}(\{x\})$ combined (via union) with any member of $\hat{\Lambda}^{\text{rev}}_{K,Z_K \setminus S,\text{con}}(P \setminus \{x\})$ is a member of $\hat{\Lambda}_{K,\text{con}}(P)$. However, to form labellings in $\hat{\Lambda}_{K,\text{con}}(P)$, members of $\hat{\Lambda}^{\text{nrev}}_{K,Z_K \setminus S,\text{con}}(P \setminus \{x\})$ can only be combined with members of $\hat{\Lambda}^{\text{nrev}}_{K,S,\text{con}}(\{x\})$.

so that:

$$\begin{aligned}
|\hat{\Lambda}_{K,\text{con}}(P)| &= \sum_{s=1}^{K-\mu(\overset{\circ}{P})} \binom{Z_K}{s} \Big[(2^s+1) \cdot \big|\hat{\Lambda}^{\text{rev}}_{K-s,\text{con}}(\overset{\circ}{P})\big| + 2^s \cdot \big|\hat{\Lambda}^{\text{nrev}}_{K-s,\text{con}}(\overset{\circ}{P})\big|\Big] \\
&= \sum_{s=1}^{K-\mu(\overset{\circ}{P})} \binom{Z_K}{s} \Big[\big|\hat{\Lambda}^{\text{rev}}_{K-s,\text{con}}(\overset{\circ}{P})\big| + 2^s \cdot \big|\hat{\Lambda}_{K-s,\text{con}}(\overset{\circ}{P})\big|\Big].
\end{aligned}$$

Proof. Let $s \in Z_{K-\mu(\overset{\circ}{P})}$ and $S \in \binom{Z_K}{s}$. Let $\lambda_x \in \hat{\Lambda}_{K,S,\text{con}}(\{x\})$ and $\lambda_y \in \hat{\Lambda}^{\text{rev}}_{K,Z_K \setminus S,\text{con}}(\overset{\circ}{P})$. Set $\lambda = \lambda_x \cup \lambda_y$. Let $y_1, y_2 \in P$. Since $\operatorname{supp}(\lambda(\{x\})) = \operatorname{supp}(\lambda_x(\{x\}))$ and $\operatorname{supp}(\lambda(\overset{\circ}{P})) = \operatorname{supp}(\lambda_y(\overset{\circ}{P}))$ are disjoints, intersecting projections of labels of y_1 and y_2 imply that $y_1 = y_2 = x$, or $y_1, y_2 \in \overset{\circ}{P}$. Suppose $\lambda(y_1) \cap \lambda(y_2) \neq \emptyset$. If both y_1 and y_2 are members of $\overset{\circ}{P}$, then consistency of λ_y implies that $\bar{y_1}, \bar{y_2} \subseteq \overset{\circ}{P}$ and $y_1 \cong^{\text{fwd}}_{\overset{\circ}{P},\lambda_y} y_2$. Hence, $y_1 \cong^{\text{fwd}}_{P,\lambda} y_2$. If on the other hand $y_1 = y_2 = x$, then of course $y_1 \cong^{\text{fwd}}_{P,\lambda} x \cong^{\text{fwd}}_{P,\lambda} y_2$. Now, suppose $\lambda(y_1) \cap -\lambda(y_2) \neq \emptyset$. If both y_1 and y_2 are members of $\overset{\circ}{P}$, then analogous to the previous case, $y_1 \cong^{\text{rev}}_{P,\lambda} y_2$. If on the other hand $y_1 = y_2 = x$, then consistency of λ_x implies that $\lambda(x) = -\lambda(x)$. Furthermore, since λ_y is l-reverse-complementary, the mapping $\sigma^{\text{rev}}_{Z_N,Z_N}$ is a graph isomorphism of $G(\overset{\circ}{P})$ with itself which negates labels. Since $\sigma^{\text{rev}}_{Z_N,Z_N}$ maps the end points of x into the end points of x, and the label of x is its own negation, $\sigma^{\text{rev}}_{Z_N,Z_N}$ is a graph isomorphism of $G(P) = G(\bar{x})$ with itself which negates labels. Thus, $x \cong^{\text{rev}}_{P,\lambda}$. Then λ is consistent. Since $\operatorname{supp}(\lambda_x(\{x\}) = S$ and $\operatorname{supp}(\lambda_y(\overset{\circ}{P})) = Z_K \setminus S$, we have that $\operatorname{supp}(\lambda(P)) = Z_K$. Therefore, $\lambda \in \hat{\Lambda}_{K,\text{con}}(P)$.

The case where $\lambda_x \in \hat{\Lambda}^{\text{nrev}}_{K,S,\text{con}}(\{x\})$ and $\lambda_y \in \hat{\Lambda}^{\text{nrev}}_{K,Z_K \setminus S,\text{con}}(\overset{\circ}{P})$, works the same way except that $\lambda(y_1) \cap -\lambda(y_2) \neq \emptyset$ implies that $y_1, y_2 \neq x$.

To show the reverse containment, let $\lambda \in \hat{\Lambda}_{K,\text{con}}(P)$, set $S = \operatorname{proj}(\lambda(x))$ and set $\lambda_x = \lambda|_{\{x\}}$ and $\lambda_y = \lambda|_{\overset{\circ}{P}}$. Proposition 2.3.10 and completeness of λ imply that $\operatorname{supp}(\lambda_y(\overset{\circ}{P})) = Z_K \setminus S$ and then $K - |S| \geq \mu(\overset{\circ}{P})$, so that $|S| \in Z_{K-\mu(()\overset{\circ}{P})}$. We now have $\lambda = \lambda_x \cup \lambda_y$, $\lambda_x \in \hat{\Lambda}_{K,S,\mathrm{coh}}(\{x\})$, and $\lambda_y \in \hat{\Lambda}_{K,Z_K \setminus S,\mathrm{coh}}(\overset{\circ}{P})$.

What remains to be shown is that if λ_y is not reverse-complementary, then λ_x is not reverse-complementary, too. Suppose λ_x is l-reverse-complementary. Then $\lambda_x(x) = -\lambda_x(x)$. Consistency of λ implies that $\sigma^{\mathrm{rev}}_{Z_N,Z_N}$ is a graph isomorphism of $\bar{x} = P$ with itself, which negates labels. Then it is also a graph isomorphism of $\overset{\circ}{P}$ with itself which negates labels, so that λ_y is l-reverse-complementary.

The second part of the result follows from the first and the fact that $\{x\}$ can be consistently labelled with support S in $|\operatorname{pal}(S)|$ ways and consistently and non-reverse-complementary labelled with the same support in $|\operatorname{apal}(S)|$ ways. □

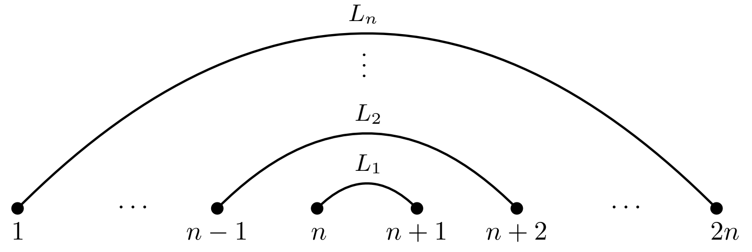


Figure 2.14: The pairing $SN(n)$. For any two pairs $x, y \in SN(n)$, either $x \sqsupseteq y$, or $y \sqsupseteq x$. If the labels $L_1, \ldots, L_n$ describe a consistent K-labelling, then their projections must be disjoint and whenever any of the L_i is palindromic, all labels below must be palindromic, as well.

As a special case of Theorem 2.3.14 we consider the pairing $SN(n) = \big\{\{1, 2n\}, \{2, 2n-1\}, \ldots, \{n, n+1\}\big\}$ of Z_{2n}, which we will call the **simply nested set of n pairs**. The pairing $SN(n)$ is depicted in Figure 2.14. Observe that for any two pairs $x, y \in SN(n)$, either $x \sqsupseteq y$, or $y \sqsupseteq x$, so that by Proposition 2.3.10, any consistent labelling of $SN(n)$ must label the pairs with labels that have pairwise disjoint projections. Furthermore, if λ is a consistent K-labelling of $SN(n)$ and $\lambda(\{x\}) = \pm\lambda(x)$, for some $x \in SN(n)$, then $\lambda(y) = \pm\lambda(y)$, for any $y \in SN(n)$, where $x \sqsupseteq y$.

Proposition 2.3.15. *Denote the pairs in $SN(n)$ by $x_i = \{i, 2n-i+1\}$.*

$$\hat{\Lambda}_{K,\mathrm{coh}}(SN(n)) = \bigsqcup_{\mathcal{S} \in \left\{ {Z_K \atop n} \right\}} \; \bigsqcup_{\gamma : P \xrightarrow{bij} \mathcal{S}} \; \bigsqcup_{i=0}^{n} \Big(\mathop{\circ}_{j=1}^{i} \hat{\Lambda}^{\mathrm{nrev}}_{K,\gamma(x_j),\mathrm{con}}(\{x_j\}) \Big) \circ \Big(\mathop{\circ}_{j=i+1}^{n} \hat{\Lambda}^{\mathrm{rev}}_{K,\gamma(x_j),\mathrm{con}}(\{x_j\}) \Big),$$

so that:

$$|\hat{\Lambda}_{K,\mathrm{coh}}(SN(n))| = \sum_{\mathcal{S} \in \left\{ {S \atop n} \right\}} \; \sum_{\gamma : P \xrightarrow{bij} \mathcal{S}} \Big(1 + \sum_{i=1}^{n} \prod_{j=1}^{i} 2^{|\gamma(x_j)|} \Big)$$

Proof. Suppose $\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(SN(n))$. For all $x_i, X_j \in SN(n)$, we have that either $x_i \sqsupseteq x_j$, or $x_j \sqsupseteq x_i$. Then by completeness of λ, the projections $\operatorname{proj}(x_1), \ldots, \operatorname{proj}(x_n)$ of the pairs in $SN(n)$ partition Z_K. Furthermore, $x_i \sqsupseteq x_{i+1}, \ldots, x_n$, for every $i \in Z_n$, so that whenever $\lambda(x_i)$ is palindromic, the restriction of λ to $\{x_{i+1}, \dots, x_n\}$ is l-reverse-complementary, which is the case if and only if all the labels of all of $x_{i+1}, \dots, x_n$ are palindromic. Let i be the smallest index for which $\lambda(x_i)$ is non-l-reverse-complementary, or set $i = 0$ when no such index exists. Then we must have that $\{\lambda(x_j) : j \leq i\}$ consists of anti-palindromic labels and $\{\lambda(x_j) : j > i\}$ consists of palindromic labels. Thus, if $j \leq i$, the restriction of λ to x_j is a member of $\hat{\Lambda}^{\text{nrev}}_{K,\text{proj}(\lambda(x_j)),\text{con}}(\{x_j\})$, and if $j > i$, it is a member of $\hat{\Lambda}^{\text{rev}}_{K,\text{proj}(\lambda(x_j)),\text{con}}(\{x_j\})$. Then λ is a member of

$$\bigsqcup_{\mathcal{S} \in \left\{ {Z_K \atop n} \right\}} \bigsqcup_{\gamma : P \xrightarrow{bij} \mathcal{S}} \bigsqcup_{i=0}^{n} \Big(\mathop{\circ}_{j=1}^{i} \hat{\Lambda}^{\text{nrev}}_{K,\gamma(x_j),\text{con}}(\{x_j\}) \Big) \circ \Big(\mathop{\circ}_{j=i+1}^{n} \hat{\Lambda}^{\text{rev}}_{K,\gamma(x_j),\text{con}}(\{x_j\}) \Big),$$

where $\mathcal{S} = \{\text{proj}(x_i) : i \in Z_n\}$, and γ is defined by $x_i \longmapsto \text{proj}(x_i)$.

To show reverse containment, let $\mathcal{S} \in \left\{ {Z_K \atop n} \right\}$, let $\gamma : P \longrightarrow \mathcal{S}$ be a bijection. For some $i \in \{0, 1, \dots, n\}$, let $\lambda_j \in \hat{\Lambda}^{\text{nrev}}_{K,\gamma(x_j),\text{con}}(\{x_j\})$, for $j = 1, \dots, i$, let $\lambda_j \in \hat{\Lambda}^{\text{rev}}_{K,\gamma(x_j),\text{con}}(\{x_j\})$, for $j = i+1, \dots, n$, and let $\lambda = \lambda_1 \cup \cdots \cup \lambda_n$. Since each of the λ_j is complete in $\gamma(x_j)$, γ is a bijection and $\mathcal{S}$ is a partition of Z_K, we have that λ is complete in Z_K. Let $j, j' \in Z_n$ and suppose $\lambda(x_j) \cap \text{proj}(\lambda(x_{j'})) \neq \emptyset$. By definition of λ, and by disjointness of the supports $\text{supp}(\lambda_j(x_j))$ and $\text{supp}(\lambda_{j'}(x_{j'}))$, we must have $j = j'$. Clearly, $x_j \cong^{\text{fwd}}_{SN(n),\lambda} x_j$. If $\lambda(x_j) \cap -\lambda(x_j)$, then $\lambda(x_j) = \pm\lambda(x_j$ because λ_j is consistent. Then $\lambda_j \in \hat{\Lambda}^{\text{rev}}_{K,\gamma(x_j),\text{con}}(\{x_j\})$, so that $j > i$. Since $\bar{x_j} = \{x_{j'} : j' \geq j\}$, all members of the closure of x_j are labelled palindromically, so that $\sigma^{\text{rev}}_{\text{dom}(\bar{x_j}),\text{dom}(\bar{x_j})}$ is a graph isomorphism of $G_{SN(n)}[x_j]$ which negates labels. Thus, $x_j \sigma^{\text{rev}}_{SN(n),\lambda} x_j$.

The second part of the result follows immediately from the first part and the fact that for any pair $x \in \binom{Z_N}{2}$, $\hat{\Lambda}^{\text{rev}}_{K,S,\text{con}}(\{x\})$ consists of only one K-labelling, which assigns the label $\pm S$ to x, and $\hat{\Lambda}^{\text{nrev}}_{K,S,\text{con}}(\{x\})$ consists of of the K-labellings, which assign any one of the $2^{|S|}$ members of $\text{apal}(S)$ to x. $\square$

Recall that to simplify our task, we have restricted our recursive counting strategy of all consistent K-labellings of sets of pairs to non-crossing sets of pairs. Theorems 2.3.5 and 2.3.11 allow us to count the set of consistent K-labellings of the 0th level of any set of pairs and Theorems 2.3.12 and 2.3.13 provide the counts of the subset of l-reverse-complementary K-labellings. Although it is not restricted to non-crossing sets of pairs, Theorem 2.3.14 provides a way of reducing the count of consistent K-labellings of the closure of any pair x, which is not a member of the 0th level of a set of pairs, to counting the set of consistent K-labellings of $\bar{x} \setminus x$ and the subset of l-reverse-complementary K-labellings. Combining these results we still cannot count the set $\hat{\Lambda}_{K,\text{con}}(P)$ for all non-crossing pairings P.

For example, the consistent K-labellings of the non-crossing sets of pairs P_2 and P_3 in Figure 2.15 can be counted using these results, but the consistent K-labellings of the non-crossing set of pairs $P_2 \cup P_3$ cannot be counted this way. The count of $\hat{\Lambda}_{K,\text{con}}(P_2)$ is reduced by Theorem 2.3.14 to counting the set of consistent K-labellings and the subset of reverse-complementary K-labellings of a copy of F_2, which can be done using Theorems 2.3.5, 2.3.11, and 2.3.12. Similarly, the count of $\hat{\Lambda}_{K,\text{con}}(P_3)$ can be repeatedly reduced using Theorem 2.3.14 to counting $\hat{\Lambda}_{K',\text{con}}(F_1)$ and $\hat{\Lambda}^{\text{rev}}_{K',\text{con}}(F_1)$, for various $K' < K$. These two sets can then be counted using Theorems 2.3.5, 2.3.11, and 2.3.13. Alternatively, $\hat{\Lambda}_{K,\text{con}}(P_3)$ can be counted using Proposition 2.3.15. However, none of the results so far allow us to count $P_2 \cup P_3$. The reason for that is that all of these results except for Theorem 2.3.14 only apply to pairings which have height 0. Additionally, Theorem 2.3.14 does not apply here because there is no single pair x whose closure is all of $P_2 \cup P_3$. In order to be able to recursively count the consistent K-labellings of $P_2 \cup P_3$, we need a way of combining the counts of consistent K-labellings for the two sets.

By extension, a general result which reduces counting the consistent K-labellings of a sequence $P_1, \dots, P_r$ of sets of pairs where $\max(\text{dom}(P_i)) < \min(\text{dom}(P_{i+1}))$ to counting the consistent K-labellings of the individual sets P_i would allow us to count the consistent K-labellings of any non-crossing pairing P. However, obtaining such a result is challenging due to the fact that in a consistent K-labelling λ of $P_1 \cup \cdots \cup P_r$, the supports of the $\lambda(P_i)$ are not necessarily disjoint. A consistent K-labelling of $P_2 \cup P_3$ in Figure 2.15, for example, may assign the same labels or negations thereof to any subset of the pairs $\{6, 7\}$, $\{8, 9\}$ and $\{13, 14\}$.

Denote with $\kappa_K(P_1, \dots, P_r)$ the set of partitions $\mathcal{Z_K} = (S_1, \dots, S_r) \in \left\{ {S \atop r} \right\}$ of Z_K into r parts, ordered in such a way that for each i^{th} member S_i of $\mathcal{S}$, $|S_\ell| \geq \mu(P_i)$:

$$\kappa_K(P_1, \dots, P_r) = \left\{ (S_1, \dots, S_r) \in \left\{ {Z_K \atop r} \right\} : |S_i| \geq \mu(P_i), \text{ for } i \in Z_r \right\}.$$

For the pairings P_1, P_2, P_3 in Figure 2.15, we have $\mu(P_1) = 1, \mu(P_2) = 2$, and $\mu(P_3) = 3$, so that any partition of Z_K into a sequence of parts (S_1, S_2, S_3) where $S_i \geq i$, for $i \in Z_3$ is a member of $\kappa_K(P_1, \dots, P_3)$.

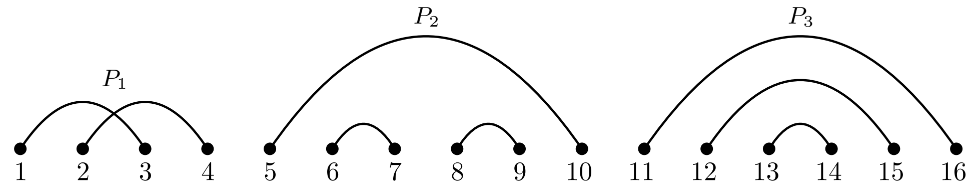


Figure 2.15: Three sets of pairs P_1, P_2, and P_3 whose union is a pairing of Z_{16}. The sets P_2, and P_3 are non-crossing, while P_1 is not. Using Theorems 2.3.5, 2.3.11, 2.3.12, 2.3.13, and 2.3.14, the consistent K-labellings of the non-crossing sets of pairs $\hat{\Lambda}_{K,\text{con}}(P_2)$ and $\hat{\Lambda}_{K,\text{con}}(P_3)$ can be counted. Even though $P_2 \cup P_3$ is non-crossing these results do not provide the necessary tools to count $\hat{\Lambda}_{K,\text{con}}(P_2 \cup P_3)$.

Theorem 2.3.16. *Let $r \geq 2$ and let $P_1, \dots, P_r$ be sets of pairs of Z_N, where* $\max(\text{dom}(P_i)) < \min(\text{dom}(P_{i+1}))$*, for $i \in Z_{r-1}$. Let $K \geq \sum_{i=1}^{r} \mu(P_i)$ and denote with Λ the set of all complete and consistent K-labellings λ of $P_1 \cup \cdots \cup P_r$, where* $\text{supp}(\lambda(P_i)) \cap \text{supp}(\lambda(P_j)) = \emptyset$*, for all $\{i, j\} \in \binom{Z_r}{2}$. Then every* *member of* Λ *is non-l-reverse-complementary and:*

$$\Lambda = \mathop{\circ}_{(S_1,\ldots,S_r)\in\kappa_K(P_1,\ldots,P_r)} \hat{\Lambda}_{K,S_i,\mathrm{con}}(P_i),$$

so that:

$$|\Lambda| = \sum_{(S_1,\ldots,S_r)\in\kappa_K(P_1,\ldots,P_r)} \prod_{i=1}^{r} \left|\hat{\Lambda}_{|S_i|,\mathrm{con}}(P_i)\right|.$$

Proof. Let $\lambda \in \Lambda$ and for $i \in Z_r$, set λ_i to be the restriction of λ to P_i and set S_i to be the support of $\lambda(P_i)$. Then by definition of Λ, $S_1 \cup \cdots \cup S_r = Z_K$. Since $\max(\mathrm{dom}(P_i)) < \min(\mathrm{dom}(P_{i+1}))$, for $i \in Z_{r-1}$, the closures of any two pairs x, y from different sets P_i and P_j are disjoint. Then consistency of λ implies consistency each of the λ_i. Thus $\lambda_i \in \hat{\Lambda}_{K,S_i,\mathrm{con}}(P_i)$, for $i \in Z_r$ and thus $|S_i| \geq \mu(P_i)$.

For the reverse containment, let $(S_1, \ldots, S_r) \in \kappa_K(P_1, \ldots, P_r)$, let $\lambda_i \in \hat{\Lambda}_{K,S_i,\mathrm{con}}(P_i)$, for $i \in Z_r$ and set $\lambda = \lambda_1 \circ \cdots \circ \lambda_r$. Since the sets $S_1, \ldots, S_r$ partition Z_K and each of the λ_i is complete in S_i, we have that λ is complete in Z_K. If for any two pairs $x, y \in P_1 \cup \cdots \cup P_r$, we have that $\lambda(x) \cap \pm\lambda(y)$, then disjointness of the S_i implies that x and y are members of the same set P_j, for some $j \in Z_r$. Then since $\max(\mathrm{dom}(P_i)) < \min(\mathrm{dom}(P_{i+1}))$, their closures are contained in P_j, so that by consistency of λ_j, we have that $\lambda(x) \cap \lambda(y)$ implies $x \cong^{\mathrm{fwd}}_{P,\lambda} y$ and $\lambda(x) \cap -\lambda(y)$ implies $x \cong^{\mathrm{rev}}_{P,\lambda} y$. Thus, λ is consistent. By definition, $\mathrm{supp}(\lambda(P_i)) \cap \mathrm{supp}(\lambda(P_j)) = S_i \cap S_j$, for $\{i, j\} \in \binom{Z_r}{2}$, so that $\lambda \in \Lambda$.

The second part of the result follows immediately from the first. $\square$

2.4 Relationship of Pairings and Labellings to Older Models

In this section, attention returns to rearrangement maps where no two MDSs in the mic overlap, or correspond to the same regions in the mac (except for pointer overlap at the ends of successive regions in the mac). A variety of models of rearrangements is defined here and their connection to pairings and labellings is established by a set of bijective, surjective and injective relationships summarized in Figure 2.17. An example for each of the models is given in Figure 2.16, all of which correspond to the rearrangement schematic in Figure 2.16(f).

Under the simplifying assumption a rearrangement is readily described by a sequence of symbols representing the MDSs in their order of appearance in the mic, where each symbol carries two pieces of information about an MDS:

1. The position of the matching region in the mac relative to the other matched regions,

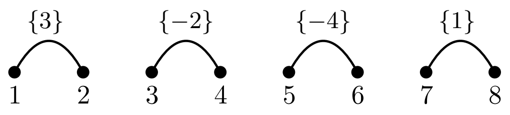
2. The relative orientations of the matching segments in the mic and mac.

Definition 2.4.1. An **MDS sequence of length** N is a sequence of symbols $X_{k_1} \cdots X_{k_N}$, where $\{k_1, \ldots, k_N\} = Z_N$, and $X_{k_i} \in \{M_{k_i}, \overline{M}_{k_i}\}$, for $i \in Z_N$.

In this representation, the subscripts of the symbols in an MDS sequence indicate the position of the matching region in the mac relative to the others and the bar above the symbol is added when the matched micronuclear and macronuclear regions are located on opposite strands. An MDS sequence is shown in Figure 2.16(a) and the corresponding rearrangement schematic in Figure 2.16(f).

$M_3\ \overline{M}_2\ \overline{M}_4\ M_1$

(a)



(b)

2 3 2 1 4 3 0 1

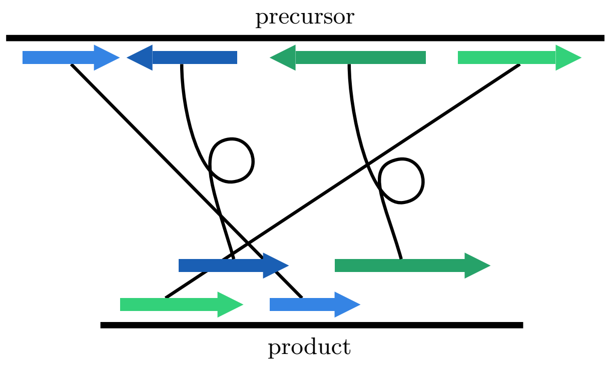
(c)

2 3 2 1 3 1

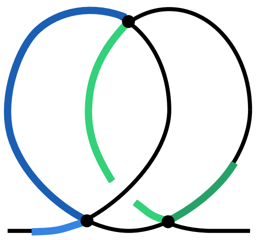
(d)

$[1\ 2\ 1\ 3\ 2\ 3]_{\sim}$

(e)



(f)



(g)

Figure 2.16: Various models of the same rearrangement. An MDS sequence (Definition 2.4.1) is shown in (a), the corresponding simple arrangement (Definition 2.4.2) in (b), the augmented pointer sequence (Definition 2.4.5) in (c), the pointer sequence (Definition 2.4.4) in (d), the assembly word (Definition 2.4.8) in (e), and the assembly graph (Definition 2.4.12) in (g). A schematic of the modelled rearrangement is shown in (f). The colors of the MDSs in (f) correspond to the respective segments in (g). The notation $[w]_{\sim}$ in (e) denotes the equivalence class of w under the relation $\sim$.

MDS sequences are in one-to-one correspondence with certain pairs (π, λ) consisting of a pairing π and a labelling λ of π.

Definition 2.4.2. An **arrangement of** Z_N is a pair (π, λ), where $\pi \in \Pi(2N)$ is a pairing π and λ a labelling of π. An arrangement (π, λ) of Z_N is **simple** if π is the flat pairing, λ is complete in Z_N, and $\lambda(\pi)$ consists of singletons with pairwise distinct projections.

After establishing the bijective correspondence between simple arrangements and MDS sequences in Proposition 2.4.3, their relationship to **assembly words**, as defined in [3, 9], is discussed. Due to the lack of a straightforward one-to-one correspondence between the two structures, the differences between simple arrangements and assembly words are described by presenting the relationships between successive intermediates between the two structures as shown in Figure 2.17. Finally, observations made during the investigation of these relationships are combined to a result that provides some insight into **assembly numbers** which are heavily studied in [3, 9].

To each MDS sequence $X = X_{k_1} \cdots X_{k_N}$, a simple arrangement $\mathcal{A}_X = (\pi_X, \lambda_X)$ can be associated by setting $\pi_X = F_N$ and letting

$$\lambda_X(\{2i-1, 2i\}) = \begin{cases} \{k_i\}, & \text{if } X_{k_i} = M_{k_i} \\ \{-k_i\}, & \text{if } X_{k_i} = \overline{M}_{k_i}, \end{cases}$$

for $i \in Z_N$.

Proposition 2.4.3. *The mapping $X \longmapsto \mathcal{A}_X$ is a bijection between the collection of MDS sequences of length N and simple arrangements of Z_N.*

Proof. Suppose $X = X_{k_1} \cdots X_{k_N}$ and $X' = X_{l_1} \cdots X_{l_N}$ are distinct MDS sequences. Then for some $i \in Z_N$, either $k_i \neq l_i$, or $k_i = l_i$ and $(X_{k_i}, X_{l_i}) \in \{(M_{k_i}, \overline{M}_{l_i}), (\overline{M}_{k_i}, M_{l_i})\}$. In the first case, $\lambda_X(\{2i-1, 2i\}) = \{k_i\} \neq \{l_i\} = \lambda_{X'}(\{2i-1, 2i\})$. In the latter case, either $\lambda_X(\{2i-1, 2i\}) = \{k_i\} \neq \{-l_i\} = \lambda_{X'}(\{2i-1, i\})$, or $\lambda_X(\{2i-1, 2i\}) = \{-k_i\} \neq \{l_i\} = \lambda_{X'}(\{2i-1, i\})$.

Next, let $\mathcal{A} = (\pi, \lambda)$ be a simple arrangement. Let $X = X_{k_1} \cdots X_{k_N}$, where:

$$X_{k_i} = \begin{cases} M_{k_i} & \text{if } \lambda(\{2i-1, 2i\}) = \{k_i\} \\ \overline{M}_{k_i} & \text{if } \lambda(\{2i-1, 2i\}) = \{-k_i\}. \end{cases}$$

Then $\mathcal{A}_X = \mathcal{A}$. □

The intermediate structures between simple arrangements and assembly words considered here are derived from the sequence of pointers in their order of appearance in the mic. The difference between the two intermediate structures is that one contains two additional symbols which disambiguate the rearrangement modelled by the pointer sequence, if any.

Definition 2.4.4. A **pointer sequence on** N **letters** is a word w over the alphabet Z_N in which every symbol of Z_N appears exactly twice.

Definition 2.4.5. An **augmented pointer sequence** on N letters is a word w over the alphabet $Z_N \cup \{0\}$ obtained from a pointer sequence on $N-1$ letters by inserting two additional letters 0 and N.

Remark 2.4.6. The mapping associating to every augmented pointer sequence w on N letters, the pointer sequence $\overset{\circ}{w}$ obtained from w via removal of letters 0 and N is a surjection onto the set of pointer sequences on $N-1$ letters.

A pointer sequence represents the sequence of pointers at the ends of MDSs in their order of appearance on the mic, labelled according to their order of appearance in the mac. Since the assumption is made here that no two MDSs overlap, or correspond to the same regions in the mac, the order of the pointers is well-defined and each pointer appears exactly twice. The symbols added to a pointer sequence in an augmented pointer sequence can be thought of as the positions of telomere addition sites at the ends of the terminal MDSs. The same pointer sequence w can be obtained from multiple augmented pointer sequences $v_1, \ldots, v_k$ (i.e. $w = \overset{\circ}{v}_1 = \cdots = \overset{\circ}{v}_k$). By disregarding the locations of the pointer-less ends of terminal MDSs, information about the exact position of these terminal MDSs is lost in a pointer sequence in contrast to an augmented pointer sequence. In fact, it is possible that the same pointer sequence corresponds to both ordered and disordered augmented pointer sequences. For example, the word 11 may correspond to the augmented pointer sequences 0121, and 0112. The former, corresponds to the scrambled MDS sequence $M_1\overline{M}_2$, whereas the latter corresponds to the non-scrambled MDS sequence M_1M_2.

Let $\mathcal{A} = (\pi, \lambda)$ be a simple arrangement of Z_N. For $i \in Z_N$ denote with k_i the symbol in the projection $\text{proj}(\lambda(\{2i-1, 2i\})) = \{k_i\}$. An augmented pointer sequence $w_{\mathcal{A}}$ on N letters can be associated to $\mathcal{A}$ by setting $w_{\mathcal{A}} = a_1 a_2 \cdots a_{2N}$, where for all $i \in Z_N$,

$$(a_{2i-1}, a_{2i}) = \begin{cases} (k_i - 1, k_i), & \text{if } \lambda(\{2i-1, 2i\}) = \{k_i\}, \\ (k_i, k_i - 1), & \text{if } \lambda(\{2i-1, 2i\}) = \{-k_i\}. \end{cases}$$

Note that for each augmented pointer sequence $w_{\mathcal{A}} = a_1 \cdots a_{2N}$ on N letters associated to a simple arrangement $\mathcal{A}$, and for every even index $i \in Z_{2N}$, the symbols a_{i-1}, a_i in $w_{\mathcal{A}}$, are immediate successors in the integer ordering (i.e. $|a_{i-1} - a_i| = 1$).

Proposition 2.4.7. *The mapping $\mathcal{A} \longmapsto w_{\mathcal{A}}$ is an injection from the set of simple arrangements of Z_N into the set of augmented pointer sequences on N letters.*

Proof. Let $\mathcal{A} = (\pi, \lambda), \mathcal{A}' = (\pi', \lambda')$ be two simple arrangements of Z_N. Suppose $w_{\mathcal{A}} = w_{\mathcal{A}'} = a_1 \cdots a_{2N}$. Then for all $i \in Z_N$,

$$\begin{cases} \lambda(\{2i-1, 2i\}) = \lambda'(\{2i-1, 2i\}) = a_{2i}, & \text{if } a_{2i-1} < a_{2i}, \\ \lambda(\{2i-1, 2i\}) = \lambda'(\{2i-1, 2i\}) = -a_{2i-1}, & \text{if } a_{2i-1} > a_{2i}. \end{cases}$$

□

Non-surjectivity of the mapping $\mathcal{A} \longmapsto w_{\mathcal{A}}$ can be observed by noting that there are pointer sequences $w = a_1 \cdots a_{2N}$ where $|a_i - a_{i-1}| \neq 1$, for some number of indices $i \in Z_N$. For example, in the word 1351352424, no two consecutive letters are immediate successors in the integer ordering. Augmenting such a pointer sequence can compensate for this issue by inserting 0 next to a 1 and $N+1$ next to an N. However, only two letters of the original word would have their immediate successors as neighbors in the augmented pointer sequence, while the remaining symbols do not.

Note that in an augmented pointer sequence, there is only one way the MDSs can appear between the pointers. If $w = a_1 \cdots a_{2N}$ is an augmented pointer sequence, then the MDSs must fall between every other pair of consecutive symbols, i.e. the first MDS is defined by pointers a_1, a_2, the second by pointers a_3, a_4, etc. The coupling of every other pair of consecutive symbols in an augmented pointer sequence which correspond to pointers of the same MDSs is consistent with the model of MDS descriptors presented in [19]. However, there are many ways to add symbols 0 and N to a pointer sequence on $N-1$ letters, causing different pairs of pointers in the resulting augmented pointer sequence to correspond to the ends of MDSs between them.

Definition 2.4.8. Two pointer sequences $w_1 = a_1 \cdots a_{2N}$ and $w_2 = b_1 \cdots b_{2N}$, are **equivalent** if for some permutation f of Z_N, either $w_1 = f(b_1) \cdots f(b_{2N})$, or $w_1 = f(b_{2N}) \cdots f(b_1)$. When w_1 and w_2 are equivalent, we write $w_1 \sim w_2$. An **assembly word** is the equivalence class of a pointer sequence under the equivalence relation $\sim$.

Remark 2.4.9. The mapping associating to every pointer sequence w its equivalence class under $\sim$ is a surjection.

Considering only equivalence classes under the relation defined in 2.4.8 leads to further loss of information. In some instances, some pointer sequences in an equivalence class W may admit augmented pointer sequences w such that $w = w_{\mathcal{A}}$, for some simple arrangement $\mathcal{A}$, whereas other pointer sequences from the same equivalence class do not admit any augmented pointer sequences that can be obtained from a simple arrangement via the mapping $\mathcal{A} \longmapsto w_{\mathcal{A}}$. For example, the pointer sequences 113322 and 112233 are equivalent. The word 112233 admits both scrambled and non-scrambled augmented pointer sequences (such as

10122334, and 01122334), whereas no augmented pointer sequence obtained from 113322 corresponds to an MDS sequence.

Figure 2.17 summarizes the relationship between MDS sequences, simple arrangements, and assembly words discussed here. The bijection between MDS sequences and simple arrangements shows that simple arrangements model rearrangement maps which do not contain MDSs that overlap in the mic or correspond to the same regions in the mac. The successive mapping from simple arrangements via intermediates to assembly words, illustrates the various facets of the divergence between the models. Augmented pointer sequences contain the same information as simple arrangements, but not all augmented pointer sequences can be expressed as simple arrangements. pointer sequences miss the information of the exact positioning of terminal MDSs. Finally, assembly words have no knowledge of the correspondence between symbols in the word and pointers in the rearrangement. It should be noted that augmented pointer sequences, pointer sequences, or assembly words, which do not correspond to MDS sequences still model rearrangements of MDSs, but of MDSs from more than one mac contig.

$$\begin{tikzcd}
\text{simple arrangements} \arrow[r, leftrightarrow, "\text{bijection}", "\text{Proposition 2.4.3}"'] \arrow[d, "\text{injection}"', "\text{Proposition 2.4.7}"] & \text{MDS sequences} \\
\text{augmented pointer words} \arrow[d, "\text{surjection}"', "\text{Remark 2.4.6}"] & \\
\text{pointer words} \arrow[d, "\text{surjection}"', "\text{Remark 2.4.9}"] & \\
\text{assembly words} \arrow[r, leftrightarrow, "\text{bijection}", "{[3]}"'] & \text{Isomorphism classes of assembly graphs}
\end{tikzcd}$$

Figure 2.17: The relationship between the various models discussed in this section.

Assembly words are derived in [3, 9] from a spatial graph model called assembly graph. Central to an assembly graph is the notion of rigid vertices which represent the alignments of recombination sites.

Definition 2.4.10. A **cyclic equivalence** $(x_1,\dots,x_d)^{\text{cyc}}$ on a d-tuple $(x_1,\dots,x_d)$ is the set of all cyclic permutations of the tuple and their reverses:

$$\begin{aligned}(x_1,\dots,x_d)^{\text{cyc}} =\big\{&(x_1,\dots,x_d),(x_2,\dots,x_d,x_1),\dots,(x_d,x_1,\dots x_{d-1}),\\ &(x_d,x_{d-1},\dots,x_1),(x_{d-1},\dots,x_1,x_d),\dots,(x_1,x_d,\dots,x_2)\big\}\end{aligned}$$

A **rigid vertex** in a multigraph with loops $G=(V,E)$ is a pair $\big(u,(e_1,\dots,e_d)^{\text{cyc}}\big)$, where $u\in V$, and $(e_1,\dots,e_d)^{\text{cyc}}$ is the cyclic equivalence of a d-tuple of all edges incident to u in some ordering (loops incident to u are listed twice). Two edges e_i, e_j incident to a rigid vertex $(u, (e_1, \ldots, e_d))^{\text{cyc}}$ are called **neighbors with respect to** u if $|i - j| = 1$, or $\{i, j\} = \{1, d\}$.

Figure 2.16(g) depicts a rigid vertex graph. Each vertex is represented by a small disk and the its cyclic equivalence is illustrated by the order in which its incident edges intersect the segment boundary. Listing the members of the cyclic equivalence of a vertex amounts to listing all possible ways of choosing an edge incident to a vertex, then choosing a direction clockwise or counter-clockwise, and finally putting the edges incident to the vertex in order starting from the chosen edge in the chosen direction.

Definition 2.4.11. A **transverse path** in a rigid vertex graph is a sequence $(v_0, e_1, v_1, \ldots, e_n, v_n)$ of vertices v_i, and edges e_j, where no vertex or edge is repeated, except possibly for the first and last vertex, and where no two successive edges e_i, e_{i+1} are neighbors with respect to the vertex v_i between them.

A **polygonal path** in a rigid vertex graph is a sequence $(e_0, v_1, e_1, \ldots, v_n, e_n)$ of vertices v_i, and edges e_j, where no vertex or edge is repeated, except possibly for the first and last edge, and where every two successive edges e_{i-1}, e_i are neighbors with respect to the vertex v_i between them.

For any polygonal, or transverse path γ in a graph G, denote with $E(\gamma)$ the sequence of edges in the order they appear in γ.

The colored segments on the rigid vertex graph depicted in Figure 2.16(g) form a polygonal path. Two polygonal paths are considered disjoint if they have no edges, or vertices in common, except possibly for the first and last edges.

Definition 2.4.12. An **assembly graph** is a finite multigraph with loops consisting of rigid 1- and 4-valent vertices. An assembly graph with a Eulerian transverse path is called **simple**.

Two assembly graphs are considered isomorphic if there exists a graph isomorphism between them which preserves the cyclic equivalences of rigid vertices. Informally, the Eulerian transverse path in a simple assembly graph can be thought of as the micronuclear DNA molecule folding in on itself to align recombination sites (matching pointer sequences). MDSs and IESs are encountered along the Eulerian transverse path in an alternating manner. A polygonal path connects the MDSs in the right order and orientation. The first and last edge of a polygonal path correspond to terminal MDSs which have only one pointer and are thought of as taking up only a portion of the edge closest to the vertex neighboring the edge in the path. When a polygonal path begins and ends with the same edge, the terminal MDSs are thought of as taking up disjoint portions of the edge at opposite ends. In the trivial case of a polygonal path consisting of only one edge, the corresponding MDS rearrangement consists of a single MDS without pointers and is thought of as a taking up a small portion in the middle of the edge which does not touch the vertices incident to the edge.

An assembly graph models the simultaneous alignment of recombination sites (pointers) in the mic. More precisely, vertex-smoothing operations defined in [3] split the 4-valent vertices in an assembly graph in such a way that polygonal paths stay intact and represent the rearranged product sequences. It was shown in [3] that the isomorhpism classes of assembly graphs are in one-to-one correspondence with assembly words. In a simple assembly graph, each vertex is traversed exactly twice by its Eulerian transverse. In [3, 9], the minimum numbers of pairwise disjoint polygonal paths which form a Hamiltonian set in an assembly graph, called the **assembly number**, is heavily studied. Observations made in this section led to insights about the assembly number. Proposition 2.4.13 characterizes augmented pointer sequences that describe assembly graphs with assembly number 1. Corollary 2.4.14 derives a necessary condition for the existence of an augmented pointer sequence which corresponds to an assembly graph with assembly number 1.

Proposition 2.4.13. *Let $w = a_1 \cdots a_{2N}$ be an augmented pointer sequence. Then $w = w_{\mathcal{A}}$ for some simple arrangement $\mathcal{A}$ if and only if the graph $G_w = (V_w, E_w)$ with vertices $V_w = Z_{2N}$ and edges E_w defined by:*

$$\{i,j\} \in E_w \iff \begin{cases} \{i,j\} \neq \{t-1,t\} \textit{ for some even } t \in Z_{2N} \textit{ and } a_i = a_j, \textit{ or} \\ \{i,j\} = \{t-1,t\} \textit{ for some even } t \in Z_{2N} \textit{ and } |a_i - a_j| = 1 \end{cases}$$

is connected.

Proof. Suppose $w = w_{\mathcal{A}}$ for a simple arrangement $\mathcal{A} = (\pi, \lambda)$ of Z_N. Simplicity of $\mathcal{A}$ implies that $\text{proj}(\lambda(\pi)) = \{\{k\}, \text{ for } k \in Z_N\}$, so that the set of pairs $\{a_{i-1}, a_i\}$ for even indices $i \in Z_{2N}$ in some ordering forms the sequence of pairs $S = \big(\{0,1\},\{1,2\},\{2,3\},\dots,\{N-1,N\}\big)$. Then $\{i-1,i\} \in E_w$, for all even indices $i \in Z_{2N}$. Note also that each $k \in Z_N$, except for $k = N$ appear in the sequence S once in a pair together with $k-1$ and once together with $k+1$. Denote with t_k^+ the index of the appearance of symbol k in w which occurs in the sequence S together with $k+1$ and denote with t_k^- the index of the appearance of symbol k in w which occurs in S together with $k-1$. Additionally, let t_0 and t_N be the indices of symbols 0 and N in w, respectively. Then the sequence $(t_0, t_1^-, t_1^+, t_2^-, \dots t_{N-2}^+, t_{N-1}^-, t_{N-1}^+, t_N)$ forms a path in G_w containing all vertices.

Conversely, assume G_w is connected. V_w has $2N$ vertices. Since G_w is connected, E_w must have at least $2N-1$ edges. There are precisely two occurrences of the letters $1, \dots, N-1$ in w, so that at most $N-1$ edges are of the form $\{i,j\}$, where $a_i = a_j$. Then at least N edges must appear between vertices $\{i-1,i\}$ where $i \in Z_{2N}$ is even. Since there are exactly N even numbers in Z_{2N}, there is an edge $\{i-1,i\}$ in E_w for every even $i \in Z_{2N}$. Then $|a_{i-1} - a_i| = 1$, for all even $i \in Z_{2N}$. Since G_w is connected, and all vertices are 1- or 2-valent, the graph is linear. Thus, the pairs $\{a_{i-1}a_i\}$ can be ordered to form the the sequence S.

Then the labelling λ of F_N is defined by:

$$\lambda(\{i-1,i\}) = \begin{cases} \{k\}, & \text{if } k = a_i = a_{i-1}+1 \\ \{-k\}, & \text{if } k = a_i + 1 = a_{i-1}. \end{cases}$$

Then $\mathcal{A} = (F_N, \lambda)$ is simple and $w = w_{\mathcal{A}}$. □

In [9], the authors established that the addition of loops in an assembly graph affects its assembly number. This can be attributed to the fact the a polygonal path can traverse a loop in only one way, unless it is the end point of the path. Loops in assembly graphs correspond to factors of the form aa in pointer sequences. In terms of augmented pointer sequences, an assembly number higher than 1 corresponds to the lack of a simple arrangement which maps to the word. The effect a factor of the form aa has on the existence of a simple arrangement mapping to an augmented pointer sequence is summarized in Corollary 2.4.14.

Corollary 2.4.14. *Let $w = a_1 \cdots a_{2N}$ be an augmented pointer sequence. If $a_{i_0-1} = a_{i_0}$ for some even $i_0 \in Z_{2N}$, then there is no simple arrangement $\mathcal{A}$, such that $w = w_{\mathcal{A}}$.*

Proof. Since $a_{i_0-1} = a_{i_0}$ and i_0 is even, there is no edge between $i_0 - 1$ and i_0. Then because there are precisely N even numbers in Z_{2N}, and $N-1$ symbols appearing exactly twice in w, there must be less than $2N-1$ edges in G_w. Since $|V_w| = 2N$, G_w is not connected. □

Chapter 3

Algorithms for Annotating DNA Rearrangements in Ciliates

3.1 Introduction

Sequence alignment tools such as BLAST and LFASTA are usually insufficient for the analysis of DNA rearrangements in ciliates. Such tools are able to establish a correspondence between matching segments between the micronuclear and macronuclear genomes in the form of a large collection of local sequence alignments. Automation is required to systematically make the various rearrangement maps accessible for further analyses, such as the detection of repeating MDSs, overlapping MDSs or scrambled rearrangement maps. Furthermore, these tools are typically designed for the comparison of sequences on an evolutionary scale. Precursor and product genomes analyzed in the context of DNA rearrangements are a single generation apart (i.e. mother to daughter cell), as opposed to one or more steps on the phylogenetic tree. By default, the popular BLAST algorithm adds gaps to produce long high-scoring alignments. On an evolutionary scale such gaps correspond to mutation events, where sections of DNA were inserted or deleted. Such events occur rarely and are not expected to be observed between mother and daughter cell. Frequently, consecutive MDSs in the macronucleus with little pointer overlap appear in non-scrambled order in the mic. When these mac-consecutive MDSs appear close to each other in the mic, the combined region might falsely be identified as a long gapped alignment as opposed to two separate MDSs. BLAST can be instructed to only return ungapped alignments. However, in some cases combining the alignments to larger gapped alignments via the introduction of very short gaps may be more desirable than keeping them separate. Gaps introduced in this way must be very short to be able to qualify as sequencing, or assembly errors, or allelic variations, as opposed to mutational events on an evolutionary scale. A framework for combining ungapped alignments to gapped alignments in a controlled way is beneficial.

Past analyses of the rearrangement maps of *O. trifallax* used custom scripts to obtain the required annotations [11, 16]. These scripts are unavailable for reuse or adaptation and address repeating and overlapping MDSs by filtering them out, leading to a loss of information potentially significant to the underlying Biology. The only available software tool dedicated to annotating features relevant for DNA unscrambling known to the author is MIDAS [12]. MIDAS lacks transparency in the method it uses to generate rearrangement maps which present the correspondence between MDSs in the micronuclear and macronuclear genomes. The rearrangement maps do not address overlapping MDSs and uses a definition of srambling that works best without the presence of repeating or overlapping MDSs.

So far, *O. trifallax* has been the model organism for the types of DNA rearrangements studied here. This is due in part to the high degrees of scrambling detected by [16] after sequencing the entire micronucleus. However, other ciliate organisms that had not been studied may be subject to comparably severe DNA rearrangements. It is desireable to sequence the genomes of other ciliates in the future and compare the rearrangements taking place between the various species. A tool that is capable of handling a variety of organisms, must be adaptable to the diverse characteristics.

In Section 3.2, an algorithm for the annotation of features relevant to DNA rearrangements in ciliates is provided. Additionally, a generalization of the current notion of scrambling that can be applied to rearrangement maps in the presence of repeating and overlapping MDSs is proposed. An algorithm to detect scrambling in the output of the annotation algorithm is presented, as well. Next, the implementation of a pipeline incorporating the two algorithms is discussed. A variety of parameters is provided along with the algorithms and their implementation to allow for adaptability and user control over the computation. The effect of the parameters on the pipeline computation is tested and the results are discussed. The pipeline is executed with a carefully chosen set of parameter values for a comparison of the annotation it produces with the analyses in [11] and [16]. Finally, the pipeline is run on a more recent macronuclear genome assembly of *O. trifallax* published in [28].

The annotation algorithm includes a method for combining ungapped alignments to larger gapped alignments in a controlled way. This method is developed further in Section 3.3 and its potential in the more general context of gapped sequence alignment algorithms is explored. A command-line tool implementing the algorithm is introduced. The software is rigorously tested and compared to the gapped alignment step executed by BLAST.

3.2 Annotation algorithm for DNA rearrangements in ciliates

Existing computational tools for analyzing sequence data from the micronuclear and macronuclear genomes of *O. trifallax* in the context of DNA rearrangements are based on simplifying assumptions. When applying these tools, some more complex cases which do not satisfy the assumptions are addressed insufficiently or must be filtered out [12, 16, 42]. These tools also lack in transparency and flexibility which complicates the reproducibilty of resulting annotations, and hinders cross-species comparisons. In this section, an algorithm that computes annotations of a precursor genome and its product is presented. The produced annotations describe a rearrangement of segments analogous to the extraction and recombination of MDSs from the micronucleus to form a functioning macronucleus in ciliate biology. Additionally, we specify a notion of scrambling that generalizes upon previous definitions and can be applied to more complex arrangements. An algorithm to detect these scrambled arrangements among those produced by the first algorithm is given, as well. An implementation of the algorithms called **Scrambled DNA Rearrangement Annotation Pipeline (SDRAP)** is presented with a choice of various parameters, such that other genomes undergoing similar rearrangements but with specific differences can still be accommodated and processed with this software. Altogether, this annotation tool facilitates the analysis of scrambled pairs of genomes, such as those found in ciliates, in a consistent, automated, adaptable and reproducible way. The effects the software parameters have on the computation is assessed in a series of test runs with various parameter value sets. Furthermore, results obtained from SDRAP using a carefully chosen set of parameter values and the same genomes from [42] (mac) and [16] (mic), are crossreferenced with results given in [11, 16] based on existing annotations. Finally, the pipeline is applied to the most recent macronuclear genome published in [28] and the results are discussed.

3.2.1 Rearrangement Annotations and Properties

The annotation algorithm implemented by SDRAP accepts as input nucleotide sequences which make up the precursor (micronucleus or germline) and product (macronucleus or somatic) genomes. By convention, we read nucleotide sequences as oriented from the 5' end to the 3' end. Specifically, we define **precursor sequences** and **product sequences** to be words over the alphabet $\Sigma_{\mathrm{DNA}} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. In this section we establish the terminology which we use to describe the severity of disorder in which segments of product sequences are observed inside precursor sequences.

For the remainder of the section, let $\mathcal{C}$ and $\mathcal{G}$ denote a precursor and a product sequence, respectively. The algorithm locates substrings of the precursor sequence $\mathcal{C}$ that resemble substrings of the product sequence $\mathcal{G}$ to determine how the product is distributed along the precursor. In ciliate biology these substrings found in the precursor and product are called **macronuclear destined sequences**, or **MDSs**. They describe the fragments of DNA that are spliced out of the micronucleus, possibly inverted, ligated and retained in the macronucleus during the genome rearrangement process.

Definition 3.2.1. Define a **match of $\mathcal{G}$ on $\mathcal{C}$** to be a triple $M = (\mathrm{Prec}, \mathrm{Prod}, \sigma)$, where $\mathrm{Prec} \subset [1, |\mathcal{C}|]$ and $\mathrm{Prod} \subset [1, |\mathcal{G}|]$ are integer intervals specifying the coordinates of the homologous substrings of $\mathcal{C}$ and $\mathcal{G}$, and where $\sigma \in \{0, 1\}$ indicates their relative orientation. The first component, Prec, is the **precursor interval** of M, and is denoted $\mathrm{Prec}(M)$. The second component, Prod, is the **product interval** of M, denoted $\mathrm{Prod}(M)$, and the third component, σ, is the **orientation** of M, denoted $\sigma(M)$. The **starting points** of the precursor and product intervals of M are $\min(\mathrm{Prec}(M))$ and $\min(\mathrm{Prod}(M))$, respectively. The **ending points** are $\max(\mathrm{Prec}(M))$ and $\max(\mathrm{Prod}(M))$, respectively.

A match describes the location of an MDS in the precursor and its corresponding matching region in the product, as well as the relative orientation of the two segments. A match does not describe the actual nucleotide sequences at the matched regions in the precursor and product sequence. We follow the convention that $\sigma(M) = 1$ indicates that the homologous subsequences of $\mathcal{C}$ and $\mathcal{G}$ defined by $\text{Prec}(M)$ and $\text{Prod}(M)$ appear in the same orientations, whereas $\sigma(M) = 0$ indicates that the reverse complementary sequence of one matches the other.

Collections of matches between two sequences are used to describe the rearrangement map from $\mathcal{C}$ to $\mathcal{G}$. Define $\mathcal{M}(\mathcal{C}, \mathcal{G}) = \{M : M \text{ is a match of } \mathcal{G} \text{ on } \mathcal{C}\}$ to be the set of all matches of $\mathcal{G}$ on $\mathcal{C}$.

Definition 3.2.2. A subset $\mathcal{A} \subseteq \mathcal{M}(\mathcal{C}, \mathcal{G})$ is an **arrangement of $\mathcal{G}$ on $\mathcal{C}$** if for any $M_1, M_2 \in \mathcal{A}$, we have that $\text{Prod}(M_1) \subseteq \text{Prod}(M_2)$ implies $\text{Prod}(M_1) = \text{Prod}(M_2)$. A **subarrangement** of $\mathcal{A}$ is simply a subset of $\mathcal{A}$.

Denote the set $\{\text{Prod}(M) : M \in \mathcal{A}\}$ of product intervals of matches in an arrangement $\mathcal{A}$ with $\mathcal{A}_{\text{Prod}}$ and similarly write $\mathcal{A}_{\text{Prec}} = \{\text{Prec}(M) : M \in \mathcal{A}\}$.

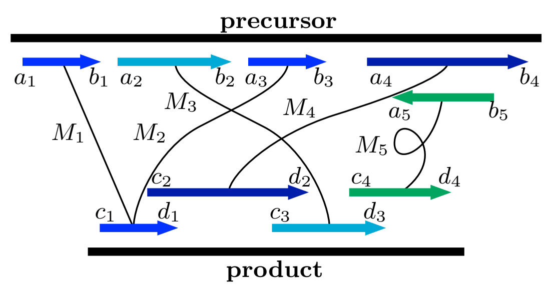
An example of an arrangement is illustrated in Figure 3.1(a). This particular arrangement has five matches, four (M_1, M_2, M_3 and M_4) with equally oriented ($\sigma = 1$), and one (M_5) with oppositely oriented precursor and product intervals.

Precursor intervals of matches found by the annotation algorithm may only be close approximations of the actual regions which survive the rearrangement and make it into the product genome. Hence, we allow for leniency in the scrambling detection algorithm for the extend of the overlap between two precursor intervals necessary to be considered relevant, as explained below.

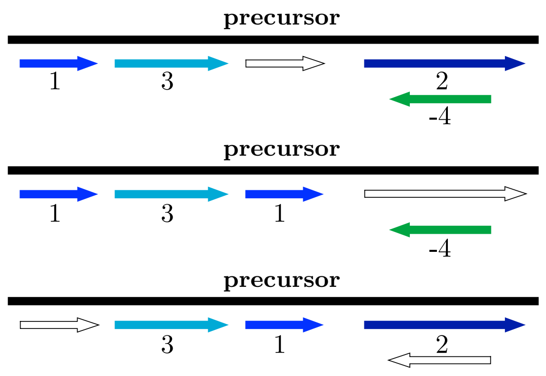
Definition 3.2.3. Let $\mathcal{B} \subseteq \mathcal{A}$ be a subarrangement of an arrangement $\mathcal{A}$ and let s be a positive integer. Two matches $M_1, M_2 \in \mathcal{B}$ are **s-overlapping**, if $|\text{Prec}(M_1) \cap \text{Prec}(M_2)| \geq \min\{s, |\text{Prec}(M_1)|, |\text{Prec}(M_2)|\}$. $\mathcal{B}$ is said to be **s-overlapping** if it contains two distinct matches which are s-overlapping.

Note that the s-overlapping property does not depend on the product intervals of M_1 and M_2. The parameter s gives a threshold that defines when the matches corresponding to two overlapping precursor intervals are considered to be overlapping matches. Small overlaps ($< s$) at the ends of precursor intervals are disregarded by the algorithm depending on the magnitude of s. In particular, when $s = 1$, any nonzero intersection between the precursor intervals of two matches to be considered s-overlapping. The exact value for s is left as input parameter for the algorithm.

Since the product intervals of an arrangement do not contain one another, the intervals can be ordered in a natural way according to their starting points. According to this ordering, define an index of a match



(a)



(b)

Figure 3.1: **(a):** A representation of an arrangement of a product sequence on a precursor sequence (introduced in Definition 3.2.2). The horizontal black lines at the top and bottom signify the precursor and product sequences, respectively. Line segments annotated on the precursor sequence depict the precursor intervals of matches labelled with their starting points a_1, a_2, a_3, a_4, a_5 and end points b_1, b_2, b_3, b_4, b_5. Similarly, the segments annotated on the product sequence represent the product intervals of matches with their start points c_1, c_2, c_3, c_4 and end points d_1, d_2, d_3, d_4 labelled. Each of the black lines connecting a precursor and a product interval corresponds to one of the matches M_1, M_2, M_3, M_4, M_5 in the arrangement. The intervals of the match M_5 are oppositely oriented ($\sigma(M_5) = 0$) illustrated by the loop in the line connecting the corresponding intervals. All other matches M_j $(1 \leq j \leq 4)$ in this arrangement have orientation $\sigma(M_j) = 1$. Thus, depicted in this arrangement are the matches $M_1 = ([a_1, b_1], [c_1, d_1], 1)$, $M_2 = ([a_3, b_3], [c_1, d_1], 1)$, $M_3 = ([a_2, b_2], [c_3, d_3], 1)$, $M_4 = ([a_4, b_4], [c_2, d_2], 1)$, and $M_5 = ([a_5, b_5], [c_4, d_4], 0)$. **(b):** A simplified representation of three subarrangements of the arrangement shown in Figure 3.1(a). Only the precursor intervals of the matches are shown, labelled by the indices (Definition 3.2.4) of the corresponding matches in the arrangement (i.e. the product interval corresponding to a precursor interval with label i occurs in the i'th position on the product sequence relative to the other product intervals of the matches in the arrangement). The index of the match associated with the rightmost product interval is negated to mark that the match has orientation 0. White unlabelled segments indicate matches from the original arrangement that were left out of the subarrangements. Observe that the subarrangement at the top is s-overlapping (Definition 3.2.3) since the matches corresponding to the precursor intervals labelled by the indices -4 and 2 are s-overlapping. The subarrangement in the middle is repeating (Definition 3.2.5) since the matches corresponding to the two precursor intervals labelled by the index 1 have the same product interval. The subarrangement at the bottom is maximal non-repeating and non-s-overlapping. It is maximal with these properties since addition of any more matches from the original arrangement results in the presence of repeating or s-overlapping matches.

in an arrangement which indicates the position of the product interval of a match relative to the positions of the product intervals of the other matches in the arrangement. More precisely, the relation $<$ defined on the set $\mathcal{A}_{\text{Prod}}$ by:

$$\text{Prod}(M) < \text{Prod}(M') \text{ if } \min(\text{Prod}(M)) < \min(\text{Prod}(M'))$$

is a well defined order.

Definition 3.2.4. Let $\mathrm{Prod}(\mathcal{A})$ be the ordered sequence $(\mathrm{Prod}_1, \ldots, \mathrm{Prod}_t)$, where $\{\mathrm{Prod}_1, \ldots, \mathrm{Prod}_t\} = \mathcal{A}_{\mathrm{Prod}}$, and where $\mathrm{Prod}_1 < \mathrm{Prod}_2 < \cdots < \mathrm{Prod}_t$. If $M \in \mathcal{A}$ and $\mathrm{Prod}_i = \mathrm{Prod}(M)$, then i is said to be the **index of M in $\mathcal{A}$**, denoted $i_{\mathcal{A}}(M)$.

Definition 3.2.5. If some $M \neq M'$ in a subarrangement $\mathcal{B} \subseteq \mathcal{A}$ have the same product interval (i.e. $\mathrm{Prod}(M) = \mathrm{Prod}(M')$), then $i_{\mathcal{A}}(M) = i_{\mathcal{A}}(M')$ and we say M and M' **repeat in $\mathcal{B}$**. In this case, the two matches M and M' are said to be **repeats of each other**. A subarrangement $\mathcal{B} \subseteq \mathcal{A}$ is **repeating** if it contains a match which repeats in $\mathcal{B}$.

In the example arrangement $\mathcal{A}$ depicted in Figure 3.1(a), the sequence of product intervals is $\mathrm{Prod}(\mathcal{A}) = ([c_1, d_1], [c_2, d_2], [c_3, d_3], [c_4, d_4])$ (since $c_1 < c_2 < c_3 < c_4$), so that $i_{\mathcal{A}}(M_1) = i_{\mathcal{A}}(M_2) = 1$, $i_{\mathcal{A}}(M_3) = 3$, $i_{\mathcal{A}}(M_4) = 2$ and $i_{\mathcal{A}}(M_5) = 4$. Figure 3.1(b) depicts three subarrangements of $\mathcal{A}$. The subarrangement at the top is s-overlapping since the matches corresponding to the precursor intervals labelled by the indices -4 and 2 are s-overlapping. The subarrangement in the middle is repeating since the matches corresponding to the two precursor intervals labelled by the index 1 have the same product interval. The subarrangement at the bottom is maximal non-repeating and non-s-overlapping. It is maximal with these properties since addition of any more matches from the original arrangement results in the presence of repeating or s-overlapping matches.

As demonstrated above, deciding whether or not a given arrangements is scrambled is complicated by s-overlapping and repeating matches because they pose ambiguities. However, the underlying biological process somehow resolves these ambiguities. Since the precise mechanism is not yet fully understood, any computational decision of one choice over another risks being arbitrary and lead to loss of information. The general approach taken here is to consider as many maximal non-repeating and non-s-overlapping subarrangements of each arrangement as possible and not hide any information. A subarrangement in consideration is maximal with these properties in the sense that the addition of any other match from the arrangement would result in a repeating or s-overlapping subarrangement. To shorten the phrase, call a maximal non-repeating and non-s-overlapping subarrangement **unambiguous**. For unambiguous subarrangements, intuitive notions of scrambling as in [12, 16] can be applied. Despite the maximality of a given subarrangement in consideration, some product intervals may be missing in the set of product intervals of the subarrangement. Hence, considering possibly proper subsets of an arrangement suggests the need to include a notion of completeness of product intervals in the definition of scrambling. To determine whether or not an unambiguous subarrangement is scrambled, three elementary properties are considered. These arrangement properties reflect the state of disorder of matches in the unambiguous subarrangement and the completeness of the product intervals with respect to its parent arrangement.

Definition 3.2.6. An unambiguous subarrangement $\mathcal{B} \subseteq \mathcal{A}$ is **complete in** $\mathcal{A}$ if $\mathcal{B}_{\text{Prod}} = \mathcal{A}_{\text{Prod}}$, and it is considered **consecutive in** $\mathcal{A}$ if $\{i_{\mathcal{A}}(M) : M \in \mathcal{B}\}$ forms a consecutive sequence of integers. Since $\mathcal{B}$ is non s-overlapping, the set $\mathcal{B}_{\text{Prec}}$ forms a sequence $\text{Prec}(\mathcal{B}) = (\text{Prec}_1, \ldots, \text{Prec}_{|\mathcal{B}|})$, ordered by their starting points the same way as the sequence $\text{Prod}(\mathcal{A})$ is ordered. $\mathcal{B}$ is said to be **ordered in** $\mathcal{A}$ if all members of $\mathcal{B}$ have the same orientation σ and $\text{Prec}_j = \text{Prec}(M)$, $\text{Prec}_k = \text{Prec}(M')$, for $j < k$ either implies $i_{\mathcal{A}}(M) < i_{\mathcal{A}}(M')$ and $\sigma = 1$, or $i_{\mathcal{A}}(M) > i_{\mathcal{A}}(M')$ and $\sigma = 0$.

Note that completeness of an unambiguous subarrangement implies consecutivity. The property of being ordered is similar to what [12] consider non-scrambled, except that product intervals may be missing in the arrangement.

In Figure 3.2(a), the two unambiguous subarrangements are $\{1, 3\}$ and $\{2, 3\}$. Both are ordered, neither is complete, and $\{2, 3\}$ is consecutive. The unambiguous subarrangements of (b) are $\{1, -2\}$ and $\{1, 3\}$ and neither is complete while only the latter of the two is ordered and the former is consecutive. Here, $\{1, -2\}$ is not ordered because the orientations of the two matches differ. In (c), the unambiguous subarrangements are $\{-1, 2, 3\}$ and $\{2, 1, 3\}$, both of which are complete, but not ordered.

Definition 3.2.7. For a nonempty subset $\mathcal{S} \subseteq$ {ordered, consecutive, complete}, define an unambiguous (sub-)arrangement to be $\mathcal{S}$-scrambled if at least one of the properties listed in $\mathcal{S}$ does not hold. An arbitrary arrangement is then considered **weakly $\mathcal{S}$-scrambled** if at least one of its unambiguous subarrangements is $\mathcal{S}$-scrambled and it is considered **strongly $\mathcal{S}$-scrambled** if all of its unambiguous subarrangements are $\mathcal{S}$-scrambled.

Weak and strong versions of the properties ordered, consecutive and complete for arbitrary arrangements are defined analogously. The choice for the combination of properties that comprise the set $\mathcal{S}$ is left as input parameter for the algorithm.

Three examples of arrangements are depicted in Figure 3.2. All depicted overlaps are assumed to be of size $\geq s$. The arrangement in 3.2(a) is strongly $\mathcal{S}$-scrambled if the property "complete" is in the set $\mathcal{S}$. The arrangement is neither weakly not strongly $\mathcal{S}$-scrambled if $\mathcal{S} =$ {ordered} and it is weakly but not strongly $\mathcal{S}$-scrambled if the property "consecutive" but not the property "complete" is in the set. In 3.2(b), one of the two unambiguous subarrangements $\{1, 3\}$ is ordered but not consecutive and the other $\{1, -2\}$ is consecutive but not ordered. Thus the depicted arrangement is strongly $\mathcal{S}$-scrambled if both properties "ordered" and "consecutive" are in $\mathcal{S}$. The arrangement in 3.2(c) is strongly complete and consecutive, but strongly $\mathcal{S}$-scrambled if the property "ordered" is in the set $\mathcal{S}$.

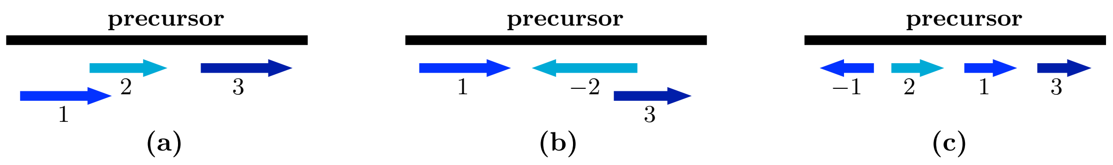


Figure 3.2: The precursor intervals of the matches of three arrangements labelled by their corresponding indices. The overlaps in (a) and (b) are assumed to be of length at least s. Each of the three arrangements has exactly two unambiguous subarrangements. Neither of the unambiguous subarrangements of the arrangement in (a) is complete, so it is neither weakly nor strongly complete. One of the two unambiguous subarrangements is ordered, and the other is both ordered and consecutive. The arrangement depicted in (b) is weakly ordered and weakly consecutive, but strongly $\mathcal{S}$-scrambled if $\{\text{ordered, consecutive}\} \subseteq \mathcal{S}$. All unambiguous subarrangements of the arrangement shown in (c) are complete and consecutive, but none are ordered, so that the arrangement is strongly $\mathcal{S}$-scrambled if "ordered" is a member of $\mathcal{S}$.

3.2.2 Mergeability of Matches

The arrangement annotation algorithm takes as input a set of local alignments between a precursor sequence and a product sequence and returns an arrangement constructed from it. Such alignments can be obtained using softwares such as BLAST [1, 2, 14, 33, 45] or LFASTA [29, 35].

In ciliate biology, the comparison between micronucleus and macronucleus in the context of DNA rearrangements can be expected to show little change on an evolutionary scale. Very strict sequence similarity is expected for the alignments between the two genomes. Popular sequence alignment tools, such as BLAST and LFASTA, however, are optimized for sequence alignments with the purpose of detecting homology over evolutionary distances. To ensure a high degree of similarity between the aligned regions, the input alignments are assumed to be ungapped. While ungapped alignments do not allow for insertions and deletions, slight discrepancies caused, for example, by sequencing artifacts or allelic variations, are to be expected. Thus, by forbidding gaps, long alignments that reflect true biological relationships may be fragmented in the data into smaller ungapped alignments separated by numbers of nucleotides that differ from precursor to product sequence. Due to the differing distances between the ungapped fragments of a gapped alignment, a large ungapped alignment encompassing both fragments may not achieve high sequence similarity. This is caused by the alignment of at least one of the two fragments in the encompassing alignment to be shifted, not matching up the same basepairs as were matched up originally. The introduction of even very small gaps could drastically improve sequence similarity and describe more accurately the actual distribution of matches across the micronuclear and macronuclear genomes. A specific way of combining alignments to larger possibly gapped alignments is discussed here. This technique of merging alignments is designed to offer a high level of control over the lengths of the gaps and the potentially dissimilar regions between ungapped alignments that are incorporated into a larger gapped alignment.

Definition 3.2.8. Define the **shift** between two equally oriented matches $M_1 = ([a_1, b_1], [c_1, d_1], \sigma_1)$ and $M_2 = ([a_2, b_2], [c_2, d_2], \sigma_2)$ as:

$$\begin{cases} \max\{|(a_1 - a_2) - (c_1 - c_2)|, |(b_1 - b_2) - (d_1 - d_2)|\}, & \text{if } \sigma_1 = \sigma_2 = 1 \\ \max\{|(a_1 - a_2) - (d_2 - d_1)|, |(b_1 - b_2) - (c_2 - c_1)|\}, & \text{if } \sigma_1 = \sigma_2 = 0. \end{cases}$$

In Definition 3.2.8, the maximum is taken over the difference of positions based on starting points and based on end points to be able to apply the notion to gapped alignments, where these two differences may not be the same.

Definition 3.2.9. For nonnegative integers t and d, we say that two matches are **(t, d)-mergeable** if they have the same orientation, their shift does not exceed t, and their precursor and product intervals are at most d base pairs apart. To say that $M_1 = ([a_1, b_1], [c_1, d_1], \sigma)$ **is merged with** $M_2 = ([a_2, b_2], [c_2, d_2], \sigma)$ means that M_1 is extended to the match $([\min\{a_1, a_2\}, \max\{b_1, b_2\}], [\min\{c_1, c_2\}, \max\{d_1, d_2\}], \sigma)$.

The shift between two equally oriented matches represents the amount by which the relative positions of the precursor intervals differ from the relative positions of the product interval. When the orientation of the two matches is zero, then the difference of the end points of the precursor intervals is compared to the difference of the starting points of the product intervals because the regions of the precursor sequence defined by the precursor intervals correspond to the reverse complement of the DNA substrings in the product sequence defined by the product intervals. Similarly, with reversed orientations, the distance between the difference of the starting points of the precursor intervals and the difference of the end points of the product intervals is computed and limited by t to account for the inverted orientation while determining how alike the precursor and product intervals are relatively positioned. Observe that the shift also represent the difference in length of the precursor and product intervals of the resulting merged match. Any end-to-end sequence alignment between two different length sequences must insert gaps of total length at least as much as the sequences differ in length. Thus, limiting the shift by the parameter t directly controls the possible gap length introduced via merging. Note that when precursor and product intervals of two (t, d)-mergeable matches do not overlap, the resulting merged alignment aligns some of the characters between the intervals. Frequently such characters may be mismatches, especially if the original alignments already are locally optimal. Thus, the parameter d limits the number of mismatches that can be introduced in this way. The values for t and d are left as input parameters for the algorithm.

Various configurations of equally oriented matches $M_1 = ([a_1, b_1], [c_1, d_1], \sigma_1)$ and $M_2 = ([a_2, b_2], [c_2, d_2], \sigma_2)$ in Figure 3.3 illustrate examples and non-examples of (t, d)-mergeable matches. In (a),

$\sigma_1 = \sigma_2 = 1$, and a_1 and a_2 are positioned relative to each other identically to how c_1 and c_2 are positioned (i.e. $a_1 - a_2 = c_1 - c_2$) and analogously b_1, b_2 are positioned the same way as d_1, d_2, the shift is 0. Similarly, in (b), $\sigma_1 = \sigma_2 = 0$, and (a_1, a_2) and (b_1, b_2) are positioned the same way as (d_2, d_1) and (c_2, c_1), respectively, so that the shift is 0, as well. The distance between precursor intervals differs from the distance between product intervals in (c) and (f) implying a non-zero shift. These two cases are (t, d)-mergeable if the shift is not greater than t and if the space between precursor intervals as well as product intervals is not larger than d. The intervals in (d) have no basepairs in between them, but there is a difference in the length of the overlap. If this non-zero shift exceeds t, the two matches are not (t, d)-mergeable. The intervals in (e) are not in a favorable configuration for merging since the order of precursor intervals and corresponding product intervals coincide even though the matches have orientation $\sigma_1 = \sigma_2 = 0$. The shift of these two matches here is the larger of the two distances $|(a_1 - a_2) - (d_2 - d_1)|$ and $|(b_1 - b_2) - (c_2 - c_1)|$. Since $(a_1 - a_2)$ and $(b_1 - b_2)$ are negative, but $(d_2 - d_1)$ and $(c_2 - c_1)$ are positive due to the configuration of the intervals, the shift is much larger than if the intervals appeared in opposite order in one of the two sequences. Even for relatively high values of t, the two matches are not (t, d)-mergeable.

3.2.3 Arrangement Annotation Algorithm

The input set of alignments $\mathcal{H}$ may not be an arrangement because the product intervals of some may properly contain the product intervals of others. Consequently, the set $\mathcal{H}_{\text{Prod}}$ of product intervals of members of $\mathcal{H}$ may not be totally ordered under the desired order relation and the indices of the members of $\mathcal{H}$ are not well defined. Without the indices of the members of $\mathcal{H}$ defined, the arrangement properties introduced in Section 3.2.1 cannot be determined. As a solution to this problem, the algorithm first computes a smaller preliminary set $\mathcal{A}_0$ that is an arrangement by giving precedence to alignments of higher quality according to bitscore and percent identity, which are given for all local alignments as part of the output of sequence alignment tools such as BLAST. Next, the algorithm combines $\mathcal{A}_0$ with a set $\mathcal{A}_1$ of additional matches computed from $\mathcal{H}$ whose product intervals may contain or be contained in product intervals of members of $\mathcal{A}_0$. The resulting set $\mathcal{A}_0 \cup \mathcal{A}_1$ is treated by the pipeline to be the final arrangement of the current product on the precursor. To the members of $\mathcal{A}_0$ the algorithm assigns their index in $\mathcal{A}_0$ and to the members of $\mathcal{A}_1$ it assigns the index of those members of $\mathcal{A}_0$ whose product intervals they intersect sufficiently, as explained in the detailed algorithmic description of this procedure below.

To obtain the set of preliminary matches $\mathcal{A}_0$, the input alignments are sorted lexicographically by bitscore and percent identity as primary and secondary sort keys, respectively, so that alignments of better quality are given precedence over those of lower quality. The arrangement $\mathcal{A}_0$ is built from the sorted sequence $\mathcal{H}_0$ of members of $\mathcal{H}$ by deciding for each $H \in \mathcal{H}_0$, one by one, whether or not to add H to the current arrangement,

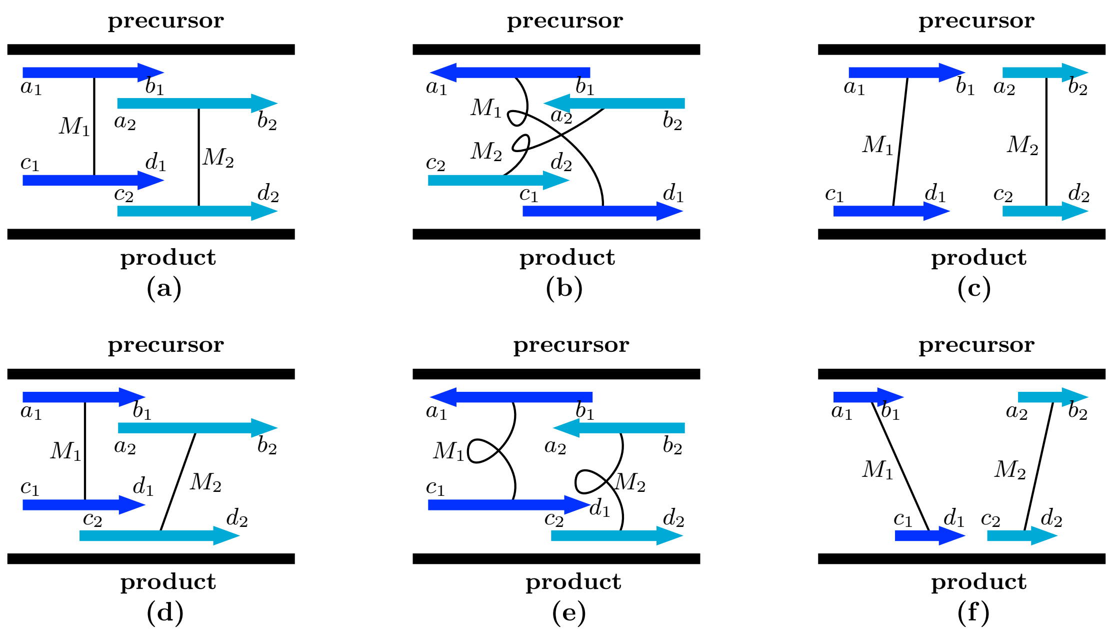


Figure 3.3: Various configurations of equally oriented matches $M_1 = ([a_1, b_1], [c_1, d_1], \sigma_1)$ and $M_2 = ([a_2, b_2], [c_2, d_2], \sigma_2)$ illustrating examples and non-examples of (t, d)-mergeable matches. In (a), $\sigma_1 = \sigma_2 = 1$, and a_1 and a_2 are positioned relative to each other identically to how c_1 and c_2 are positioned (i.e. $a_1 - a_2 = c_1 - c_2$) and analogously b_1, b_2 are positioned the same way as d_1, d_2, the shift is 0. Similarly, in (b), $\sigma_1 = \sigma_2 = 0$, and (a_1, a_2) and (b_1, b_2) are positioned the same way as (d_2, d_1) and (c_2, c_1), respectively, so that the shift is 0, as well. Since the shift is 0 and there are no basepairs between the precursor or product intervals, these two cases illustrate examples of (t, d)-mergeable matches. The distance between precursor intervals differs from the distance between product intervals in (c) and (f) implying a non-zero shift. These two cases may still be (t, d)-mergeable if the shift is not greater than t and if the space between precursor intervals as well as product intervals is not larger than d. The intervals in (d) have no basepairs in between them, but there is a difference in the length of the overlap. If this non-zero shift exceeds t, the two matches are not (t, d)-mergeable. The intervals in (e) are not in a favorable configuration for merging since the order of precursor intervals and corresponding product intervals coincide even though the matches have orientation $\sigma_1 = \sigma_2 = 0$. The shift of these two matches here is the larger of the two distances $|(a_1 - a_2) - (d_2 - d_1)|$ and $|(b_1 - b_2) - (c_2 - c_1)|$. Since $(a_1 - a_2)$ and $(b_1 - b_2)$ are negative, but $(d_2 - d_1)$ and $(c_2 - c_1)$ are positive due to the configuration of the intervals, the shift is much larger than if the intervals appeared in opposite order in one of the two sequences. Even for relatively high values of t, the two matches are not (t, d)-mergeable.

and if so, whether or not it should be merged with or replace members of the current arrangement, as explained below.

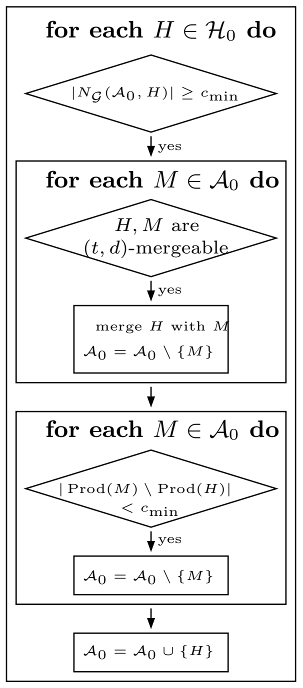
Let $\mathcal{A}'_0$ be the current arrangement, let $H \in \mathcal{H}_0$ be the next alignment being considered and let

$$N_{\mathcal{G}}(\mathcal{A}'_0, H) = \text{Prod}(H) \cap \Big(\mathcal{G} \setminus \bigcup_{M \in \mathcal{A}'_0} \text{Prod}(M)\Big).$$

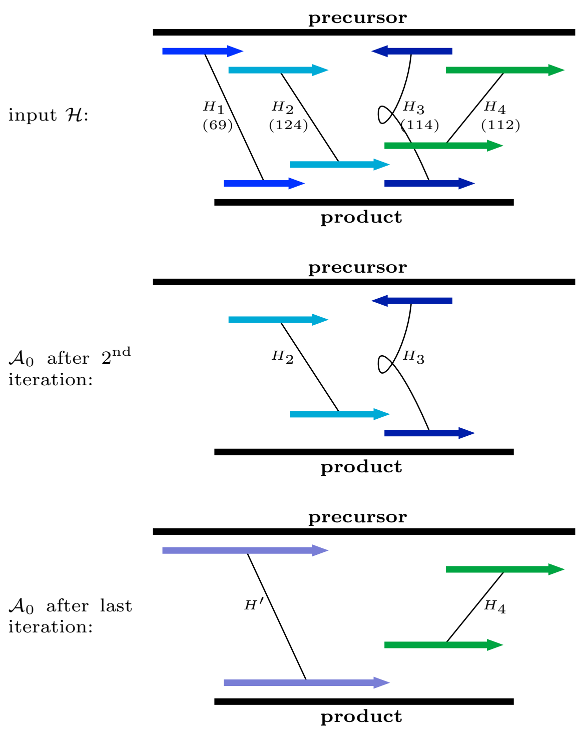
The set $N_{\mathcal{G}}(\mathcal{A}'_0, H)$ represents the subset of $\text{Prod}(H)$ that is not contained in the product interval of any of the members of the current arrangement $\mathcal{A}'_0$. When $|N_{\mathcal{G}}(\mathcal{A}'_0, H)| \geq c_{\min}$, for some predefined positive number $c_{\min}$, the alignment H continues to the next phase in which it is added to $\mathcal{A}'_0$ after potential removal of redundant alignments in $\mathcal{A}'_0$ and potential merging with alignments of $\mathcal{A}'_0$. When $|N_{\mathcal{G}}(\mathcal{A}'_0, H)| < c_{\min}$, the alignment is disregarded, and the algorithm continues with the next alignment. The parameter $c_{\min}$ establishes the threshold of how many base pairs in $\mathcal{G}$ that are not already covered by product intervals of members of $\mathcal{A}'_0$ must be covered by H to qualify for placement in $\mathcal{A}'_0$. The value $c_{\min}$ is left as an input parameter of the algorithm. To place H in $\mathcal{A}'_0$ appropriately, when it qualifies, each member $M \in \mathcal{A}'_0$ is checked for (t, d)-mergeability with H. Whenever that is the case, H is merged with M and M is removed from $\mathcal{A}'_0$. Next, the algorithm checks for each M in the new set $\mathcal{A}'_0$ whether $|\operatorname{Prod}(M) \setminus \operatorname{Prod}(H)| < c_{\min}$, so as to test whether M is now made redundant to H. Whenever that is the case, M is removed from $\mathcal{A}'_0$. Finally, H is added to $\mathcal{A}'_0$ and the algorithm continues with the next alignment in $\mathcal{H}_0$. A flowchart for this algorithm and stages of its execution when finding $\mathcal{A}_0$ from an example input set $\mathcal{H}$ of alignments is illustrated in Figure 3.4.

The returned set of preliminary matches $\mathcal{A}_0$ is sorted according to the order relation for its product intervals defined in Section 3.2.1. Indices of the preliminary matches in $\mathcal{A}_0$ can easily be computed at this point via a linear traversal of the list $\mathcal{A}_0$. The set $\mathcal{A}_1$ of additional matches is obtained from the subset $\mathcal{H}' \subseteq \mathcal{H}$ of alignments which do not make up members of $\mathcal{A}_0$ already. The algorithm for additional match annotation begins with the input set $\mathcal{H}'$ and an empty set $\mathcal{A}'_1$. One by one, the algorithm considers the next member H from $\mathcal{H}'$ in any order. First, H is merged with any (t, d)-mergeable member currently contained in $\mathcal{A}'_1$. Each member of $\mathcal{A}'_1$ that H is merged with in this way is removed from the set. Next, for each member M of $\mathcal{A}_0$, the size of the intersection between the product intervals of M and H is computed. If $|\operatorname{Prod}(M) \cap \operatorname{Prod}(H)|/|\operatorname{Prod}(H)|$ meets or exceeds a predefined threshold r, a copy of H is added to $\mathcal{A}'_1$ with the same index as M. Note that multiple copies of H may be added to $\mathcal{A}'_1$ with different indices assigned to them. While not technically part of the definition, here two matches are considered different if their (artifically assigned) index differs. When $|\operatorname{Prod}(M) \cap \operatorname{Prod}(H)|/|\operatorname{Prod}(M)|$ is smaller than the threshold r, it is still added to $\mathcal{A}'_1$, but flagged as insufficient match. None of the insufficient matches are considered additional matches, but during the execution of the algorithm, they may still merge into other alignments. At the end, the set $\mathcal{A}'_1$ without the matches flagged as insufficient is returned as the set $\mathcal{A}_1$ of additional matches. The threshold r determines the proportion of the product interval of a preliminary match that must be covered by an additional match for it to be considered a match of the same index. The value for r is left as input parameter for the algorithm.

At the end of the additional match annotation step, the combined sets $\mathcal{A}_0 \cup \mathcal{A}_1$ is returned as the arrangement of $\mathcal{G}$ on $\mathcal{C}$. In total the algorithm, takes $O(|\mathcal{H}|^2)$ time. The parameters $t, d, c_{\min}$ and r, offer substantial control over the decisions made during the computation.


for each $H \in \mathcal{H}_0$ do
$|N_{\mathcal{G}}(\mathcal{A}_0, H)| \geq c_{\min}$
yes
for each $M \in \mathcal{A}_0$ do
H, M are (t,d)-mergeable
yes
merge H with M
$\mathcal{A}_0 = \mathcal{A}_0 \setminus \{M\}$
for each $M \in \mathcal{A}_0$ do
$|\operatorname{Prod}(M) \setminus \operatorname{Prod}(H)| < c_{\min}$
yes
$\mathcal{A}_0 = \mathcal{A}_0 \setminus \{M\}$
$\mathcal{A}_0 = \mathcal{A}_0 \cup \{H\}$

(a)



(b)

Figure 3.4: **(a):** The flowchart for the preliminary arrangement annotation algorithm. Beginning with a sequence of alignments $\mathcal{H}_0$ sorted lexicographically by (bitscore, percent identity), and the output set $\mathcal{A}_0$ empty, the algorithm processes one member H of $\mathcal{H}_0$ at a time. First, the size of the set $N_{\mathcal{G}}(\mathcal{A}_0, H)$ of positions in $\mathcal{G}$ covered by Prod(H) but none of the product intervals of members of $\mathcal{A}_0$ is computed. If that number is less than $c_{\min}$, then the algorithm continues with the next member of $\mathcal{H}_0$. If $|N_{\mathcal{G}}(\mathcal{A}_0, H)|$ does meet or exceed the threshold $c_{\min}$, then H is merged with any member of $\mathcal{A}_0$ that is (t,d)-mergeable with H. Each match in $\mathcal{A}_0$ that was merged into H, is removed. Next, any member of $\mathcal{A}_0$ whose product interval does not cover more than $c_{\min}$ positions that are not already cover by the product interval of the merged match H is removed. Lastly, H is added to $\mathcal{A}_0$. The set $\mathcal{A}_0$ is the set of preliminary matches returned at the end of the algorithm. **(b):** An example input $\mathcal{H}$ (top) and $\mathcal{A}_0$ at the ends of the second (middle) and last (bottom) iterations of the outermost loop shown in (a). The numbers in parentheses below the labels of the alignments in the representation of $\mathcal{H}$ at the top, are their bitscores. The alignments H_1 and H_2 are assumed to be (t,d)-mergeable. The members of the sequence $\mathcal{H}_0$ are considered in the order of decreasing bitscore: H_2, H_3, H_4 and then H_1. After the second iteration, $\mathcal{A}_0$ consists of H_2 and H_3. Neither of the conditionals in the two inner loops evaluates to true at any time during the first two iterations. During the third iteration ($H = H_4$), the conditional in the second inner loop evaluates true when $M = H_3$, since $|\operatorname{Prod}(H_3) \setminus \operatorname{Prod}(H_4)| = 0 < c_{\min}$. Thus, H_4 replaces H_3 in $\mathcal{A}_0$. During the fourth iteration ($H = H_1$), the conditional in the first inner loop evaluates true when $M = H_2$, so that H_1 and H_2 are merged to form H' which replaces H_2 in $\mathcal{A}_0$. Observe that we assume that during every iteration of the outermost loop, the conditional at the top of the loop body evaluates true. This means that at any point during the execution of the algorithm, at least $c_{\min}$ base pairs fall into the product interval of the alignments currently in consideration but not into the product intervals of current members of $\mathcal{A}_0$.

3.2.4 Property Computation

Given an unambiguous subarrangement of an arrangement of a product on a precursor sorted by the order of starting points of precursor intervals, a straightforward linear scan suffices to determine whether or not the subarrangement is complete, consecutive, or ordered in the parent arrangement. Note that the number of matches in an arrangement of a product on a precursor is at most the number of HSPs between the two sequences, so that $O(|\mathcal{A}|) = O(|\mathcal{H}|)$. Therefore, computing the weak and strong versions of these properties for an arrangement $\mathcal{A}$ with all its m unambiguous subarrangements $\mathcal{B}_1, \ldots, \mathcal{B}_m$ given sorted by starting points of precursor intervals, takes $O(\sum_{j=1}^{m} |\mathcal{B}_j|) = O(m|\mathcal{A}|) = O(m|\mathcal{H}|)$ time.

In order to find the unambiguous subarrangements of an arrangement $\mathcal{A}$, we reduce the problem to the problem of listing all maximal cliques in a graph. Consider the graph $G = (\mathcal{A}, E)$ with vertex set $\mathcal{A}$ (the set of matches), where two vertices (matches) $M_1 \neq M_2 \in \mathcal{A}$ are adjacent if and only if they are not repeats of each other and they are not s-overlapping. We call G the **graph induced by the arrangement** $\mathcal{A}$. Then two vertices in G do not have an edge between them if and only if the corresponding matches are repeats of each other or are s-overlapping. Thus, the unambiguous subarrangements of $\mathcal{A}$ are precisely the maximal cliques in G.

The algorithm listing the maximal cliques in the graph induced by a given arrangement used in this algorithm is based on the results by Makino et al. [30] and it is output sensitive in the number of maximal cliques being computed. Arbitrary graphs can have a exponential number of maximal cliques [32]. Thus, an upper bound u ($1 \leq u \in \mathbb{Z}$) is imposed to limit the computational complexity of the algorithm. The computation of maximal cliques in each of the t connected components of a graph induced by an arrangement is terminated after u cliques were obtained. The algorithm finds the maximal cliques of an arrangement $\mathcal{A}$ in $O(t|\mathcal{A}|^3 \cdot u)$ time.

Forming the graph $G = (\mathcal{A}, E)$ induced by an arrangement $\mathcal{A}$ can be done in $O(|\mathcal{A}|^2)$ time via pairwise comparison of the indices of the matches and the overlaps between their precursor intervals. Next, the pipeline obtains the connected components $G_1, \ldots, G_t$ of G using a simple recursive DFS algorithm, which takes $O(|\mathcal{A}| + |E|)$ time. Thus, computing the properties of $\mathcal{A}$ takes $O(t|\mathcal{H}|^3 \cdot u) = O(|\mathcal{H}|^4 \cdot u)$ time. For the algorithm computing the properties of the arrangements control over the computation is provided by specifying the values of s, u, and $\mathcal{S}$.

3.2.5 Scrambled DNA Rearrangement Annotation Pipeline

The arrangement annotation and arrangement property computation algorithms were implemented as a web application called **Scrambled DNA Rearrangement Annotation Pipeline**, or **SDRAP**, using PHP 5.3.3 and MySQL 5.6.31 with Apache 2.2.15 on a linux server running with the CentOS 6.7 operating system. The user interface was implemented using HTML, CSS and javascript and can be accessed at `https://knot.math.usf.edu/SDRAP`. The application accepts the input genomes in FASTA format. The code for SDRAP and documentation are available at `https://github.com/JasperBraun/SDRAP`. The steps of the pipeline are:

1. Detect and mask telomeric sequences at the ends of product sequences.
2. BLAST product sequences as query against precursor sequences as subject.
3. Annotate arrangements.
4. Compute arrangement properties.

SDRAP implements its own telomere detection algorithm which is a heuristic adaptation of the Smith-Waterman gapped local sequence alignment algorithm [41]. Since many such variants exist which use the same or a similar strategy (such as BLAST and LFASTA), the algorithm is not discussed here. BLAST is run internally after telomeres are masked as the second step of the pipeline. SDRAP uses Nucleotide-Nucleotide BLAST 2.2.31+ with parameters `-task megablast -ungapped -lcase_masking -word_size 18 -dust no -max_hsps 10000 -max_target_seqs 10000`. In the remainder of the computation, SDRAP only considers the portions of the product sequences between the detected telomeres at either end, if any. When a telomere at the 5' or 3' end is missing, the product sequence included in subsequent steps includes the entire prefix or suffix, respectively.

SDRAP applies the arrangement annotation algorithm discussed in Section 3.2.3 to the set of local alignments returned by BLAST during the second step. It returns annotations of precursor and product intervals of the resulting matches. In addition to $t, d, c_{\min}$, and r discussed in Section 3.2.3, the software accepts minimum thresholds $\ell_{\text{pre}}, b_{\text{pre}}, p_{\text{pre}}$ for length, bitscore and percent identity that alignments returned by BLAST must satisfy to be included in the set $\mathcal{H}$ of alignments used for preliminary match annotation. Another set of parameters $b_{\text{add}}, p_{\text{add}}$ is provided which define minimum bitscore and percent identity thresholds that alignments must satisfy to be included in the set $\mathcal{H}'$ of alignments used for additional matches.

Precursor and product interval annotations are complemented with annotations of **pointers**, **gaps**, **fragments** and **eliminated sequences**, which are defined here. Let $\mathcal{A}_0$ be the preliminary arrangement of a precursor and a product sequence. A **pointer** is the region of overlap, if any, between two consecutive product intervals in the sequence $\text{Prod}(\mathcal{A}_0)$, together with the corresponding two regions in the precursor. These repeats in the precursor genome take a guiding role for the recombination during macronuclear formation. Since no two alignments in the preliminary arrangement $\mathcal{A}_0$ determined by the algorithm described in Section 3.2.3 have the same product interval, the overlapping region between two product intervals of alignments A_i, A_j that are immediate successors in $\text{Prod}(\mathcal{A}_0)$ corresponds to exactly two regions in the precursor sequence. These two regions are located at the ends of $\text{Prec}(A_i)$ and $\text{Prec}(A_j)$. Pointer annotations for both precursor and product are returned. The **gaps** in a product sequence with respect to a precursor sequence are the regions in the product complementary to the product intervals of the preliminary matches. Gaps represent regions that are not covered by the precursor sequence, i.e. gaps in coverage. The regions complementary to the precursor intervals of the union of all arrangements on a precursor sequence are annotated as **eliminated sequences**. Eliminated sequences represent the sequences in a precursor that are separated from the DNA that contributes to the macronuclear DNA, i.e. the sequences that are eliminated during macronuclear formation. Minimum length thresholds ℓ_{ptr}, ℓ_{gap}, and ℓ_{elim} define the minimum lengths of pointers, gaps, and eliminated sequences, respectively, to be annotated as such. When one of these features is shorter in length than its respective threshold, it is not annotated. All three length thresholds are provided as input parameters. **Fragments** of matches are those matches in the set $\mathcal{H}'_1$ that are not returned as additional matches at the end of the additional match annotation algorithm described in Section 3.2.3. A match H in $\mathcal{H}'_1$ is not included in the additional match annotation if for all preliminary matches M, $|\operatorname{Prod}(H) \cap \operatorname{Prod}(M)|/|\operatorname{Prod}(M)| < r$. All such matches H are annotated as fragments. Fragments represent statistically significant alignments that were not captured by the annotation because they are redundant to larger or higher scoring alignments. Whenever a fragment satisfies the inequality:

$$|\operatorname{Prod}(H) \cap \operatorname{Prod}(M)|/|\operatorname{Prod}(M)| \geq r'$$

for some preliminary match M and a smaller threshold $r' < r$, then a copy of it is annotated as a fragment with the index of M. When a fragment does not satisfy the inequality for any preliminary match M, it is annotated as a fragment without index. The threshold r' is provided as another input parameter of the software. All annotations are given labels providing as much information as possible to establish the correspondence of the annotated features in one genome with location of corresponding annotations in the other genome.

Arrangement properties of the arrangements are returned as a simple table listing each arrangement and its various arrangement properties. To be able to further limit computational complexity the parameters s, u, and $\mathcal{S}$ of the arrangement property computation algorithm discussed in Section 3.2.4, are supplemented with a threshold q that determines the proportion of the product sequence that must be covered by preliminary matches of a precursor sequence for arrangement properties of the arrangement between the two sequences to be computed.

One of the goals of the software is to offer a uniform way to analyze not only the DNA rearrangements of *O. trifallax* but also of other genome pairs undergoing a similar process, such as the genomes of other ciliates. Many parameter values should be chosen on a context-specific basis, so an optimal choice of default parameter values is difficult. To assess the effect different parameter values have on the computation, the pipeline was run with various parameter values on the precursor and product genomes of the organism *Oxytricha trifallax*, obtained from [16] and [42], respectively. An outdated version of the macronuclear genome was chosen so that the results can be corroborated with previous analyses that used the same genome without having to account for differences in the genome assembly. For each parameter, a low, a center and a high value were chosen and an initial test run was performed with all parameters at their center value. Next, one test run was carried out for each parameter at their low and high value with all other parameters fixed at their center values. Only the parameter ℓ_{pre} was held fixed at its lower value instead of its center value and its lower value was also used for the initial center run. Since there is no natural concept of low or high values for $\mathcal{S}$, the values for this parameter were chosen to reflect different levels of stringency for the properties that an unambiguous subarrangement must satisfy to be considered scrambled. The "low" value was chosen as $\mathcal{S} = \{\text{ordered}\}$, the "mid" value as $\mathcal{S} = \{\text{ordered, consecutive}\}$, and the "high" value as $\mathcal{S} = \{$ordered, consecutive, complete$\}$. Since "ordered" was a member in all three choices for $\mathcal{S}$, an additional run with $\mathcal{S} = \{\text{consecutive, complete}\}$ was conducted. High sequence similarity for aligned precursor and product segments is desirable for DNA rearrangements in ciliates. Consequently, the percent identity threshold for preliminary match annotation p_{pre} was additionally tested for two stricter values than chosen as the "high" value. The exact parameter values used are listed in Appendix A.

A variety of descriptive statistics was obtained from the annotations to evaluate the effect of the individual parameter variations. For all test runs the execution time of the program and the number of arrangements was obtained. The test runs for parameters ℓ_{pre}, b_{pre}, p_{pre}, c_{min}, t, d, b_{add}, p_{add}, r, r', ℓ_{gap}, and ℓ_{ptr} are considered relevant to the arrangement annotation step implementing the algorithm described in Section 3.2.3. For these tests, two subsets $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ of the set of arrangements were extracted. $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ consist of those arrangements where the product sequence without gaps takes up 30% and 90%, respectively, of the total product sequence (only the product region between telomeric ends, if any, is considered). The alignments between precursor and product sequences that have arrangements in $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ were counted. From these, the number of alignments which satisfy the three thresholds $\ell_{\text{pre}}, b_{\text{pre}}$ and p_{pre}, and the number of alignments which satisfy the two thresholds b_{add} and p_{add} were extracted. Next, the numbers of preliminary and additional matches were counted. Additionally, the number of matches derived from two or more merged alignments was counted. Finally, the numbers of gaps, pointers, and fragments was obtained. To describe the effect the parameters $\mathcal{S}$, s, u, and q have on the outcome, the sets of arrangements $\mathbb{A}_q$, and $\mathbb{A}_{90}$ with respective coverages at least q, and 90%, were computed. Next, the numbers of arrangements which had more unambiguous subarrangements than u was obtained. The remaining counts were restricted to the sets $\mathbb{A}_q^*$ and $\mathbb{A}_{90}^*$ of arrangements with coverage at least q and 90%, which did not have more than u unambiguous subarrangements. From the arrangements in $\mathbb{A}_q^*$ and $\mathbb{A}_{90}^*$, the numbers of repeating, s-overlapping, as well as both repeating and s-overlapping arrangements were counted. Additionally the numbers of weakly and strongly scrambled arrangements were obtained.

The results of the test runs are stored in Tables A.2 through A.6 in Appendix A. Some observations made from the tables are discussed for the remainder of this section. In Tables A.2, A.3, and A.4, it can be observed that the test runs of ℓ_{pre} resulted in a significant change in the total number of arrangements (22.5 to 35.3% decrease compared to center run), but had little impact on the sizes of $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ (0.1 to 1.2% decrease compared to center run). This suggests that the majority of arrangements in $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ achieved their high coverages through long alignments, and that many of the shorter alignments align sequence pairs that otherwise do not have alignments. For all other statistics, the test runs of parameter ℓ_{pre} deviated only moderately (0.4 to 6.5%) from the center run.

Running SDRAP with b_{pre} at its low value resulted in an extreme increase of the total number of arrangements (>1800% more than center), but only a small increase in the sizes of $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ ($< 1.4\%$ increase compared to center). At its low value, a considerable decrease of the total number of arrangements was observed (57.0% less than center). Despite this large impact on the total number of arrangements, the sizes of $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$ deviated only moderately (3.4 to 9.7%) from the center run. This suggests that a the majority of high coverage arrangements is captured by the "mid" value for b_{pre}. The remaining statistics were impacted by b_{pre} only moderately when at its "low" value (0.2 to 4.2% deviation from center). The "high" value for this parameter resulted in significant reductions of the counts of preliminary matches, merged matches and pointers (9.6 to 23.2% decrease compared to center), indicating an overall significant loss of information. Note that the "high" value for b_{pre} is 100 which requires a length of at least 52 base pairs (using default megablast scoring parameters), so that setting b_{pre} to 100 led to the exclusion of previously discovered shorter MDSs [16].

The parameter p_{pre} had a high impact on all statistics (>14% deviation from center for $\mathbb{A}_{30}$, or $\mathbb{A}_{90}$ column in "high", or "low" test run). A higher value for this parameter generally decreased the counts, except the numbers of additional matches and merged matches in members of $\mathbb{A}_{30}$. These counts are larger than the center value for p_{pre} at its "high" value and the additional run with the parameter set to 95%, but are much lower (>75% less than center) when $p_{\text{pre}} = 99\%$. Thus, enforcement of high sequence similarity using high values for p_{pre} must be approached with care.

The parameter c_{min} had a moderate impact on the numbers of additional and merged matches in both $\mathbb{A}_{30}$ and $\mathbb{A}_{90}$, as well as on the number of fragments in $\mathbb{A}_{30}$ (0.7 to 3.1% deviation from center) its impact on all other statistics was insignificant (<1% deviation from center). The test runs of t and d deviated moderately from the center run in the numbers of preliminary, additional and merged matches (>1% deviation for $\mathbb{A}_{30}$, or $\mathbb{A}_{90}$ when at "low" or "high" value) with d having more significant impact on the numbers of merged matches

(8.8 to 13.3% deviation from center). This suggests that many merged alignments only have a small overlap at the ends of their precursor or product intervals. The two parameters also notably affected the numbers of fragments (up to 14.0% deviation from center). Parameters specific to the additional match annotation step ($b_{\text{add}}, p_{\text{add}}$, and r) had a significant impact (>10.6% deviation from center at "low" or "high" value) on the numbers of additional matches, merged matches and fragments. As intended, b_{add} and p_{add} also had a significant impact (14.3 to 25.6% deviation from center at "high" value) on the numbers of alignments that satisfy the two thresholds. These three parameters had no significant effect on any other statistic (<1% deviation from center). The parameters r', ℓ_{gap} and ℓ_{ptr} had no significant impact (<1% deviation from center) in all statistics except for the numbers of fragments, gaps and pointers, respectively.

From Tables A.5, and A.6, it can be observed that s had no significant impact on the number of arrangements which had more than u unambiguous subarrangements (<1% deviation from center). On the other hand, the numbers of s-overlapping arrangements, and consequently the numbers of arrangements that are both repeating and s-overlapping were affected by s considerably (48.6 to 85.5% increase compared to center when at its "low" value and 12.1 to 15.4% decrease when at its "high" value). The strong increase in s-overlapping arrangements when s is at its "low" value indicates frequent short overlaps of length at most 5 ("mid" value for s). The test runs of $\mathcal{S}$ had no impact on any of the statistics except for the numbers of weakly and strongly $\mathcal{S}$-scrambled arrangements in $\mathbb{A}_q^*$ and $\mathbb{A}_{90}^*$. When "complete" was added to $\mathcal{S}$ (the "high" run for $\mathcal{S}$), the number of strongly $\mathcal{S}$-scrambled alignments increased considerably (29.6 to 30.3% increase compared to center). Less significant deviations from the center run (6.3 to 9.9%) where observed for the number of weakly $\mathcal{S}$-scrambled arrangements and the "low" value test run of $\mathcal{S}$. Removing the property "ordered" from the set $\mathcal{S}$ led to a drastic decrease in the number of weakly and strongly $\mathcal{S}$-scrambled arrangements (58.6 to 66.1% fewer than center). Arrangements can only have incomplete or non-consecutive unambiguous subarrangements when they have repeats, or are s-overlapping. Less than 20% of the arrangements in $\mathbb{A}_q^*$ and $\mathbb{A}_{90}^*$ are repeating or s-overlapping (in center run). Thus, a much lower number of scrambled arrangements when $\mathcal{S}$ does not include the property "ordered" is not surprising. Setting the parameter u to its "high" value caused a reduction of the sizes of $\mathbb{A}_q \setminus \mathbb{A}_q^*$ and $\mathbb{A}_{90} \setminus \mathbb{A}_{90}^*$ by 29.8 and 31.8%, respectively, compared to the center run. With the increase in arrangements in $\mathbb{A}_q^*$ and $\mathbb{A}_{90}^*$, the number of repeating and s-overlapping arrangements increased by 27.6 to 29.6% compared to center. Interestingly, the numbers of arrangements in $\mathbb{A}_q^*$ and $\mathbb{A}_{90}^*$ which are both repeating and s-overlapping increased by 93.4 and 97.6%, respectively, suggesting that many arrangements with more than 4 ("mid value for u) unambiguous subarrangements are among these. The "high" value for u also significantly increased the numbers of weakly scrambled arrangements (~11% increase compared to center). The numbers of strongly scrambled arrangements, on the other hand, increased only moderately (4.3 to 4.5% compared to center). Setting u to its "low" value had the opposite effect but similar in magnitude. As expected, the parameter q had no effect on $\mathbb{A}^*_{90}$, but had a high impact on $\mathbb{A}^*_q$ (>17% deviation from center in all counts). This is not surprising since the size of $\mathbb{A}_q$ varies between 76569 and 124066 across the test runs of q.

3.2.6 Comparison of Annotations

To demonstrate how the software SDRAP measures up against previous annotation procedures [12, 16], the program was run two more times. Both of the additional tests used the micronuclear genome published in [16] and the macronuclear genome published in [42]. Except for $p_{\text{pre}}, p_{\text{add}}, \ell_{\text{ptr}}, q$, and u, all parameters were kept at the "mid" values discussed in Section 3.2.5 and listed in Table A.1 in Appendix A. To enforce high sequence similarity appropriate for the close relationship between two genomes of the same organism, p_{pre} and p_{add} were set to 99% and 95%, for one of the tests and to 95% and 90%, respectively, for the other. Since pointers as short as two base pairs were observed in *O. trifallax*, $\ell_{\text{ptr}} = 2$ for both tests. In [12, 16], arrangements with coverage of the product sequence less than 30% were excluded from their observations, so that $q = 30$ was used in the SDRAP test runs. To maximize the number of unambiguous subarrangements explored, the parameter u was set to 10 for all test runs.

Table 3.1: Set of parameter values used for additional test runs of the software SDRAP. For the parameters $\ell_{\text{pre}}, b_{\text{pre}}, c_{\text{min}}, t, d, b_{\text{add}}, r, r', \ell_{\text{gap}}$ and s, the "mid" values listed in Table A.1 in Appendix A are used. The two parameter value sets differ only in the percent identity thresholds for alignments considered in the preliminary and additional match annotation steps.

Parameter	strict	lenient
p_{pre}	99.0	95.0
p_{add}	95.0	90.0
ℓ_{ptr}	2	2
u	10	10
q	30.0	30.0

Tables A.7 and A.8 organize various statistics obtained from the SDRAP test runs and corresponding results presented in [16] and [12], respectively. Due to the different approaches taken in the extraction of rearrangement maps from sequence data, the statistics listed for [16] and [12] are not always perfect equivalents of the counterparts listed for SDRAP. The attempt was made to choose values as close as possible in meaning to the compared results. In some cases, more than one statistic that may be relevant for the comparison is listed. When an alignment considered during the additional match annotation algorithm intersects more than one preliminary match sufficiently, it may be annotated multiple times as additional match, each time inheriting the index of the intersecting preliminary match. To ensure the bias in the number of matches introduced by these double counted alignments is taken into consideration, the number of times alignments are annotated in excess of the first time (i.e. an alignment annotated n times as an additional match, is counted $n-1$ times here), is listed in the tables.

The results in [16] are restricted to arrangements and matches of 1- or 2-telomeric product sequences in arrangements with product sequence coverage of at least 90%. Thus, all statistics from SDRAP in Table A.7 are taken only from this restricted set of arrangements and matches. In addition, repeats were filtered in [16], and no concept analogous to "preliminary" and "additional" matches exists. For that reason, several statistics are listed twice, once obtained solely from preliminary matches and once from all matches together. It can be observed that even with the more lenient of the two choices of percent identity thresholds, SDRAP appears to find much fewer matches than [16] (over 15% less matches). However, [16] used ungapped alignments from BLAST whereas SDRAP sometimes merges alignments to form matches. Counting the number alignments that constitute the matches returned by SDRAP, there are only 7.3% less compared to the number of MDSs reported in [16] SDRAP obtains a much higher number of arrangements with a single preliminary match. In the more lenient test run, over 45% of these preliminary matches consist of merged matches. Hence, some of the discrepancy between the numbers of single alignment product sequences can be attributed to the fragmentation of alignments in [16] due to the use of ungapped alignments. Smaller numbers of matches per product sequence kb were obtained from SDRAP (both for weakly scrambled and strongly non-scrambled arrangements). The fragmentation due to the use of ungapped alignments in [16] may be a contributing factor to this difference. Similarly, the median length of matches in both weakly scrambled and strongly nonscrambled arrangements is longer than in [16] It should be emphasized that these test runs used b_{pre} and b_{add} values of 49, which effectively led to the disregard of all alignments of length less than 25. In [16], the word size parameter for BLAST was set to 20, indicating that the minimum alignment length in their data is 20. No other threshold imposed on the alignment lengths is documented in [16].

Table A.8 compares the annotation of SDRAP with the results in [12]. As observed in Section 3.2.5, the results of SDRAP vary greatly with the choice of p_{pre} and p_{add}. SDRAP obtains more preliminary matches but only the lenient percent identity thresholds resulted in more matches overall (preliminary and additional). The number of arrangements with coverage at least 30% was significantly lower for the two SDRAP test runs (47 and 30% less than [12]) Additionally, SDRAP provides notably lower values in almost all other counts, except for the numbers of strongly complete arrangements with coverage at least 90%. In fact, SDRAP detects 36% more scrambled arrangements among strongly complete high coverage arrangements than [12]. However, this comparison must be viewed with caution. A different mac genome was used in [12] and the statistics listed for the SDRAP test runs are not necessarily equivalent to their counterparts in [12]. Note that the number of times additional matches were double counted is relatively large (contributing between 8.5 and 14.9% of the matches). These double counted matches are investigated in Section 3.2.7.

3.2.7 Classification of Extended Data

The models introduced in Chapter 2 are applied to four annotations produced by SDRAP. These annotations were generated by applying each of the two parameter value sets described in Table 3.1 to the micronuclear genome from [16] and the macronuclear genome from [42] and from [28].

For each arrangement, a pairing and its labelling was derived from the data. Three subsets of the arrangements in each annotation were chosen representative sets. The set $\mathbb{A}$ is simply the set of all arrangements. The set $\mathbb{A}_{30}$ is the set of all arrangements where the complement of the gaps in the product sequence (between telomeres, if any) takes up at least 30%. Similarly, the set $\mathbb{A}_{90}^{2\mathrm{T}}$ contains all arrangements where the product sequence is covered at least 90%, but further restricts itself to arrangements of 2-telomeric product sequences. The proportions of unidirectional, crossing and flat pairings, and consistent and coherent labellings derived from arrangements from the three different sets is listed in Table A.9. As expected, the vast majority of pairings ($> 99\%$ in most cases) are unidirectional. Only in the lenient test with the genome from [42] does the proportion of unidirectional pairings derived from arrangements in $\mathbb{A}$ and $\mathbb{A}_{30}$ drop below 99% to 98.7 and 97.3%, respectively. The proportion of coherent and consistent labellings unexpectedly drop as low as 92.4%. This is unexpected since the properties of coherence and consistency were expected to apply to all labellings obtained from arrangements in real-world data. This unexpectedly low proportion of coherent and consistent labellings is potentially a result of the additional match annotation algorithm In this algorithm, an alignment is annotated as a match for each preliminary match whose precursor interval it intersects sufficiently. When the alignment contains the precursor interval of multiple preliminary matches, it is annotated as additional match multiple times with different indices. In the derived pairing and labelling, such a match would translate into a pair x with a label containing all the indices of the intersected preliminary matches. Furthermore, the preliminary matches whose precursor intervals it intersected translate into pairs $x' \neq x$ where $x \sqsupseteq x'$ and $\lambda(x') \subset \lambda(x)$. Such a setup violates the property of consistence. The issue of double counted additional matches is investigated further below. The proportions of flat and non-crossing pairings are relatively high indicating that the majority of arrangements do not have overlapping precursor intervals. The absolute numbers of arrangements and product sequences in $\mathbb{A}_{90}^{2\mathrm{T}}$ as well as the subsets of non-flat arrangements, and repeating arrangements, and product sequences which can only be found in non-flat arrangements and product sequences which can only be found in repeating arrangements are displayed in Table 3.2. The table suggests that SDRAP is capable of providing a meaningful annotation of up to 7.2% more mac chromosomes and 7.4% more arrangements than previous annotation procedures.

The high density and height values observed in pairings derived from the annotations produced by SDRAP deserve further attention. Based on previous experiences with the data, densities as high as reported in

Table 3.2: Total numbers of arrangements and product sequences, as well as non-flat arrangements and product sequences found only in non-flat arrangements and repeating arrangements and product sequences found only in repeating arrangements in the SDRAP test runs applying parameter values listed in Table 3.1 to the micronuclear genome from [16] and the macronuclear genomes from from [42] and [28].

	mac genome from [42]		mac genome from [28]	
	(99.0, 95.0)	(95.0, 90.0)	(99.0, 95.0)	(95.0, 90.0)
# arrangements*	10071	15683	23718	33705
# non-flat arrangements*	110	512	842	1552
# arrangements with repeats*	181	628	409	1062
# product sequences*	9166	12701	21580	27204
# product sequences only in non-flat arrangements*	94	313	776	1182
# product sequences only in arrangements with repeats*	150	488	343	850

* Only arrangements and product sequences from arrangements in $\mathbb{A}_{90}^{2T}$ are considered.

Table A.9 come as a surprise. To further investigate, the frequencies of nontrivial height (> 0) and density (> 1) were plotted in Figures A.1, A.3, A.2, and A.4 in Appendix A for pairings derived from the arrangement sets of interest in the four test runs. It is evident from the figures that only a small fraction of arrangements produces high density, or height values. Furthermore, the maximum observed heights and densities drop severly when restricting attention to $\mathbb{A}_{90}^{2T}$, which can also be observed in Table A.9. To show that such high values in height and density are due to double counting of additional matches, the annotations were filtered according to a set of thresholds $\mathcal{T} = \{0, 5, 25, 50, 75, 95, 100\}$. For each threshold $T \in \mathcal{T}$, a filtered set $\mathbb{A}_T$ was obtained by excluding all additional matches which intersect the preliminary match of the same index by less than T% of the size of the precursor interval of the additional match (i.e. an additional match M_{add} which has the same index as preliminary match M_{pre} is filtered out if $100 \cdot \frac{|\operatorname{Prec}(M_{\text{add}}) \cap \operatorname{Prec}(M_{\text{pre}})|}{|\operatorname{Prec}(M_{\text{add}})|} < T$). The various maximum densities and heights that can be observed in the filtered data sets are plotted in Figure 3.5. The maximum densities and heights drop significantly as the intersection threshold increases and no change is observed between the data sets filtered by thresholds $T = 75, 95$, and 100. Thus, combining this observation with the low observed frequencies of high density and height values, it can be concluded that the surprisingly high values in density and height are caused by a double counting of additional matches in a few isolated arrangements.

To further investigate how big the impact of the double counting of matches is on the quality of the annotation, the numbers of arrangements in $\mathbb{A}, \mathbb{A}_{30}$, and $\mathbb{A}_{90}^{2T}$ where alignments were annotated more than once during the additional arrangement annotation step were obtained and are listed in Table A.10. Additionally, the table lists the average number of times per arrangement in $\mathbb{A}_{30}$ and $\mathbb{A}_{90}^{2T}$ that an alignment was double counted. Furthermore, these counts were duplicated for arrangements with high density and high height, where high density and height are defined by the max height and density values that remained stable for high thresholds in Figure 3.5. The numbers of 1- and/or 2-telomeric sequences in arrangements where additional matches are double counted are presented in Table A.11. These numbers were also duplicated for high density and high height arrangements. From Table A.10 it can be observed that the majority of arrangements which contain alignments that were double counted has low coverage, or lacks telomeres. Furthermore, the average number of double counted matches in high density and high height arrangements are up to two times higher for the macronuclear genome from [42] and over sixty times higher for the macronuclear genome from [28], than the overall average number of double countings in arrangements in $\mathbb{A}_{30}$. From Table A.11 it can be observed that the number of product sequences which appear in high density and height arrangements is negligible.

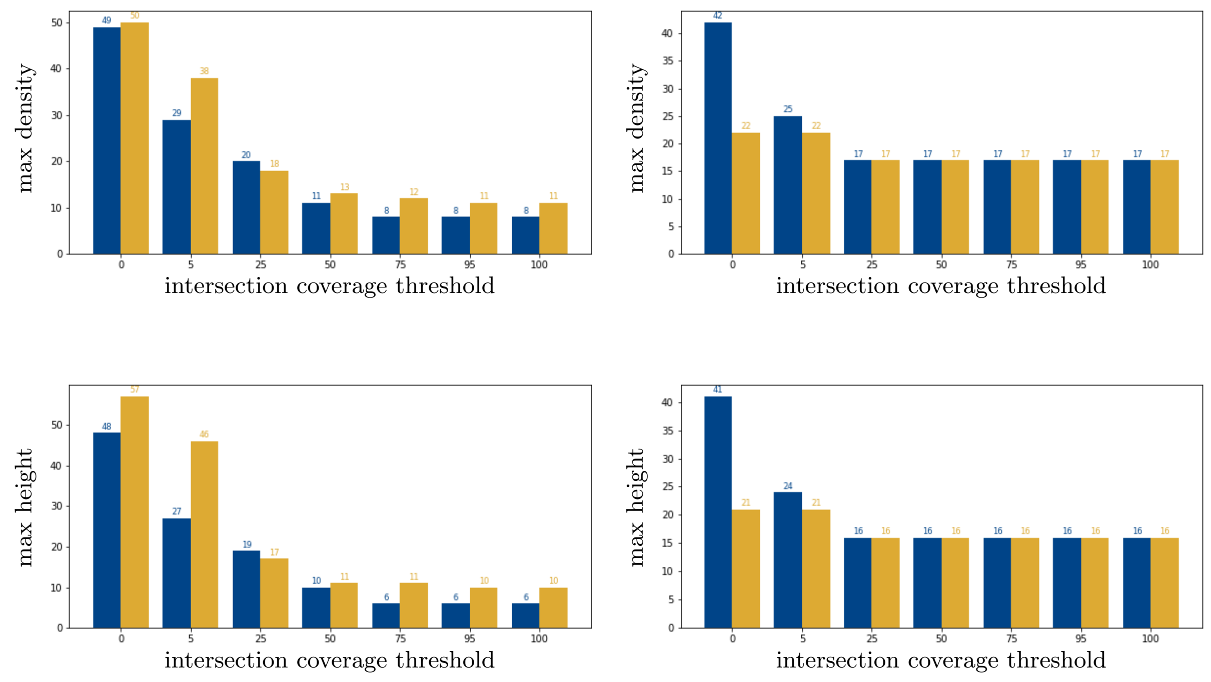


Figure 3.5: The maximum density values (top) and maximum height values (bottom) for SDRAP test runs with $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in blue and $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in yellow, applied to the macronuclear genomes from [42] (left) and [28] (right). The x-axes indicate thresholds used to filter out additional matches that are counted more than once. For each threshold T, any additional match M_{add} which is annotated as a repeat of a preliminary match M_{pre} is filtered out if $100 \cdot \frac{|\operatorname{Prec}(M_{\text{add}}) \cap \operatorname{Prec}(M_{\text{pre}})|}{|\operatorname{Prec}(M_{\text{add}})|} < T$, i.e. if the intersection of the precursor intervals covers the interval of M_{add} by less than T%. When $T = 0$, then no match is filtered out.

3.3 Alignment pasting

A frequent computational task in the field of Bioinformatics is the search for regions of high similarity between two sequences. It typically requires the arrangement of the two sequences alongside each other in a way that maximizes the similarity between two regions within the sequences. If the aligned regions must include both sequences end-to-end, the task is referred to as *global sequence alignment*. When alignments of possibly proper sub-regions of the sequences are sought, the task is referred to as *local sequence alignment*. The aligned regions are only interesting if they strongly resemble each other by some chosen similarity assessment, which typically takes into consideration each aligned pair of characters in two sequences of equal length. Usually, not all characters of the two regions have to match. In the context of *gapped* sequence alignments, characters may, in fact, be added to either sequence to improve similarity and/or obtain sequences of equal length. Of course, such insertions need to be taken into account by the similarity assessment of the alignment. The space of possible alignments between two sequences is drastically inflated by allowing insertions. Ungapped alignments only involve two regions of equal length. Once a pair of regions is chosen, there is only one way the two regions can be aligned. Gapped alignments, on the other hand, do not limit themselves to pairs of regions of equal length since insertion of characters allows for modifying the lengths. Furthermore, given a pair of regions, there is no one unique way of inserting characters in those regions to obtain equal length sequences.

An algorithm solving this problem was introduced by Smith and Waterman in [41], which is based on a dynamic programming algorithm for global sequence alignment by Needleman and Wunsch in [34]. Similarity of alignments in this algorithm is evaluated by assigning a score to alignments in which identical matches contribute positively and mismatches as well as insertions contribute negatively. Informally, this algorithm explores a subset of the possible local alignments between two sequences and returns a set of alignments which optimize the assigned alignment scores. The Smith-Waterman algorithm has time and space complexity proportional to the product of the lengths of the two sequences, which is better than brute force, but may be considered insufficient for large data sets. In order to achieve lower complexity, many algorithms decreasing the size of the space of alignments explored. Two popular softwares - BLAST [1, 2, 14, 33, 45] and LFASTA [29, 35] - implement variations of the Smith-Waterman algorithm which apply various heuristics to reduce the space of possible alignments searched at the cost of the small possibility of producing a suboptimal result. These softwares begin by obtaining a set of ungapped alignments. Next, heuristic variations of the Smith-Waterman algorithm are applied to extend ungapped alignments to larger gapped alignments. Despite the heuristic nature of their variations of the Smith-Waterman algorithm, these softwares still explore a large space of possible alignments which are not included in the output.

A fast method for combining ungapped alignments to obtain larger high-scoring gapped alignments inspired by the technique for merging alignments described in Section 3.2.2 is discussed in this section. The potential of this method in the more general context of gapped sequence alignment algorithms is explored. An algorithm which applies this method to a set of alignments and its implementation are presented. A command-line tool implementing the algorithm is introduced and the software is rigorously tested and compared to the gapped alignment step executed by BLAST.

3.3.1 Formal Definitions of Sequence Alignments and Alignment Scores

For a word $u = a_1 \cdots a_n$ over any alphabet and positive integers $i < j \leq n$, denote with $u_{[i,j]}$ the **factor** $a_i \cdots a_j$ of u. Denote with Σ_{DNA} the alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Call the symbol `-` the **gap character** and let $\mathcal{L}^-_{\text{DNA}} = \Sigma^*_{\text{DNA}} \sqcup \{\mathtt{-}\}$ be the set of shuffles of words over Σ_{DNA} with the gap character. All definitions in this section are illustrated by Figure 3.6.

Definition 3.3.1. Let u, v be two words over the alphabet Σ_{DNA}. An **alignment** of (u, v) is a pair $(\hat{u}, \hat{v})$ of shuffles $\hat{u} = \hat{a}_1 \cdots \hat{a}_k \in u \sqcup \{\mathtt{-}\} \subseteq \mathcal{L}^-_{\text{DNA}}$ and $\hat{v} = \hat{b}_1 \cdots \hat{b}_k \in v \sqcup \{\mathtt{-}\} \subseteq \mathcal{L}^-_{\text{DNA}}$ of u and v, respectively, of equal length, where for all $i = 1, ..., k$, $(\hat{a}_i, \hat{b}_i) \neq (\mathtt{-}, \mathtt{-})$. Each maximal factor in $\hat{u}$ or $\hat{v}$ of the form $(\mathtt{-})^m$ is called a **gap-opening**. An **ungapped** alignment is an alignment without gap-openings. A **local alignment** between two words $u, v \in \Sigma^*_{\text{DNA}}$ is an alignment between a pair of factors of u and v.

Figure 3.6 shows a local alignment between two words u and v. The two words u and v as well as the aligned factors $u_{[i,j]}$ and $v_{[k,l]}$ are located at the top and bottom, respectively. The shuffles $\hat{u}$ and $\hat{v}$ of $u_{[i,j]}$ and $v_{[k,l]}$ with `-`shown stacked on top of each other in-between u and v constitute the local alignment $(\hat{u}, \hat{v})$. The gap-openings in the alignments are the factors `--` and `----` of $\hat{u}$ and $\hat{v}$, respectively.

In the context of DNA sequence alignments, we treat opposite strands of the same DNA molecule as two distinct words over the alphabet Σ^*_{DNA} and assume that all alignments referred to in this discussion are alignments of strands from distinct molecules.

Definition 3.3.2. For an alignment $A = (\hat{u}, \hat{v}) \in \mathcal{L}^-_{\text{DNA}} \times \mathcal{L}^-_{\text{DNA}}$ of a pair of words $(u, v) \in \Sigma^*_{\text{DNA}} \times \Sigma^*_{\text{DNA}}$, define the **matches** $\phi(A)$ of A to be the multiset $\{(\hat{a}_i, \hat{b}_i) : i = 1, \ldots, |\hat{u}|\}$. A match $(a, b) \in \phi(A)$ is called an **identity** if $a = b$, a **gap-extension** if one of a or b is the gap character and a **mismatch** if $a \neq b$ and neither a nor b are the gap character.

Denote with $id(A), m(A)$, and $g(A)$ the numbers of identities, mismatches and gap-extensions, respectively, in $\phi(A)$ and denote with $\Delta(A)$ the number of gap-openings in A.

Definition 3.3.3. Given positive integers id_0, m_0, g_0, Δ_0, the **alignment score** of an alignment A is defined as:

$$s(A) = id_0 \cdot id(A) - m_0 \cdot m(A) - g_0 \cdot g(A) - \Delta_0 \cdot \Delta(A).$$

For the remainder of the chapter we keep id_0, m_0, g_0, and Δ_0 fixed and call these constants the **match reward**, **mismatch penalty**, **gap-extension cost**, and **gap-opening cost**, respectively.

Definition 3.3.4. The **percent identity** pid(A) of A is defined as $100 \cdot \frac{id(A)}{|\phi(A)|}$.

In Figure 3.6, the matches of A are the pairs of symbols of $\hat{u}$ and $\hat{v}$ stacked above each other. Vertical lines between the two symbols of a match indicate identities. In this alignment, $id(A) = 9$, $m(A) = 2$, $g(A) = 6$, and $\Delta(A) = 2$. Thus, $s(A) = id_0 \cdot 9 - m_0 \cdot 2 - g_0 \cdot 6 - \Delta_0 \cdot 2$, and pid$(A) = 100 \cdot \frac{9}{17} \approx 52.9$.

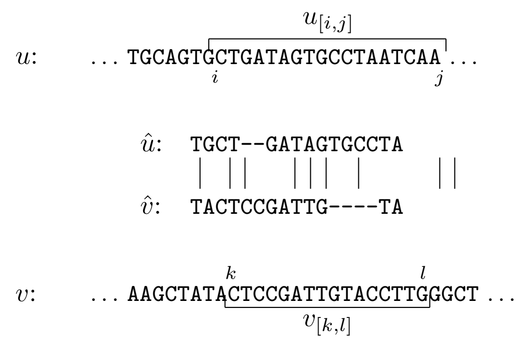

$u_{[i,j]}$
u: ... TGCAGTGCTGATAGTGCCTAATCAA...
i
j
$\hat{u}$: TGCT--GATAGTGCCTA
$\hat{v}$: TACTCCGATTG----TA
k
l
v: ... AAGCTATACTCCGATTGTACCTTGGGCT ...
$v_{[k,l]}$

Figure 3.6: A gapped local alignment $A = (\hat{u}, \hat{v})$ between two sequences u and v. Coordinates i, j and k, l indicate beginning and end of the two aligned factors $u_{[i,j]}$, and $v_{[k,l]}$, respectively. The two gap-openings in this alignment are the two factors $\texttt{-}^2$ of $\hat{u}$ and $\texttt{-}^4$ of $\hat{v}$, (so that $\Delta(A) = 2$). The matches of A consist of the pairs of symbols of $\hat{u}$ and $\hat{v}$ stacked on top of each other ($\phi(A) = \{$(T, T), (G, A), (A, C), (T, T), (-, C), (-, C), (G, G), (A, A), (T, T), (A, T), (G, G), (T, -), (G, -) (A, -), (A, -), (T, T), (A, A)$\}$). Identities are indicated via vertical connecting lines between symbols. $\phi(A)$ contains 9 identities ($id(A) = 9$). The 2 mismatches are the matches (G, A) and (A, T). The 6 gap-extensions ($g(A) = 6$) are those matches which contain -. Thus, the alignment score of A is $s(A) = id_0 \cdot 9 - m_0 \cdot 2 - g_0 \cdot 6 - \Delta_0 \cdot 2$, and its percent identity is pid$(A) = 100 \cdot \frac{9}{17}$.

3.3.2 Pasting a Pair of Ungapped Alignments

High-scoring sequence alignments generally consist of sections of relatively high-scoring ungapped alignments interrupted by comparatively short gaps and regions of low similarity. The higher the score of an alignment, the fewer and shorter are the gapped regions and regions of low similarty and the longer and higher scoring are the ungapped regions. Ungapped local sequence alignment search algorithms can find the ungapped subregions of high-scoring gapped alignments. With large portions of the gapped alignments already determined, the method described here can be used to quickly connect the ungapped alignments and form larger high-scoring gapped alignments. While this method may not produce optimal alignments, the results may still indicate a high level of similarity between the aligned regions, especially when the original ungapped alignments are already highly similar and they are favorably positioned. When determining how to connect two alignments, only one possible ungapped alignment enveloping the two is considered, so that the space of explored gapped sequence alignments is minimized drastically. In fact, the estimates of the numbers of identities, mismatches, gap-openings and gap-extensions for the resulting gapped alignment is derived solely from the relative positions and the underlying sequences can be ignored entirely. Thus, if only the locations, scores and percent identities of the gapped alignments are of interest, alignments can be represented computationally as a combination of coordinates of the aligned factors and counts of the different types of matches. As opposed to having to store and examine arbitrarily long sequences, such a fixed size memory footprint further decreases computational time and space complexity.

Two local alignments may not always be positioned favorably for being combined to a larger alignment.

Definition 3.3.5. Two pairs of factors $(u_1 = u_{[i_1,j_1]}, v_1 = v_{[k_1,l_1]})$ and $(u_2 = u_{[i_2,j_2]}, v_2 = v_{[k_2,l_2]})$ of a pair of words $(u, v) \in \Sigma^*_{\text{DNA}} \times \Sigma^*_{\text{DNA}}$ are said to be in **legal configuration** if:

(i) $i_1 < i_2$ and $j_1 < j_2$.

(ii) $k_1 < k_2$ and $l_1 < l_2$.

Two local alignments between u and v are said to be in **legal configuration** if the pairs of factors of u and v they align are in legal configuration.

Three examples of pairs of legally configured pairs of factors (u_1, v_1) and (u_2, v_2) of two sequences $u, v \in \Sigma^*_{\text{DNA}}$ are shown in Figure 3.7. Non-examples would involve containment of one of the aligned factors in another (i.e. $u_1 \subseteq u_2$, or $v_1 \subseteq v_2$, or vice versa), or reversed order of appearance of the aligned factors in u, or v (i.e. u_1 appearing before u_2 in u, but v_2 appearing before v_1 in v, or vice versa).

The method for combining ungapped local alignments to larger gapped local alignments discussed here only applies to pairs of alignments in legal configuration. For the remainder of this section, let $A_1 = (\hat{u}_1, \hat{v}_1)$ and $A_2 = (\hat{u}_2, \hat{v}_2)$ be two legally configured ungapped local alignments, of two pairs of factors $(u_1, v_1) = (u_{[i_1,j_1]}, v_{[k_1,l_1]})$ and $(u_2, v_2) = (u_{[i_2,j_2]}, v_{[k_2,l_2]})$ of a pair of words $(u, v) \in \Sigma^*_{\text{DNA}} \times \Sigma^*_{\text{DNA}}$. We wish to define a **pasted alignment** $A_1 \circ A_2 = (\hat{u}', \hat{v}')$, as an alignment of the pair of factors $(u' = u_{[i_1,j_2]}, v' = v_{[k_1,l_2]})$ enveloping the original factors. The method by which $\hat{u}'$ and $\hat{v}'$ are determined is easily understood when split into three cases which are illustrated by Figure 3.7.

Case 1: $j_1 < i_2$ and $l_1 < k_2$. Let $\mathcal{D}_u = i_2 - j_1 - 1$ be the distance between u_1 and u_2 and let $\mathcal{D}_v = k_2 - l_1 - 1$ be the distance between v_1 and v_2. Denote with u_0 the factor of u between u_1 and u_2 and denote with v_0

the factor of v between v_1 and v_2:

$$u_0 = u_{[j_1+1, i_2-1]} \qquad\qquad v_0 = v_{[l_1+1, k_2-1]}.$$

Let $\mathcal{G} = |\,\mathcal{D}_u - \mathcal{D}_v\,|$. When $\mathcal{D}_u = |u_0| \leq |v_0| = \mathcal{D}_v$, define $\hat{u}', \hat{v}'$ as the concatenations:

$$\hat{u}' = \hat{u}_1 u_0 (\text{-})^{\mathcal{G}} \hat{u}_2 \qquad\qquad \hat{v}' = \hat{v}_1 v_0 \hat{v}_2.$$

When $\mathcal{D}_v \leq \mathcal{D}_u$, then the roles are simply reversed by placing the gap $(\text{-})^{\mathcal{G}}$ in $\hat{v}'$ between v_0 and $\hat{v}_2$, instead. In the worst case, $\mathcal{G} > 0$ and the characters of the shorter of u_0 and v_0 aligned with the prefix of the longer are all mismatches, so that:

$$s(A_1 \circ A_2) \geq s(A_1) + s(A_2) - m_0 \cdot \min\{\mathcal{D}_u, \mathcal{D}_v\} - g_0 \cdot \mathcal{G} - \Delta_0. \tag{3.1}$$

Case 2: $j_1 \geq i_2$ **and** $l_1 < k_2$**.** The case where $j_1 < i_2$ and $l_1 \geq k_2$ works analogously. Let $\mathcal{O} = j_1 - i_2 + 1$ be the length of the overlap between u_1 and u_2 and let $\mathcal{D} = k_2 - l_1 - 1$ be the the distance between v_1 and v_2. Recall that A_1 and A_2 are assumed to be ungapped, so that $\hat{u}_z = u_z$ and $\hat{v}_z = v_z$ for $z = 1, 2$. Denote with $\text{suff}(\hat{u}_2)$ the suffix of $\hat{u}_2$ obtained by omitting the first $\mathcal{O}$ symbols and denote with v_0 the factor of v between v_1 and v_2:

$$\text{suff}(\hat{u}_2) = u_{[j_1+1, j_2]} \qquad\qquad v_0 = v_{[l_1+1, k_2-1]}.$$

Let $\mathcal{G} = \mathcal{D} + \mathcal{O}$. Define $\hat{u}', \hat{v}'$ as the concatenations:

$$\hat{u}' = \hat{u}_1 (\text{-})^{\mathcal{G}}\, \text{suff}(\hat{u}_2) \qquad\qquad \hat{v}' = \hat{v}_1 v_0 \hat{v}_2.$$

In the worst case, the aligned characters between the prefices of $\hat{u}_2$ and $\hat{v}_2$ of length $\mathcal{O}$ cut out of the alignment are all identities, so that:

$$s(A_1 \circ A_2) \geq s(A_1) + s(A_2) - id_0 \cdot \mathcal{O} - g_0 \cdot \mathcal{G} - \Delta_0. \tag{3.2}$$

Case 3: $j_1 \geq i_2$ **and** $l_1 \geq k_2$**.** Let $\mathcal{O}_u = j_1 - i_2 + 1$ be the length of the overlap between u_1 and u_2 and let $\mathcal{O}_v = l_1 - k_2 + 1$ be the length of the overlap between v_1 and v_2. Recall again that A_1 and A_2 are assumed to be ungapped, so that $\hat{u}_z = u_z$ and $\hat{v}_z = v_z$ for $z = 1, 2$. Denote with $\text{suff}(\hat{u}_2)$ the suffix of $\hat{u}_2$ obtained by omitting the first $\mathcal{O}_u$ symbols and denote with $\text{suff}(\hat{v}_2)$ the suffix of $\hat{v}_2$ obtained by omitting the first $\mathcal{O}_v$

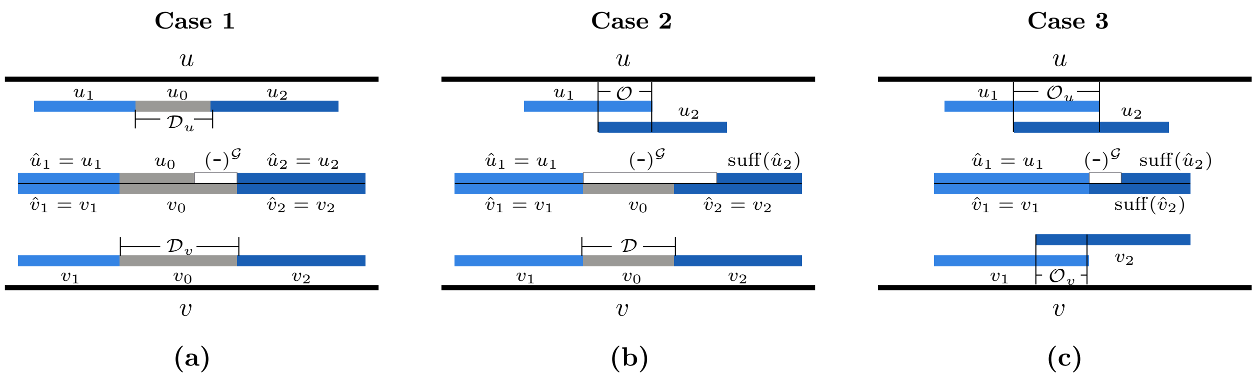


Figure 3.7: Three cases of legally configured local alignments $A_1 = (\hat{u}_1, \hat{v}_1)$, $A_2 = (\hat{u}_2, \hat{v}_2)$ between words $u, v \in \Sigma^*_{\text{DNA}}$. The thick black lines at the top and bottom represent u and v, respectively. The pairs of factors (u_1, v_1) and (u_2, v_2) of u and v aligned by A_1 and A_2 are depicted in light blue and dark blue, respectively, along u and v. The actual sequence alignments $(\hat{u}_1, \hat{v}_1)$, and $(\hat{u}_2, \hat{v}_2)$ are stacked on top of each other in the middle and color coded the same way. In **(a)**, the length of the gap introduced by combining the alignments is $\mathcal{G} = |\,\mathcal{D}_u - \mathcal{D}_v\,|$, in **(b)**, $\mathcal{G} = |\,\mathcal{D} + \mathcal{O}\,|$, and in **(c)**, $\mathcal{G} = |\,\mathcal{O}_u - \mathcal{O}_v\,|$.

symbols:

$$\text{suff}(\hat{u}_2) = u_{[j_1+1, j_2]} \qquad\qquad \text{suff}(\hat{v}_2) = v_{[k_1+1, k_2]}.$$

Let $\mathcal{G} = |\,\mathcal{O}_u - \mathcal{O}_v\,|$. When $\mathcal{O}_u \geq \mathcal{O}_v$, define $\hat{u}', \hat{v}'$ as the concatenations:

$$\hat{u}' = \hat{u}_1 (\text{-})^{\mathcal{G}}\, \text{suff}(\hat{u}_2) \qquad\qquad \hat{v}' = \hat{v}_1\, \text{suff}(\hat{v}_2).$$

When $\mathcal{O}_v \leq \mathcal{O}_u$, then the roles are simply reversed by placing the gap $(\text{-})^{\mathcal{G}}$ in $\hat{v}'$ between v_0 and $\text{suff}(\hat{v}_2)$, instead. In the worst case, $\mathcal{G} > 0$ and the aligned characters between the prefixes of $\hat{u}_2$ and $\hat{v}_2$ cut out of the alignment, are all identities, so that:

$$s(A_1 \circ A_2) \geq s(A_1) + s(A_2) - id_0 \cdot \max\{\mathcal{O}_u, \mathcal{O}_v\} - g_0 \cdot \mathcal{G} - \Delta_0. \tag{3.3}$$

The precision of the bounds in all three cases depends on the quality of the alignments A_1 and A_2 and their relative positions. Unless, u_1 and u_2 as well as v_1 and v_2 are immediately adjacent, these bounds are all equalities if $\text{pid}(A_1) = \text{pid}(A_2) = 1$. When $\mathcal{G} = 0$ in cases 1 and 3, then no gap is introduced and Δ_0 does not need to be subtracted on the right-hand-side of the bounds in 3.1 and 3.3. When u_1 and u_2 as well as v_1 and v_2 are immediately adjacent (i.e. $j_1 + 1 = i_2$ and $l_1 + 1 = k_2$), then the score of the pasted alignment A_0 is precisely the sum of the scores of A_1 and A_2 (assuming neither alignment ends in a gap).

Note that only the overlapping portions of u_1, u_2 or v_1, v_2 needed to be aligned without gaps in the original two alignments for the bounds on the pasted alignment scores to be valid. Thus, pairs of gapped alignments may be pasted using the same method as long as the overlapping prefices/suffices do not contain gaps. In all three cases, the alignment of A_1 was incorporated into $A_1 \circ A_2$ without modifications, whereas in cases 2 and 3, only portions of A_2 made it into $A_1 \circ A_2$. This could have been done the other way around, leaving A_2 intact and modifying A_1 as needed, but would not have impacted the lower bounds on the scores. Additionally, in case 1, the gap could have been placed before u_0 in the alignment without affecting the upper bound on the score. Deciding where to place the gap in case 1, and whether to keep A_1, or A_2 intact in cases 2 and 3 can affect the length of the longes ungapped suffix, or prefix of the pasted alignment.

The three Inequalities 3.1, 3.2, and 3.3 can be expressed as a single lower bound for $s(A_1)A_2$. All three bounds consist of the sum of the scores of A_1 and A_2 minus some penalty for introducing a gap, a penalty for adding possible mismatches (case 1) and a penalty for removing possible identities (cases 2 and 3). The penalties depend on the size of the introduced gap as well as the numbers of potential mismatches added, or identities removed. These numbers can all be immediately obtained from the end coordinates j_1, l_1 of u_1, v_1 and the start coordinates i_2, k_2 of u_2, v_2, respectively.

Definition 3.3.6. The **offset of** (A_1, A_2) **in** u is defined by:

$$o_u(A_1, A_2) = i_2 - j_1 - 1,$$

and similarly, the **offset of** (A_1, A_2) **in** v is defined by:

$$o_v(A_1, A_2) = k_2 - l_1 - 1.$$

In case 1, $o_u(A_1, A_2) = \mathcal{D}_u$ and $o_v(A_1, A_2) = \mathcal{D}_v$, which are the lengths of u_0 and v_0, respectively, in Figure 3.7(a). In case 2, $o_u(A_1, A_2) = -\mathcal{O}$ and $o_v(A_1, A_2) = \mathcal{D}$. Here, $o_u(A_1, A_2)$ is the negation of the length of the overlap between u_1 and u_2 and $o_v(A_1, A_2)$ is the length of v_0 in Figure 3.7(b). In case 3, $o_u(A_1, A_2) = -\mathcal{O}_u$ and $o_v(A_1, A_2) = -\mathcal{O}_v$. The offsets $o_u(A_1, A_2)$ and $o_v(A_1, A_2)$ are depicted in Figure 3.7(c) as the negations of the overlap lengths between u_1, u_2 in u and v_1, v_2 in v, respectively. In general, if the offset of (A_1, A_2) in $w \in \{u, v\}$ is positive, the two aligned factors in w are a distance of $o_w(A_1, A_2)$ apart, and when it is negative, the two factors have an overlap of length $|o_w(A_1, A_2)|$. The exact relationship between the offsets of (A_1, A_2) in u and v and the values $\mathcal{D}_u, \mathcal{D}_v, \mathcal{D}, \mathcal{O}, \mathcal{O}_u, \mathcal{O}_v$, and $\mathcal{G}$ described in the three cases is established in Definition 3.3.7.

Definition 3.3.7. Let $w \in \{u, v\}$. Define the **distance** $\mathcal{D}_w(A_1, A_2)$ **of** (A_1, A_2) **in** w and simply the **distance** $\mathcal{D}(A_1, A_2)$ **of** (A_1, A_2) by:

$$\mathcal{D}_w(A_1, A_2) = \max\{0, o_w(A_1, A_2)\} \qquad \mathcal{D}(A_1, A_2) = \min\{\mathcal{D}_u(A_1, A_2), \mathcal{D}_v(A_1, A_2)\}.$$

Define the **overlap** $\mathcal{O}_w(A_1, A_2)$ **of** (A_1, A_2) **in** w and simply the **overlap** $\mathcal{O}(A_1, A_2)$ **of** (A_1, A_2) by:

$$\mathcal{O}_w(A_1, A_2) = |\min\{0, o_w(A_1, A_2)\}| \qquad \mathcal{O}(A_1, A_2) = \max\{\mathcal{O}_u(A_1, A_2), \mathcal{O}_v(A_1, A_2)\}.$$

Define the **shift** of (A_1, A_2) as the distance between the offsets:

$$\mathcal{G}(A_1, A_2) = |o_u(A_1, A_2) - o_v(A_1, A_2)|.$$

In all three cases, $\mathcal{G}(A_1, A_2) = \mathcal{G}$. In case 1, $\mathcal{D}_u(A_1, A_2) = \mathcal{D}_u$, $\mathcal{D}_v(A_1, A_2) = \mathcal{D}_v$, and $\mathcal{O}_u(A_1, A_2) = \mathcal{O}_v(A_1, A_2) = 0$. In case 2, $\mathcal{D}_u(A_1, A_2) = 0$, $\mathcal{D}_v(A_1, A_2) = \mathcal{D}$, $\mathcal{O}_u(A_1, A_2) = \mathcal{O}$, and $\mathcal{O}_v(A_1, A_2) = 0$. In case 3, $\mathcal{D}_u(A_1, A_2) = \mathcal{D}_v(A_1, A_2) = 0$, $\mathcal{O}_u(A_1, A_2) = \mathcal{O}_u$, and $\mathcal{O}_v(A_1, A_2) = \mathcal{O}_v$. By the constructions outlined in the three cases, we have:

$$\begin{aligned}
id(A_1 \circ A_2) &\geq id(A_1) + id(A_2) - \mathcal{O}(A_1, A_2)\\
m(A_1 \circ A_2) &\leq m(A_1) + m(A_2) + \mathcal{D}(A_1, A_2)\\
g(A_1 \circ A_2) &\leq g(A_1) + g(A_2) + \mathcal{G}(A_1, A_2)\\
\Delta(A_1 \circ A_2) &\leq \begin{cases} \Delta(A_1) + \Delta(A_2), & \text{if } \mathcal{G}(A_1, A_2) = 0\\ \Delta(A_1) + \Delta(A_2) + 1, & \text{otherwise.} \end{cases}
\end{aligned} \tag{3.4}$$

Thus:

$$s(A_1 \circ A_2) \geq s(A_1) + s(A_2) - \mathcal{O}(A_1, A_2) \cdot id_0 - \mathcal{D}(A_1, A_2) \cdot m_0 - \mathcal{G}(A_1, A_2) \cdot g_0 - \mathbb{1}_{\mathbb{Z}_{>0}}(\mathcal{G}(A_1, A_2)) \cdot \Delta_0, \tag{3.5}$$

where $\mathbb{1}_{\mathbb{Z}_{>0}}$ is the characteristic function of the set of positive integers.

Definition 3.3.8. The right-hand side of Inequality 3.5 is called the **pasted alignment score of** $A_1 \circ A_2$, denoted $\widetilde{s}(A_1 \circ A_2)$.

A lower bound on the percent identity of $A_1 \circ A_2$ can be obtained using the bounds in 3.4. Recall that the percent identity of an alignment is the number of percentage of identities in the set of matches of an alignment. At most $\mathcal{O}(A_1, A_2)$ identities from A_1 and A_2 are removed when pasting the alignments. Furthermore, $\mathcal{G}(A_1, A_2)$ many gap-extensions and at most $\mathcal{D}(A_1, A_2)$ many mismatches are added to the multiset of pairs of $A_1 \circ A_2$ in addition to the mismatches and gap-extensions of A_1 and A_2. Thus:

$$\mathrm{pid}(A_1 \circ A_2) \geq \frac{id(A_1) + id(A_2) - \mathcal{O}(A_1, A_2)}{|\phi(A_1)| + |\phi(A_2)| + \mathcal{G}(A_1, A_2) + \mathcal{D}(A_1, A_2)}. \tag{3.6}$$

Definition 3.3.9. The right-hand side of Inequality 3.6 is called the **pasted percent identity of** $A_1 \circ A_2$, denoted $\widetilde{\mathrm{pid}}(A_1 \circ A_2)$.

By pasting alignments, we generally wish to improve the overall alignment quality. With Inequalities 3.5, and 3.6 testable without much computational effort, imposing thresholds for the score and percent identity comes naturally. While the score of a pasted alignment may be better than the average of the two original alignments, the average percent identity of the original alignments always meets or exceeds that of the pasted alignment. Thus, it may be desirable to enforce improvement of the average alignment score and impose a fixed threshold on the percent identity that must be satisfied for two alignments to be pasted. Additionally, in some contexts, such as that of DNA rearrangements in ciliates, it may be desirable to limit the shift of A_1 and A_2. In particular, when two successive MDSs in a macronuclear chromosome only have a short pointer overlap, and appear in the correct order only a small distance apart in the micronucleus, they may be mistakenly pasted and information can be lost. This situation can be expected to occur with high frequency in the genome of *O. trifallax* due to its tendency for short pointers and IESs and mostly unscrambled MDS arrangements [16]. Additionally, limiting the shift during pasting offers direct control over the maximum gap lengths introduced through pasting.

Definition 3.3.10. Let s, p, γ be non-negative. A_1 and A_2 are (s, p, γ)**-pastable** if the resulting pasted alignment $A_1 \circ A_2$ satisfies:

(I) $\widetilde{s}(A_1 \circ A_2) \geq s$,

(II) $\widetilde{\mathrm{pid}}(A_1 \circ A_2) \geq p$, and

(III) $\mathcal{G}(A_1, A_2) \leq \gamma$.

$A_1 \circ A_2$ is said to **improve the average score** if $\widetilde{s}(A_1 \circ A_2) \geq \frac{1}{2}(s(A_1) + s(A_2))$.

3.3.3 Pasting Multiple Gapped Alignments

Gapped alignments may contain more than two ungapped alignments connected by gaps and regions of low similarity. Repeated pasting of alignments using the method for pasting two ungapped alignments introduced in Section 3.3.2 is problematic because gaps may be introduced along the way. Thus, a generalization of the method from Section 3.3.2 that applies to gapped alignments is desirable. Note that for the construction outlined in Section 3.3.2 and the associated Inequalities 3.4, 3.5, and 3.6 the only part of A_1 and A_2 that needed be ungapped was the prefix of A_2 of length $\mathcal{O}(A_1, A_2)$ in cases 2 and 3. This is necessary to ensure that this prefix of the alignment corresponded to the longer of the potentially overlapping regions between u_1 and u_2, or v_1 and v_2. If there were gaps within that prefix, a longer portion of the alignment would need to be removed to avoid double-counting the scores of symbols within the overlapping region that are aligned twice, once in A_1 and once in A_2. Do determine the exact length of the prefix of A_2 that would require removal, closer examination of the shuffles $\hat{u}_2$ and $\hat{v}_2$ would be necessary. Much of the computational efficiency of the pasting method presented in Section 3.3.2, however, stems from the ability to avoid examinations of the actual sequences in the alignment.

A generalized version of the alignment pasting method discussed in Section 3.3.2 might require that only the prefix of A_2 must be ungapped. However, it is possible to achieve the same bounds established in Section 3.3.2, by keeping A_2 intact during pasting and trimming the suffices of length $\mathcal{O}(A_1, A_2)$ off A_1. In that case, the generalized version of alignment pasting might only require that the suffix of A_1 be ungapped, instead. The tradeoff between these two choices is that when trimming the suffix of A_1 the resulting pasted alignment may potentially have a shorter maximal ungapped prefix, and a longer maximal ungapped suffix than the pasted alignment obtained by trimming the prefix of A_2, and vice versa. To distinguish between these two choices for pasting strategies, two generalized versions of alignment pasting are defined. Denote with $\mathrm{pref}_k(w)$ and $\mathrm{suff}_k(w)$ the prefix and suffix, respectively, of a word w of length k. Here, the same notation for A_1, A_2 from Section 3.3.2 is used, except that the alignments now may have gaps.

Definition 3.3.11. A_1 and A_2 are considered to be in **left-legal configuration** if they are in legal configuration and the prefix of length $\mathcal{O}(A_1, A_2) + 1$ of A_2 does not contain any gap-extensions. Suppose A_1 and A_2 are in left-legal configuration. The **left-pasted alignment** $A_1 \circ_L A_2$ is the alignment:

$$(\hat{u}_1 u_0 (\text{-})^{\mathcal{G}_u}\, \mathrm{suff}_{s_u}(\hat{u}_2),\ \ \hat{v}_1 v_0 (\text{-})^{\mathcal{G}_v}\, \mathrm{suff}_{s_v}(\hat{v}_2)),$$

where for $w \in \{u, v\}$, $s_w = |\hat{w}_2| - \mathcal{O}_w(A_1, A_2)$, and w_0 is the factor of w of length $\mathcal{D}_w(A_1, A_2)$ starting immediately after w_1, and where:

$$(\mathcal{G}_u, \mathcal{G}_v) = \begin{cases} (\mathcal{G}(A_1, A_2), 0), & \text{if } o_u(A_1, A_2) \leq o_v(A_1, A_2) \\ (0, \mathcal{G}(A_1, A_2)), & \text{otherwise} \end{cases}$$

Note that an ungapped prefix of length $\mathcal{O}(A_1, A_2) + 1$ is required in Definition 3.3.11 instead of a prefix of length $\mathcal{O}(A_1, A_2)$ to ensure that the gap introduced by pasting, if any, is not adjacent to a gap internal to A_1. With this requirement, the parameter γ in the definition of (s, p, γ)-pastability still represents the maximum length that could possibly be introduced via pasting.

Definition 3.3.12. A_1 and A_2 are considered to be in **right-legal configuration** if they are in legal configuration and the suffix of length $\mathcal{O}(A_1, A_2) + 1$ of A_1 does not contain any gap-extensions. Suppose A_1 and A_2 are in right-legal configuration. The **right-pasted alignment** $A_1 \circ_R A_2$ is the alignment:

$$(\text{pref}_{p_u}(\hat{u}_1)(\text{-})^{\mathcal{G}_u} u_0 \hat{u}_2, \;\; \text{pref}_{p_v}(\hat{v}_1)(\text{-})^{\mathcal{G}_v} v_0 \hat{v}_2),$$

where for $w \in \{u, v\}$, $p_w = |\hat{w}_2| - \mathcal{O}_w(A_1, A_2)$, and w_0 is the factor of w of length $\mathcal{D}_w(A_1, A_2)$ ending immediately before w_2, and where

$$(\mathcal{G}_u, \mathcal{G}_v) = \begin{cases} (\mathcal{G}(A_1, A_2), 0), & \text{if } o_u(A_1, A_2) \leq o_v(A_1, A_2) \\ (0, \mathcal{G}(A_1, A_2)), & \text{otherwise} \end{cases}$$

The maximal ungapped prefices and suffices of $A_1 \circ_L A_2$ and $A_1 \circ_R A_2$ depend on the maximal prefix and suffix of A_1 and A_2 and whether $\mathcal{G}(A_1, A_2) > 0$. When A_1 and A_2 are ungapped, then left-pasting introduces the gap, if any, as far as possible toward the end of the alignment, and right-pasting introduces it as far as possible towards the beginning of the alignments. Thus, the length of the maximal ungapped prefix of $A_1 \circ_L A_2$ is maximized, whereas the length of the maximal ungapped suffix of $A_1 \circ_R A_2$ is maximized. When A_1 and/or A_2 are gapped, however, this behavior cannot be guaranteed.

3.3.4 Alignment Pasting Algorithm

The task of the algorithm described in this section is to improve the average score of a set $\mathcal{A}$ of ungapped local alignments between two sequences u and v by pasting pairs of alignments where appropriate without introducing gaps of length longer than some fixed length γ_0, and without dropping below some fixed score s_{fin}, and percent identity p_{fin}. It should be noted that pasting alignments in different orders may yield different results. For example, given three alignments A_1, A_2, A_3 positioned favorably for pasting, $A_1 \circ_R (A_2 \circ_R A_3)$

may not be the same as $(A_1 \circ_L A_2) \circ_L A_3$. Also, it is possible that (A_1, A_2), and (A_2, A_3) are left-(s, p, γ)-pastable, but $(A_1 \circ_L A_2, A_3)$ are not.

Informally, the strategy chosen here begins with sorting $\mathcal{A}$ in three ways: by alignment score, starting coordinate in u, and ending coordinate in u. Next, alignments are processed one-by-one from the score-sorted list from high score to low. Each alignment is extended to the left and right via pasting with the help of the other two lists. During this process, lower intermediate thresholds $s_{\text{int}}, p_{\text{int}}$ are used to test for left- and right-$(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastability. Whenever the alignment being extended satisfies the higher thresholds $s_{\text{fin}}, p_{\text{fin}}$, and also improves average score, it is marked as stable. Once extension halts, the alignment is reverted to its last stable version. By temporarily allowing satisfaction of lower intermediate bounds, regions of low similarity may be absorbed by surrounding regions of higher similarity. Alignments that were processed this way are marked and candidates for pasting with the alignment currently being processed are chosen only from the unmarked alignments, which all have a lower score. The main reason to paste alignments in this order is that it enables the calculation of a bound on the distance in u from the alignment being extended, past which alignments are not $(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastable.

Lemma 3.3.13. *If $s(A_1) \geq s(A_2)$ and A_1, A_2 are left- or right-(s_0, p_0, γ)-pastable, then:*

$$\mathcal{D}_u(A_1, A_2) \leq \frac{2 \cdot s(A_1) - s_0}{m_0} + \gamma.$$

Proof. By pastability, $|\mathcal{D}_u(A_1, A_2) - \mathcal{D}_v(A_1, A_2)| \leq \mathcal{G}(A_1, A_2) \leq \gamma$. Then:

$$\mathcal{D}(A_1, A_2) = \min\{\mathcal{D}_u(A_1, A_2), \mathcal{D}_v(A_1, A_2)\} \geq \mathcal{D}_u(A_1, A_2) - \gamma. \tag{3.7}$$

Also, $s(A_1) \geq s(A_2)$ implies that:

$$2 \cdot s(A_1) - \mathcal{D}(A_1, A_2) \cdot m_0 \geq \widetilde{s}(A_1 \circ A_2) \geq s_0.$$

Then by Inequality 3.7,

$$\frac{2 \cdot s(A_1) - s_0}{m_0} \geq \mathcal{D}(A_1, A_2) \geq \mathcal{D}_u(A_1, A_2) - \gamma.$$

□

By processing higher scoring alignments first, and removing processed alignments from consideration, Lemma 3.3.13 can be used to limit how far to the left and right of an alignment the algorithm should look to find candidates for pasting. This strategy is particularly efficient if the alignments in $\mathcal{A}$ are spread far apart resulting in only a small fraction of the members of $\mathcal{A}$ to populate the search space at any given time. On the other hand, when the factors aligned by members of $\mathcal{A}$ are crowded near the same position in u, then the bound does not narrow down the search space by much.

To detect the next candidates for left- and right-pasting onto an alignment A, the algorithm uses two procedures named NextLeft, and NextRight. Pseudocode for NextLeft is given in Appendix B. Informally, the procedure takes the list L_{end} of alignments in $\mathcal{A}$ sorted by decreasing end coordinate in u and an integer l and continues incrementing l until either it is out of range (i.e. $l > |L_{\text{end}}|$), or $\mathcal{D}_u(L_{\text{end}}[l], A) > \frac{2 \cdot s(A) - s_{\text{int}}}{m_0}$, or $L_{\text{end}}[l]$ is unmarked, $(L_{\text{end}}[l], A)$ is in left-legal configuration, and left-$(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastable. When the distance threshold was exceeded, the procedure returns $|L_{\text{end}}|+1$; otherwise it returns l. NextRight works analogously, searching the list L_{start} sorted by increasing start coordinate and tests for right-pastability.

Whenever the algorithm finds two candidates A_l and A_r for left- and right-pasting, respectively, onto A, the algorithm decides which to prefer using a simple relation $\succ_A$ defined on the pairs of alignments (A_l, A_r) that are left-, right-$(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastable with A, respectively. This relation is defined by, $A_l \succ_A A_r$ if and only if one of the following is true:

- A_l, A are left-$(s_{\text{fin}}, p_{\text{fin}}, \gamma_0)$-pastable, but A, A_r are not right-$(s_{\text{fin}}, p_{\text{fin}}, \gamma_0)$-pastable, or
- A_l, A and A, A_r are left-, right-$(s_{\text{fin}}, p_{\text{fin}}, \gamma_0)$-pastable, respectively, and $(\widetilde{s}(A_l \circ_L A), \widetilde{\text{pid}}(A_l \circ_L A)) \geq (\widetilde{s}(A \circ_R A_r), \widetilde{\text{pid}}(A \circ_R A_r))$, lexicographically.
- Neither A_l, A nor A, A_r are left-, right-$(s_{\text{fin}}, p_{\text{fin}}, \gamma_0)$-pastable, respectively, and $(\widetilde{s}(A_l \circ_L A), \widetilde{\text{pid}}(A_l \circ_L A)) \geq (\widetilde{s}(A \circ_R A_r), \widetilde{\text{pid}}(A \circ_R A_r))$, lexicographically.

Deciding whether $A_l \succ_A A_r$ requires a constant number of comparisons.

Algorithm 1: Alignment pasting algorithm.

```
$L_{\text{pasted}} \leftarrow [\,]$
$L_s \leftarrow \mathcal{A}$ sorted by decreasing score
$L_{\text{start}} \leftarrow \mathcal{A}$ sorted by increasing starting coordinate in $u$
$L_{\text{end}} \leftarrow \mathcal{A}$ sorted by decreasing end coordinate in $u$
for $A$ in $L_s$ do
    Mark $A$
    $l, r \leftarrow$ position of $A$ in $A_{\text{end}}, A_{\text{start}}$, respectively
    $L_{\text{unmark}} \leftarrow [\,]$
    $l \leftarrow$ NextLeftLegal($L_{\text{end}}$, $l$, $A$)
    $r \leftarrow$ NextRightLegal($L_{\text{start}}$, $r$, $A$)
    $A' \leftarrow A$
    $A_{\text{stable}} \leftarrow$ None
    while $l \leq |L_{\text{end}}|$ or $r \leq |L_{\text{start}}|$ do
        if $r > |L_{\text{start}}|$ or $L_{\text{end}}[l] \succ_A L_{\text{start}}[r]$ then
            $A' \leftarrow L_{\text{end}}[l] \circ_L A'$
            Mark $L_{\text{end}}[l]$
            Append $L_{\text{end}}[l]$ to $L_{\text{unmark}}$
        else
            $A' \leftarrow A' \circ_R L_{\text{start}}[r]$
            Mark $L_{\text{start}}[r]$
            Append $L_{\text{start}}[r]$ to $L_{\text{unmark}}$
        end
        if $\widetilde{s}(A') \geq s_{\text{fin}}$ and $\widetilde{\text{pid}}(A') \geq p_{\text{fin}}$ and $A'$ improves average score then
            $A_{\text{stable}} \leftarrow A'$
            Empty $L_{\text{unmark}}$
        end
        $l \leftarrow$ NextLeftLegal($L_{\text{end}}$, $l$, $A'$)
        $r \leftarrow$ NextRightLegal($L_{\text{start}}$, $r$, $A'$)
    end
    Unmark each member of $L_{\text{unmark}}$
    if $A_{\text{stable}}$ not None then
        Append $A_{\text{stable}}$ to $L_{\text{pasted}}$
    end
end
```

In order to decide whether two gapped alignments are $(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastable, it must be determined if the overlapping regions contain gaps. In general, this task may take time linear in the length of the alignments. However, the length of the longest prefix and suffix without gaps in each alignment can be stored internally, and adjusted whenever an alignment is pasted with another, so that determining pastability can be determined in constant time. At the beginning of the algorithm, all alignments are ungapped. Whenever left-pasting two alignments A_1, A_2, either the the longest ungapped prefix of A_1 becomes the longest ungapped prefix of $A_1 \circ_L A_2$ unless A_1 was ungapped to begin with. When A_1 is ungapped and $\mathcal{G}(A_1, A_2) > 0$, the longest ungapped prefix of $A_1 \circ_L A_2$ is the prefix of length $|A_1| + \mathcal{D}(A_1, A_2)$. When A_1 is ungapped and $\mathcal{G}(A_1, A_2) = 0$, then the length of the longest ungapped prefix depends on where gaps appear in A_2 and on how big $\mathcal{O}(A_1, A_2)$ is. Denote with ℓ_{pref} and ℓ_{suff} the lengths of the largest ungaped prefix and suffix of A_2, respectively. Recall that during left-pasting, the prefix of length $\mathcal{O}(A_1, A_2)$ is removed from A_2. Therefore, if $\mathcal{O}(A_1, A_2) \leq \ell_{\text{pref}}$, then the length of the longest ungapped prefix of $A_1 \circ_L A_2$ is $|A_1| + \mathcal{D}(A_1, A_2) + (\ell_{\text{pref}} - \mathcal{O}(A_1, A_2))$. When $\mathcal{O}(A_1, A_2) > |A_2| - \ell_{\text{suff}}$, then all gaps from A_2 are removed before left-pasting, so that $A_1 \circ_L A_2$ is ungapped. When $\ell_{\text{pref}} < \mathcal{O}(A_1, A_2) \leq |A_2| - \ell_{\text{suff}}$, then the longest ungapped prefix of $A_1 \circ_L A_2$ cannot be determined from ℓ_{pref} and ℓ_{suff}. The simplest solution then is to air on the safe side by simply assigning $|A_1| + \mathcal{D}(A_1, A_2)$ as the longest ungapped prefix length. Determining the lengths of the longest ungapped suffix of $A_1 \circ_L A_2$, as well as the longest ungapped prefix and suffix of two right-pasted alignments works analogously. This method of tracking longest ungapped prefix and suffix lengths is not always perfectly accurate, but ensures that they are not overestimated without causing computational overhead. Additionally, testing whether an alignment improves average score in constant time can be made possible if each alignment is equipped with a counter for the number of alignments that were pasted onto it and a cumulative score of these alignments. With these methods in mind, the computational complexity of Algorithm 1 is

$$O(|\mathcal{A}| \cdot \log(|\mathcal{A}|) \cdot D_{\text{avg}}),$$

where D_{avg} is the average number of alignments within a distance of $\frac{2 \cdot s(A) - s_{\text{int}}}{m_0}$ in u for members of $\mathcal{A}$. Typically, Smith-Waterman derived heuristics have complexity at least linear in the size of the longer of the two sequences. Algorithm 1 ignores the sequences entirely and does not depend on their lengths. In most cases, the number of high-scoring ungapped local alignments between two sequences should be much smaller than the lengths of the sequences. However, Algorithm 1 requires a set of ungapped local alignments as its input.

3.3.5 Implementation of Pasting Algorithm

Algorithm 1 was implemented as a C++ command line application named **PasteAlignments** available at `https://github.com/JasperBraun/PasteAlignments`. The software accepts as input a set of ungapped local alignments between a set of **query sequences** and a set of **subject sequences** in a format that can be obtained from BLAST. The total size (in basepairs) of the subject sequence set is also required in order to compute alignment evalues. PasteAlignments returns a set of alignments obtained from the input alignments by pasting where appropriate according to Algorithm 1. Intermediate and final thresholds can be specified as parameter values. The same scoring parameter values supported by gapped blastn and megablast are also supported by PasteAlignments. By default, pasted alignment sequences are constructed by chopping and combining input alignments. The character `N` is used to align with a gap, or to fill in regions between pasted alignments (which are treated as mismatches) to avoid the necessity of looking up nucleotides at corresponding positions of the underlying sequences. To improve computational time and space efficiency, the software can be run in "blind mode" in which case it does not require alignment sequences in the input and does not construct the pasted alignment sequences. The requirement that pasted alignments must improve average score to be considered stable (line 23 in Alorithm 1), is optional. If average score improvement is not specified, pasted alignments must still satisfy final percent identity and score thresholds, in order to be considered stable. PasteAlignments can be instructed to output some descriptive statistics for each pair of query and subject sequences, as well as a summary for the entire output data set.

The software was tested on the set of subject sequences comprising the micronuclear genome and the set of query sequences comprising the macronuclear genome of *O. trifallax*. The micronuclear genome was obtained from mds_ies_db (data originally published in [16]) and the macronuclear genome from GenBank (originally published in [28]. Three sets of scoring parameters $(id_0, m_0, \Delta_0, g_0) \in \{(1, 2, 0, 2.5), (1, 5, 0, 5.5), (2, 7, 4, 2)\}$, seven values for $\gamma_0 \in \{2, 4, 6, 8, 10, 20, 40\}$, and four values for the xdrop parameter of the final gapped alignment step of gapped blast (40, 60, 80, and 100) were chosen. For each set of scoring parameters, nucleotide blast (BLAST+ 2.2.31 [45]) was run with the ungapped and megablast flags. Using the ungapped alignment output, the PasteAlignments software was run with each of the 7 choices for γ_0. In all runs PasteAlignments was instructed to enforce that pasted alignments improve average score. For each combination of scoring parameters and xdrop parameters, BLAST was run with the megablast flag only gapped alignments. All runs were done 5 times and average runtime and its standard deviation were measured. These steps were repeated, instructing BLAST and PasteAlignments to omit the actual alignment sequences. The results are summarized in Tables B.1, B.2, and B.3 in Appendix B, which contain statistics for all test runs with scoring parameter values $(id_0, m_0, \Delta_0, g_0) = (1, 2, 0, 2.5), (1, 5, 0, 5.5)$, and $(2, 7, 4, 2)$, respectively.

As expected, larger values for γ result in fewer and on average longer alignments and a larger number of (stable) pastings that were performed. For all 3 scoring parameter sets, the highest average percent identity is achieved by the ungapped alignments. The average percent identities of gapped alignments returned by BLAST and PasteAlignments are about the same, where alignments produced by PasteAlignments have slightly larger percent identity, except for gapped blast runs with xdrop parameter value at 40 compared to PasteAlignments with $\gamma = 40$. Average alignment scores increase, as leniency for the introduction of gaps increases. PasteAlignments increases the average score of the set of ungapped input alignments by 5-15% and gapped BLAST increases the average score by 7-32%. Gapped BLAST always achieves higher average score values than PasteAlignments except when run with the xdrop parameter value at 40 and compared to PasteAlignments with $\gamma = 40$. Note that PasteAlignments only returns an underestimate of the actual optimal score between aligned regions.

To compare the runtime of the gapped alignment step of BLAST, the execution time of PasteAlignment was compare with the difference of the execution times of ungapped and gapped BLAST. While this difference may not accurately reflect the time the gapped alignment step in BLAST takes, it does provide a point of comparison for the two alternative workflows - gapped BLAST, or ungapped BLAST together with PasteAlignments - for generating gapped alignments from scratch. For scoring parameter values $(id_0, m_0, \Delta_0, g_0) = (1, 2, 0, 2.5)$, PasteAlignments ran on average 31% faster than the gapped alignment step of BLAST. When run in blind mode, PasteAlignments ran 50% faster. For scoring parameter values $(2, 7, 4, 2)$, PasteAlignments ran 94% faster (95% in blind mode). For scoring parameter values $(1, 5, 0, 5.5)$, gapped BLAST outperformed ungapped BLAST. On average, it was 22.6 seconds faster to run gapped BLAST than ungapped BLAST followed by PasteAlignments. When running in blind mode, gapped BLAST was 17.6 seconds faster. The average performance of ungapped BLAST with the three different scoring parameter value sets falls between 415 and 477 seconds (387 and 451 without sequences), and PasteAlignments falls on average between 20 and 26 seconds (16 and 22 in blind mode). Gapped BLAST however, was much less consistent and varied between 418 and 1194 seconds (393 and 1171 without sequences), which suggests that the combination of ungapped BLAST and PasteAlignments has a more reliable runtime than gapped BLAST.

PasteAlignments allows for direct control over the size of the gaps that may be introduced via the parameter γ. Gapped BLAST allows for control over the size of the gaps less directly by use of the xdrop parameter. The xdrop parameter limits how far below the bitscore associated with a path in the dynamic programming matrix may drop below the highest bitscore achieved along the path, so far. When the bitscore drops too far, the current path is not explored further. Thus, the longest gap in gapped alignments returned by BLAST must be short enough, so that the corresponding gap penalty in bits does not exceed the xdrop parameter value. For reference, the maximum gap lengths for the different runs of gapped BLAST is shown in Table 3.3. Since the more lenient scoring systems $(1, 2, 0, 2.5)$ and $(2, 7, 4, 2)$ allowed for much longer gaps than the strict scoring system $(1, 5, 0, 5.5)$, the space of possible alignments considered during the gapped alignment step is much larger, and thus, the algorithm takes longer.

Table 3.3: The maximum gap length in alignments obtained via gapped BLAST for different scoring systems and final gapped xdrop parameter values.

	$(id_0, m_0, \Delta_0, g_0)$		
xdrop	$(1, 2, 0, 2.5)$	$(1, 5, 0, 5.5)$	$(2, 7, 4, 2)$
40	19	5	30
60	21	7	41
80	30	9	54
100	43	11	70

Finally, it should be emphasized that the outcome of this analysis varied greatly based on the different scoring parameter values. Since only three scoring systems were tested, conclusions must be drawn with caution. However, if these three scoring systems are any indication, then gapped BLAST seems to be better suited for stricter scoring system, such as $(1, 5, 0, 5.5)$, which enforces higher sequence similarity. For more lenient scoring systems, such as $(1, 2, 0, 2.5)$ and $(2, 7, 4, 2)$, gapped BLAST still finds more optimal alignments, but using PasteAlignments is faster with a subjectively small decrease in quality.

Chapter 4

Concluding Remarks

4.1 Modelling interactions between multiple genes

In [11, 16], it was pointed out that MDSs of different genes often interleave, or overlap. In [6], particularly interesting cases of interleaving genes, involving several levels of nested genes in the precursor were investigated. The theoretical models in Chapter 2 do not address the interactions of multiple genes, but rather look at individual genes in isolation. A pairing only describes the relative locations of MDSs in the precursor. Labellings, on the other hand, take on the responsibility of indicating where in the macronuclear genomes the MDSs belong. It seems natural to extend the definition of labellings to add the capability of modelling rearrangements of MDSs of multiple mac sequences. This can be done, for example, by partitioning the symbols in labels do indicate MDSs in different mac sequences.

4.2 Ambiguities in the data

The sequence data of the mac and mic genomes of *O. trifallax* contains many ambiguities, such as multiple regions in the mic matching the same region in the mac and vice versa. Extracting an unambiguous rearrangement map for each macronuclear chromosome requires preference of some alignments over others. No method of making all decisions in a meaningful way exists. Therefore, theoretical models need to be applicable to rearrangement maps containing these ambiguities. Chapter 2 introduces a model that is applicable to rearrangements where there is no clear 1-1 correspondence between the segments in the precursor and product. In Chapter 3, an algorithm extracting rearrangement maps from the data is developed and implemented. Finally, it was shown that the amount of data that the new model applies to is over 7% larger than what conventional models could be applied to.

However, in Sections 3.2.7, cases where some alignments contain multiple other alignments in the mac are investigated. These cases represent additional ambiguities that are not anticipated by the theoretical models and algorithms introduced in this work. Deciding between one long alignment or several shorter alignments represents the decision between different fragmentations of a mac sequence. While the general observation in Section 3.2.7 is that only an insignificant portion of the data exhibits this behavior, these alignments may suggest multiple ways to partition the product sequence into MDSs. Furthermore, the only observations made here are based on the genomes of *O. trifallax* and the algorithm and software is meant to be adaptable to genomes of other species where this behaviour could be more common. It would be useful to adjust the algorithms and models presented in this work to respect the existence of multiple different possible fragmentations of mac sequences in a meaningful way. All in all, the various ambiguities in the data represent possible choices of how unambiguous subsets of alignments can be obtained from the data.

While the strategy presented in this work is to show and analyze all of the unambiguous subarrangements, a method for rating the different possible choices for downstream analyses is desirable. Such methods may be of quantitative nature, or probabilistic. Quantitative ways of rating different unambiguous subsets, may naturally include an overall sequence alignment similarity assessment, but could also take other statistics into consideration. For example, a measure for the mutual distance between matches in a set could be an indicator of likelihood of the biological mechanism behind the rearrangement choosing one subset over another. When matches are in close proximity, it is reasonable to assume that the rearrangement takes less effort. Many quantitative measures could play a role in the rating and comparison of unambiguous subsets, so one challenge with this approach likely involves how such measures can be combined appropriately. A natural probabilistic rating of an unambiguous subset of an arrangement is its chance to have appeared at random in the data. Such method requires an appropriate probabilistic model, and can benefit from a high-confidence or computer generated data set.

4.3 Coverage of macronuclear chromosomes by multiple mic sequences

In [16], the authors point out that about 10% of the 1- to 2-telomeric mac sequences are not covered 90% or more by MDSs from a single mic locus. This may be a result from the incomplete assembly of the mic chromosomes, or the spread of MDSs of a single mac chromosome across multiple mic chromosomes. Using the MDS arrangements produced by applying SDRAP to the most resent macronuclear assembly from [28] and the mic assembly from [16] of *O. trifallax* with the parameters presented in Table 3.1 with $(p_{\text{pre}}, p_{\text{add}}) = (95.0, 90.0)$, coverage of mac chromosomes by matches from multiple mic sequences was compared to the coverage by matches from single mic sequences. Only the regions on mac contigs between telomeres were considered and coverage was obtained using bedtools v2.29.0 [40]. The results are pictured in Figure 4.1.

By combining MDSs from different mic loci, more mac chromosomes can be included in downstream analyses. Figure 4.1 suggests that over 12% of the mac contigs which would have been filtered out by analysis requiring 90% coverage or higher by MDSs from a single mic locus (as was done in [16]), are covered by at least 90% by a combination of matches from multiple mic loci. Furthermore, this observation suggests close proximity of the chromosomes which recombine from MDSs from different mic contigs, or the common origin of the contigs from the same molecule. The information could potentially be used to further assemble the mic genome, or draw conclusions about the spatial arrangement of the micronuclear chromosomes.

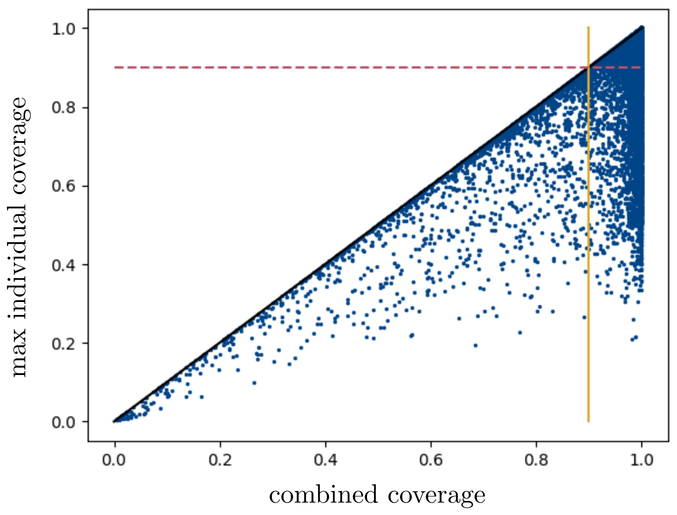


Figure 4.1: The coverages of macronuclear contigs of *O. trifallax* by the combined collection of matches from all micronuclear contigs (x-axis) plotted against the maximum coverage by matches from single micronuclear contigs (y-axis). Each point corresponds to a mac contig. Points on the diagonal achieve their highest coverage by the matches from a single mic locus. Points below the diagonal achieve higher coverage through the combined collection of matches from all mic loci. Points on or above the dashed red line ($y = 0.9$) indicate mac contigs which are covered by the matches of a single mic locus by at least 90%. Points on or to the right of the yellow line ($x = 0.9$) indicate mac contigs which achieve at least 90% coverage by the matches from all mic loci combined. Out of a total of 36,721 mac contigs, 4,599 (12.5%) are covered at least 90% by matches from all mic contigs combined, but less than 90% by matches from single mic loci (points in the bottom right rectangle).

4.4 DNA rearrangements in other settings

The extreme magnitude of rearrangements that take place during conjugation of ciliates such as *O. trifallax* make them a model organisms for the general study of DNA rearrangements that appear in various settings. The software tool SDRAP offers a way to apply the same algorithms to a variety of ciliates and compare the DNA rearrangements they undergo. By extracting rearrangement maps in a uniform way, a comparison does not need to account for differences between the extraction of data from the genomes. Furthermore, the numerous parameters provide a high level of control over the computation increasing the spectrum of situations it can be applied to. Ultimately, SDRAP can be applied to any situation that involves the comparison of two genomes, a precursor and a product, and the organization of data relevant to the mapping of highly similar regions between the two. Such contexts appear in many areas of study, such as VDJ combinations in immunology, or chromosomal rearrangements in the study of cancer.

4.5 Pasting as a gapped alignment algorithm

The alignment pasting algorithm and implementation presented in 3.3 provide a fast method to obtain gapped alignments from a set of ungapped alignments via the introduction of gaps. The resulting set of gapped alignments will be less fragmented and on average have a higher alignment score. The test data described in Section 3.3.5 suggests that the gapped alignment step by BLAST results in better alignment scores and a less fragmented set of alignments, but at the cost of less predictable speed. In the extreme case, the gapped alignment step performed by BLAST takes up to 20 times longer than the pasting algorithm. Therefore, in general, the pasting alignment algorithm currently provides an alternative to the gapped alignment step of the popular sequence alignment software BLAST that trades predictable fast run time for relative small sacrifices in optimality of the results.

However, the algorithm leaves room for improvements. For example, when constructing pasted alignment sequences (not running in blind mode), the computation incurs a cost linear in the distance between alignments. Due to its ignorance towards the underlying sequences, the constructed gapped alignments often contain placeholder `N-N` matches which are treated as mismatches. Since the algorithm takes the time to iterate through possibly long runs of `N-N` matches to include them in constructed alignments, it may as well test the underlying sequences for potential matches. The overall alignment scores obtained by the algorithm will improve in this way without increasing asymptotic time complexity.

Notations

Symbol	Description	Page
Z_N	$\{1, 2, \ldots, N\}$	9
$\vec{a}_i$	i^{th} coordinate of $\vec{a}$	10
$\lvert\vec{A}\rvert$	$(\lvert\vec{A}_1\rvert, \ldots, \lvert\vec{A}_{\lvert\vec{A}\rvert}\rvert)$	10
$G(R)$	The graph with vertex-set domain of relation R and edge-set R.	10
$N_R(a)$	The neighborhood of a in $G(R)$	10
$\Pi(N)$	The set of all pairings of Z_N	10
$\Pi_{\text{uni}}(N)$	The set of all unidirectional pairings of Z_N	12
$\vec{L}(\pi), \vec{R}(\pi)$	The unique tuples $\vec{L}(\pi) = (L_1, \ldots, L_{\nu(\pi)})$ and $\vec{R}(\pi) = (R_1, \ldots R_{\nu(\pi)})$, where $L_1 < R_1 < L_2 < R_2 < \cdots < L_{\nu(\pi)} < R_{\nu(\pi)}$, ($A < B$ for any $A, B \subseteq \mathbb{Z}$, whenever $a < b$, for all $a \in A, b \in B$)	12
$\nu(\pi)$	The length of the tuples $\vec{L}(\pi)$ and $\vec{R}(\pi)$	12
$\vec{l}(\pi)$	$\lvert\vec{L}(\pi)\rvert$	12
$\vec{r}(\pi)$	$\lvert\vec{R}(\pi)\rvert$	12
$M(\pi)$	Biadjacency matrix of π	13
$\mathcal{T}(\vec{l}, \vec{r})$	The set of all $\nu \times \nu$ block upper triangular binary block matrices where ν is the number of coordinates of $\vec{l}$ and $\vec{r}$ and where each $(p, q)^{\text{th}}$ block has is a $\vec{l}_p \times \vec{r}_q$ matrix	14
$\mathcal{T}^*(\vec{l}, \vec{r})$	The subset of $\mathcal{T}(\vec{l}, \vec{r})$ of all members with no zero rows and columns	14
$\pm A$	$\bigcup_{a \in A}\{a, -a\}$	19
$\Lambda_K(P)$	The set of K-labellings of P	19
$\text{proj}(A)$	$\{\lvert a\rvert : a \in A\}$	19
$\text{proj}(\mathcal{F})$	$\{\text{proj}(A) : A \in \mathcal{F}\}$	19
$\text{supp}(\mathcal{F})$	$\bigcup_{A \in \mathcal{F}} \text{proj}(A)$	19

Symbol	Description	Page
$\Lambda_{K,S}(P)$	$\{\lambda \in \Lambda_K(P) : \mathrm{supp}(\lambda(P)) \subseteq S\}$	20
$\hat{\Lambda}_{K,S}(P)$	$\{\lambda \in \Lambda_K(P) : \mathrm{supp}(\lambda(P)) = S\}$	20
$\hat{\Lambda}_{K}(P)$	$\{\lambda \in \Lambda_K(P) : \mathrm{supp}(\lambda(P)) = Z_K\}$	20
$-A$	$\{-a : a \in A\}$	21
$\Lambda_{K,\mathrm{coh}}(P)$	The set of coherent K-labellings of P	21
$\Lambda_{K,S,\mathrm{coh}}(P)$	$\Lambda_{K,\mathrm{coh}}(P) \cap \Lambda_{K,S}(P)$	21
$\hat{\Lambda}_{K,S,\mathrm{coh}}(P)$	$\Lambda_{K,\mathrm{coh}}(P) \cap \hat{\Lambda}_{K,S}(P)$	21
$\hat{\Lambda}_{K,\mathrm{coh}}(P)$	$\Lambda_{K,\mathrm{coh}}(P) \cap \hat{\Lambda}_{K}(P)$	21
$\equiv_\lambda^{\mathrm{proj}}$	$x \equiv_\lambda^{\mathrm{proj}} y \iff \mathrm{proj}(x) = \mathrm{proj}(y)$	22
$\mathrm{pal}(S)$	$\{\pm S\}$	22
$\mathrm{apal}(S)$	$\{A : \mathrm{proj}(A) = S; A \cap -A = \emptyset\}$	22
$\mathrm{coh}(S)$	$\mathrm{pal}(S) \cup \mathrm{apal}(S)$	22
$\hat{\Lambda}_{K,\mathrm{coh}}(\mathcal{P}, \mathcal{S}, f)$	$\Big\{\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(\cup_{X\in\mathcal{P}} X) : \Big(\cup_{X\in\mathcal{P}} X \big/ \equiv_\lambda^{\mathrm{proj}}\Big) = \mathcal{P}$, and $\mathrm{proj}(\lambda(\cup_{X\in\mathcal{P}} X)) = \mathcal{S}$, and $\mathrm{proj} \circ \lambda = f\Big\}$	24
P_λ^{pal}	$\{x \in P : \lambda(x) = \pm\lambda(x)\}$	25
$P_\lambda^{\mathrm{apal}}$	$\{x \in P : \lambda(x) \cap -\lambda(x) = \emptyset\}$	25
$\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{pal}}(P)$	$\{\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(P) : P = P_\lambda^{\mathrm{pal}}\}$	25
$\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{apal}}(P)$	$\{\lambda \in \hat{\Lambda}_{K,\mathrm{coh}}(P) : P = P_\lambda^{\mathrm{apal}}\}$	25
$\ell_{\mathrm{coh}}^{\mathrm{pal}}(K,p)$	$\lvert\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{pal}}(P)\rvert$, where $\lvert P\rvert = p$	26
$\ell_{\mathrm{coh}}^{\mathrm{apal}}(K,p)$	$\lvert\hat{\Lambda}_{K,\mathrm{coh}}^{\mathrm{apal}}(P)\rvert$, where $\lvert P\rvert = p$	26
$\ell_{\mathrm{coh}}(K,p)$	$\lvert\hat{\Lambda}_{K,\mathrm{coh}}(P)\rvert$, where $\lvert P\rvert = p$	26
$\sqsupseteq$	$\{i_1, j_1\} \sqsupseteq \{i_2, j_2\} \iff i_1 \le i_2 < j_2 \le j_1$	27
$\bar{x}$	$\{y \in P : x \sqsupseteq y\}$	27
$\mathrm{dom}(X)$	$\bigcup_{x\in X} x$	27
$G_P[x]$	The subgraph of $G(P)$ induced by $\mathrm{dom}(\bar{x})$	28
$\sigma_{A,B}^{\mathrm{rev}}$	Order isomorphism between $(A, >)$ and $(B, <)$	28
$\Lambda_{K,\mathrm{con}}(P)$	The set of consistent K-labellings of P	29
$\Lambda_{K,S,\mathrm{con}}(P)$	$\Lambda_{K,\mathrm{con}}(P) \cap \Lambda_{K,S}(P)$	29
$\hat{\Lambda}_{K,S,\mathrm{con}}(P)$	$\Lambda_{K,\mathrm{con}}(P) \cap \hat{\Lambda}_{K,S}(P)$	29
$\hat{\Lambda}_{K,\mathrm{con}}(P)$	$\Lambda_{K,\mathrm{con}}(P) \cap \hat{\Lambda}_{K}(P)$	29
$h_P(x)$	$\max\{h : \exists$ distinct $y_1, \dots, y_h \in P \setminus \{x\}$ where $x \sqsupseteq y_1 \sqsupseteq \cdots \sqsupseteq y_h\}$	30

Symbol	Description	Page
$h(P)$	$\max\{h_P(x) : x \in P\}$	30
$L_h(P)$	$\{x \in P : h_P(x) = h\}$	30
$\hat{\Lambda}^{\text{rev}}_{K,S,\text{con}}(P)$	$\{\lambda \in \hat{\Lambda}_{K,S,\text{con}}(P) : \lambda$ is l-reverse-complementary$\}$	32
$\hat{\Lambda}^{\text{nrev}}_{K,S,\text{con}}(P)$	$\{\lambda \in \hat{\Lambda}_{K,S,\text{con}}(P) : \lambda$ is non-l-reverse-complementary$\}$	32
$\hat{\Lambda}^{\text{rev}}_{K,\text{con}}(P)$	$\{\lambda \in \hat{\Lambda}_{K,\text{con}}(P) : \lambda$ is l-reverse-complementary$\}$	32
F_n	The unique non-crossing pairing of Z_{2n} containing no two distinct pairs x, y, where $x \sqsupseteq y$	33
$\lambda_1 \cup \lambda_2(x)$	$\begin{cases} \lambda_1(x) & x \in P_1 \\ \lambda_2(x) & x \in P_2. \end{cases}$, where $\lambda_1 \in \Lambda_K(P_1), \lambda_2 \in \Lambda_K(P_2)$, and $P_1 \cap P_2 = \emptyset$	36
$\Lambda_1 \circ \Lambda_2$	$\{\lambda_1 \cup \lambda_2 : \lambda_1 \in \Lambda_1, \lambda_2 \in \Lambda_2$	36
$\mu(P)$	$\min\{K : \Lambda_{K,\text{con}}(P) \neq \emptyset$	37
$SN(n)$	$\big\{\{1, 2n\}, \{2, 2n-1\}, \ldots, \{n, n+1\}\big\}$	39
$\kappa_K(P_1, \ldots, P_r)$	$\Big\{(S_1, \ldots, S_r) \in \left\{ {Z_K \atop r} \right\} : \lvert S_i \rvert \geq \mu(P_i), \text{ for } i \in Z_r\Big\}$	41

Bibliography

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, October 1990.

[2] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389 – 3402, September 1997.

[3] Angela Angeleska, Nataša Jonoska, and Masahico Saito. DNA recombination through assembly graphs. *Discrete Applied Mathematics*, 157(14):3020 – 3037, July 2009.

[4] Angela Angeleska, Nataša Jonoska, Masahico Saito, and Laura F Landweber. RNA-guided DNA assembly. *Journal of Theoretical Biology*, 248(4):706 – 720, October 2007.

[5] Ryan Arredondo. Reductions on double occurrence words. *arXiv preprint arXiv:1311.3543*, 2013.

[6] Jasper Braun, Lukas Nabergall, Rafik Neme, Laura F Landweber, Masahico Saito, and Nataă Jonoska. Russian doll genes and complex chromosome rearrangements in *Oxytricha trifallax*. *G3: Genes, Genomes, Genetics*, 8(5):1669 – 1674, May 2018.

[7] Robert Brijder, Hendrik Jan Hoogeboom, Nataša Jonoska, and Masahico Saito. Graphs associated with dna rearrangements and their polynomials. In *Algebraic and Combinatorial Computational Biology*, pages 61 – 87. Elsevier, 2019.

[8] Dorothy Buck, Egor Dolzhenko, Nataša Jonoska, Masahico Saito, and Karin Valencia. Genus ranges of 4-regular rigid vertex graphs. *The Electronic Journal of Combinatorics*, 22(3):1 – 26, September 2015.

[9] Jonathan Burns, Egor Dolzhenko, Nataša Jonoska, Tilahun Muche, and Masahico Saito. Four-regular graphs with rigid vertices associated to DNA recombination. *Discrete Applied Mathematics*, 161(10):1378 – 1394, July 2013.

[10] Jonathan Burns, Nataša Jonoska, and Masahico Saito. Genus ranges of chord diagrams. *Journal of knot theory and its ramifications*, 24(04):1550022, May 2015.

[11] Jonathan Burns, Denys Kukushkin, Xiao Chen, Laura F Landweber, Masahico Saito, and Nataša Jonoska. Recurring patterns among scrambled genes in the encrypted genome of the ciliate *Oxytricha trifallax*. *Journal of Theoretical Biology*, 410:171–180, December 2016.

[12] Jonathan Burns, Denys Kukushkin, Kelsi Lindblad, Xiao Chen, Nataša Jonoska, and Laura F Landweber. <mds_ies_db>: a database of ciliate genome rearrangements. *Nucleic Acids Research*, 44(D1):D703 – D709, January 2016.

[13] Jonathan Burns and Tilahun Muche. Counting irreducible double occurrence words. *arXiv preprint arXiv:1105.2926*, 2011.

[14] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. Blast+: architecture and applications. *BMC Bioinformatics*, 10, December 2009.

[15] Andre RO Cavalcanti and Laura F Landweber. Insights into a biological computer: detangling scrambled genes in ciliates. In *Nanotechnology: Science and Computation*, pages 349 – 359. Springer, 2006.

[16] Xiao Chen, John R Bracht, Aaron D Goldman, Egor Dolzhenko, Derek M Clay, Estienne C Swart, David H Perlman, Thomas G Doak, Andrew Stuart, Chris T Amemiya, Robert P Sebra, and Laura F Landweber. The architecture of a scrambled genome reveals massive levels of genomic rearrangement during development. *Cell*, 158(5):1187 – 1198, August 2014.

[17] Daniel A Cruz, Margherita Maria Ferrari, Nataša Jonoska, Lukas Nabergall, and Masahico Saito. Insertions yielding equivalent double occurrence words. *Fundamenta Informaticae*, 171(1-4):113 – 132, October 2020.

[18] Mark Daley and Ian McQuillan. Template-guided dna recombination. *Theoretical Computer Science*, 330(2):237 – 250, 2005.

[19] Andrzej Ehrenfeucht, Tero Harju, Ion Petre, David M Prescott, and Grzegorz Rozenberg. *Computation in living cells: gene assembly in ciliates.* Springer Science & Business Media, 2003.

[20] Wenwen Fang, Xing Wang, John R Bracht, Mariusz Nowacki, and Laura F Landweber. Piwi-interacting rnas protect dna against loss during oxytricha genome rearrangement. *Cell*, 151(6):1243 – 1255, December 2012.

[21] Olivier Garnier, Vincent Serrano, Sandra Duharcourt, and Eric Meyer. Rna-mediated programming of developmental genome rearrangements in paramecium tetraurelia. *Molecular and cellular biology*, 24(17):7370 – 7379, 2004.

[22] Tom Head. Formal language theory and dna: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of mathematical biology*, 49(6):737 – 759, November 1987.

[23] Nataša Jonoska and Masahico Saito, editors. *Discrete and topological models in molecular biology.* Springer, 2013.

[24] Nataša Jonoska, Lukas Nabergall, and Masahico Saito. Patterns and distances in words related to DNA rearrangement. *Fundamenta Informaticae*, 154:225 – 238, August 2017.

[25] Stefan Juranek, Sina Rupprecht, Jan Postberg, and Hans J Lipps. snrna and heterochromatin formation are involved in dna excision during macronuclear development in stichotrichous ciliates. *Eukaryotic cell*, 4(11):1934 – 1941, 2005.

[26] Nazifa Azam Khan and Ian McQuillan. Descrambling order analysis in ciliates. In *International Conference on Unconventional Computation and Natural Computation*, pages 206 – 219. Springer, 2017.

[27] Jaspreet S Khurana, Derek M Clay, Sandrine Moreira, Xing Wang, and Laura F Landweber. Small rna-mediated regulation of dna dosage in the ciliate oxytricha. *RNA*, 24(1):18 – 29, 2018.

[28] Kelsi A Lindblad, Jananan S Pathmanathan, Sandrine Moreira, John R Bracht, Robert P Sebra, Elizabeth R Hutton, and Laura F Landweber. Capture of complete ciliate chromosomes in single sequencing reads reveals widespread chromosome isoforms. *BMC Genomics*, 20(1037), December 2019.

[29] David J Lipman and William R Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435 – 1441, March 1985.

[30] Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In Torben Hagerup and Jyrki Katajainen, editors, *Algorithm Theory - SWAT 2004*, Lecture Notes in Computer Science, vol 3111, pages 260 – 272. Springer, Berlin, Heidelberg, July 2004.

[31] Kazufumi Mochizuki, Noah A Fine, Toshitaka Fujisawa, and Martin A Gorovsky. Analysis of a piwi-related gene implicates small rnas in genome rearrangement in tetrahymena. *Cell*, 110(6):689 – 699, 2002.

[32] John W Moon and Leo Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23 – 28, March 1965.

[33] Aleksandr Morgulis, George Coulouris, Yan Raytselis, Thomas L Madden, Richa Agarwala, and Alejandro A Schäffer. Database indexing for production megablast searches. *Bioinformatics*, 24(16):1757 – 1764, June 2008.

[34] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, March 1970.

[35] William R Pearson and David J Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444 – 2448, April 1988.

[36] Dennis Pixton. Regularity of splicing languages. *Discrete Applied Mathematics*, 69(1-2):101 – 124, August 1996.

[37] David M Prescott, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. Molecular operations for dna processing in hypotrichous ciliates. *European Journal of Protistology*, 37(3):241 – 260, 2001.

[38] David M Prescott and Arthur F Greslin. Scrambled actin i gene in the micronucleus of oxytricha nova. *Developmental Genetics*, 13(1):66 – 74, 1992.

[39] Gheorghe Păun. On the splicing operation. *Discrete Applied Mathematics*, 70(1):57 – 79, September 1996.

[40] Aaron R Quinlan and Ira M Hall. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841 – 842, January 2010.

[41] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, March 1981.

[42] Estienne C Swart, John R Bracht, Vincent Magrini, Patrick Minx, Xiao Chen, Yi Zhou, Jaspreet S Khurana, Aaron D Goldman, Mariusz Nowacki, Klaas Schotanus, Seolkyoung Jung, Robert S Fulton, Amy Ly, Sean McGrath, Kevin Haub, Jessica L Wiggins, Donna Storton, John C Matese, Lance Parsons, Wei-Jen Chang, Michael S Bowen, Nicholas A Stover, Thomas A Jones, Sean R Eddy, Glenn A Herrick, Thomas G Doak, Richard K Wilson, Elaine R Mardis, and Laura F Landweber. The *oxytricha trifallax* macronuclear genome: A complex eukaryotic genome with 16,000 tiny chromosomes. *PLOS Biology*, 11(1):1 – 29, January 2013.

[43] S Lorraine Tausta and Lawrence A Klobutcher. Detection of circular forms of eliminated dna during macronuclear development in e. crassus. *Cell*, 59(6):1019 – 1026, 1989.

[44] Vedia T Yerlici, Michael W Lu, Carla R Hoge, Richard V Miller, Rafik Neme, Jaspreet S Khurana, John R Bracht, and Laura F Landweber. Programmed genome rearrangements in oxytricha produce transcriptionally active extrachromosomal circular dna. *Nucleic acids research*, 47(18):9741 – 9760, 2019.

[45] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A greedy algorithm for aligning dna sequences. *Journal of Computational Biology*, 7(1-2):203 – 214, 2000.

Appendix A

SDRAP Test Data

Table A.1: Values used for parameters during test runs of SDRAP. For each parameter a “low”, “mid”, and “high” value was chosen. An initial run with all parameters at their “mid” value was conducted. Only the parameter ℓ_{pre} was set to its “low” value during the initial run. Next, two additional runs for each parameter were performed. Each run differed from the initial test run only in one parameter value which was set to its “low”, or “high” value. The parameter ℓ_{pre} was tested at its “mid” and “high” values instead. The “low”, “mid”, and “high” values for the parameter $\mathcal{S}$ were {ordered}, {ordered, consecutive}, and {ordered, consecutive, complete}. Additional test runs were conducted for some parameters, as indicated.

Parameter	low	mid	high	additional	description
ℓ_{pre}	18*	30*	35		Minimum length of alignments for preliminary match annotation
b_{pre}	20	49	100		Minimum bitscore of alignments for preliminary match annotation
p_{pre}	60	80	90	95, 99	Minimum percent identity of alignments for preliminary match annotation
c_{min}	2	4	6		Minimum contribution to coverage by alignments for preliminary match annotation
t	4	8	12		Maximum shift allowed for merging alignments
d	1	3	6		Maximum distance between alignments for merging
b_{add}	25	49	100		Minimum bitscore of alignments for additional match annotation
p_{add}	60	80	90		Minimum percent identity of alignments for additional match annotation
r	0.5	0.8	0.9		Minimum proportion of product interval alignment must cover to be annotated as additional match and inherit index
r'	0.1	0.2	0.5		Minimum proportion of product interval alignment must cover to be annotated as fragment that inherits index
ℓ_{gap}	2	4	8		Minimum length of non-covered region in product to be considered length
ℓ_{ptr}	1	3	5		Minimum overlap length between successive product intervals to qualify as pointer
$\mathcal{S}$	**				Collection of properties indicating $\mathcal{S}$-scrambling for unambiguous subarrangements
s	0	5	10		Maximum size of precursor interval overlap tolerated for two matches in an unambiguous subarrangement
u	2	4	10		Maximum number of unambiguous subarrangements extracted
q	25	50	75		Minimum proportion of product sequence that must be covered by product intervals of preliminary matches for properties of arrangement to be computed

* $\ell_{\text{pre}} = 18$ was used as the fixed value across test runs for other parameters instead of 30.

** low: {ordered}, mid: {ordered, consecutive}, high: {ordered, consecutive, complete}, additional: {consecutive, complete}

Table A.2: Counts of arrangements and alignments of test runs of parameters related to the MDS annotation step by SDRAP. When a value does not deviate from the value in the row for the center run, it is left blank. The counts of tests of parameters r, r', and ℓ_{ptr} do not deviate from the center run and were omitted entirely.

		# arrangements			# alignments satisfying $\ell_{\text{pre}}, b_{\text{pre}}, p_{\text{pre}}$ thresholds**		# alignments satisfying $b_{\text{add}}, p_{\text{add}}$ thresholds**	
		total	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$
center*		429566	117612	59885	1740567	891293	1740567	891293
ℓ_{pre}	mid*	332994	117510	59586	1720298	877743	1720298	877743
	high	277861	117316	59168	1686785	856651	1686785	856651
b_{pre}	low	8167735	119159	60303	1813861	923872	1744673	895206
	high	184836	113585	54076	1435953	711786	1710538	814542
p_{pre}	low	446673	130942	61289	1835682	927143	1772034	908144
	high	366054	91517	47265	1220892	647596	1535790	712381
	95%	283489	65588	27941	676173	329977	1186096	400808
	99%	160400	30010	13485	269350	127346	423126	168381
c_{min}	low	429582	117603	59829	1740479	890265	1740479	890265
	high	429560	117612	59908	1740682	891611	1740682	891611
t	low		117613	59933	1740576	891874	1740576	891874
	high		117608	59870	1740533	891153	1740533	891153
d	low		117627	60010	1740707	893098	1740707	893098
	high		117597	59855	1740387	890457	1740387	890457
b_{add}	low						1808171	919285
	high						1451418	763590
p_{add}	low						1782502	906120
	high						1294164	746688
ℓ_{gap}	low		117588	59825	1740450	890390	1740450	890390
	high		117655	60130	1740751	893959	1740751	893959

* The chosen low value for ℓ_{pre} was used as the fixed value across test runs for other parameters instead of the mid value.
** Numbers of alignments in the columns for $\mathbb{A}_{30}$, $\mathbb{A}_{90}$ are taken from those between precursor and product sequences whose arrangement is in the respective set.

Table A.3: Counts of matches for test runs of parameters related to MDS annotation step by SDRAP. When a value does not deviate from the value in the row for the center run, it is left blank. The counts of tests of parameters r', and ℓ_{ptr} do not deviate from the center run and were omitted entirely.

		# of preliminary matches		# of additional matches		# of merged matches	
		in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$
center*		711568	355115	407070	155486	250315	149623
ℓ_{pre}	mid*	703325	349741	405119	154115	246390	147117
	high	691449	341336	400540	152568	237872	144062
b_{pre}	low	728280	361458	408032	155952	254664	151778
	high	605583	287584	390740	146576	226193	133778
p_{pre}	low	757659	365121	404361	157424	252858	151544
	high	552112	292340	479820	135543	256311	125787
	95%	410835	199519	446861	82703	255385	66045
	99%	242705	108125	81088	28199	61009	21955
c_{min}	low	714603	356373	419859	159276	254675	152112
	high	709178	354278	396533	153519	247391	148508
t	low	717737	358855	402021	150824	243753	144244
	high	709400	353934	407429	155907	252757	151072
d	low	733384	366978	388375	143358	217169	129691
	high	693918	346318	420928	163891	275573	162754
b_{add}	low			412789	157385	262582	152567
	high			359004	138948	216096	133541
p_{add}	low			425026	159502	252549	150586
	high			226556	107972	187068	118154
r	low			585559	239051	306226	191833
	high			341837	132860	230577	136854
ℓ_{gap}	low	711474	354894	407068	155154	250307	149424
	high	711726	356323		155891	250331	150044

* The chosen low value for ℓ_{pre} was used as the fixed value across test runs for other parameters instead of the mid value.

Table A.4: Counts of gaps, pointers, and fragments for test runs of parameters related to MDS annotation step by SDRAP. When a value does not deviate from the value in the row for the center run, it is left blank.

		# gaps		# pointers		# fragments	
		in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$	in $\mathbb{A}_{30}$	in $\mathbb{A}_{90}$
center*		310040	95093	401030	237895	502134	264506
ℓ_{pre}	mid*	312596	96650	391851	231515	500301	263534
	high	315783	98293	379350	222514	495499	259483
b_{pre}	low	314681	94555	411635	243815	503777	265273
	high	305089	92489	324314	182680	618642	284911
p_{pre}	low	355106	98754	420481	242795	522816	272595
	high	199826	67434	344141	205098	404580	183182
	95%	136408	35558	270504	147755	280656	69603
	99%	60454	16813	165625	79982	91449	20597
c_{min}	low	309939	94911	403583	239182	506619	265979
	high	310753	95562	398652	236846	496788	263610
t	low	310043	95161	402813	239024	515172	276604
	high	310044	95071	400185	237384	498660	261775
d	low	309962	95257	401836	238643	556494	301596
	high	298076	90155	400119	237280	463539	240079
b_{add}	low					545623	282192
	high					443980	240485
p_{add}	low					533474	277987
	high					382779	210390
r	low					323645	180941
	high					567367	287132
r'	low					544885	296028
	high					393634	201707
ℓ_{gap}	low	322583	100747	400999	237807	502123	264031
	high	279535	83193	401077	238334	502140	265328
ℓ_{ptr}	low			438964	262094		
	high			304497	178187		

* The chosen low value for ℓ_{pre} was used as the fixed value across test runs for other parameters instead of the mid value.

Table A.5: Counts of arrangements for test runs of parameters related to the property computation step by SDRAP. When a value does not deviate from the value in the row for the center run, it is left blank. The total number of arrangements was 429566 for all test runs described in this table. The sizes of $\mathbb{A}_q$ and $\mathbb{A}_{90}$ are 97846, and 59885, respectively, throughout, except for the test runs of q. With q at its "low" and "high" value, the two sets have sizes 124066 and 76569, respectively. The counts of tests of parameters $\mathcal{S}$ does not deviate from the center run and was omitted entirely.

		# arrangements which exceeded u		# repeating		# s-overlapping		# repeating and s-overlapping	
		$\mathbb{A}_q \setminus \mathbb{A}_q^*$	$\mathbb{A}_{90} \setminus \mathbb{A}_{90}^*$	in $\mathbb{A}_q^*$	in $\mathbb{A}_{90}^*$	in $\mathbb{A}_q^*$	in $\mathbb{A}_{90}^*$	in $\mathbb{A}_q^*$	in $\mathbb{A}_{90}^*$
center		10380	5457	7748	4835	8165	4287	1550	888
s	low	10453	5483	7709	4824	12133	6603	2538	1647
	high	10355	5450	7772	4839	6924	3625	1363	763
u	low	14102	7612	4774	3040	6309	3305	442	266
	high	7285	3721	10039	6217	10416	5508	2997	1755
q	low	12198		9262		10002		1845	
	high	8031		6110		6327		1248	

Table A.6: Counts of some arrangement properties for test runs for parameters related to the property computation step by SDRAP. When a value does not deviate from the value in the row for the center run, it is left blank.

		# weakly scrambled		# strongly scrambled	
		in $\mathbb{A}_q^*$	in $\mathbb{A}_{90}^*$	in $\mathbb{A}_q^*$	in $\mathbb{A}_{90}^*$
center		23534	13383	17308	9730
$\mathcal{S}$	low	21198	12077	15685	8968
	high	25055	14225	22554	12609
	other	8580	4534	7170	3675
s	low	25401	14308	17970	9979
	high	22858	13047	17071	9648
u	low	20202	11453	15874	8967
	high	26193	14850	18058	10168
q	low	29531		22102	
	high	17867		12960	

Table A.7: Statistics for the comparison of two SDRAP test runs with results from [16]. A full description of the parameter values used is given in Table 3.1. Whenever, a directly comparable number could not be obtained due to discrepancies in the annotation procedures, results from [16] are compared to multiple descriptive statistics obtained form the output of SDRAP.

description	$(p_{\text{pre}}, p_{\text{add}})$ (99.0, 95.0)	$(p_{\text{pre}}, p_{\text{add}})$ (95.0, 90.0)	reported in [16]
# arrangements of 1- or 2- telomeric product sequences with coverage of at least 90%	12665	22844	16220
# preliminary matches*	104069	182981	>225000
# matches*	105277	189513	
# merged preliminary matches*	861	7252	
# alignments merged into preliminary matches*	1795	18244	
# double counted additional matches**	258	1386	
# arrangements that have only one preliminary match*	1039	2896	548
# arrangements whose only preliminary match is merged*	81	1309	
# weakly scrambled arrangements*	1709	4140	2818
# weakly scrambled arrangements containing matches of both orientations*	978 994	2249 2355	1676
avg # matches over length per product kb in weakly scrambled arrangements*	4.1/kb 4.3/kb	4.3/kb 5.2/kb	4.9/kb
avg # matches per product kb in strongly nonscrambled arrangements*,**	3.0/kb 3.0/kb	3.0/kb 3.0/kb	3.7/kb
median length of preliminary matches in weakly scrambled arrangements*,***	153bp	132bp	81bp
median length of preliminary matches in strongly non-scrambled arrangements*,**,***	196bp	191bp	181bp

* restricted to matches and arrangements of 1- or 2- telomeric product sequences with coverage at least 90%

** excluding arrangements with more than u unambiguous subarrangements

*** length of preliminary matches does not include pointers

no equivalent to additional matches exists in [16] since repeating MDSs were filtered

Table A.8: Statistics for the comparison of two SDRAP test runs with results from [12]. A full description of the parameter values used is given in Table 3.1. Double counted additional matches are alignments annotated as multiple additional matches due to sufficient overlap with multiple preliminary matches. Such alignments are counted in the numbers of all matches for SDRAP each time they are annotated as additional match.

description	$(p_{\text{pre}}, p_{\text{add}})$ (99.0, 95.0)	(95.0, 90.0)	reported in [12]
# 2-telomeric contigs	16027		17198
# product preliminary matches	422704	701725	278656
# all matches	643712	1279163	752901
# double counted additional matches	54823	190948	N/A
# arrangements with coverage at least 30%	20713	27189	39128
# weakly scrambled arrangements with coverage at least 30%*	2869	4176	5909
# strongly complete arrangements with coverage at least 90%*	9925	15095	15210
# weakly scrambled and strongly complete arrangements with coverage at least 90%*	1254	2109	1548

* restricted to matches and arrangements of 2-telomeric product sequences which had no more than u unambiguous subarrangements

Table A.9: Overview of the numbers of arrangements with properties derived from SDRAP test runs on the macronuclear genomes from [42] and [28] using parameter values described in Table 3.1. The set $\mathbb{A}$ consists of all arrangements, $\mathbb{A}_{30}$ consists of all arrangements covering the product sequence by at least 30%, and $\mathbb{A}_{90}^{2T}$ consists of all arrangements of 2-telomeric mac sequences which cover the product sequence by at least 90%.

		mac genome from [42]		mac genome from [28]	
		(99.0, 95.0)	(95.0, 90.0)	(99.0, 95.0)	(95.0, 90.0)
% unidirectional	$\mathbb{A}$	99.20	98.69	99.36	99.30
	$\mathbb{A}_{30}$	99.27	97.32	99.80	99.56
	$\mathbb{A}_{90}^{2T}$	99.92	99.67	99.75	99.64
% coherent	$\mathbb{A}$	98.79	97.32	98.79	98.60
	$\mathbb{A}_{30}$	98.39	93.63	99.94	99.79
	$\mathbb{A}_{90}^{2T}$	99.97	99.78	99.98	99.85
% consistent	$\mathbb{A}$	96.29	92.99	92.91	92.36
	$\mathbb{A}_{30}$	98.28	93.06	99.92	99.76
	$\mathbb{A}_{90}^{2T}$	99.94	99.75	99.97	99.83
% crossing	$\mathbb{A}$	17.88	17.96	18.93	17.20
	$\mathbb{A}_{30}$	3.35	14.15	2.29	3.76
	$\mathbb{A}_{90}^{2T}$	0.92	2.72	2.63	3.54
% flat	$\mathbb{A}$	81.60	81.33	80.34	82.06
	$\mathbb{A}_{30}$	96.45	85.50	96.94	95.37
	$\mathbb{A}_{90}^{2T}$	98.91	96.74	96.45	95.40
max density	$\mathbb{A}$	49	50	42	22
	$\mathbb{A}_{30}$	49	50	42	22
	$\mathbb{A}_{90}^{2T}$	2	4	5	5
max height	$\mathbb{A}$	48	57	41	21
	$\mathbb{A}_{30}$	48	57	41	21
	$\mathbb{A}_{90}^{2T}$	1	3	4	4

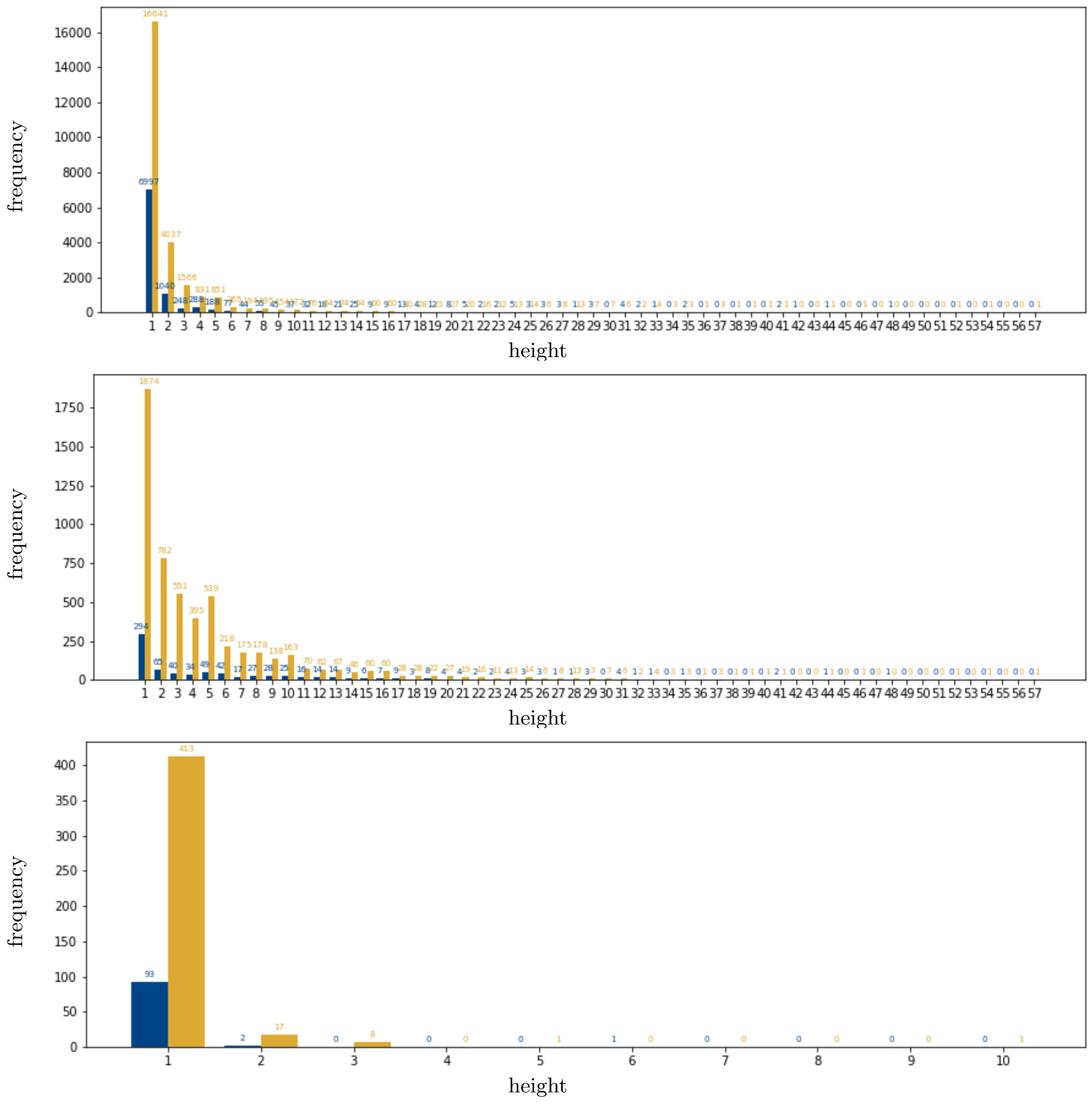


Figure A.1: The frequencies of different height values for SDRAP test runs with $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in blue and $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in yellow, applied to the macronuclear genome from [42]. All arrangements are included in the plot at the top. The middle plot is restricted to arrangements with product sequence coverage at least 30%. The bottom plot only contains the height frequencies for arrangements of 2-telomeric product sequences with at least 90% coverage. The majority of arrangements has zero height which is omitted from this figure.

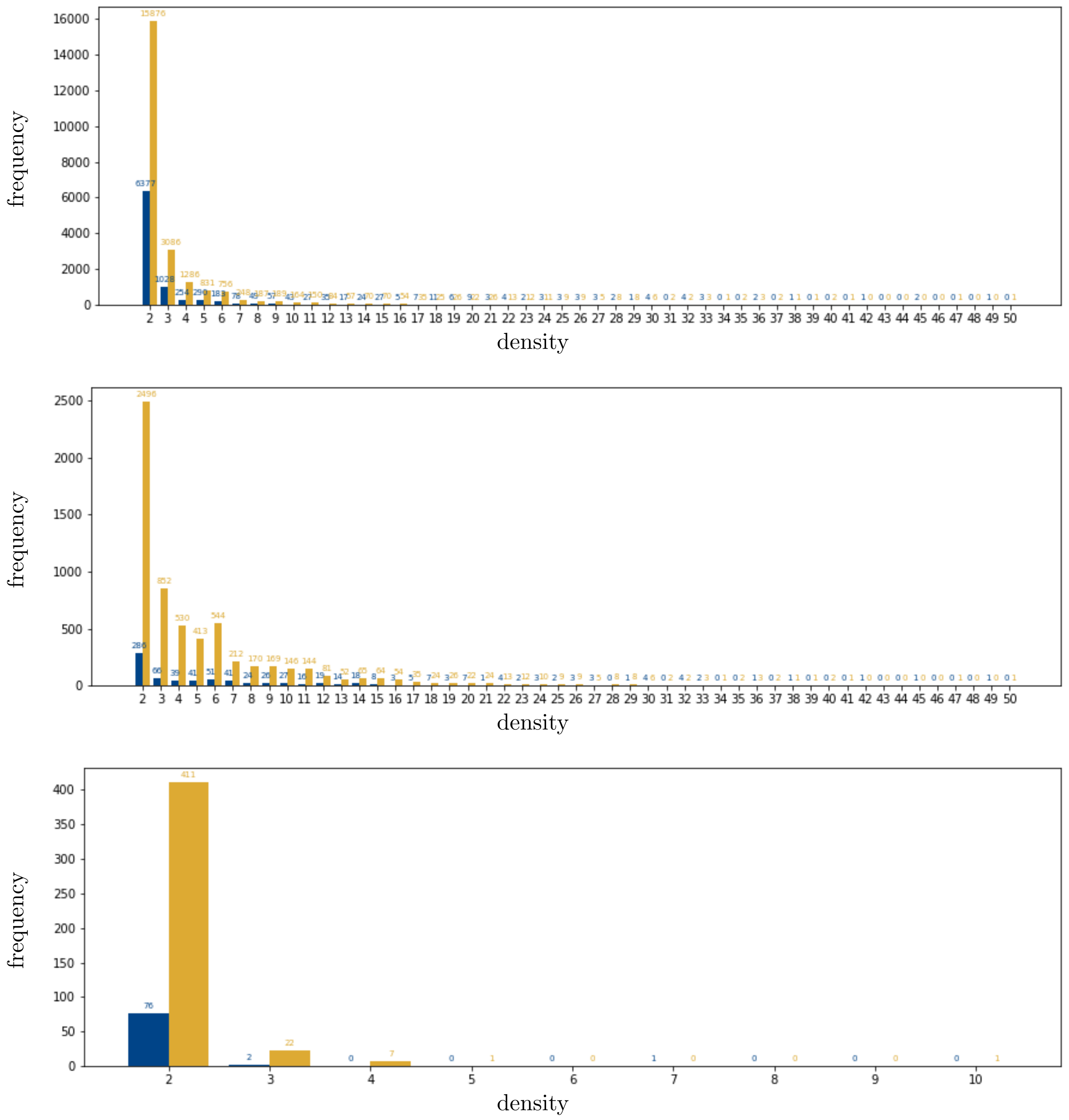


Figure A.2: The frequencies of different density values for SDRAP test runs with $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in blue and $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in yellow, applied to the macronuclear genome from [42]. All arrangements are included in the plot at the top. The middle plot is restricted to arrangements with product sequence coverage at least 30%. The bottom plot only contains the density frequencies for arrangements of 2-telomeric product sequences with at least 90% coverage. The majority of arrangements has density one which is omitted from this figure.

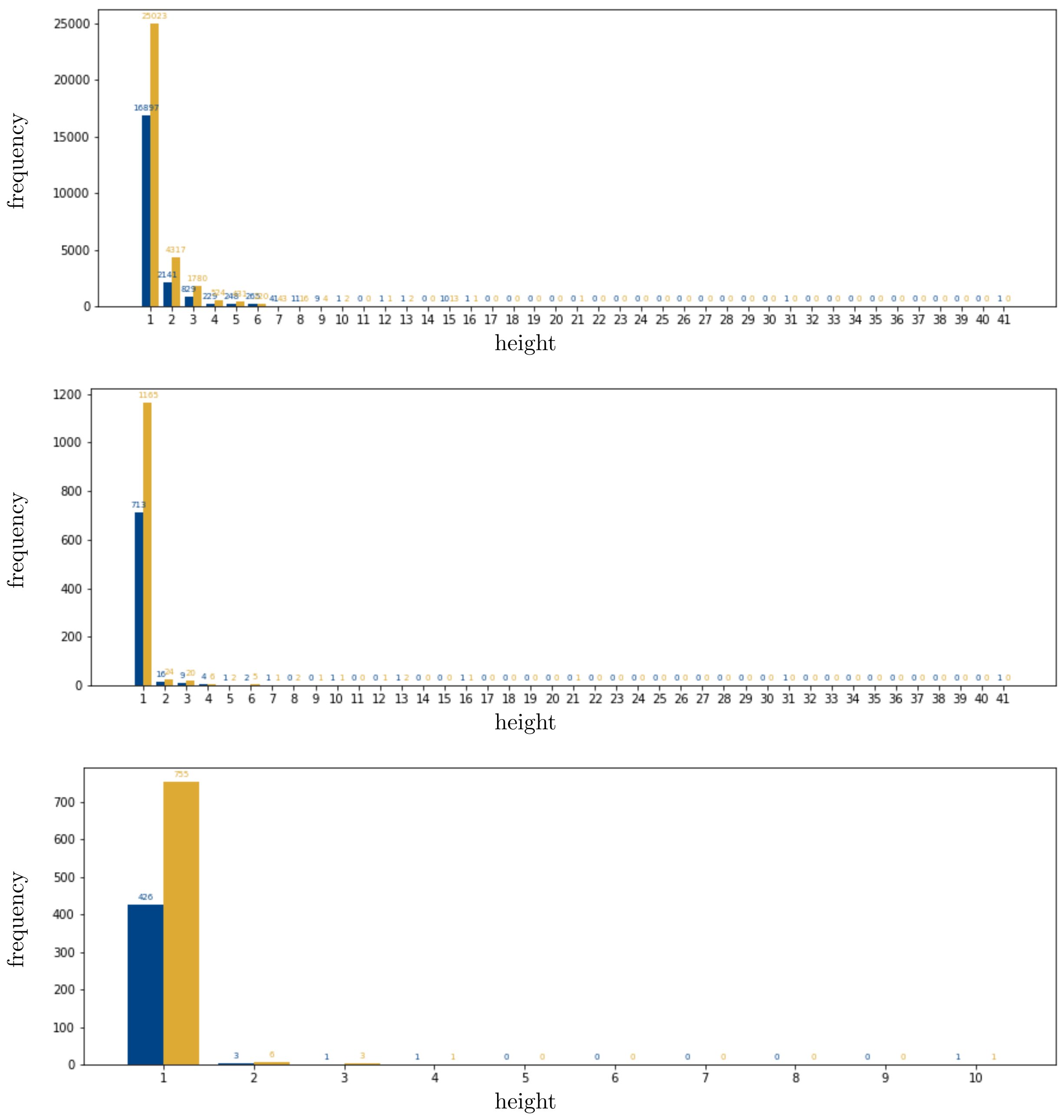


Figure A.3: The frequencies of different height values for SDRAP test runs with $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in blue and $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in yellow, applied to the macronuclear genome from [28]. All arrangements are included in the plot at the top. The middle plot is restricted to arrangements with product sequence coverage at least 30%. The bottom plot only contains the height frequencies for arrangements of 2-telomeric product sequences with at least 90% coverage. The majority of arrangements has zero height which is omitted from this figure.

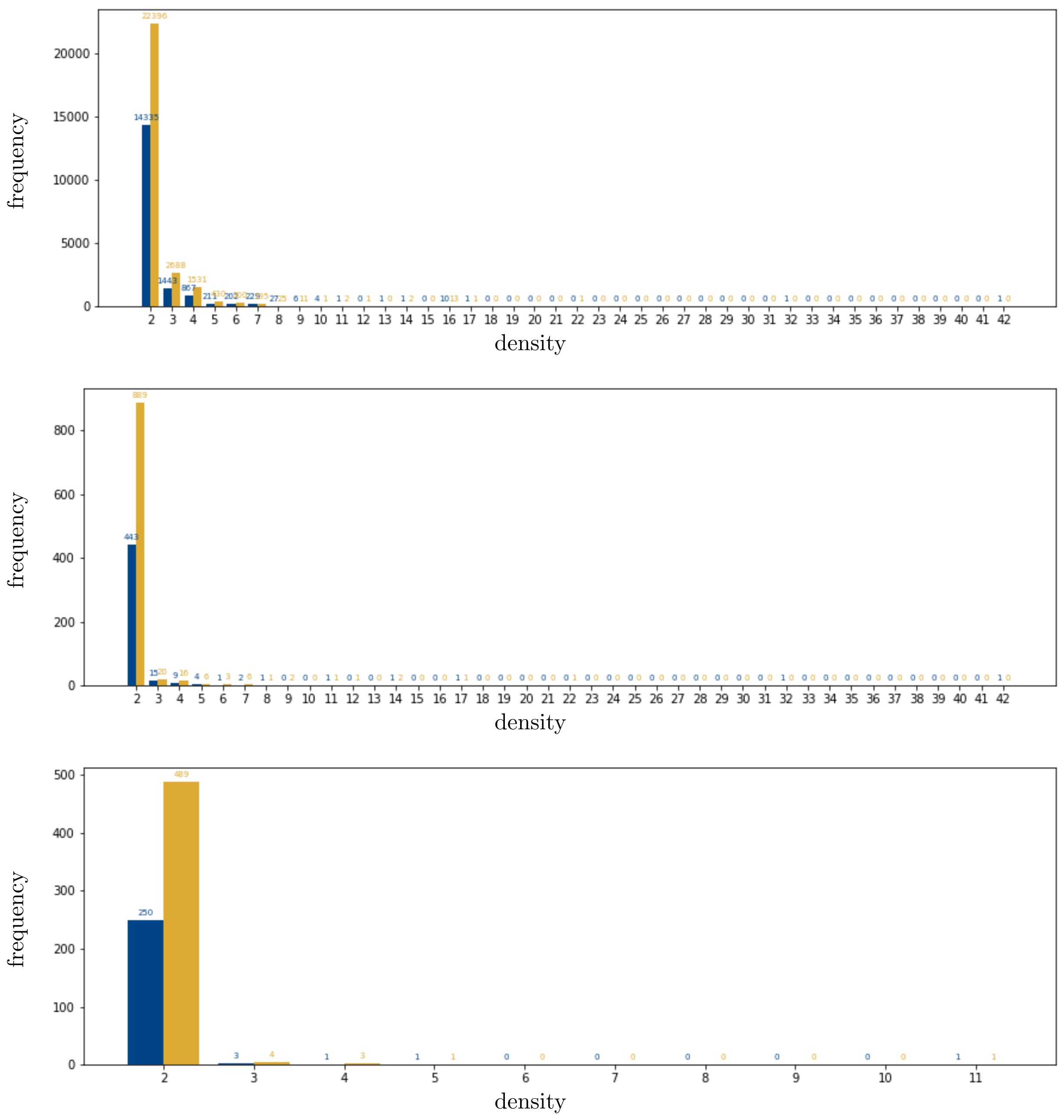


Figure A.4: The frequencies of different density values for SDRAP test runs with $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in blue and $(p_{\text{pre}}, p_{\text{add}}) = (99.0, 95.0)$ in yellow, applied to the macronuclear genome from [28]. All arrangements are included in the plot at the top. The middle plot is restricted to arrangements with product sequence coverage at least 30%. The bottom plot only contains the density frequencies for arrangements of 2-telomeric product sequences with at least 90% coverage. The majority of arrangements has density one which is omitted from this figure.

Table A.10: Analysis of the relationship double counted additional matches and high height and density values for SDRAP test runs with mac genomes from [42] and [28] using parameter values described in Table 3.1. High height and density thresholds for the two genomes were obtained from the max height and density remaining stable in the right half of the corresponding bar plots in Figure 3.5. For the mac genome from [42], the density and height considered high are 8, and 6, respectively. For the mac genome from [28], high density and height thresholds are 17, and 16, respectively. The numbers of double counting arrangements with high density and high height are particularly low in the last two columns because only very few arrangements had high density and high height for the mac genome from [28], which can be seen in Figures A.3 and A.4.

		mac genome from [42]		mac genome from [28]	
		(99.0, 95.0)	(95.0, 90.0)	(99.0, 95.0)	(95.0, 90.0)
# double counting arrangements*	$\mathbb{A}$	2267	8453	2743	4895
	$\mathbb{A}_{30}$	858	4996	607	795
	$\mathbb{A}_{90}^{2T}$	91	210	219	374
Avg # double counted**	$\mathbb{A}_{30}$	43.4	33.6	2.5	2.4
	$\mathbb{A}_{90}^{2T}$	1.8	2.3	2.3	2.7
# double counting arrangements* with high density	$\mathbb{A}$	309	1075	2	1
	$\mathbb{A}_{30}$	187	1008	2	1
	$\mathbb{A}_{90}^{2T}$	0	0	0	0
Avg # double counted** in high density arrangements	$\mathbb{A}_{30}$	139.2	113.4	170.0	91.0
# double counting arrangements* with high height	$\mathbb{A}$	373	1357	2	1
	$\mathbb{A}_{30}$	221	1268	2	1
	$\mathbb{A}_{90}^{2T}$	0	0	0	0
Avg # double counted** in high height arrangements	$\mathbb{A}_{30}$	129.0	100.7	170.0	91.0

* A double counting arrangement is an arrangement which contains multiple additional matches derived from the same alignment.

** Counts the number of times an alignment is annotated as additional match in excess of the first (i.e. an alignment annotated n times as an additional match, is counted here $n-1$ times).

Table A.11: Analysis of the relationship double counted additional matches and presence of telomeres for SDRAP test runs with mac genomes from [42] and [28] using parameter values described in Table 3.1. High height and density thresholds for the two genomes were obtained from the max height and density remaining stable in the right half of the corresponding bar plots in Figure 3.5. For the mac genome from [42], the density and height considered high are 8, and 6, respectively. For the mac genome from [28], high density and height thresholds are 17, and 16, respectively.

	mac genome from [42]		mac genome from [28]	
	(99.0, 95.0)	(95.0, 90.0)	(99.0, 95.0)	(95.0, 90.0)
# 1- or 2-telomeric sequences in double counting arrangements*	424	638	667	901
# 2-telomeric sequences in double counting arrangements*	312	438	592	762
# 1- or 2-telomeric in high density double counting arrangements*	2	2	0	0
# 2-telomeric in high density double counting arrangements*	0	1	0	0
# 1- or 2-telomeric in high height double counting arrangements*	1	2	0	0
# 2-telomeric in high height double counting arrangements*	0	1	0	0

* A double counting arrangement is an arrangement which contains multiple additional matches derived from the same alignment.

Appendix B

Alignment Pasting Procedures and Data

Algorithm 2: Procedure NextLeft.

Input: Local Alignment A between u and v,
Index l of member of L_{end} with end coordinate in u less than or equal to that of A, and
List L_{end} of local alignments between u, v sorted by decreasing end coordinate in u and where all alignments with score larger than $s(A)$ are marked.

Output: Smallest index l' of member of L_{end}, such that $L_{\text{end}}[l']$ and A are in left-legal configuration and left-$(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastable, or $|L_{\text{end}}| + 1$ if none exist.

```
1  $\mathcal{D}_{\max} \leftarrow \frac{2 \cdot s(A) - s_{\text{int}}}{m_0}$
2  while $l \leq |L_{\text{end}}|$ do
3      if $\mathcal{D}_u(L_{\text{end}}[l], A) > \mathcal{D}_{\max}$ then
4          return $|L_{\text{end}}| + 1$
5      end
6      if $L_{\text{end}}[l]$ not marked and $L_{\text{end}}[l], A$ are in left-legal configuration and left-$(s_{\text{int}}, p_{\text{int}}, \gamma_0)$-pastable
        then
7          return $l$
8      end
9      $l \leftarrow l + 1$
10 end
11 return $l$
```

Table B.1: Performance comparison between gapped BLAST and the combination of ungapped BLAST with PasteAlignments to obtain gapped alignments using scoring parameters $(id_0, m_0, \Delta_0, g_0) = (1, 2, 0, 2.5)$. The micronuclear genome assembly of *O. trifallax* published in [16] was used as database genome and the macronuclear genome assembly published in [28] was used as query in all BLAST searches. The only difference between the various PasteAlignment tests is the parameter value for the gap_tolerance parameter. The BLAST searched differed only in their xdrop_gap_final parameter value.

Description	total # alignments	avg. length	avg. score	avg. % identity	avg. bitscore	avg. evalue	total # pastings	avg. time (s) $\pm\sigma$	avg. time (s) $\pm\sigma$ blind
blastn -megablast -xdrop_gap_final 40	782963	277.63	246.06	96.91	455.50	6.99e-6	N/A	435.54 ±14.61	408.81 ±9.75
blastn -megablast -xdrop_gap_final 60	775111	282.71	248.78	96.84	460.53	7.03e-6	N/A	442.35 ±28.19	417.07 ±6.48
blastn -megablast -xdrop_gap_final 80	767974	286.79	250.94	96.79	464.52	7.08e-6	N/A	451.64 ±23.44	431.67 ±7.72
blastn -megablast -xdrop_gap_final 100	757082	292.16	253.83	96.72	469.86	7.22e-6	N/A	454.21 ±34.30	430.38 ±23.78
blastn -megablast -ungapped	871639	231.82	214.26	97.40	396.79	7.08e-6	N/A	414.56 ±1.74	387.39 ±1.42
paste_alignments –gap_tolerance 2	816087	248.12	227.37	97.36	420.99	7.41e-6	55552	19.98 ±1.17	16.13 ±0.27
paste_alignments –gap_tolerance 4	802362	252.60	230.65	97.35	427.04	7.50e-6	69277	20.08 ±0.99	16.21 ±0.22
paste_alignments –gap_tolerance 6	795584	254.93	232.17	97.32	429.87	7.53e-6	76055	20.12 ±1.15	16.23 ±0.17
paste_alignments –gap_tolerance 8	792461	256.06	232.81	97.30	431.04	7.55e-6	79178	20.82 ±2.23	16.16 ±0.17
paste_alignments –gap_tolerance 10	789449	257.14	233.46	97.29	432.24	7.57e-6	82190	20.08 ±1.06	16.18 ±0.13
paste_alignments –gap_tolerance 20	779268	260.91	235.50	97.23	436.01	7.65e-6	92371	24.86 ±4.35	19.64 ±2.52
paste_alignments –gap_tolerance 40	725727	282.38	246.22	96.75	455.80	8.14e-6	145912	23.93 ±0.64	18.41 ±0.35

Table B.2: Performance comparison between gapped BLAST and the combination of ungapped BLAST with PasteAlignments to obtain gapped alignments using scoring parameters $(id_0, m_0, \Delta_0, g_0) = (1, 5, 0, 5.5)$. The micronuclear genome assembly of *O. trifallax* published in [16] was used as database genome and the macronuclear genome assembly published in [28] was used as query in all BLAST searches. The only difference between the various PasteAlignment tests is the parameter value for the gap_tolerance parameter. The BLAST searched differed only in their xdrop_gap_final parameter value.

Description	total # align- ments	avg. length	avg. score	avg. % identity	avg. bitscore	avg. evalue	total # pastings	avg. time (s) $\pm\sigma$	avg. time (s) $\pm\sigma$ blind
blastn -megablast -xdrop_gap_final 40	837675	234.06	208.10	98.16	417.74	6.17e-7	N/A	421.33 ±6.25	393.87 ±0.89
blastn -megablast -xdrop_gap_final 60	814138	244.97	214.01	98.09	429.58	6.23e-7	N/A	420.83 ±18.74	394.27 ±1.67
blastn -megablast -xdrop_gap_final 80	804037	249.87	216.43	98.07	434.44	6.29e-7	N/A	428.20 ±4.83	395.47 ±8.02
blastn -megablast -xdrop_gap_final 100	798680	252.48	217.64	98.07	436.87	6.32e-7	N/A	418.49 ±17.85	396.16 ±4.98
blastn -megablast -ungapped	900394	212.31	192.79	98.34	387.03	6.11e-7	N/A	423.39 ±8.31	394.99 ±2.56
paste_alignments –gap_tolerance 2	840231	228.13	203.05	98.29	407.60	6.39e-7	60163	20.24 ±1.04	16.55 ±0.19
paste_alignments –gap_tolerance 4	831220	230.76	204.38	98.27	410.28	6.43e-7	69174	20.06 ±1.22	16.51 ±0.17
paste_alignments –gap_tolerance 6	827299	231.94	204.87	98.27	411.25	6.45e-7	73095	21.18 ±1.87	16.78 ±0.49
paste_alignments –gap_tolerance 8	825800	232.40	205.02	98.26	411.56	6.46e-7	74594	20.90 ±0.85	16.66 ±0.14
paste_alignments –gap_tolerance 10	824294	232.86	205.19	98.26	411.90	6.47e-7	76100	19.69 ±0.93	16.48 ±0.20
paste_alignments –gap_tolerance 20	819112	234.46	205.69	98.24	412.91	6.51e-7	81282	25.64 ±3.15	22.02 ±2.74
paste_alignments –gap_tolerance 40	787448	245.03	206.62	98.08	414.76	6.75e-7	112946	22.16 ±1.21	17.74 ±1.75

Table B.3: Performance comparison between gapped BLAST and the combination of ungapped BLAST with PasteAlignments to obtain gapped alignments using scoring parameters $(id_0, m_0, \Delta_0, g_0) = (2, 7, 4, 2)$. The micronuclear genome assembly of *O. trifallax* published in [16] was used as database genome and the macronuclear genome assembly published in [28] was used as query in all BLAST searches. The only difference between the various PasteAlignment tests is the parameter value for the gap_tolerance parameter. The BLAST searched differed only in their xdrop_gap_final parameter value.

Description	total # alignments	avg. length	avg. score	avg. % identity	avg. bitscore	avg. evalue	total # pastings	avg. time (s) $\pm\sigma$	avg. time (s) $\pm\sigma$ blind
blastn -megablast -xdrop_gap_final 40	780340	272.11	468.48	97.33	456.91	0.00	N/A	606.67 ±12.38	571.98 ±6.89
blastn -megablast -xdrop_gap_final 60	747869	288.54	487.20	96.98	475.13	0.00	N/A	721.93 ±29.15	694.57 ±23.20
blastn -megablast -xdrop_gap_final 80	705661	310.10	511.96	96.50	499.25	0.00	N/A	890.67 ±52.25	866.81 ±48.58
blastn -megablast -xdrop_gap_final 100	660257	336.38	540.45	95.93	526.99	0.00	N/A	1193.59 ±34.02	1171.23 ±28.09
blastn -megablast -ungapped	879711	224.11	407.58	97.89	397.60	0.00	N/A	477.43 ±10.83	451.40 ±4.39
paste_alignments –gap_tolerance 2	823314	239.98	431.18	97.86	420.33	0.00	56397	22.42 ±1.38	17.22 ±0.75
paste_alignments –gap_tolerance 4	811529	243.67	435.94	97.84	424.97	0.00	68182	24.59 ±3.58	16.98 ±0.42
paste_alignments –gap_tolerance 6	805858	245.52	438.15	97.82	427.12	0.00	73853	21.38 ±1.22	17.40 ±0.75
paste_alignments –gap_tolerance 8	803279	246.39	439.06	97.81	428.01	0.00	76432	20.61 ±0.97	16.54 ±0.21
paste_alignments –gap_tolerance 10	800741	247.25	439.98	97.80	428.91	0.00	78970	19.92 ±1.03	16.73 ±0.56
paste_alignments –gap_tolerance 20	791343	250.48	443.75	97.73	432.58	0.00	88368	22.68 ±1.80	17.74 ±0.66
paste_alignments –gap_tolerance 40	730525	273.77	472.78	97.05	460.84	0.00	149186	24.06 ±2.66	17.71 ±0.90