Algebraic Models of Large Scale Genome Rearrangement Events

A thesis submitted for the degree of Doctor of Philosophy

Supervisors

Author
Sangeeta Bhatia

Prof. Andrew Francis
Associate Prof. Volker Gebhardt
Associate Prof. Mark Tanaka

February 10, 2018
ABSTRACT

...I’d put up the four words that anyone telling a story wants to hear. The ones that show that it’s working, and that pages will be turned:
“... and then what happened?”

Neil Gaiman

Variations in genome arrangements are an important source of phylogenetic information and have been used to inform phylogenetic studies since the 1930s. The arrangement of genes along chromosomes is roughly analogous to that of beads on a necklace. It is no surprise therefore that in designing algorithms for comparison of genome arrangements, permutations have constituted the construct of choice for representing genomes. However, notwithstanding the widespread use of permutations in genome rearrangement studies, the rich theory of permutation groups has not been exploited to tackle the problems in evolutionary genomics. Our thesis attempts to address this gap in the existing methods. We describe two algebraic models for the evolution of genomes through large scale rearrangement events.

Our first model is an algebraic translation of a universal genome rearrangement operator called double cut and join (DCJ). In going from a graph theoretic to an algebraic setting, we provide new proofs for some results related to the DCJ and discuss the potential for adapting the DCJ operator to specific biological models.
The second part of our thesis makes use of the well developed theory of rewriting systems to design a flexible framework for determining weighted rearrangement distances. The proposed framework can be adapted to a number of different rearrangement models and presents an important contribution to the study of weighted distances in genome rearrangement literature. Our work is the first to utilise the theory of rewriting systems to a problem in phylogenetics thereby linking these two separate fields.

Science flourishes best when scientific disciplines are not isolated in silos but like teens at a party, freely mingle and converse with each other. Contributing to the conversation between algebra and biology is the primary goal of this thesis. The significance of our work lies in bringing new methods and approaches to biology and raising new mathematical questions motivated from biology.
To Mom. We miss you every day.
ACKNOWLEDGEMENTS

I want to thank Andrew for his guidance and patience. In the four years I have known him, I am yet to see him frown. I also thank both my co-supervisors for their feedback which helped improve my understanding and writing. I acknowledge the support extended to me by Western Sydney University through the HDR Scholarship.

To my best friend - in living up to your opinion of me, I have become a better person. Thank you, my friend.

My life partner - thank you for keeping me sane for the last four years and for sharing the burden of my dreams.

I am a product of the sweat, blood and blessings of too many people to name. Suffice it to say, I remember and am grateful to each one of them.

I’ve often heard it said that it is a lucky researcher whose thesis is read by more than three people on the planet. Thank you dear Reader, for being one of the three of my audience. R.A. Fisher apologises to his readers in the preface of his book *The Genetical Theory of Natural Selection*, ‘No efforts of mine could avail to make the book easy reading.’ I follow his example and apologise in advance for the probable dryness of my thesis. If it lacks what Terry Pratchett called ‘narrativium’, believe me, it is not for want of trying.

Settle in reader, for this will be a long read.
DECLARATION

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. I, Sangeeta Bhatia, hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution.
CONTENTS

1 INTRODUCTION 13
   1.1 Phylogenetics 14
   1.2 Large scale genome rearrangement events 16
   1.3 Mechanisms of large scale rearrangement events 17
   1.4 Significance of large scale rearrangement events in phylogeny 20
   1.5 Thesis organisation and project aims 21

2 PRELIMINARIES 23
   2.1 Permutation multiplication 24
   2.2 Left and right multiplications 24
   2.3 Inverse of a permutation 25
   2.4 Cycles and cycle decomposition 26
   2.5 Conjugation 27
   2.6 Permutation as product of transpositions 27
   2.7 Group Action 28

3 MATHEMATICAL MODELS OF GENOME REARRANGEMENT 29
   3.1 Overview of Rearrangement Models 32
      3.1.1 Inversions 32
      3.1.2 Inversions on circular genomes 36
      3.1.3 Translocation 37
      3.1.4 Transpositions 38
      3.1.5 Rearrangement models with multiple operators 38
   3.2 Genome Representation Paradigms 39
3.2.1 The position paradigm – Genomes as maps between positions and regions 40
3.2.2 The content paradigm – Genomes as maps from regions to regions 42
3.3 Genome rearrangement events 45
3.3.1 Actions 46
3.3.2 Rearrangements in the position paradigm 47
3.3.3 Rearrangements in the content paradigm 49
3.3.4 Rearrangement operators on adjacency lists 51
3.4 The genome rearrangement distance problem 54
3.5 Going from one genome representation to the other 55
3.5.1 From position to content representation and back 55
3.5.2 From adjacency lists to chromosomes 58
3.5.3 Combined approaches 59
3.6 Summary and conclusions 61
4 Algebraic double-cut-and-join 62
4.1 Introduction 62
4.2 The double cut and join model 64
4.2.1 Genome as a graph 64
4.2.2 The double cut and join operator 65
4.2.3 The Double Cut and Join Distance 67
4.3 Genomes as permutations 70
4.4 The DCJ operator as an action on a permutation 73
4.4.1 DCJ action is a group action 76
4.4.2 Algebraic DCJ is equivalent to the “classical” DCJ 76
4.5 Products of involutions 79
4.6 Determining the DCJ distance 84
4.6.1 A lower bound on the DCJ distance 84
LIST OF FIGURES

Figure 1  Phylogenetic Tree  15
Figure 2  Breakpoint Graph  34
Figure 3  Genome representation as an adjacency list and the corresponding permutation representation  45
Figure 4  Composing the genome and the rearrangement maps  48
Figure 5  Action of the DCJ operator on a genome graph.  52
Figure 6  A circular genome and a corresponding permutation in the content paradigm  57
Figure 7  A circular genome can be encoded by more than one permutation  57
Figure 8  A signed permutation encoding the orientation of regions  58
Figure 9  The relationship between permutations arising from different genome representations.  59
Figure 10 Genome graph  65
Figure 11 DCJ operation when both the adjacencies being cut lie on the same chromosome  67
Figure 12 DCJ operation when the adjacencies being cut lie on different chromosomes  68
Figure 13 Adjacency graph  70
Figure 14 A genome graph and the encoding permutation  71
Figure 15 Algebraic DCJ operation is equivalent to the classical DCJ operation.  77
Figure 16  Loops on a Cayley Graph  129
Figure 17  Confluence  131
Figure 18  A phylogeny drawn using the weighted distances on linear genomes  136
Figure 19  A phylogeny drawn using the weighted distances on circular genomes  138
LIST OF TABLES

Table 1 Correct ways to compose a rearrangement map with a genome map 49
Table 2 Different ways to represent the genome and the action of rearrangements, with a focus on the composition of rearrangement operators. In the (right hand) Structure column the placement of an additional rearrangement $\rho_{n+1}$ is shown in bold. Note that when the side of the group action and the side the rearrangement acts on the genome clash (one right, one left), new rearrangements must be added into the middle of the expression. This is not ideal, and why we recommend use of the structures in the first and last rows, both involving representations of genomes mapping regions to positions. 50
Table 3 The number of genomes on $n$ regions 73
Table 4 Number of rewriting rules 140
INTRODUCTION

O ye powers! (for powers ye are, and great ones too) which enable mortal man to tell a story worth the hearing – that kindly shew him, where he is to begin it – and where he is to end it – what he is to put into it – and what he is to leave out – how much of it he is to cast into a shade – and whereabouts he is to throw his light! – Ye, who preside over this vast empire of biographical freebooters, and see how many scrapes and plunges your subjects hourly fall into; – will you do one thing? I beg and beseech you (in case you will do nothing better for us) that wherever in any part of your dominions it so falls out, that three several roads meet in one point, as they have done just here – that at least you set up a guide-post in the centre of them, in mere charity, to direct an uncertain devil which of the three he is to take.

Tristram Shandy, Laurence Sterne

Human beings are probably unique in the animal world in seeking answers that confer no immediate survival advantage. Our earliest stories contain seeds of explanation for the natural world that could be terrifying and beautiful at the same time. The urge to organise and to classify everything we can is an expression of this impulse to make up and when possible, discover
explanations. Faced with the bewildering complexity of life around them, the earliest scientists attempted to make sense of it by imposing on it some order and in an all too human hubris, envisaged all life as a great chain of beings with humans at the top.

It was the French natural historian Georges Cuvier who proposed in the late 18\textsuperscript{th} century that there are only four basic anatomical types of animals, thus shattering the hierarchical underpinnings of a scientific understanding of life. The scientific developments over the next century in fields as wide ranging as paleontology to geology to political economics, culminated in Alfred Wallace’s and Charles Darwin’s theory of evolution through natural selection\cite{27}. Since then, this theory has become a strong explanatory thread unifying diverse biological phenomena. In the oft-quoted words of the biologist Theodosius Dobzhansky, nothing in biology makes sense except in the light of evolution\cite{31}.

1.1 Phylogenetics

In the book \textit{The Science of Discworld}\cite{79} the authors succinctly express the essence of scientific endeavour when they say, ‘sometimes the best answer is a more interesting question’. Of the many different questions spawned by the growing evolutionary understanding, one that has assumed great importance is discovering the relationships among the many different plants, animals and microscopic life forms. The study of evolutionary relationships among entities is known as phylogenetics. The entities in question may be genes, groups of organisms or proteins. The output of a phylogenetic study is usually expressed as a “phylogenetic tree” (see Figure 1). In this thesis we are primarily concerned with phylogenetic relationships among various taxa and hence we will continue to refer to taxa when talking of the labels on the leaves of trees.
The leaf nodes of a phylogenetic tree are the observed taxa and the intermediate nodes are the inferred ancestors. Since a phylogenetic tree represents the evolutionary relationships among the taxa, it seems obvious that the more closely related taxa should be placed closer together on the tree and the distantly related further apart. This is analogous to drawing up a family tree. The evolutionary past however, is messy and difficult to infer. In order to place the taxa on a phylogenetic tree, we need tools that can inspect the clues in the present to bring to light events that have been hidden by time and enable us to assign some measure to the “relatedness” of the various taxa.

A set of clues is derived from the similarity or lack thereof between different features of an organism or a group of organisms from the taxa under study. One point of comparison could be the morphology of the organisms. Before the advent of the genomics era, the morphological characteristics were in fact the only observations available to biologists and natural historians to infer relations among organisms. For instance, Darwin’s comparison of the beak shapes of finches on Galápagos islands played an important role in shaping the theory of natural selection [26].

![Unrooted Tree](a) Unrooted Tree ![Rooted Tree](b) Rooted Tree

Figure 1: The left figure shows an unrooted phylogenetic tree. The tree clusters nodes A and C indicating that A and C are more closely related to each other than to the other nodes on the tree. An unrooted tree is a hypothesis only about the relatedness of the nodes. In contrast, a rooted tree such as one in the right subfigure is also a hypothesis about the direction of the ancestry. The non leaf nodes of a rooted tree are the inferred ancestors.
However a more reliable source of phylogenetic signal is the comparison of “genomic sequences”. Even before the physical structure of the genomic sequence was fully understood, linkage maps of genes were being used to differentiate between different species [30] and strains within the same species [32]. When genomic sequences for various groups of organisms started becoming available, the comparison of these sequences became the primary source of discovering evolutionary history.

The processes of DNA replication, which underlies the proliferation and continuation of life on earth, usually produce very high fidelity copies of the DNA. Yet, in spite of the many cellular mechanisms to prevent them, ‘mistakes’ in copying a sequence do occur. As Lewis Thomas beautifully puts it [96], “The capacity to blunder slightly is the real marvel of the DNA. Without this special attribute, we would still be anaerobic bacteria and there would be no music.” The tiny differences that result from such errors form the fodder for natural selection and lead to the evolution of the genomic sequences, as they increasingly diverge from the ancestor sequences. A mutation (a change in a replicated sequence compared to its parent) may be the replacement of a single base nucleotide or the insertion or deletion of a single base pair. Modern phylogenetic analysis relies to a large extent on such single nucleotide substitutions along with single site insertions and deletions (Single Nucleotide Polymorphisms or SNPs).

1.2 LARGE SCALE GENOME REARRANGEMENT EVENTS

While SNPs are very important sources of phylogenetic information, the availability of entire genomic sequences for a number of organisms has also made it possible to compare genome level features rather than focusing on the comparison of small parts of a genome. Examples of such features include gene
order, utilisation of variants of the genetic code, and insertion and deletion of introns. The present work focuses on the evolution of genomes through large scale changes to the order of the genes along a chromosome – shuffling of big chunks of genomes rather than changes to single sites. The gross changes to genomic content or the order of genes can take the form of inversions, translocations, and/or duplications and may involve a few hundred to a million base pairs [46]. Such changes are believed to be key agents of genetic variation in a number of different organism groups. For instance, the addition and deletion of chromosomal segments is believed to have contributed significantly to the genetic variation and hence to the rapid evolution within the human and the great ape lineage [53]. Another example comes from the bacterial family where inversions of chromosomal fragments are thought to be the main type of rearrangement event [9].

In the next two sections, we will briefly touch upon the biological mechanisms underlying the rearrangements of genomes (Section 1.3) and then discuss the significance of gene order from the perspective of extracting phylogenetic information (Section 1.4). The final section of this chapter will bring the focus back on our work and delineate the aims of the current project.

1.3 A QUICK STROLL THROUGH BIOLOGY: MECHANISMS OF LARGE SCALE REARRANGEMENT EVENTS

In the intricate dance of cell survival and replication, an important move is recombination – an umbrella term for the processes that enable the exchange of genetic material between a pair of chromosomes or even between regions of the same chromosome.

Homologous Recombination (HR) is a kind of recombination process that is primarily guided by the presence of homologous sequences. Homologous
DNA sequences are sequences that are similar due to a shared ancestry. Homologous recombination is an important step in sexual reproduction as well as DNA damage repair. What is of greater importance to us, it is one of the most important facilitators of genome shuffling and it is this avatar of homologous recombination that we discuss here.

The recombination process could lead to an even exchange of the genetic material between chromosomes, producing new combinations of the genes on the two chromosomes, or to an unequal exchange of genetic material. In the latter case, a part of one of the sequences involved in the recombination is lost, resulting in the deletion of genes on one chromosome and the addition of genes on the other. Unequal recombination is thus one of the processes underlying the deletion and insertion of large chunks of the genome. Recombination between homologous sequences on the same chromosome that are oriented in opposite directions could lead to the inversion of the part of the chromosome flanked by the two sequences (e.g. see [44]).

As mentioned earlier, homologous recombination requires the presence of long homologous sequences. In contrast to this, another kind of recombination process known as site-specific recombination utilises short stretches of sequences that bear little to no similarity to each other. Site-specific recombination is also known to facilitate several different rearrangements of the genome. For instance, similar to intra-chromosomal homologous recombination, site-specific recombination between oppositely oriented sites will invert the intermediate section. Recombination between sites on separate molecules causes one molecule to be integrated into another [47].

The word ‘recombination’ is generally used for the processes that shuffle the genomic content by bringing together existing sequences. Yet another category of rearrangements is mediated by “jumping genes” or transposable elements (TEs). These mobile elements of the genomic sequence were discov-
ered by Barbara McClintock in the 1940s and challenged the then prevailing picture of genes occupying fixed positions on a chromosome like the carriages of a train.

As the name “jumping genes” suggests, transposons are discrete DNA segments that have the ability to move between different chromosomal locations. This movement may be of the cut-and-paste variety where the transposable element is excised from one location on the chromosome and integrated at another, or it may involve copying the TE resulting in a new copy of the element. Through these two alternate but defining processes, TEs are involved in changing the content and/or the order of the genome.

The impact of TEs however, is not limited to these insertion and duplication events and there are some subtle ways in which TEs contribute to genome rearrangement. For instance, the excision of a TE is often imprecise, leading to the addition of new sequences near the site of insertion or the deletion of sequences near the original site. Another example of the indirect influence exerted by TEs is that by increasing their copies, they facilitate intrachromosomal recombination that in turn leads to inversion or deletion events [61].

To summarise, a number of cellular processes involved in critical cell functions such as DNA replication and damage repair are also implicated in the reorganization of the genome through the reordering of genomic content. Our knowledge of the physical mechanisms bringing about a change in gene order is being expanded through active research. Even with our current rudimentary understanding of the underlying physical processes, the variations in gene order have been mined as a source of phylogenetic information. We will examine the reasons for the use of gene order to infer phylogenies in the next section.
A note about the sources

In addition to the sources already noted, a number of textbooks [45, 66] and review papers [37, 47, 58, 61, 73, 98] were consulted to gain an understanding of the molecular processes underlying genomic rearrangements.

1.4 WHY DO PHYLOGENETICISTS CARE ABOUT LARGE SHUFFLES

We mentioned in Section 1.1 that some measure of their “relatedness” is needed to be able to place the various taxa on a phylogenetic tree. To obtain this measure of closeness between taxa, scientists make use of shared traits that differ within a group. The fact that the traits are shared asserts the common ancestry of the members of the group. The differences in the traits point to the evolutionary past.

The order of the genes is one such trait that is used as a phylogenetic marker. The differences in the order of genes along a chromosome were used as early as 1938 to identify different strains of Drosophila melanogaster [32]. In order to understand the reasons that make gene order an effective source of phylogenetic signal, we need to understand the challenges associated with teasing out evolutionary history from sequence data.

As already mentioned, molecular sequences are the most important source of phylogenetic information. Any trait or character, and in particular DNA sequences, that is used to derive a phylogeny suffers from problems that obscure the signal in the data. The two problems are masked and convergent evolution.

The first problem is that the evolutionary past is masked because at any instance, a character can be in only one state. For example, a site in a molecular sequence may have changed multiple times, undergoing several insertions or
deletions. It might in fact have reverted to an ancestral state having undergone these changes. Thus any measure of closeness that we assign is likely to be a severe underestimate.

The second problem, convergent evolution, refers to the acquisition of traits for reasons other than common ancestry. The presence of shared traits is used to infer that the groups possessing these traits have descended from a common ancestor. However, in some instances this inference may be incorrect. It is possible for two different traits, especially ones with a small number of possible states, to be the same merely by chance or if natural selection favours a particular state over other possibilities. Perhaps the most well known and striking example of convergent evolution is the presence of wings on birds and bats. Like two mathematicians arriving at the same result independently, the process of evolution and natural selection led both birds and bats to wings as a solution to the problem of flight and contrary to superficial appearances, this solution was not passed down to them by a common ancestor.

The use of more complex genome level features such as gene order alleviates these issues to a large extent [18, 82] because the large number of possible arrangements of genes makes it highly improbable that an arrangement would revert to an ancestral one. For instance, mitochondrial genome content is known to be highly preserved across a number of different species with very few rearrangements [17, 19] and this is why mitochondrial gene order has been used to resolve multiple phylogenetic relationships [72, 76].

1.5 THESIS ORGANISATION AND PROJECT AIMS

The aim of this thesis is to bring the structural approach at the core of algebra to biological problems. Algebraic methods and tools have been used in different fields with great success. Group theory forms an essential part of the
tool-kit of modern chemists and physicists for the study of symmetry. Similarly the study of Lie algebras and their representation theory is intimately connected with quantum mechanics [99]. However, barring a few examples, an algebraic approach has not found a prominent place in biology.

Inferring the evolutionary relationships among the extant life forms is an important problem in biology. As we have discussed in Section 1.4, genome rearrangements can be an extremely valuable source of phylogenetic information and hence, development of mathematical and algorithmic approaches to study large scale rearrangements is an important scientific endeavour.

The field of genome rearrangements has received significant attention in the last few decades. While the language of permutations has been used extensively in this field, the opportunity to use the repertoire of tools from the theory of permutation groups has remained under-exploited. In our work we aim to address this gap. We describe two algebraic models for large scale rearrangement events. The first of these is an algebraic formulation of the popular Double Cut and Join rearrangement operator and will be discussed in Chapter 4. The second model makes a novel use of the theory of rewriting systems to determine weighted rearrangement distance. This work is presented in Chapter 5. Before we discuss these two models however, we first provide a summary of the key algebraic concepts used in this thesis in Chapter 2 and then present in Chapter 3 a brief survey of the current paradigms in the field. A reader familiar with permutations and elementary group theory could skip Chapter 2.
MATHEMATICAL PRELIMINARIES

“Just the place for a Snark!” the Bellman cried,
As he landed his crew with care;
Supporting each man on the top of the tide
By a finger entwined in his hair.
“Just the place for a Snark! I have said it twice:
That alone should encourage the crew.
Just the place for a Snark! I have said it thrice:
What I tell you three times is true.”

The Hunting of the Snark, Lewis Carroll

This thesis uses some standard results on symmetric groups that we collect here for ease of reference. More details on these results can be found in many undergraduate group theory textbooks, for example [39].

A permutation is a bijection from a set $S$ to itself. $S$ is usually taken to be a set of natural numbers $n = \{1, 2, \ldots, n\}$. A permutation can be written by specifying the value of the map on all the points.

Example

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 1 & 6 & 2 & 5 \end{pmatrix}$$
is a permutation on the set \{1, 2, 3, 4, 5, 6\} that sends 1 to 3, 2 to 4, etc. The set of all permutations on the set \(n\) forms a group called the symmetric group and denoted by \(S_n\).

The image of \(i\) under the bijection \(\pi\) is usually written as \(\pi(i)\). In this way of writing, the operand is written on the right and the operator is on the left. It is also not uncommon to write \((i)\pi\) to denote the image of \(i\) under \(\pi\). The difference in the two ways of writing functions and the consequences for permutation models are discussed in detail in Chapter 3. In this chapter, we will use the former notation and write the functions on the left.

### 2.1 Permutation Multiplication

Since permutations are simply bijective functions, permutation multiplication is function composition. That is, to find the image of \(i\) in the product \(\pi_2\pi_1\), we do \(\pi_2(\pi_1(i))\).

**Example** Let \(\pi_1 = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{smallmatrix}\right)\) and \(\pi_2 = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{smallmatrix}\right)\). The image of 1 in the product \(\pi_2\pi_1\) is \(\pi_2(\pi_1(1)) = \pi_2(3) = 4\). So for each \(i\), we have to “follow the string” – \(\pi_1\) send \(i\) to \(j\), \(\pi_2\) sends \(j\) to \(k\), so \(i\) gets sent to \(k\) by \(\pi_2\pi_1\).

\[
\left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{smallmatrix}\right) \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{smallmatrix}\right).
\]

### 2.2 Left and Right Multiplications

The binary operation in a group (usually referred to as multiplication) may “act” on the left or right. Since the distinction between the two kinds of action is important for the development of this thesis, we go into some detail to
clarify this topic. Let $G$ be a group with the binary operation $\ast$. For $\rho, \phi \in G$, we define operators $\ell_\rho$ and $r_\rho$ as follows:

$$
\ell_\rho(\pi) := \rho \ast \pi,
$$

and

$$
r_\rho(\pi) := \pi \ast \rho.
$$

That is, $\ell_\rho$ multiplies $\rho$ on the left with $\pi$ while $r_\rho$ multiplies $\rho$ on the right with $\pi$. Consider the composition of $\ell_\rho$ and $\ell_\phi$.

$$
\ell_\phi(\ell_\rho(\pi)) = \phi \ast (\rho \ast \pi).
$$

On the other hand,

$$
r_\phi(r_\rho(\pi)) = (\pi \ast \rho) \ast \phi.
$$

That is, $\ell_\phi \ell_\rho = \ell_{\phi \rho}$ while $r_\phi r_\rho = r_{\rho \phi}$.

### 2.3 Inverse of a Permutation

Informally, a permutation $\pi \in S_n$ scrambles the elements of $n$. The inverse of $\pi$ is the permutation that “undoes” the scrambling. Formally we define the identity permutation $i$ to be the permutation that maps $i$ to $i$ for all $i \in n$.

**Definition 2.3.1 (Inverse).** Let $\pi \in S_n$. Then the inverse of $\pi$ is the permutation $\pi^{-1}$ such that

$$
\pi \pi^{-1} = i \quad \text{and} \quad \pi^{-1} \pi = i.
$$

If $\pi^{-1}$ is the inverse of $\pi$ then $\pi$ is the inverse of $\pi^{-1}$. That is, $(\pi^{-1})^{-1} = \pi$.

In general, $(\pi_1 \pi_2)^{-1} = \pi_2^{-1} \pi_1^{-1}$.
2.4 CYCLES AND CYCLE DECOMPOSITION

For a permutation $\pi \in S_n$, if we repeatedly apply $\pi$ to any $i \in n$,

$$i \rightarrow \pi(i) \rightarrow \pi^2(i) \rightarrow \ldots,$$

we must eventually (say after $k$ steps) reach $i$ again since $n$ is a finite set. If there is some $j \in n$ which does not occur in this sequence, then we can form a similar sequence for $j$, and keep doing this until every element of $n$ occurs in some sequence.

**Definition 2.4.1 (Cycle).** Let $i_1, i_2, \ldots, i_k$ be $k$ distinct integers in $n$. A cycle $\pi_c$ written as $(i_1, i_2, \ldots, i_k)$ is a permutation in $S_n$ defined as

$$\pi_c(i_s) := \begin{cases} 
i_{s+1} & \text{if } i_s \in \{i_1, i_2, \ldots, i_{k-1}\}, \\
i_1 & \text{if } i_s = i_k, \\
i_s & \text{otherwise}. \end{cases}$$

A 2-cycle is a cycle of length 2. That is, $\pi = (i, j)$ means that $\pi(i) = j, \pi(j) = i$ and $\pi(k) = k$ if $k \neq i, j$. A cycle of length 2 is also called a transposition. The word ‘transposition’ is also used in biology to refer to a rearrangement event that involves the exchange of contiguous intervals of arbitrary length. This thesis does not focus on transposition models and we use ‘transposition’ primarily to refer to 2-cycles in a permutation. The sense in which we are using it would be clear from the context.

Two cycles are said to be disjoint if they have no elements in common.

**Theorem 2.4.2.** Any permutation $\pi \in S_n$ can be written as a product of disjoint cycles.
Example Let $\pi = (\begin{array}{cccccc}1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 1 & 6 & 2 & 5 \end{array})$. $\pi$ can be written as

$$\pi = (1,3)(2,4,6,5).$$

This way of writing a permutation is referred to as cycle notation. There is a unique way of writing a permutation as a product of disjoint cycles, up to the ordering of the cycles (they commute) and cyclic equivalence of each cycle (e.g. $(1,2,3) = (2,3,1) = (3,1,2)$). Since the sizes of the disjoint cycles will always add to $n$ (including if necessary some 1-cycles), we can define the cycle type as follows.

**Definition 2.4.3** (Cycle type). The cycle type of a permutation $\pi$ is the partition $\lambda \vdash n$ whose components are the sizes of the cycles in the disjoint cycle decomposition of $\pi$.

*Example* The cycle type of $\pi = (1,3)(2,4,6,5)$ is $(4,2)$ since it has one cycle of length 2 and one cycle of length 4.

2.5 Conjugation

**Definition 2.5.1.** Let $\pi, g \in S_n$. The conjugate of $\pi$ by $g$ is defined to be the permutation $g\pi g^{-1}$, and we say that $\pi$ and $g\pi g^{-1}$ are conjugate permutations.

**Theorem 2.5.2.** Let $\pi_1$ and $\pi_2$ be permutations on the set $n$, then $\pi_1$ and $\pi_2$ are conjugate in $S_n$ if and only if they have the same cycle type.

2.6 Permutation as product of transpositions

**Theorem 2.6.1.** Any permutation $\pi \in S_n$ can be written as a product of transpositions.
Example The permutation \( \pi = (1,3)(2,4,6,5) \) can be written as

\[
\pi = (1,3)(2,4,6,5) = (1,3)(2,5)(2,6)(2,4).
\]

While the decomposition of a permutation into a product of disjoint cycles is unique, the decomposition of a permutation into a product of transpositions is not unique. However the number of transpositions used must be either always be even, or always be odd.

**Theorem 2.6.2.** A permutation \( \pi \in S_n \) can be expressed as a product of either an even number of transpositions or an odd number of transpositions, but not both.

**Definition 2.6.3.** A permutation is said to even if it can written as a product of an even number of transpositions. Otherwise it is said to be an odd permutation.

### 2.7 Group Action

**Definition 2.7.1.** Let \( G \) be a group and let \( X \) be a set. Then the group \( G \) is said to have a (left) action on the set \( X \) if there is a function \( \phi : G \times X \rightarrow X \) such that the following is true for all elements of \( X \):

- \( \phi(e, x) = x \) where \( e \) is the identity element of the group; and

- \( \phi(gh, x) = \phi(gh, x) \) for all \( g, h \in G \).
MATHEMATICAL MODELS OF GENOME REARRANGEMENT

Could a historiographer drive on his history, as a muleteer drives on his mule,—straight forward;—for instance, from Rome all the way to Loretto, without ever once turning his head aside, either to the right hand or to the left,—he might venture to foretell you to an hour when he should get to his journey’s end;—but the thing is, morally speaking, impossible: For, if he is a man of the least spirit, he will have fifty deviations from a straight line to make with this or that party as he goes along, which he can no ways avoid. He will have views and prospects to himself perpetually soliciting his eye, which he can no more help standing still to look at than he can fly;

Tristram Shandy, Laurence Sterne

While the order of genes along a chromosome was being used to extract phylogenetic information when the field of genetics was its infancy, it wasn’t until almost half a century later that the problem of determining distance between gene arrangements was formalized [97]. Since then, several different approaches ranging from stochastic [54] to the use of genetic algorithms [1] have been applied to this problem. In this chapter, we will survey the combinatorial and graph-theoretic approaches as our work branches off directly from these.
In Chapter 1, we discussed briefly how the different genome arrangements arise due to various large scale changes in the genome. Examples of such changes include the reversal of a segment of the genome (called an inversion or reversal), the movement of a segment to a different location on the genome (transposition), events that split a chromosome in two or join two chromosomes into one single chromosome (fission and fusion) and the exchange of segments between different chromosomes (translocation).

In the study of genome rearrangements, we are interested in the set of genomic regions that are the same across all genomes under study. These conserved regions are called ‘genes’ for simplicity, although they may or may not be genes in the biological sense. The genomes are modeled as being composed of these conserved regions and each genome may correspond to a different arrangement of the regions. It is no surprise therefore that in looking for a list-like construct to represent genomes, researchers used permutations since a permutation is a mathematical abstraction of an arrangement of things.

The set of rearrangement events through which genomes are hypothesised to have changed could consist of a single type of event such as inversion, or a combination of events such as inversion, translocation, fusion and fission. These choices of allowable operations constitute models of rearrangement, in which the genomes in the data are assumed to only change according to specific rearrangement operators being considered.
A rearrangement operator acts on a genome $A$ to give a new genome $B$. A rearrangement operator therefore acts on the set of genomes. A concrete representation of the genomes, for instance as permutations, determines the representation of the rearrangement operator as we will see in Section 3.3.

The philosophies of representing genomes as permutations can be broadly categorised under two headings we call “position” and “content” paradigms. While both paradigms use permutations, the permutations have significantly different meanings in each case, and rearrangement events are consequently implemented differently in each.

The discussion in this chapter is structured around these paradigms. The notation and some results about permutations that are used have been introduced in Chapter 2 but for convenience we will provide brief notes to remind the readers of these as required. Section 3.1 gives a bird’s eye view of the major approaches to the problem of determining rearrangement distances. We then narrow our focus to the aforementioned paradigms and in the three ensuing sections describe in detail for each paradigm respectively how genomes are represented (Section 3.2); how rearrangements are represented (Section 3.3); and how the paradigms are related (Section 3.5).

**About this work**

The discussion presented in Section 3.2 and later sections has been submitted to the *Bulletin of Mathematical Biology* and is available as a preprint at arXiv [16]. The idea for this review arose out of the discussions between Prof Andrew Francis and Dr Pedro Feijão who was visiting Western Sydney University at the time. I was intimately involved in writing this review. For this chapter, I have modified the presentation to align it with the flow of the chapter.
3.1 A BRIEF OVERVIEW OF REARRANGEMENT MODELS

Rearrangement distance between genomes $A$ and $B$ is usually defined as the minimum number of allowed rearrangement events needed to transform $A$ into $B$. When permutations are being used, the target genome can be taken to be the identity permutation and finding the sequence of moves that transform $A$ is also called “sorting” the permutation. The problem of sorting a permutation can be thought of as a combination of two related problems – finding the smallest number of moves that change $A$ to $B$ and finding a sequence of moves that achieve this.

3.1.1 Inversions

Of the different operators referred to in the introduction, inversion as an order-disrupting operator has been the focus of genome rearrangement studies. One of the reasons for this attention is that inversions are believed to be one of the most important agents of genome rearrangement [17, 75]. As our thesis is also focused on inversion models, we will talk about these at greater length than about the other rearrangement operators. In a genome rearrangement model that consists of only inversions as the allowed operation, the inversion distance of a genome $A$ is the minimum number of inversions required to transform it into the identity permutation.

Since the DNA is inherently directional, the reversal of a segment of the genome reverses the order of the genes as well as their direction on a chromosome. A mathematical model however, may or may not incorporate the change in the direction of genes. When it does, the usual way of indicating the direction is by using signed rather than unsigned permutations. A signed permutation is a permutation of the set $\{-n, \ldots, -1, 1, \ldots, n\}$, which satisfies
An unsigned inversion operator reverses the order of the elements in the genome and in the signed version, the order as well as sign of the elements is reversed.

Watterson et al. [97] are credited with setting up the chromosome inversion problem in 1982. Although intuition might suggest that the problem of sorting a linear permutation would be easier than its circular variant, it was first formulated for a circular chromosome. The genome was written as a list of genes around a circle although permutation notation was not used explicitly. The authors proposed an intuitive heuristic for solving the problem which is worst-case optimal.

Sankoff et al. [86] were the first to formalize the inversion distance problem and treat it as a combinatorial optimization problem. Again, their model was not a permutation based model, but they worked with circular genomes (mitochondrial DNA) and a combination of rearrangement events (inversions, transpositions, insertion and deletion).

The first step in modeling a unichromosomal, linear genome as a permutation was taken by Kececioglu et al. [59]. They introduced the notion of a breakpoint which has been used extensively throughout the literature.

Let \( \pi : i \to \pi_i \) be a permutation on \( 1, 2, \ldots, n \). A pair of consecutive positions \( (i, i + 1) \) \( 0 \leq i \leq n \) of \( \pi \) is a breakpoint if \( |\pi_{i+1} - \pi_i| \neq 1 \). Most of the work done in computing rearrangement distance treats the number of breakpoints as a count of obstacles in sorting the permutation. This count has therefore been used to bound the rearrangement distance and there have been some suggestions to use it as a simplistic measure of the distance between the genomes [84].

Returning to the discussion of computing inversion distance, the identity permutation which is treated as the target permutation has 0 breakpoints. So the goal in rearranging the starting permutation through inversions is to min-
Figure 2: Breakpoint graph for the permutation \([3, 1, 5, 2, 6, 4]\). The black edges connect the vertices as they are connected in the permutation. The gray dashed edges connect the vertices as we would like them to be connected i.e., \(i\) connects to \(i+1\).

Imimize the number of breakpoints. Inversion of a section of the permutation \(\pi\) can affect the breakpoints only at the ends of the section being inverted. Thus at every inversion step, the number of breakpoints can reduce by at most 2.

The approximation algorithm proposed by Kececioglu et al. [59] chooses the inversion which removes the most breakpoints of \(\pi\). This algorithm has a guaranteed error bound of 2, that is, the number of inversions found by it is guaranteed to be no more than twice the optimal number.

Most of the later algorithms employ a graph construct called “breakpoint graph” that was introduced by Bafna and Pevzner in [5]. It has been more colorfully labeled as the graph of “desire and reality”. For a permutation \(\pi\) on \(n\), define an edge colored graph \(G(\pi)\) with \(n+2\) vertices \(\{0, 1, \ldots, n+1\}\). We join vertices \(i\) and \(j\) by a black edge if \(i\) and \(j\) occupy neighboring positions in \(\pi\). We join vertices \(i\) and \(j\) by a gray edge if \(i = j+1\) or \(i = j-1\). The black edges depict the reality i.e. they connect the vertices as they are connected in the permutation. The gray edges connect the vertices as in the desired permutation, which is the identity permutation. Figure 2 shows the breakpoint graph for the permutation \([3, 1, 5, 2, 6, 4]\) where the gray edges are shown as dashed.

A cycle in a graph is a sequence of vertices \(\{v_1, v_2 \ldots v_k\}\) such that \(v_k = v_1\) and \(v_{i-1}\) and \(v_i\) are connected by an edge in the graph. In an edge-colored graph such as the breakpoint graph, a cycle is called alternating if the consec-
utive edges have different colors. The key observation is that the number of alternating cycles in a breakpoint graph is maximal when the current genome is the same as the target genome. Using this observation, Bafna and Pevzner [5] were able to establish a lower bound for the inversion distance of a permutation in terms of the number of breakpoints and the number of cycles in its breakpoint graph.

Bafna and Pevzner also extended this analysis to sorting signed permutations, devising an approximation algorithm with a performance guarantee of 1.5. Their strategy for sorting a signed permutation was to transform it into an unsigned permutation. This is done by replacing every positive integer in the signed permutation by an in-order pair of integers and every negative integer by a pair of integers out of order. Let \( \pi : n \to \{-n, \ldots, -1, 1, \ldots n\} \) be a one-to-one function where \(|\pi(i)| \neq |\pi(j)|\) for \(|i| \neq |j|\), that is, each position \(i\) points to a different gene \(\pi(i)\), and therefore \(\pi\) represents a signed chromosome. \(\pi\) can be associated with a permutation \(\pi\) on the set \(2n = \{1, 2, \ldots, 2n\}\) where \(\pi\) is related to \(\pi\) by Equation (3.1.1).

\[
\pi := \begin{cases} 
\pi(2i - 1) = 2\pi(i) - 1, & \pi(2i) = 2\pi(i) \quad \text{if} \quad \pi(i) > 0 \\
\pi(2i - 1) = 2\pi(i), & \pi(2i) = 2\pi(i) - 1 \quad \text{if} \quad \pi(i) < 0.
\end{cases}
\] (3.1.1)

It is important to emphasize that this transformation preserves the inversion distance of \(\pi\) if certain restrictions on positions on which inversions act are respected. Basically, because every label in \(\pi\) is replaced by two labels in \(\pi\), we must ensure an inversion “cuts” only after even positions in \(\pi\).

Hannenhalli and Pevzner [49] improved upon the bound on the inversion distance by identifying another parameter associated with \(\pi\) which they called a ‘hurdle’. They also gave a characterization of permutations that were difficult to sort by inversions. A ‘hurdle’ was identified by constructing an
‘overlap graph’ for a permutation and analyzing the connected components of this graph. Their algorithm was the first polynomial (quadratic) time algorithm for computing the inversion distance between two signed permutations.

The theory proposed by Hannenhalli and Pevzner effectively set the tone for further research on the problem of computing inversion distance. Various authors improved upon their algorithm, for example Berman and Hannenhalli gave a $O(n \alpha(n))$ algorithm ($\alpha$ is the inverse Ackerman function) [14]. Bader et al. [3] gave the first linear time algorithm for this problem, by simplifying the analysis of connected components of the overlap graph. Bergeron et al. [11] gave another linear time algorithm, again by tidying up the associated theory and using efficient data structure to encode the features of a permutation.

3.1.2 Inversions on circular genomes

As already mentioned, the problem of computing inversion distance was first set up for a circular genome. Kececioglu and Sankoff in [86] presented an exact branch-and-bound algorithm for inversion distance on signed circular genomes. A subtlety associated with a circular genome is that rotations and reflections of a circular chromosome do not alter the arrangement of genes. These symmetries therefore need to be taken into consideration in determining the minimal distance between circular genomes. This was done for example in [35, 91, 97] while Chen and Skiena [23] factor in the rotational symmetry only. Meidanis et al. [68] treat any combination of reflections and rotations of a permutation as belonging to an equivalence class and proceed by choosing a canonical representative of the class. The reversal algorithms were run on this canonical representative. A similar attempt was made in [91] which
also established that computing the inversion length of an unsigned circular permutation is NP-hard.

Most of the efforts to sort a circular genome proceed by linearizing the circular chromosome. The distance computations on a linearized circular chromosome can vary depending on the point at which it has been cut open. This raises the question of choosing the point that gives the minimal distance. If on the other hand, we search over all possible cut points, we increase the search space considerably. A more serious drawback of this approach is that in some cases, it does not yield the minimal distance. Egri-Nagy et al. present an example of this in [35] and by lifting the problem from circular permutations to the affine symmetric group, give a polynomial time algorithm for finding the exact distance between circular genomes.

Other rearrangement models

Similar analysis has been brought to bear on computing distances with respect to other rearrangement events as well. The rest of this section contains an abbreviated discussion of the developments related to these various operators.

3.1.3 Translocation

The first of these rearrangement events we discuss is translocation. In computational biology, a translocation event is defined as exchange of chromosome ends within a genome, and hence it makes sense only for multi-chromosomal genomes.

The algorithmic study of sorting by translocations was initiated by Kececioglu and Ravi [60] who gave a 2-approximation algorithm for determining translocation distance. Hannenhalli [48] propose an algorithm for the signed
Their analysis is based on transforming a signed permutation to an unsigned permutation and constructing its breakpoint graph (which is now called a cycle graph) with respect to a target genome. A lower bound on the translocation distance is given in terms of the number of cycles in this graph. The paper gives a polynomial algorithm, which gives a shortest sequence of translocations. However there was an error in their algorithm which was corrected by Bergeron et al. in [12]. There have been continuous improvements in algorithms for sorting by translocations and recently an approximation algorithm for sorting unsigned translocation was proposed which can achieve a performance ratio $1.375$ in polynomial time [80].

3.1.4 Transpositions

Another important agent of genome arrangement variation is transposition. A transposition is defined as the exchange of contiguous intervals of arbitrary length.

By using the cycle graph structure mentioned earlier, Bafna and Pevzner [8] established a lower bound for the transposition distance of a permutation and gave a quadratic time approximation algorithm for sorting by transpositions.

3.1.5 Rearrangement models with multiple operators

Considerable work has been done into considering a combination of mutational operators. For example, a 1.75 approximation algorithm was presented in [67] for sorting signed permutations by inversions and transpositions (i.e. exchange of consecutive segments). Yancopoulos et al. [100] introduced a simple operator that simulates inversions, translocations, fissions and fusions on a genome, depending on the arguments it acts upon and called it the Dou-
ble Cut and Join operator. This model was simplified in [13] and an elegant characterization of the distance between two genomes with respect to the DCJ operator, in terms of the number of cycles and paths in the comparison graph of the genomes was given. This work also devised an algorithm for optimally sorting a multi-chromosomal genome into another. An algebraic formulation of the double cut and join model was independently presented by [38] and by us [15] which we will discuss in greater detail in Chapter 4.

There have been very few models that can accommodate the biological reality of having multiple copies of a gene in the genome. Bader [4] extended their previous algorithms to sort multi-chromosomal genomes using a number of operations, including duplication and loss of arbitrary sized sections. An algorithm for determining the exact DCJ distance between a pair of genomes containing two copies of each gene was presented in [52].

3.2 THE TWO PARADIGMS OF GENOME REPRESENTATION

The reader would have already noticed some recurrent themes such as the analysis of graphs derived from genomes, underlying the investigation of rearrangement distances. Another feature common to the various approaches, as we have emphasised, is the use of permutations to model genomes and rearrangement events. However, the modeling assumptions are often implicit, including assumptions about the way permutations are written down and composed. As a result, researchers often struggle to interpret each others’ works. In the rest of this chapter, we have attempted to bring together the various strands of modeling genomes and genome rearrangement events using permutations, underscoring the commonalities across different approaches and highlighting the differences. We also discuss the relationships between the various models and how to translate one model into another.
3.2.1 The position paradigm – Genomes as maps between positions and regions

The first paradigm we discuss is the explicit use of position to describe a chromosome. This approach has been used widely in the literature beginning with the seminal papers by Bafna and Pevzner [6], Sankoff et al. [86] to recent developments such as Egri-Nagy et al. [35]. The description of a chromosome here consists of a map between positions and regions. While positions are naturally numbered from 1 through $n$, it is common to denote the set of regions also by the integers $1, \ldots, n$, which, while natural, can lead to confusion because the map $\pi$ then looks like a bijection on the set $n$. It is important to realize that, in this genome representation, this map is not a bijection on the set $n$, but a one-to-one correspondence between two different sets: one positions, one regions. Multi-chromosomal genomes are usually modelled as a collection of permutations where each permutation encodes a chromosome (e.g. Hannenhalli and Pevzner [50], Kececioglu and Ravi [60]).

The dominant viewpoint has been to describe a genome as a map from positions to regions [8, 51, 86]. The starting point, adopted widely, is to use the two-line notation described in Chapter 2, to express a genome as a permutation. This involves writing the position numbers along the top of an array, and the region numbers along the bottom:

$$\pi = \begin{pmatrix}
1 & 2 & \cdots & n \\
\pi_1 & \pi_2 & \cdots & \pi_n
\end{pmatrix}.$$

If the positions are labelled in sequence around each chromosome, then the bottom row of the “positions to regions” version is just the labels of the regions read along the genome.
A variation on this way of representing a genome is to write it as a map from regions to positions. This means that the regions are listed in the top row and the positions in the bottom row. The two representations produce permutations that are inverses of each other: one maps a position to the region that is in that position, and the other maps a region back to its position.

The representation as a map from regions to positions is not as widely adopted as the map from positions to regions. An example where the former representation was used is Watterson et al. [97], although this work does not make explicit use of permutations. Similarly Egri-Nagy et al. [35] also write the arrangements with the set of regions constituting the domain and the positions as the co-domain.

Note that this paradigm of genome representations requires referring to an absolute position for each region. This must be chosen a priori in order to write down the genome. This choice must then be taken into account when considering the distance between two genomes, as mentioned in Section 3.3.2.

3.2.1.1 Incorporating orientation

To incorporate the orientation of DNA into the models, various approaches have been used. The most common is to use signed permutations, where the sign of a region represents its orientation. We touched upon signed permutations in Section 3.1. Recall that a signed permutation is a permutation of the set \{-n, \ldots, -1, 1, \ldots, n\}, with a constraint that \(\pi_{-i} = -\pi_i\).

For instance, the genome
has two-line form

\[
\text{pos} \rightarrow \text{reg} : \begin{pmatrix}
  -6 & -5 & -4 & -3 & -2 & -1 & 1 & 2 & 3 & 4 & 5 & 6 \\
  -5 & 6 & 3 & -1 & 4 & -2 & 2 & -4 & 1 & -3 & 6 & 5 
\end{pmatrix}
\]

which in cycle notation form is

\((1, 2, -4, 3)(-1, -2, 4, -3)(5, 6)(-5, -6)\).

Note that the cycles come in pairs, and given the constraint that \(\pi_i = -\pi_i\), one of each is redundant. Consequently, this permutation may be abbreviated to \((1, 2, -4, 3)(5, 6)\). The representation in the “reg \(\rightarrow\) pos” form is the inverse of this permutation.

Another way of dealing with orientation is to translate the problem into the realm of unoriented or unsigned permutations as discussed in Section 3.1.

3.2.2 The content paradigm – Genomes as maps from regions to regions

The second modeling approach used widely is to view a chromosome as a map from the set of regions to itself. Within this idiom, two broad categories emerge – representing genomes as disjoint cycles, and using adjacencies.

3.2.2.1 Genomes as cycles

This view of a chromosome was first formalised in terms of permutations by Meidanis and Dias \[69\] who use the cycle structure of a permutation to capture a circular chromosome. The notion of \(i\) being mapped to \(j\) in a cycle is interpreted as region \(i\) being followed by region \(j\) on the chromo-
some. Therefore, a circular chromosome is represented by the single cycle \( \pi = (\pi_1, \pi_2, \ldots, \pi_n) \). Multi-chromosomal genomes are represented by disjoint cycles, one for each chromosome. Meidanis and Dias also used signed permutations to model orientation, representing each chromosome by two cycles, one for the direct orientation and other for the reverse complement, with the property that \( \pi_{-i} = -\pi_i^{-1} \) (note this is different from the convention for signed permutations described in Section 3.1).

For instance, the circular signed chromosome from earlier in this section, reproduced here

![Circular chromosome diagram](image)

is modelled by

\[
\pi = (2, -4, 1, -3, 6, 5)(-5, -6, 3, -1, 4, -2).
\]

This representation has been used to study some rearrangement distances, such as fission, fusion and transposition [29], block-interchange [57], and 2-break operations [38].

Although multi-chromosomal genomes are easier to write down, when compared to the “positional” permutations, it is clear that this formalism is more suitable for dealing with circular genomes than linear genomes, since chromosomes are permutation cycles.

3.2.2.2 Genomes as adjacencies

In order to develop a permutation model that allows for multi-chromosomal genomes, with both linear and circular chromosomes, it was necessary to use an alternative formulation, where the focus is shifted from the ordering of
the genes to the connections between genes. This model became one of the most common ways of representing genomes in the combinatorial community (e.g. Bergeron et al. [13], Tannier et al. [94]). A gene is defined as an oriented section of the DNA, and its two ends are called its extremities. To represent a genome, considered as an arrangement of oriented genes, it is sufficient to note which extremities are adjacent on the genome. An (unordered) pair of extremities that are adjacent is referred to as an adjacency. An extremity that is not adjacent to any other is the end point of a linear chromosome and is called a telomere.

This model can easily describe a multi-chromosomal genome uniquely by its set of adjacencies and telomeres, even when the genome contains both linear and circular chromosomes. The genome graph is a graph where the vertices are the adjacencies and telomeres of a genome (sets of one or two gene extremities), and for each gene there is a directed edge from the vertex with the tail of the gene to the vertex with the head of the gene. A path in this graph corresponds to a linear chromosome, and a cycle to a circular chromosome. It is easy to recover the gene order and orientation in this graph by traversing the components, labelling the edges with the gene label and a plus or minus sign, depending on the direction of the traversal and edge orientation. In the rest of this chapter, we refer to this genome representation as the adjacency list model.

A genome graph such as that in Figure 3 and the intuitive pictorial presentation of a genome we have used earlier in this chapter illustrate the two alternative representations of a genome – as a map between regions or a map between positions and regions.

An algebraic formulation of the adjacency list model was presented by Feijão and Meidanis [38] and independently by Bhatia et al. [15]. In the algebraic framework, the gene extremities may be represented by the set of signed in-
Figure 3: The genome graph of a genome with one linear chromosome containing genes numbered 1, 2, 3 and 4. The vertex set of the graph is the set of adjacencies and extremities, \( \{1_t, 1_h, 3_t, 3_h, 2_t, 2_h, 4_t, 4_h\} \). Directed edges are drawn from the tail to the head of the same gene. To express this chromosome as a permutation \( \pi \), the set of extremities \( \{1_t, 1_h, 2_t, 2_h, \ldots, 4_h\} \) is mapped into \( \{1, 2, \ldots, 8\} \) (Bhatia et al. [15]) or into \( \{+1, -1, \ldots, +4, -4\} \) (Feijão and Meidanis [38]). \( 1_h \) is connected to \( 3_h \) which is captured by the 2-cycle \( (2\ 6) \) in the permutation encoding. The other 2-cycles can be similarly interpreted. The above genome is thus encoded as the permutation \((2, 6)(5, 3)(4, 7)\) or \((−1, −3)(3\ 2)(−2\ 4)\).

tegers \( \{-n, \ldots, -1, 1, \ldots n\} \) [38] or by mapping the set of \( 2n \) gene extremities into the set \( \{1, 2, \ldots, 2n\} \) [15]. An adjacency is represented as a 2-cycle and a genome is expressed as a product of disjoint 2-cycles. 1-cycles implicitly represent the telomeres and are usually omitted, since they represent fixed points in the permutation. The translation from a a graph-theoretic to an algebraic representation is dealt with in detail in Chapter 4.

Figure 3 shows a genome graph consisting of a linear chromosome, together with its permutation representation as an adjacency list.

### 3.3 Genome Rearrangement Events

So far in Section 3.2, we have seen that a genome may be viewed as a map between sets of positions and regions, or from a set of regions to itself. In the introduction to this chapter, we also alluded to the fact that a rearrangement event and a genome should be composable as functions. We now elaborate this point further and show how the choice of genome representation determines the order in which the rearrangements act on the genome but before
we can do that, we take a brief digression into a discussion of group actions and how they are written, as related to the topic of rearrangement events.

3.3.1 *Actions*

The set of all permutations on \( n \) regions constitutes the symmetric group \( S_n \). Thus when permutations model genome rearrangement events, we refer to the composition of rearrangement events as group multiplication. The rearrangement events act on a collection of permutations that encode genomes and we will often talk of ‘group actions’ when speaking of a rearrangement event acting on a genome.

The first complexity in using permutations in this context comes when one chooses whether the permutation acts on the *left* or on the *right*. While it is common in much of mathematics to think of functions acting on the *left*, so that we might write \( f(x) \) for \( f \) acting on an element \( x \) of the domain, it is also common in the study of group actions to write functions acting on the *right*, so that we might write \( (x)f \) or simply \( xf \) for \( f \) acting on \( x \).

The difference between the two ways of writing becomes more prominent when considering function composition: when acting on the left, \( fg \) means we “do \( g \) first”, while when acting on the right it means “do \( f \) first”, which accords with the way we read (in “left-to-right” languages).

Both these conventions appear in the literature on genome rearrangements, and so we will describe both, taking care to make it clear at each point which convention is in play.

The choice of whether our permutations act on the right or on the left has a bearing on the implementation of the rearrangement events as permutations as we will see in this section.
3.3.2 Rearrangements in the position paradigm

We begin with the position paradigm. A rearrangement in this paradigm is an action on the *positions* of the genome. So, for instance, an inversion swapping adjacent positions 2 and 3 swaps those two positions regardless of what regions are in those positions. This is really the logic behind this paradigm, as an operation that acted on *regions* 2 and 3 would affect those two regions regardless of where they physically were located on the genome (this is a feature of rearrangements in the “content” paradigm, described in Section 3.3.3).

Thus, in this view of modeling genomes, a rearrangement event is a permutation on the set of positions.

The observation that the position paradigm involves a map between positions and regions, while a rearrangement is a map from positions to positions, imposes constraints on how they may be composed. When a genome is written as a map from positions to regions, the rearrangement event that maps positions to positions, must always act *first* for the composite map to make sense as a genome. Similarly, if the genome maps regions to positions, then the rearrangement operator must act second so that the composite map represents a genome. Figure 4 illustrates this argument and Table 1 lists the permissible compositions.

The complexity becomes two-fold when more than one rearrangement acts in succession on a genome. A rearrangement operator may act from the left or the right on a genome and the rearrangement operators themselves may be composed with each other from the left or the right. This leads to four different combinations of these two different function compositions. These considerations affect the way in which new actions are tacked on an existing sequence of operators.
(a) If the genome is a map $\pi$ from positions to regions, the rearrangement event $\rho$ acts first (written as $\pi \rho$ in right-to-left notation).

(b) If the genome $\pi$ is a map from regions to positions, the rearrangement operator $\rho$ acts second (written as $\rho \pi$ in the right-to-left notation).

Figure 4: $\rho$ is a rearrangement operator that maps positions to positions and $\pi$ is a genome. If our functions act on the left so that we write $f(x)$ for $f$ acting on $x$, then in the first case (Fig 4a), we must write $\pi \rho$ while in the latter case (Fig 4b), we write $\rho \pi$. For functions acting on the right, the rearrangement event corresponds to the permutations written in reverse order. Also see Table 1.
Table 1: Correct ways to compose a rearrangement such as inversion ("inv") with a genome (permutation). The constraints on the action arise because of the needs for codomain of the first function to match the domain of the second function. For example, after applying a genome permutation representation that takes positions to regions, one cannot then act by a rearrangement function that takes positions to positions.

<table>
<thead>
<tr>
<th>Genome Map</th>
<th>Composition</th>
<th>Legal action</th>
</tr>
</thead>
<tbody>
<tr>
<td>reg → pos</td>
<td>([reg]) (reg → pos) · inv</td>
<td>Only on the right</td>
</tr>
<tr>
<td>pos → reg</td>
<td>(reg ← pos) · inv ([pos])</td>
<td>Only on the left</td>
</tr>
<tr>
<td>pos → reg</td>
<td>([pos]) inv · (pos → reg)</td>
<td>Only on the right</td>
</tr>
<tr>
<td>reg → pos</td>
<td>inv · (pos ← reg) ([reg])</td>
<td>Only on the left</td>
</tr>
</tbody>
</table>

As an example of this, consider the case where the genome is a map from positions to regions, the rearrangement operators multiply from right to the left, and the rearrangement operator acts on a genome on the left. In this set-up when we have an operator $\rho_1$ acting on the genome $\pi$, we write $\rho_1(\pi)$. Now if want $\rho_2$ to act after $\rho_1$, we must write this as $(\rho_1 \rho_2)(\pi)$ since in the first set of parenthesis, the multiplication is from right to left. On the contrary, if both the group multiplication and the rearrangement action are from right to left, the same sequence is written as $(\pi)(\rho_1 \rho_2)$. In other words, if the side of the group action and the side the rearrangement acts on the genome clash (one right, one left), new rearrangements must be added into the middle of an expression when composing rearrangement operators (see Table 2). The reader might find, as we do, that the latter way of writing is easier to understand and more transparent in conveying meaning.

3.3.3 Rearrangements in the content paradigm

When a genome is modeled as a map from a set of regions to itself, a rearrangement event must be modeled as a similar map and may act on either
Table 2: Different ways to represent the genome and the action of rearrangements, with a focus on the composition of rearrangement operators. In the (right hand) Structure column the placement of an additional rearrangement $\rho_{n+1}$ is shown in bold. Note that when the side of the group action and the side the rearrangement acts on the genome clash (one right, one left), new rearrangements must be added into the middle of the expression. This is not ideal, and why we recommend use of the structures in the first and last rows, both involving representations of genomes mapping regions to positions.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Side of Structure</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{group action}$</td>
<td>$\text{rearrangement action}$</td>
<td>$\pi \cdot \rho_1 \cdots \rho_n \rho_{n+1}$ (x is a region)</td>
</tr>
<tr>
<td>$\text{right}$</td>
<td>$\text{right}$</td>
<td>$\pi \cdot \rho_{n+1} \rho_n \cdots \rho_1(x)$ (x is a position)</td>
</tr>
<tr>
<td>$\text{left}$</td>
<td>$\text{right}$</td>
<td>$\rho_1 \cdots \rho_{n+1} \pi$ (x is a region)</td>
</tr>
<tr>
<td>$\text{left}$</td>
<td>$\text{left}$</td>
<td>$\rho_{n+1} \rho_n \cdots \rho_1 \cdot \pi(x)$ (x is a position)</td>
</tr>
</tbody>
</table>

left or right. The effect of left and right actions will of course be different, but both these formulations have been used in the literature.

Consider for example the model of a circular genome as a cycle [69] where $i$ is mapped to $j$ if region $i$ is adjacent to region $j$ on the genome (Section 3.2.2.1). Suppose the genome is represented as the $k$-cycle $\pi$, and $u$ is a region on the genome.

A transposition event on $\pi$ that exchanges blocks of length $i$ and $j$ starting at region $u$, is exchanging the block of length $i$ from $u$ to $\pi^{i-1}(u)$ with the block of length $j$ from $\pi^i(u)$ to $\pi^{i+j-1}(u)$ (with $i+j<k$).

To implement this as permutation multiplication, we simply multiply the cycle $\rho = (u, \pi^i(u), \pi^{i+j}(u))$ on the left of $\pi$ (thinking of the permutations as acting on the left).
For instance, for the genome representation $\pi = (1, 2, 3, 4, 5, 6)$, the cycle $\rho = (2, 4, 5)$ will transpose regions 2 and 3 with region 4 (recalling here we are acting on the left):

$$\rho \pi = (2, 4, 5)(1, 2, 3, 4, 5, 6) = (1, 4, 2, 3, 5, 6).$$

In this system, the permutation $\rho$ encoding the transposition event depends on the permutation $\pi$ (the genome) it is acting on. This was commented on by Meidanis and Dias [69] who note that this makes the problem of determining the rearrangement distance different from the group theoretic problem of determining the word length of a group element under a (fixed) generating set (more on this in Section 3.4).

### 3.3.4 Rearrangement operators on adjacency lists

A popular operator acting on the adjacency list representation of a genome is the “double cut and join” (DCJ) operator introduced by Yancopoulos et al. [100] and Bergeron et al. [13]. The popularity of this operator can be attributed to its simplicity and the fact that it is able to simulate most of the common rearrangement events that have been observed in comparison of genomes.

Recall the definition of a genome graph from Section 3.2.2. A DCJ operator acts on a genome graph by choosing two adjacencies at which to “cut” the genome graph. The four cut extremities may now be joined in one of two possible ways, each way resulting in a different rearrangement event on the genome. Other possibilities are: to cut one adjacency and join one of the extremities with a telomere; to cut one adjacency into two telomeres; or the inverse operation of joining two telomeres. All the possible actions of
Figure 5: The double cut and join operator can simulate several rearrangement events. In this figure, a double cut is applied at adjacencies $1_h3_t$ and $2_h4_t$, and these extremities are joined to form $1_h2_h$ and $3_t4_t$. In the chromosome at the bottom, the segment between $1_h$ and $4_t$ is reversed.

The DCJ operator are explained in detail in Chapter 4 (see Section 4.2 and Definition 4.2.1).

Figure 5 shows a signed reversal of a section being obtained through the double cut and join operator.

In Section 3.2.2, we also discussed an equivalent description of a genome as a product of disjoint 2-cycles. Recall (Section 3.2.2.2) that in the algebraic framework, the gene extremities may be represented by the set of signed integers $\{-n, \ldots, -1, 1, \ldots n\}$ [38] or by mapping the set of $2n$ gene extremeties into the set $\{1, 2, \ldots, 2n\}$ [15]. An adjacency is represented as a 2-cycle and a genome is expressed as a product of disjoint 2-cycles.

In this representation, a double cut and join operation takes the form of multiplication by two 2-cycles. Feijão and Meidanis [38] for example present it as multiplication on the left by two 2-cycles while Bhatia et al. [15] formulate the double cut and join operation as conjugation by a 2-cycle. For instance, consider a linear chromosome with adjacencies $(i, j)$ and $(k, l)$. A DCJ that cuts these adjacencies and joins the adjacencies $(i, k)$ and $(j, l)$ can be obtained by conjugating the genome $\pi$ by the 2-cycle $(k, j)$, since
\[(j, k)[(i, j)(k, l)](j, k) = (i, k)(j, l).\]

The same effect can be obtained by left or right multiplication by two 2 cycles,
\[(i, l)(j, k)[(i, j)(k, l)] = (i, k)(j, l),\]
and equivalently,
\[[ (i, j)(k, l)](j, k)(i, l) = (i, k)(j, l).\]

In fact, there is a simple relationship between these two ways of applying a DCJ operator. In general, when multiplied on the left of a genome \(\pi\), a DCJ operation is of the format \((u, v)(\pi u, \pi v)\) [38]. It is easy to see that \((u, v)(\pi u, \pi v)\pi = (u, v)\pi(u, v)\), which is the DCJ formulation of Bhatia et al. [15]. The effect of the operator on the genome depends on where the regions \(u\) and \(v\) lie on the genome. The algebraic formulation of the DCJ operator (Bhatia et al. [15]) constitutes a major contribution of the present thesis and has been presented in detail in Chapter 4.

Feijão and Meidanis [38] generalise this operator to a \(k\)-break operator, where \(k\) adjacencies are cut, and \(k\) new ones are created from the \(2k\) free extremities. As a left operator, the general form of the \(k\)-break operator is \((a_1, a_2, \ldots, a_k)(\pi(a_k), \pi(a_{k-1}), \ldots, \pi(a_1))\), which means that a DCJ operation is a 2-break, the special case for \(k = 2\). Interestingly, the \(k\)-break operator can also be seen as a conjugation of a genome \(\pi\) by a \(k\)-cycle:

\[(a_1, a_2, \ldots, a_k)(\pi(a_k), \pi(a_{k-1}), \ldots, \pi(a_1))\pi = (a_1, a_2, \ldots, a_k)\pi(a_k, a_{k-1}, \ldots, a_1).\]
Given two genomes \( \pi \) and \( \sigma \), we want to find a sequence of rearrangement operations that minimally transforms \( \pi \) into \( \sigma \). If we consider rearrangements applied on the left (see row 3 of Table 2), we want to find \( \rho_1, \rho_2, \ldots, \rho_k \) such that

\[
\rho_k \rho_{k-1} \cdots \rho_2 \rho_1 \pi = \sigma,
\]

and \( k \) is minimal. The rearrangement distance between \( \pi \) and \( \rho \) is defined as \( d(\pi, \rho) = k \).

If the rearrangement operators are invertible, then they generate a group. If starting with a single permutation (genome), say the identity permutation, all (unsigned) permutations of the genes can be obtained by the repeated application of the operators under consideration, then the operators generate the symmetric group. Otherwise, the operators generate a subgroup of the symmetric group i.e., a group structure that is a subset of the symmetric group.

In the position paradigm of genome representation, the problem of calculating the rearrangement distance is equivalent to finding the length of a reduced word for a group element with respect to a fixed generating set. To see this, recall from above that the distance problem amounts to having two genomes, written \( \pi \) and \( \sigma \), and wanting to know the minimal number of rearrangements \( \rho \) such that \( \rho_k \rho_{k-1} \cdots \rho_2 \rho_1 \pi = \sigma \). The properties of group multiplication allow us to solve this by considering \( \sigma \pi^{-1} \) (now a map from positions to positions), and factorising it in terms of the rearrangement operators. This is precisely the problem of finding a minimal expression for a group element in terms of the generators of the group. Various researchers have noted the connection between the problems of determining rearrangement distance and
finding a minimal word for a group element (e.g. see Bafna and Pevzner [6, 8], Egri-Nagy et al. [35]).

3.5 Going from one genome representation to the other

As can be expected, the different representations of a genome described in Section 3.2 are related to each other through mathematical transformations and in this section, we will show how to translate genomes from one representation to another. Unless otherwise noted, the rearrangement operator is acting on the left. That is, we write $\rho(\pi)$ for a rearrangement operator $\rho$ acting on a genome $\pi$. Clearly, all results could be rewritten with right acting compositions.

To the best of our knowledge, our presentation in this thesis is the first explicit description of the relationship between different genome representations and how to move from one paradigm to another.

3.5.1 From position to content representation and back

If $\pi$ is a genome in the positions-to-regions representation (the position paradigm in Section 3.2.1), then we can transform it into a unique content representation $\sigma$ by conjugating the $n$ cycle $(1, 2, \ldots, n)$ by $\pi$:

$$\pi(1, 2, \ldots, n)\pi^{-1} = \sigma,$$

where $\sigma$ is the genomes-as-cycles content representation from Section 3.2.2.1, and where we write our permutations acting on the left (acting on the right swaps $\pi$ and $\pi^{-1}$). This transformation can be understood as follows. Conju-
gating a $k$-cycle by an element $g$ of $S_n$ where $S_n$ is the symmetric group on a set of size $n$, results in applying $g$ to every element of the cycle. That is,

$$g(i_1, i_2, \ldots, i_k)g^{-1} = (g(i_1), g(i_2), \ldots, g(i_k)).$$

Thus, when $\pi$ conjugates the cycle $(1 \ 2 \ \ldots \ n)$, we obtain

$$\pi(1, 2, \ldots, n)\pi^{-1} = (\pi(1), \pi(2), \ldots, \pi(n))$$

which is exactly the “genomes-as-cycles” content representation.

A similar transformation can be used for signed permutations. If $\pi$ is a signed permutation in the positions-to-regions representation, satisfying $\pi_{-i} = -\pi_i$ (from Section 3.2.1.1), then the signed permutation $\sigma$ in the genomes-as-cycles representation is obtained by

$$\pi(1, 2, \ldots \ n)(-n, -n + 1, \ldots, -1)\pi^{-1} = \sigma.$$

To go from the genomes-as-cycles to the positions-to-regions representation, note that in the genomes-as-cycles representation the information about the position of each region is available relative to the other regions. Since the labeling of the positions can begin at any region, it is easy to see that multiple positions-to-regions permutations will map into the same genomes-as-cycles permutations.

For example, consider the cycle $\sigma = (1, 2, 3, 6, 5, 4)$. Viewed as a list of adjacent regions, it encodes the chromosome in Figure 6. Any of the six regions can be thought of as being at position 1, and this assignment then determines the position-to-regions permutation. See Figure 7 for the permutations that correspond to $\sigma = (1, 2, 3, 6, 5, 4)$. The first genome $\pi$ can be obtained from $\sigma$ by reading $\sigma$ as the bottom row of a two-line notation (instead of as a cycle),

56
Figure 6: The circular genome encoded by the cycle $\sigma = (1, 2, 3, 6, 5, 4)$ in the content paradigm.

\[
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 6 & 5 & 4
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
4 & 1 & 2 & 3 & 6 & 5
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
5 & 4 & 1 & 2 & 3 & 5
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
3 & 6 & 5 & 4 & 1 & 2
\end{pmatrix}
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
2 & 3 & 6 & 5 & 4 & 1
\end{pmatrix}
\]

Figure 7: The six permutations that encode the genome in Figure 6 as a map from positions to regions, obtained by rotating the labelling of positions. The cyclic (content paradigm) notation for the permutation encoded by these permutations is $(1, 2, 3, 6, 5, 4)$.

and all the others can be obtained by multiplying $\pi$ on the right by successive powers of the $n$-cycle $(n, n - 1, \ldots, 2 1)$, which corresponds to a “rotation” on the permutation $\pi$. There are another $n$ labellings obtained by counting round the circle in the opposite direction (omitted here).

This multiplicity of representations arises from the symmetry inherent in a circular chromosome. Every rotation of a circular arrangement with respect to a fixed reference is equivalent in terms of the arrangement of genes on the chromosome. In fact this symmetry needs to be accounted for while determining rearrangement distance between a pair of circular chromosomes as has already been noted (3.1).
3.5.2 From adjacency lists to chromosomes

In the content representation, we saw that there are two interpretations for the cycles of a permutation: the original algebraic formulation of Meidanis and Dias [69], with cycles corresponding to chromosomes (Section 3.2.2.1), and the adjacency list representation [15, 38], with cycles as adjacencies (Section 3.2.2.2). Although seemingly different, there is a direct relationship between both formulations. Consider the set of signed integers $\{-n, \ldots, -1, 1, \ldots, n\}$, and the permutation $\Gamma = (-1, 1)(-2, 2) \cdots (-n, n)$. Then, going from chromosome to adjacency lists and back is achieved with a right multiplication by $\Gamma$ [38] (with permutations acting on the left). For instance, the circular signed chromosome shown in Figure 8 is modelled by $\pi_c = (2, -4, 1, -3, 6, 5)(-5, -6, 3, -1, 4, -2)$. Recall from Section 3.2.2.1 that when signed permutations are used to model gene orientation, each chromosome is represented by a product of two disjoint cycles. Multiplying by $\Gamma$ on the right we get the adjacency representation $\pi_a$:

$$\pi_c \Gamma = (2, -4, 1, -3, 6, 5)(-5, -6, 3, -1, 4, -2)(-1, 1)(-2, 2) \cdots (-6, 6)$$
$$= (-2, -4)(4, 1)(-1, -3)(3, 6)(-6, 5)(-5, 2)$$
$$= \pi_a$$

where negative and (omitted) positive signs represent the head and tail of each gene, respectively. Note that the translation from a set of extremities
\{i_{\mu}, i_{\nu}\} to a set of integer labels assumes an implicit assignment between the two sets. This step will be made explicit in a later chapter on the DCJ operator (Chapter 4).

Since \(\Gamma = \Gamma^{-1}\), this transformation works in both directions. It is also possible to apply this transformation for multi-chromosomal genomes, with both linear and circular genomes.

<table>
<thead>
<tr>
<th>Positional</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\pi_p) (pos (\rightarrow) reg) (\pi_c = \pi_p \cdot I \cdot \pi_p^{-1})</td>
<td>(\pi_c) (genomes as cycles)</td>
</tr>
<tr>
<td>(\pi_p = \pi_r^{-1})</td>
<td>(\pi_c = \pi_a \Gamma)</td>
</tr>
<tr>
<td>(\pi_r) (reg (\rightarrow) pos)</td>
<td>(\pi_a) (genomes as adjacencies)</td>
</tr>
</tbody>
</table>

Figure 9: The relationship between permutations arising from different genome representations. \(I = (1, 2, \ldots, n)\) for the unsigned case, or \(I = (1, 2, \ldots, n)(-n, -n + 1, \ldots, -1)\) for signed. Note that from \(\pi_p\) to \(\pi_c\) the translation is unique, while the opposite case has multiple possibilities (because there is choice in the labelling of positions). Also, the transformation from \(\pi_c\) to \(\pi_a\) is only possible — or even the modelling of genomes as adjacencies in general — for the signed (oriented) case. \(\Gamma\) is the permutation \((-1, 1)(-2, 2) \cdots (-n, n)\). All products here assume permutations act on the left.

3.5.3 *Combined approaches*

We argue that some approaches in the genome rearrangement literature can, in fact, be interpreted as a combination of the positional and the content paradigms. These approaches are based on interpreting the cycle graph of a permutation with an algebraic approach.

The *cycle graph* of a permutation \(\pi\), proposed by Bafna and Pevzner [8], is a widely used graph where many results on genome rearrangement prob-
lems have been obtained. Labarre [64], extending results from a previous paper [33], proposed a framework that takes a permutation \( \pi \), representing a genome in the position paradigm, and applies a transformation \( f \), that is an algebraic representation of the cycle graph of \( \pi \). The transformation \( f(\cdot) \) in question is given by

\[
f(\pi) = (0, 1, \ldots, n)(\pi_n, \pi_{n-1}, \ldots, \pi_1, 0)
\]

where \( \pi_i = \pi(i) \), and a new element 0 is added, to conform with the definition of the cycle graph, where this element is also included. Clearly \( f(\pi) \) is always an even permutation, and the authors show that the number of cycles of \( f(\pi) \) is the same as the cycle graph of \( \pi \), which allows them to find new rearrangement distance results based on the number of cycles and on decompositions of \( f(\pi) \).

In the content paradigm as defined by Meidanis and Dias [69], if we want to transform a genome \( \pi \) into another genome \( \sigma \), we need operations \( \rho_1, \rho_2, \ldots, \rho_k \) such that

\[
\rho_k \cdots \rho_2 \rho_1 \pi = \sigma.
\]

Multiplying on the left by \( \pi^{-1} \), we get

\[
\rho_k \cdots \rho_2 \rho_1 = \sigma \pi^{-1},
\]

which means that decomposing of the permutation \( \sigma \pi^{-1} \) is a way to obtain the rearrangement events, and the number of cycles of \( \sigma \pi^{-1} \) is related to many rearrangement distances (e.g., [15, 29, 38]). Now, if we complete the genomes with a 0 element, we have that \( \pi = (0, \pi_1, \pi_2 \ldots, \pi_n) \) and let the target genome \( \sigma \) be the genome where all the elements are “sorted”, that is, \( \sigma = (0, 1, 2, \ldots, n) \). Therefore, the permutation \( \sigma \pi^{-1} \) is exactly \( f(\pi) \) as
defined by Labarre. In this sense, the transformation proposed by Labarre can be thought of as acting as a translation between the position and content paradigms.

3.6 Summary and Conclusions

This chapter surveyed the landscape of genome rearrangement literature bringing together the various permutation models used in this field. The first of these paradigms models a chromosome as a map between a set of positions and a set of regions and the latter two model a genome as a map from a set of regions to itself.

In the position representation of a genome, a rearrangement operator is a map from the set of positions to itself. We saw that in this view an operator can have a constant form irrespective of the genome it is acting on. This can simplify the problem of genome rearrangement in some cases, for example if we want to apply a fixed operator such as reversal operator to a genome.

When a genome is viewed as a map from the set of regions to itself, genome rearrangement operators are also interpreted as maps from the set of regions to itself. This view of a genome has offered great modeling and algorithmic simplicity although this approach has some drawbacks.

A primary goal of the current thesis is to employ the theory of permutation groups in the field of genome rearrangements. To that end, we have translated the adjacency list model into an algebraic framework and in the next chapter, we will expand on our work on the algebraic Double Cut and Join model.
A couple of hours later he had the answer, or at least some kind of an answer. Nothing that went so far as to make any kind of actual sense, but enough to make Dirk feel an encouraging surge of excitement: he had managed to unlock a part of the puzzle. How big a part he didn’t know. As yet he had no idea how big a puzzle he was dealing with. No idea at all.

The Salmon of Doubt, Douglas Adams

4.1 INTRODUCTION

The double cut and join (DCJ) operator, introduced by Yancopoulos et al. [100] (see also Bergeron et al. [13]), provided an important simplification of rearrangement models with multiple operators acting on a more general, multi-chromosomal genome. The genome is modeled as a list of adjacencies and hence, a multi-chromosomal genome that includes both linear and circular genomes can easily be described (See Chapter 2). By choosing the points on a genome at which it acts, the DCJ operator can simulate a large number of
rearrangement events. The DCJ distance can be expressed in a very simple formula based on the features of a graph derived directly from the genome arrangements.

While the DCJ operator treats all operations in its remit as equally likely, it is possible that this operator may provide a valuable base for operators that account for differences in frequency of different operations. The DCJ operator has also been specialised to genomes with specific chromosomal structure. For instance Hasić and Gavranović [52] use the exact DCJ distance between genomes consisting of circular chromosomes in which every gene appears twice, while Shao et al. [89] give an exact algorithm to determine DCJ distance in the generalised case where genomes may contain both linear and circular chromosomes and a gene may be present multiple times.

In this chapter we present our work in translating the theory of DCJ into an algebraic setting. We show that by expressing the multi-chromosomal genome with $n$ oriented regions as a permutation of $\{1, \ldots, 2n\}$, a DCJ operator can be defined as an action on a permutation. Hence, the set of double cut and join operators generates a group acting on the entire genome space. The DCJ distance is then a path distance on the Cayley graph of this group (as described in Egri-Nagy et al. [35]). The DCJ distance between two genomes can be obtained in a very simple way from the permutation encoding of the genomes. This is derived independently of the established DCJ theory.

ABOUT THIS WORK

The work presented was carried out in collaboration with Andrew Francis and Attila Egri-Nagy and has been published in [15]. All the proofs in this chapter were worked out by the author of this thesis.
The presentation of our work in this thesis differs significantly from that in the published paper. Nearly all the proofs are more detailed and the counting of sorting scenarios for non-conjugate permutations presented in Section 4.7.2 did not form a part of the published paper.

4.2 THE DOUBLE CUT AND JOIN MODEL

In the discussion of the various genome models in Chapter 3, we discussed the use of adjacency lists in the content paradigm of genome representation as well as the DCJ model introduced by Bergeron et al. [13]. To make this chapter self-contained, we briefly discuss the adjacency list representation and the DCJ operator again. Our presentation differs slightly from the presentation in [13] and we will highlight these distinctions in our discussion.

4.2.1 Genome as a graph

In the context of rearrangement models, a gene is essentially an oriented section of the DNA and its two ends are sometimes called its extremities. One way to represent a genome is to note which gene extremities are adjacent on the genome. An extremity that is not adjacent to any other is the end point of a linear section of the genome and is called a telomere. An (unordered) pair of extremities that are adjacent on the genome is referred to as an adjacency. For instance, the adjacency \{a_t, b_h\} indicates that the tail of gene a is adjacent to the head of gene b on the genome. Note that an extremity can be adjacent to at most one other extremity.

Formally, a genome \(G\) is then represented by a partition of the set of extremeties of all the genes in the genome into subsets of cardinality 1 (telomeres) or 2 (adjacencies).
Bergeron et al. [13] define a genome graph for a genome $G$ as follows: the vertex set of the genome graph is the set of all adjacencies and telomeres, and drawing the edges between the sets containing the extremities of the same gene.

We can think of the extremities in an adjacency as being joined by edges as well and distinguish the two kind of edges – those drawn between the extremities of the same gene and those between extremities in an adjacency. For instance, Friedberg et al. [41] call the edges between extremities of the same gene white edges and those between the extremities that make up an adjacency black edges.

Each gene extremity is thus incident on a white edge and on at most one black edge. Thus every vertex of a genome graph has degree one or two. This implies that the connected components of a genome graph are paths and circles. Each connected component represents a chromosome.

In visualising the structure of a genome, we found it more helpful to not draw the edges between the extremities of an adjacency. Figure 10 illustrates a genome graph, the edges denote the genes.

4.2.2 The double cut and join operator

The double cut and join operator (DCJ) acts on a genome graph by splitting two adjacencies and joining the free extremities thereby changing the genome.
structure. It can also join two telomeres or act on an adjacency and a telomere. For ease of notation, we will not distinguish between the action of the DCJ operator on a genome graph and its action on the associated genome. A formal definition of the DCJ operator lists all possible actions.

**Definition 4.2.1** (Double cut and join). *Let G be a genome on n genes. The double cut and join operator acts on G in one of the following ways:*

1. *The adjacencies \{p, q\} and \{r, s\} of G are changed to \{p, r\} and \{q, s\} or to \{p, s\} and \{q, r\}; or,*

2. *the adjacencies \{p, q\} and the telomere \{r\} of G are changed to \{p, r\} and \{q\} or to \{q, r\} and \{p\}; or,*

3. *the telomeres r and s of G are joined to form an adjacency \{r, s\}; or,*

4. *the adjacency \{p, q\} splits into two telomeres \{p\} and \{q\}.*

The effect of a DCJ operation on a genome is determined by three factors:

1. whether the operators acts on a pair of adjacencies, or an adjacency and a telomere, or between two telomeres;

2. whether the adjacencies and/or telomeres being acted on belong to the same component of the graph; and

3. the way in which the extremities are joined.

Figure 11 illustrates a single DCJ operation on a linear chromosome when both the cut points lie on the same chromosome. Figure 12 shows the case where the cut points lie on different chromosomes. As can be seen from these examples, the DCJ operator can simulate many different rearrangement events such as inversion, excision, translocation etc. and it is this versatility of the DCJ operator that makes it powerful. Another reason for its popularity
(a) A genome graph consisting of a single linear chromosome.

(b) Adjacencies \{1_h, 2_t\} and \{2_h, 3_t\} are the cut points for a double cut and join operation. Joining 1_h to 2_h achieves an inversion of gene 2.

(c) Joining 1_h to 3_t simulates excision of a circular chromosome 2_h, 2_t.

Figure 11: The effect of a double cut and join operation at (1_h, 2_t) and (2_h, 3_t) when both the adjacencies being cut lie on the same chromosome. The two different ways in which the genome graph can be reconnected produce an inversion of a chromosomal fragment and an excision of a circular chromosome respectively. A similar effect is achieved if both the cut points lie on the same circular chromosome.

is that the DCJ distance can be calculated easily in terms of the structure of a comparison graph derived from two genomes, as we shall discuss in Subsection 4.2.3.

4.2.3 The Double Cut and Join Distance

**Definition 4.2.2** (DCJ Distance). Let G_1 and G_2 be two genomes on the same set of genes. The DCJ distance between G_1 and G_2 denoted by d_D(G_1, G_2) is the smallest integer r such that the application of r successive DCJ operations transforms genome G_1 into genome G_2.
(a) A genome graph consisting of a linear and a circular chromosome.

(b) Adjacencies $1_t, 2_t$ and $4_t, 5_t$ are the cut points for a double cut and join operation. Joining $1_t$ to $4_t$ absorbs the circular chromosome into the linear chromosome.

(c) Joining $1_t$ to $5_t$ has a similar effect. It absorbs the circular chromosome into the linear chromosome but the orientation of the inserted section is opposite relative to its orientation when $1_t$ and $4_t$ were joined.

Figure 12: The effect of a double cut and join operation when the adjacencies being cut lie on different chromosomes. The two different ways in which the genome graph can be reconnected absorb the circular chromosome into the linear chromosome.
The inverse of a DCJ operator is a DCJ operator. The reader will notice that in each case in Definition 4.2.1, cutting the adjacencies that result from a DCJ operation $\rho$ and joining in the original configuration is a valid DCJ operation that reverses the effect of the DCJ operation $\rho$. Therefore it follows that

$$d_D(G_1, G_2) = d_D(G_2, G_1).$$

Bergeron et al. [13] make use of a graph construct called the “adjacency graph” to determine the DCJ distance between two genomes.

**Definition 4.2.3** (Adjacency Graph). Let $G_1$ and $G_2$ be genomes defined on the same set of genes. The vertex set of the adjacency graph $AG(G_1, G_2)$ is the disjoint union of the sets of adjacencies and telomeres of $G_1$ and $G_2$. For an adjacency (or telomere) $u \in G_1$ and adjacency (or telomere) $v \in G_2$, there is an edge in $AG(G_1, G_2)$ between $u$ and $v$ for each gene extremity they have in common.

Figure 13 shows an example of the adjacency graph for genomes $G_1$ and $G_2$ where

$$G_1 = \{\{1_h, 1_t\}, \{2_h\}, \{2_t, 3_h\}, \{3_t, 4_h\}, \{4_t\}\}$$

and

$$G_2 = \{\{1_h, 1_t\}, \{2_h, 2_t\}, \{3_h\}, \{3_t, 4_h\}, \{4_t\}\}.$$

The vertices of the adjacency graph then have degree either one (at a telomere) or two (at an adjacency), so the graph consists of a set of cycles and a set of paths. Let $G_1$ and $G_2$ be genomes on $n$ regions and let $c$ be the number of cycles and $p$ be the number of paths of odd length in $AG(G_1, G_2)$. Bergeron et al. [13] established that the DCJ distance between two genomes can be given in terms of these adjacency graph metrics as follows:

$$d_D(G_1, G_2) = n - (c + p/2).$$
Figure 13: An adjacency graph of genomes $G_1 = \{1_h, 1_t, 2_h, 2_t, 3_h, 3_t, 4_h, 4_t\}$ and $G_2 = \{1_h, 1_t, 2_h, 2_t, 3_h, 3_t, 4_h, 4_t\}$.

4.3 GENOMES AS PERMUTATIONS

In Section 4.2.1, a model of a genome as a graph was presented. In this section, we present in detail a model of a genome as a permutation on the set of region extremities introduced in [15]. To do this, we first formalise the notion of a region and region extremities.

**Definition 4.3.1 (Extremities).** Let $\mathbf{n}$ be the set $\{1, 2, \ldots, n\}$ enumerating the $n$ regions. The Cartesian product $E = \mathbf{n} \times \{h, t\}$ is the set of extremities of $n$ regions.

To conform to the notation used earlier in this chapter (4.2.1), we will use $i_h$ and $i_t$ to denote the extremities $(i, h)$ and $(i, t)$ giving the head and tail of region $i$ respectively. We define a map that assigns numeric labels to the elements of $E$.

**Definition 4.3.2 (Assignment map).** Let $\phi : E \rightarrow 2\mathbf{n}$ be defined as follows:

\[
\phi(i_t) = 2i - 1,
\]

\[
\phi(i_h) = 2i.
\]
Definition 4.3.3 (Genome). A genome on \( n \) regions is a permutation \( \pi \) on the set \( \phi(E) \) such that
\[
\pi(i) = j \iff \pi(j) = i.
\]

The restriction in the definition of a genome captures the notion of pairing of region extremities on a genomic strand. Therefore a 2-cycle in this formulation is an adjacency. Similarly, fixed points of a permutation are telomeres of the genome. The set of adjacencies and telomeres describes a genome in the adjacency list formulation. The analogous description of a genome in the algebraic formulation is as a product of disjoint 2-cycles and 1-cycles. It is important to note that at this point the permutation \( \pi \) is a static description of the genome, not an operation. The 2-cycles therefore can be considered synonyms for unordered pairs, although for consistency in notation, we always write them with as \((i, j)\) with \(i < j\). Figure 14 illustrates an example of permutation encoding of a genome on 4 regions. A similar example was presented in Chapter 3 (see Figure 3).
4.3.0.1 Cardinality of the set of genomes

Let the set of all genomes on $n$ regions be $\Gamma_n$. Each genome is a permutation of the $2n$ extremities $E$, and hence $\Gamma_n$ is a subset of the symmetric group $S_{2n}$. The cardinality of $\Gamma_n$ can be determined as follows. Each genomic permutation must have an even number of fixed points, since it is a product of 2-cycles and 1-cycles acting on a set of even cardinality ($2n$). Let the number of fixed points be $2t$. The remaining $2n - 2t$ elements must be paired off with each other. Each such pairing of the $2n - 2t$ elements defines an involution in the symmetric group $S_{2n-2t}$ that does not have any fixed points. The number of such involutions is $(2n - 2t - 1)!! [92, pp. 15-16]$ where the double factorial function is the product of odd numbers i.e. $(2k - 1)!! = \prod_{i=1}^{k} (2i - 1)$. Therefore the cardinality of $\Gamma_n$ is given by

$$|\Gamma_n| = \sum_{t=0}^{n} \binom{2n}{2t} (2n - 2t - 1)!! = \sum_{t=0}^{n} \binom{2n}{2t} (2t - 1)!!,$$

thus the number of genomes is already almost a billion for 9 regions. It is interesting to note that this is also the number of distinct standard Young tableaux on $n$ elements of shape determined by the structure of the permutation, with a correspondence given by the Robinson-Schensted algorithm (see for instance Fulton [42]). The first nine numbers in the sequence are shown in Table 3.
### Table 3: The number of genomes on $n$ regions (also the number of tableaux on $2n$ elements).

<table>
<thead>
<tr>
<th>#regions</th>
<th>#genomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>76</td>
</tr>
<tr>
<td>4</td>
<td>764</td>
</tr>
<tr>
<td>5</td>
<td>9496</td>
</tr>
<tr>
<td>6</td>
<td>140152</td>
</tr>
<tr>
<td>7</td>
<td>2390480</td>
</tr>
<tr>
<td>8</td>
<td>46206736</td>
</tr>
<tr>
<td>9</td>
<td>997313824</td>
</tr>
</tbody>
</table>

#### 4.4 The DCJ Operator as an Action on a Permutation

In this section we define an algebraic version of the DCJ operator acting on the set $\Gamma_n$ of genomes on $n$ regions, and show that it is an involution.

As explained in Section 4.3, the genome is modeled as a set of unordered pairs of region extremities (adjacencies) and single region extremities (telomeres). A DCJ operation as defined in Bergeron et al. [13] swaps region extremities between two pairs (i.e. adjacencies), or a pair and a singleton, as described in Definition 4.2.1. It can also join singletons to form a pair or split a pair to form singletons.

Hence, the possible scenarios are that the two region extremities being swapped are: adjacent to each other; both involved in different adjacencies; one of them is a telomere and the other in an adjacency; or both of them are telomeres. When a DCJ operation acts on a pair of region extremities that form an adjacency, it splits them, producing two telomeres, and conversely when it acts on two telomeres, it combines them into an adjacency. In the two other cases, an extremity is swapped.
Thus, in the permutation representation, the DCJ operation swapping \( i \) and \( j \) changes:

\[
\begin{align*}
(i,k)(j,l) & \rightarrow (j,k)(i,l) \text{ or } (i,j)(k,l), \\
(i,k)(j) & \rightarrow (j,k)(i) \text{ or } (i,j)(k), \\
(i,j) & \rightarrow (i)(j), \quad \text{and} \\
(i)(j) & \rightarrow (i,j).
\end{align*}
\]

With this motivation in mind, we formalise the algebraic double cut and join operator acting on the set of genomes \( \Gamma_n \).

Let \( X \) be an alphabet i.e. a set of symbols, indexed by the unordered pair \( \{i,j\} \).

\[
X := \{d_{i,j} \mid i, j \in 2n\}.
\]

For an unordered pair \( \{i,j\} \), we will write \( d_{i,j} \) with \( i < j \) throughout the thesis.

We first define the action of the set \( X \) on the set of genomes \( \Gamma_n \). This action will be extended to a group action on \( \Gamma_n \).

**Definition 4.4.1.** Let \( F_\pi \) be the set of the fixed points of a genomic permutation \( \pi \).

That is,

\[
F_\pi := \{i \in 2n \mid \pi(i) = i\}.
\]

We define \( f : X \times \Gamma_n \rightarrow \Gamma_n \) as follows:

\[
f(d_{i,j}, \pi) := \begin{cases} 
(i,j)\pi & \text{if } i,j \in F_\pi \text{ or } \pi(i) = j, \text{ and} \\
(i,j)\pi(i,j) & \text{otherwise.}
\end{cases}
\]
The first condition in Definition 4.4.1 amounts to requiring \( i, j \) to be either both telomeres or adjacent to each other and the second part of the definition covers the complementary case that \( i, j \) are not both telomeres and are not adjacent to each other. From Definition 4.4.1, it is clear that for all \( i, j \in 2n \), 
\[
f(d_{i,j}, \pi) = f(d_{j,i}, \pi).
\]
Thus as elements of a set acting on the set of genomes, \( d_{i,j} \) and \( d_{j,i} \) are identical and we need only retain one of the two in the set \( X \). For consistency, we will always write \( d_{i,j} \) where \( i < j \).

In algebraic terms, the double cut and join operators as defined above are conjugations or left actions by 2-cycle involutions. This suggests that the operator may itself be an involution. This is formally established in the following lemma.

**Lemma 4.4.2.** \( f(d_{i,j}, f(d_{i,j}, \pi)) = \pi \) for all \( \pi \in \Gamma_n, d_{i,j} \in X \).

**Proof.** For any permutation \( \pi \in \Gamma_n \), if \( i, j \) are not both telomeres and do not form an adjacency of \( \pi \), then the same holds in \( f(d_{i,j}, \pi) = (i, j)\pi(i, j) \). Similarly if \( i \) and \( j \) are both telomeres in \( \pi \) then they form an adjacency in \( f(d_{i,j}, \pi) \), and if they are in an adjacency in \( \pi \), they will both be telomeres in \( f(d_{i,j}, \pi) \). Thus acting by \( d_{i,j} \) on \( f(d_{i,j}, \pi) \) will cause the same condition in the definition of \( d_{i,j} \) to be invoked which was invoked when \( d_{i,j} \) acted on \( \pi \).

The operation in each case is an involution, hence \( f(d_{i,j}, f(d_{i,j}, \pi)) = \pi \) for all \( \pi \in \Gamma_n \). \( \square \)

Let \( G_D \) be the group given by the following presentation:

\[
G_D := \langle X \mid d_{i,j}^2 \rangle.
\]

We will extend the action of the generators of the group \( G_D \) to an action of \( G_D \) on \( \Gamma_n \) in the natural way.

**Definition 4.4.3 (Algebraic DCJ).** Let \( g = d_{i_1,j_1}d_{i_2,j_2} \ldots d_{i_k,j_k} \in G_D \) and let \( e \) be the identity element of \( G_D \). \( \tilde{f} : G_D \times \Gamma_n \to \Gamma_n \) is defined recursively as
\[
\tilde{f}(g, \pi) := \begin{cases} 
\pi & \text{if } g = e \\
 f(g, \pi) & \text{if } g \in X \\
 \tilde{f}(d_{i_1,j_1}d_{i_2,j_2} \cdots d_{i_{k-1},j_{k-1}}, f(d_{i_k,j_k}, \pi)) & \text{otherwise.}
\end{cases}
\]

Since \( G_D \) is finitely generated by \( X \) and the action of \( f \) is well-defined on each element of \( X \), the recursive definition of \( \tilde{f} \) in terms of the action of \( f \) is also well-defined.

4.4.1 DCJ action is a group action

Recall from Chapter 2 that a group \( G \) is said to act on a set \( X \) when there is a function \( \psi : H \times S \to S \) such that for all \( x \in S \), the following conditions hold:

1. \( \psi(e, x) = x \), and
2. \( \psi(g, \psi(h, x)) = \psi(gh, x) \) for all \( g, h \in H \).

The algebraic DCJ map \( \tilde{f} \) from \( G_D \times \Gamma_n \) to \( \Gamma_n \) satisfies the properties of being a group action by definition and by virtue of Lemma 4.4.2.

For ease of notation, we will write \( \tilde{f}(g, \pi) \) as \( g(\pi) \).

4.4.2 Algebraic DCJ is equivalent to the “classical” DCJ

To distinguish this formulation from the standard, we would like to call \( d_{i,j} \) the algebraic double cut and join operator. But before we can do so, we must establish that the operation as defined in Definition 4.4.3 is the same as the double cut and join operation as defined in Section 4.2.2.

The vertex set of a genome graph consists of unordered pairs and singletons from the set of region extremities \( E \). The map \( \phi \) 4.3.2 simply relabels elements
of the set $E$ with the labels from the set $2n$. Therefore we can consider the genome to consist of unordered pairs and singletons from $2n$. A genomic permutation $\pi$ is a permutation on the set $2n$ satisfying the constraint

$$\pi(i) = j \iff \pi(j) = i.$$ 

Let $\rho$ be the map that writes the vertex set of the genome graph $G$ as the permutation $\pi$.

$$\rho([i,j]) = (i,j) \text{ and } \rho([i]) = (i).$$

The map $\rho$ is extended to the genome $G$ as the product of the images of the vertices. That is, let $G = \{v_i\}$ where each $v_i$ is a singleton or a pair. Then,

$$\rho(G) = \prod_{v_i \in G} \rho(v_i).$$

Since the pairs and singletons of $G$ are mutually disjoint, their images under $\rho$ are also disjoint and hence commute. Thus, $\rho$ is a well defined map on $G$.

Let $D_{\{i,j\}}(G)$ be the DCJ operator acting on the extremities $i$ and $j$ of genome $G$ and let $d_{i,j}(\pi)$ be the operator acting on the permutation $\pi$. The remarks motivating the definition of algebraic DCJ operator informally explain why we can expect the graph-theoretic and the algebraic operators to be equivalent. That is, the diagram in Figure 15 commutes.

We now prove this statement formally.

$$D_{\{i,j\}} \quad \rho \quad d_{i,j}$$

Figure 15: Rewriting genome $G$ as the permutation $\pi$, and acting on $\pi$ by $d_{i,j}$ gives the same result as acting on $G$ by the DCJ operator and then rewriting the result as permutation $\pi'$. 
Lemma 4.4.4. For all genomes $G$,

$$\rho \left( \mathcal{D}_{\{i,j\}}(G) \right) = d_{i,j}(\rho(G)).$$

Proof. We prove this by considering all the four cases in the definition of the operator $\mathcal{D}$ (Section 4.2.2).

Case 1. $i$ and $j$ are in separate pairs $\{i,k\}, \{j,l\} \in G$.

$$\rho \left( \mathcal{D}_{\{i,j\}}(G) \right) = \rho \left( \mathcal{D}(\{i,k\}, \{j,l\}) \right) = \rho(\{j,k\}, \{i,l\}) = (j,k)(i,l).$$

$$d_{i,j}(\rho(\{i,k\}, \{j,l\})) = d_{i,j}((i,k)(j,l)) = (j,k)(i,l).$$

Case 2. Exactly one of the $i, j$ is in a pair $\{i,k\}, \{j\} \in G$.

$$\rho \left( \mathcal{D}_{\{i,j\}}(G) \right) = \rho \left( \mathcal{D}(\{i,k\}, \{j\}) \right) = \rho(\{(j,k)\}, \{i\}) = (j,k)(i).$$

$$d_{i,j}(\rho(\{i,k\}, \{j\})) = d_{i,j}((i,k)(j)) = (j,k)(i).$$

Case 3. None of the $i, j$ is in a pair. $\{i\}, \{j\} \in G$.

$$\rho \left( \mathcal{D}_{\{i,j\}}(G) \right) = \rho \left( \mathcal{D}(\{i\}, \{j\}) \right) = \rho(\{i,j\}) = (i,j).$$

$$d_{i,j}(\rho(\{i\}, \{j\})) = d_{i,j}((i)(j)) = (i,j).$$

Case 4. $i, j$ are in the same pair in $G \{i,j\} \in G$

$$\rho \left( \mathcal{D}_{\{i,j\}}(G) \right) = \rho \left( \mathcal{D}(\{i,j\}) \right) = \rho(\{i\}, \{j\}) = (i)(j).$$
\[ d_{ij} (\rho(\{i,j\})) = d_{ij} ((i,j)) = (i)(j). \]

Thus in all cases, \( \rho \left( D_{\{i,j\}} (G) \right) = d_{ij} (\rho(G)). \)

With this definition of the algebraic DCJ, we can formalise the notion of DCJ distance in the algebraic framework. We make use of the length of a group element \( w \). The length of a group element is the smallest number of generators needed to write it as their product. Recall that the generators of \( G_D \) is the set \( X = \{ d_{i,j} | i, j \in 2n, i < j \} \).

**Definition 4.4.5 (DCJ Distance).** The DCJ distance \( d_D(\pi_1, \pi_2) \) between genomes \( \pi_1 \) and \( \pi_2 \) is the smallest \( r \) such that there is some element \( w \in G_D \) where the length of \( w \) is \( r \) and

\[ w(\pi_1) = \pi_2. \]

If \( w(\pi_1) = \pi_2 \) for some \( w \in G_D \), then because of the properties of a group action, it follows that \( w^{-1}(\pi_2) = \pi_1 \). As the length of a group element and its inverse is the same, it follows that the DCJ distance is symmetric i.e., \( d_D(\pi_1, \pi_2) = d_D(\pi_2, \pi_1) \).

### 4.5 Products of Involutions

As we have seen in Section 4.4, in the algebraic framework, the genome and the DCJ operator are modeled as involutions. Products of involutions will play an important role in the development of the algebraic DCJ theory and in the rest of this thesis. In this section we prove some of the results about products of involutions that are needed for later development. In the rest of this chapter, we use the notation \( F_\pi \) to denote the set of points fixed by \( \pi \) i.e. \( i \) such that \( \pi(i) = i \).
Lemma 4.5.1. Let $\alpha$ and $\beta$ be involutions acting on the set $2n = \{1, 2, \ldots, 2n\}$. If $F_\alpha = F_\beta = \emptyset$, then

1. For any $i \in 2n$, $i$ and $\alpha(i)$ are in different cycles of $\beta \alpha$. Similarly $i$ and $\beta(i)$ are in different cycles of $\beta \alpha$.

2. $\beta \alpha$ has an even number of cycles of length $k$ for any $k \in \mathbb{N}$.

Since the cycle structures of a permutation and its inverse are the same, the above statements also hold true for $\alpha \beta$.

Proof. The cycle in $\beta \alpha$ containing 1 is of the form

$$\left(1, \beta \alpha(1), \beta \alpha \beta \alpha(1), \ldots, (\beta \alpha)^k(1)\right),$$

where $k$ is the smallest positive integer such that $(\beta \alpha)^{k+1}(1) = 1$, so the length of this cycle is $k + 1$. Suppose that $\alpha(1) = (\beta \alpha)^r(1)$ for some $r$. If $r$ is even then

$$\alpha(1) = (\beta \alpha)^r(1) = (\beta \alpha)^{(r/2)-1} \beta \alpha (\beta \alpha)^{r/2}(1).$$

By multiplying on the left both sides by $(\alpha \beta)^{(r/2)-1}$, (the inverse of $(\beta \alpha)^{(r/2)-1}$), we get

$$(\alpha \beta)^{(r/2)-1} \alpha(1) = \beta \alpha (\beta \alpha)^{r/2}(1),$$

and multiplying by $\beta$ yields

$$\beta (\alpha \beta)^{(r/2)-1} \alpha(1) = \alpha (\beta \alpha)^{r/2}(1),$$

and therefore,

$$(\beta \alpha)^{r/2}(1) = \alpha \left((\beta \alpha)^{r/2}(1)\right).$$
In other words, \((\beta \alpha)^{r/2}(1) \in F_{\alpha}\), contradicting the assumption that \(F_{\alpha} = \emptyset\). Similarly, if \(r\) is odd, we find that \(\alpha(1) = (\beta \alpha)^{r}(1)\) implies that \((\alpha \beta)^{(r+1)/2} \alpha (1)\) is a fixed point of \(\beta\), another contradiction. Thus \(\alpha(1) \notin \{1, \beta \alpha(1), \ldots, (\beta \alpha)^{k}(1)\}\). This proves part (1) of the theorem.

Write the cycle in \(\beta \alpha\) containing \(\alpha(1)\) as

\[(\alpha(1), \beta(1), \beta \alpha \beta(1), \ldots, (\beta \alpha)^{s} \beta(1)),\]

where \(s\) is the smallest positive integer such that \((\beta \alpha)^{s+1} \beta(1) = \alpha(1)\). Then,

\[
\begin{align*}
\alpha(1) &= (\beta \alpha)^{s+1} \beta(1) \\
\beta \alpha(1) &= \beta (\beta \alpha)^{s+1} \beta(1) \\
\beta \alpha(1) &= \beta \beta \alpha (\beta \alpha)^{s} \beta(1) \\
\beta \alpha(1) &= \alpha (\beta \alpha)^{s} \beta(1) \\
\beta \alpha(1) &= (\alpha \beta)^{s+1}(1) \\
(\beta \alpha)^{s+1} \beta \alpha(1) &= 1 \\
(\beta \alpha)^{s+2}(1) &= 1.
\end{align*}
\]

That is, \((\beta \alpha)^{s+1} \beta(1) = \alpha(1)\) implies \((\beta \alpha)^{s+2}(1) = 1\).

\((\beta \alpha)^{k+1}(1) = 1\) and minimality of \(k\) and \(s\) imply \(s = k - 1\). Thus, the length of the cycle containing \(\alpha(1)\) is \(k + 1\), which is the same as the length of the cycle containing \(1\). Since the same argument holds for any \(i \in 2n\) there will be an even number of cycles of any given length in \(\beta \alpha\).

The next lemma concerns the cycle containing a point that is fixed by both permutations. The result is trivial but for the ease of future reference, we note it here as a lemma.
Lemma 4.5.2. Let $\alpha$ and $\beta$ be permutations on the set $2n$ such that $\alpha$ and $\beta$ are involutions. Let $i \in F_\alpha \cap F_\beta$. Then the cycle in $\beta\alpha$ containing $i$ is of length 1.

Proof. If $i \in F_\alpha \cap F_\beta$ then $\alpha(i) = i$ and $\beta(i) = i$. Consequently $\beta\alpha(i) = i$. That is, $i$ is a fixed point of $\beta\alpha$ and the lemma follows. \qed

Finally, we consider the case where a point is fixed by one but not both of the involutions.

Lemma 4.5.3. Let $\alpha$ and $\beta$ be permutations on the set $2n$ such that $\alpha$ and $\beta$ are involutions. Let $i \in (F_\alpha \cup F_\beta) \setminus (F_\alpha \cap F_\beta)$. A cycle in $\beta\alpha$ containing $i$ contains exactly 2 points from $F_\alpha \cup F_\beta$.

Proof. Without loss of generality, suppose $i \in F_\alpha$; that is, $\alpha(i) = i$. The cycle containing $i$ in $\beta\alpha$ is of the form

\[
(i, \beta\alpha(i), \beta\alpha\beta\alpha(i), \ldots, (\beta\alpha)^k(i))
\]

where $k$ is the smallest positive integer for which $(\beta\alpha)^{k+1}(i) = i$. As in the proof of Lemma 4.5.1, this cycle contains $\alpha(i) = i = (\beta\alpha)^{k+1}(i)$. We have argued in the proof of Lemma 4.5.1 that if $k + 1$ is odd then

\[
(\beta\alpha)^{k+1}(i) = i = \alpha(i) \implies (\alpha\beta)^{(k+2)/2} \alpha(i) \in F_\beta
\]

and if $k + 1$ is even then

\[
(\beta\alpha)^{k+1}(i) = i = \alpha(i) \implies (\beta\alpha)^{(k+1)/2}(i) \in F_\alpha.
\]

That is, if $i \in F_\alpha$ then the cycle containing $i$ contains $(\alpha\beta)^{(k+2)/2} \alpha(i)$ if the length of the cycle is odd, and $(\beta\alpha)^{(k+1)/2}(i)$ if it is even. This point is distinct from $i$ because if it is $(\alpha\beta)^{(k+2)/2} \alpha(i)$, then it is in $F_\beta$ and since $i \in (F_\alpha \cup F_\beta) \setminus
\((F_\alpha \cap F_\beta), i \in F_\alpha \implies i \notin F_\beta\). If the point is \((\beta \alpha)^{(k+1)/2}(i)\), then it belongs to the cycle
\[
\left( i, \beta \alpha(i), \beta \alpha \beta \alpha(i), \ldots, (\beta \alpha)^k(i) \right)
\]
and the cycle consists of distinct points.

At this point, we have established that if \(i \in F_\alpha\) then the cycle containing \(i\) contains at least one other point from \(F_\alpha \cup F_\beta\). We now need to prove that this cycle cannot contain any other point from \(F_\alpha \cup F_\beta\).

Suppose then that \(j \in F_\alpha \cup F_\beta\) and \(j\) is in the cycle containing \(i\). Since \(j\) is in the cycle, for some positive integer \(s\), \(j = (\beta \alpha)^s(i)\). Let \(s\) be the smallest such integer.

Suppose first that \(j \in F_\alpha\); that is, \(\alpha((\beta \alpha)^s(i)) = (\beta \alpha)^s(i)\). We then have
\[
i = (\alpha \beta)^s \alpha (\beta \alpha)^s(i) \quad \text{since} \quad ((\beta \alpha)^s)^{-1} = (\alpha \beta)^s
\]
\[
= \alpha(\beta \alpha)^{2s}(i).
\]
But \(\alpha(i) = i\), so acting on both sides by \(\alpha\) we have that \((\beta \alpha)^{2s}(i) = i\).

Similarly if \(j \in F_\beta\), so that \(\beta((\beta \alpha)^s(i)) = (\beta \alpha)^s(i)\), we obtain \((\beta \alpha)^{2s-1}(i) = i\).

But since \(k + 1\) is the minimal integer for which \((\beta \alpha)^{k+1}(i) = i\) it follows that \(2s\) or \(2s - 1\) is a multiple of \(k + 1\), respectively. Since \(s\) is also minimal, \(s = (k + 1)/2\) or \(s = (k + 2)/2\) according to whether \(j \in F_\beta\) or \(j \in F_\alpha\). Hence \(j\) is one of the points identified above unless
\[
k + 1 = \frac{k + 1}{2} \quad \text{or} \quad k + 1 = \frac{k + 2}{2}.
\]

The first equation does not have any nonnegative solution. The only nonnegative integer satisfying the second condition is \(k = 0\) which would mean that \(i \in F_\beta\) contradicting the assumption that \(i \in (F_\alpha \cup F_\beta) \setminus (F_\alpha \cap F_\beta)\).
Thus if \( i \in (F_\alpha \cup F_\beta) \setminus (F_\alpha \cap F_\beta) \) then the cycle of \( \beta \alpha \) containing \( i \) contains one other point from \( F_\alpha \cup F_\beta \) and therefore contains exactly two points from \( F_\alpha \cup F_\beta \). \( \square \)

Petersen and Tenner [77] also investigate the nature of the product of involutions and prove similar results. Their results are stated in terms of the structure of an involution product graph.

**Example 4.5.4** (Product of involutions). Let \( \alpha = (1,2)(3,4)(5,6) \) and \( \beta = (1,5)(2,4)(3,6) \). Then \( \alpha \beta = (1,6,4)(2,3,5) \) contains two cycles of the same length where \( i \) and \( \alpha(i) \) are not in the same cycle for any \( i \in \{1,2,3,4,5,6\} \) and neither are \( i \) and \( \beta(i) \).

## 4.6 Determining the DCJ Distance

### 4.6.1 A lower bound on the DCJ distance

**Definition 4.6.1** (Transposition Length). For any permutation \( \pi \), the transposition length of \( \pi \) denoted by \( \ell_t(\pi) \) is the minimal number of transpositions needed to express \( \pi \).

Since the \( d_{i,j} \) operation involves multiplying a permutation with transpositions, we are interested in how multiplication by a transposition affects the transposition length of a permutation. In fact this effect is easily stated: multiplication by a transposition changes the transposition length of a permutation by \( \pm 1 \).

That is,

\[
\ell_t((i,j)\pi) = \ell_t(\pi) \pm 1,
\]

\[
\ell_t(\pi(i,j)) = \ell_t(\pi) \pm 1.
\] (4.6.1)
This can be observed by noting first that a permutation can be expressed
as a product of either an odd or an even number of transpositions, but not
both. That is, the parity of the number of transpositions needed to write a
permutation as a product is unique.

Let the transposition length of a permutation $\pi$ be $r$. That is,

$$\pi = t_1 t_2 \ldots t_r,$$

where the $t_i$ are transpositions.

Suppose $r$ is odd. Then the parity of $(i, j)\pi$ is even since $(i, j)t_1 t_2 \ldots t_r$ is
one expression of the result as a product of transpositions, although it may
not be minimal. The transposition length of $(i, j)\pi$ is also therefore even. If
transposition length of $(i, j)\pi$ is less than $r - 1$, that is,

$$(i, j)\pi = s_1 s_2 \ldots s_p,$$

where the $s_i$ are transpositions and $p < r - 1$. It follows that

$$\pi = (i, j)s_1 s_2 \ldots s_p.$$

That is, $\pi$ can be written as a product of $p + 1 < r$ transpositions, contradicting
the supposition that the transposition length of a permutation $\pi$ is $r$. Hence
the transposition length of $(i, j)\pi$ is either $r + 1$ or $r - 1$. A similar argument
works if $r$ is even.

In the remaining part of this section, we will show that the DCJ distance
between $\pi_1$ and $\pi_2$ can be determined in terms of the transposition length of
the permutation product $\pi_2\pi_1$. First of all, note that if $\pi_1 = \pi_2$ then $\pi_2\pi_1 = ()$
where () is the identity permutation, hence $\ell_1 (\pi_2\pi_1) = 0$. 

85
We make the following claim regarding a lower bound on the DCJ distance between permutations $\pi_1$ and $\pi_2$.

**Lemma 4.6.2.** Let $\pi_1$ and $\pi_2$ be genomic permutations. Then

$$d_D(\pi_1, \pi_2) \geq \frac{1}{2} \ell_t(\pi_1 \pi_2).$$

**Proof.** A single DCJ operation $d_{i,j}$ acts either by conjugation of $\pi_1$ by the transposition $(i, j)$, or by multiplication of $\pi_1$ by $(i, j)$.

If $d_{i,j}(\pi_1) = (i, j)\pi_1$, then $d_{i,j}(\pi_1)\pi_2 = (i, j)\pi_1\pi_2$. Hence by equation (4.6.1)

$$\ell_t(d_{i,j}(\pi_1)\pi_2) = \ell_t(\pi_1\pi_2) \pm 1.$$

If $d_{i,j}(\pi_1) = (i, j)\pi_1(i, j)$ then

$$d_{i,j}(\pi_1)\pi_2 = (i, j)(\pi_1(i), \pi_1(j))\pi_1\pi_2.$$

By applying equation 4.6.1 twice,

$$\ell_t(d_{i,j}(\pi_1)\pi_2) = \ell_t(\pi_1\pi_2) \text{ or } \ell_t(\pi_1\pi_2) \pm 2.$$

Thus a single DCJ operation on $\pi_1$ can reduce the transposition length of $\pi_1\pi_2$ by at most 2. Since $\ell_t(\pi_2\pi_1) = 0$ when $\pi_1 = \pi_2$, the DCJ distance between $\pi_1$ and $\pi_2$ must be at least $\ell_t(\pi_1\pi_2)/2$. □

### 4.6.2 Subpermutations

We define a binary relation on $2n$ which will allow us to separate out the different components of a pair of genomic permutations, each of which we will then be able to sort independently of the others.
**Definition 4.6.3.** Let $\pi_1$ and $\pi_2$ be genomic permutations on $n$ regions. Define a binary relation $\sim$ on $2n$ by setting

$$i \sim j \iff (\pi_2\pi_1)^k(i) = j \text{ or } \pi_1(\pi_2\pi_1)^k(i) = j \text{ for some } k \in \mathbb{Z}.$$ 

The cycles of $\pi_1\pi_2$ and $\pi_2\pi_1$ are equivalent as sets (they are inverses of each other in $S_{2n}$), hence the binary relation $\sim$ defined for the pair $\pi_1, \pi_2$ would be the same as defined for the pair $\pi_2, \pi_1$. Therefore, without any ambiguity $\sim$ can be defined for an unordered pair of genomic permutations. Note that although the relation $\sim$ depends on the permutations under consideration, for ease of writing, we choose not to denote this dependence in the notation.

**Lemma 4.6.4.** $\sim$ is an equivalence relation on $2n$.

*Proof.* It is easy to verify that $\sim$ is reflexive and symmetric. To establish that $\sim$ is transitive, note that since $i$ could be related to $j$ through either of the two relations $\pi_1(\pi_2\pi_1)^k(i) = j$ or $(\pi_2\pi_1)^k(i) = j$, there are four possible cases to be checked. For example, suppose that $\pi_1(\pi_2\pi_1)^p(i) = j$ and $(\pi_2\pi_1)^q(j) = k$. Noting that $(\pi_2\pi_1)^q\pi_1 = \pi_1(\pi_2\pi_1)^{-q}$, since $\pi_i$ are involutions, we have

$$k = (\pi_2\pi_1)^q\pi_1(\pi_2\pi_1)^p(i)$$

$$= \pi_1(\pi_2\pi_1)^{p-q}(i),$$

and hence $i \sim k$. The other cases can be checked similarly. \qed

For any $i, j \in 2n$, if $i$ and $j$ are in the same cycle of $\pi_2\pi_1$ then $j = (\pi_2\pi_1)^s(i)$ for some $s$ and hence $i \sim j$. Also, $i \sim \pi_1(i)$, and $\pi_1(i)$ is related to all the elements in the cycle of $\pi_2\pi_1$ that contains $\pi_1(i)$. Therefore an equivalence class under $\sim$ contains the union of the cycles of $\pi_1\pi_2$ containing $i$ and $\pi_1(i)$. 
If \( j \) is the equivalence class containing \( i \), then by definition, either \( j = (\pi_2 \pi_1)^k(i) \) or \( j = \pi_1(\pi_2 \pi_1)^k(i) \). The cycle of \( \pi_2 \pi_1 \) containing \( i \) is of the form

\[
(i, \pi_2 \pi_1(i), \ldots, (\pi_2 \pi_1)^r(i), \ldots, (\pi_2 \pi_1)^l(i)),
\]

where \( l \) is the smallest integer such that \( (\pi_2 \pi_1)^{l+1}(i) = i \). If \( k \leq l \), then it is clear that \( j \) is in this cycle. If \( k > l \), let \( k \equiv d \pmod{l} \). Then,

\[
j = \pi_1(\pi_2 \pi_1)^k(i) = \pi_1(\pi_2 \pi_1)^d(i) = \pi_1(\pi_2 \pi_1)^{l}(i).
\]

Once again, we have that \( j \) is in the cycle of \( \pi_2 \pi_1 \) containing \( i \). A similar argument shows that if \( j = \pi_1(\pi_2 \pi_1)^k(i) \), then \( j \) is in the same cycle as \( \pi_1(i) \).

Therefore, an equivalence class under \( \sim \) is equal to the union of the cycles of \( \pi_1 \pi_2 \) containing \( i \) and \( \pi_1(i) \).

Observe that \( i \sim \pi_1(i) \) and \( i \sim \pi_2(i) \). Hence the partition of \( 2n \) under \( \sim \) will also partition the 2-cycles of \( \pi_1 \) and \( \pi_2 \). In what follows we would like to talk about the sub-permutations of \( \pi_1 \) and \( \pi_2 \) thus induced. Formally,

**Definition 4.6.5 (Sub-permutation).** Let \( \pi_1 \) and \( \pi_2 \) be genomic permutations. Let \( C_1, C_2, \ldots, C_r \subseteq 2n \) be the equivalence classes under \( \sim \) defined by \( \pi_1, \pi_2 \). For \( 1 \leq s \leq r \) and \( i = 1, 2 \), the sub-permutation \( \pi_i^{(s)} \) of \( \pi_i \) induced by \( \sim \) is defined to be the restriction of \( \pi_i \) to \( C_s \), that is,

\[
\pi_i^{(s)} := \pi_i|_{C_s}.
\]

Intuitively, we have collected in a sub-permutation all the 2-cycles that are relevant for sorting \( \pi_1^{(s)} \) into \( \pi_2^{(s)} \).

An example will help illustrate these definitions. Let \( \pi_1 \) and \( \pi_2 \) be the following genomic permutations on 4 regions:

\[
\pi_1 = (1, 6)(2, 3)(4, 5)(7, 8), \quad \pi_2 = (1, 2)(3, 4)(5, 6).
\]
The product $\pi_2\pi_1$ is

$$(1,2)(3,4)(5,6)(1,6)(2,3)(4,5)(7,8) = (1,5,3)(2,4,6)(7,8).$$

Consider the equivalence class containing 1. $\pi_2\pi_1(1) = 5$, hence $1 \sim 5$. Similarly, $1 \sim 3$. $\pi_1\pi_2\pi_1(1) = \pi_1(5) = 4$, so $1 \sim 4$. The partitions of the set $\{1,2,\ldots,8\}$ under $\sim$ are

$$C_1 = \{1,2,3,4,5,6\}, \quad C_2 = \{7,8\}.\$$

The sub-permutations $\pi_1^{(1)}$ and $\pi_1^{(2)}$ are then $\pi_1^{(1)} = (1,6)(2,3)(4,5)$, $\pi_1^{(2)} = (7,8)$. Similarly, the sub-permutations $\pi_2^{(1)}$ and $\pi_2^{(2)}$ are $\pi_2^{(1)} = (1,2)(3,4)(5,6)$, $\pi_2^{(2)} = (7)(8)$ where the cycles of length 1 are written for clarity.

As remarked earlier, an equivalence class $C_s$ under $\sim$ contains precisely those elements of $2n$ that are contained in the cycle of $\pi_1\pi_2$ containing $i$ and $\pi_1(i)$. Therefore as proved in Lemmas 4.5.1 and 4.5.3, if either of the sub-permutations $\pi_1^{(s)}$ and $\pi_2^{(s)}$ has a fixed point, their product is a single cycle containing one or two points from $F_{\pi_1} \cup F_{\pi_2}$. If neither of them has a fixed point, their product consists of (exactly) two disjoint cycles.

Suppose the sub-permutations $\pi_1^{(s)}$ and $\pi_2^{(s)}$ are distinct. If $\pi_1^{(s)}$ and $\pi_2^{(s)}$ are conjugate in $S_{2n}$, then they have the same cycle type. Hence either $C_s$ contains no points from $F_{\pi_1} \cup F_{\pi_2}$, or it contains one point each from $F_{\pi_1}$ and $F_{\pi_2}$. In the first case, it follows from Lemma 4.5.1 that $C_s$ contains an even number of points. In the latter case, the cardinality of $C_s$ is odd.

These observations are summarised in Corollary 4.6.6.

**Corollary 4.6.6.** Let $\pi_1^{(s)}$ and $\pi_2^{(s)}$ be distinct sub-permutations induced by an equivalence class $C_s$ such that $\pi_1^{(s)}$ and $\pi_2^{(s)}$ are conjugate in $S_{2n}$. Then either $F_{\pi_1^{(s)}} \cup F_{\pi_2^{(s)}} = \emptyset$ or it contains two points.
Let $i \in C_s$. If $F_{\pi_1(s)} \cup F_{\pi_2(s)} = \emptyset$, then the product $\pi_2^{(s)} \pi_1^{(s)}$ is given by

$$
\left( i, \pi_2^{(s)} \pi_1^{(s)}(i), \ldots, (\pi_2^{(s)} \pi_1^{(s)})^u(i) \right) \left( \pi_1^{(s)}(i), \ldots, (\pi_2^{(s)} \pi_1^{(s)})^{u-1} \pi_2^{(s)}(i) \right).
$$

Otherwise the product is of the form

$$
\left( i, \pi_2 \pi_1^{(s)}(i), \ldots, (\pi_2 \pi_1^{(s)})^{2u}(i) \right),
$$

and this cycle contains one point each from $F_{\pi_1(s)}$ and $F_{\pi_2(s)}$.

The sum of lengths of the cycles in the product is the cardinality of $C_s$.

Continuing our example above, we see $\pi_1^{(1)} \pi_2^{(1)} = (1,3,5)(2,6,4)$, and $\pi_1^{(2)} \pi_2^{(2)} = (7,8)$. Observe that $\pi_1^{(1)} \pi_2^{(1)}$ is a product of two disjoint cycles. $\pi_1^{(2)} \pi_2^{(2)}$ contains two points from $F_{\pi_2}$ namely 7 and 8.

If a partition $C_t$ consists of a single point, say $i$, then $i$ is a fixed point of both $\pi_1$ and $\pi_2$, hence the DCJ distance between sub-permutations induced by $C_t$ is 0.

From the definition of sub-permutations, it is clear that the sub-permutations induced by an equivalence class $C_i$ under $\sim$ can be sorted independently of the sub-permutations induced by the other equivalence classes.

The problem of determining the DCJ distance between two genomic permutations can thus be broken down into the problem of determining DCJ distances between the sub-permutations. Each sub-permutation is a genomic permutation on a restricted set of regions. In the next section, we will determine the DCJ distance between a pair of conjugate genomic permutations and a pair of non-conjugate genomic permutations, and finally sum the distances over the sub-permutations of a genomic permutation. We will establish in Theorem 4.6.2 that the DCJ distance cannot be smaller than this sum. It is worth emphasising that although for convenience, we will not use the sub-permutation notation, we are still talking about the genomic permutations.
that can be sorted independently and that have only a single equivalence class under the relation $\sim$.

4.6.3 DCJ distance between conjugate permutations

Let the permutations $\pi_1$ and $\pi_2$ be conjugate in $S_{2n}$. That is, there exists a $g \in S_{2n}$ such that

$$g\pi_1g^{-1} = \pi_2.$$ 

By writing $g$ as a product of transpositions, we obtain a sequence of DCJ operations that transforms $\pi_1$ to $\pi_2$, each of which is conjugation by a transposition. Let $d_{DCJ}^c(\pi_1, \pi_2)$ be the minimal number of DCJ conjugation operations needed to transform $\pi_1$ into $\pi_2$. Then clearly the DCJ distance between $\pi_1$ and $\pi_2$, $d_D(\pi_1, \pi_2)$ can be no greater than the conjugation distance between them. That is,

$$d_D(\pi_1, \pi_2) \leq d_{DCJ}^c(\pi_1, \pi_2).$$

**Theorem 4.6.7.** Let $\pi_1$ and $\pi_2$ be genomic permutations $\pi_1$ and $\pi_2$ on $n$ regions such that $\pi_1$ and $\pi_2$ are conjugate in $S_{2n}$. Then the conjugation distance $d_{DCJ}^c(\pi_1, \pi_2)$ is half the transposition length of $\pi_2\pi_1$, that is,

$$d_{DCJ}^c(\pi_1, \pi_2) = \frac{1}{2} \ell_t (\pi_2\pi_1).$$

**Proof.** Note that from Lemma 4.6.2, we have a lower bound on the DCJ distance and hence on the conjugation distance:

$$\frac{1}{2} \ell_t (\pi_1\pi_2) \leq d_D(\pi_1, \pi_2) \leq d_{DCJ}^c(\pi_1, \pi_2).$$
To prove the theorem therefore, it would suffice to show that

\[ d_{DCJ}^\ell (\pi_1, \pi_2) \leq \frac{1}{2} \ell_t (\pi_1 \pi_2). \]

We now proceed to establish this inequality. Recall that Corollary 4.6.6 gives the structure of the product \( \pi_2 \pi_1 \). If \( \pi_1 \) and \( \pi_2 \) have no fixed points then

\[
\begin{align*}
\pi_2 \pi_1 &= \left( 1, \pi_2 \pi_1(1), (\pi_2 \pi_1)^2(1), \ldots, (\pi_2 \pi_1)^u(1) \right) \\
&\quad \left( \pi_1(1), \pi_2(1), \pi_2 \pi_1 \pi_2(1), \ldots, (\pi_2 \pi_1)^{u-1} \pi_2(1) \right).
\end{align*}
\]

(4.6.2)

If \( \pi_1 \) and \( \pi_2 \) contain fixed points then \( \pi_2 \pi_1 \) is a single cycle,

\[
\pi_2 \pi_1 = \left( 1, \pi_2 \pi_1(1), (\pi_2 \pi_1)^2(1), \ldots, (\pi_2 \pi_1)^{2u}(1) \right).
\]

(4.6.3)

The transposition length of \( \pi_2 \pi_1 \) in either case is \( 2u \). We prove the theorem by finding an element \( g \) such that \( \ell_t (g) = u \) and \( g \pi_1 g^{-1} = \pi_2 \), which will show that \( d_{DCJ}^\ell (\pi_1, \pi_2) \leq u \).

Let \( g = (1, \pi_2 \pi_1(1), (\pi_2 \pi_1)^2(1) \ldots, (\pi_2 \pi_1)^u(1)) \). We claim that

\[ g \pi_1 g^{-1} = \pi_2. \]

The crux of the proof for the above claim is in the following assertion.

**Claim:** For any 2-cycle \((i, \pi_1(i))\) in \( \pi_1 \), either \( i \) or \( \pi_1(i) \) is moved by \( g \) but not both.

**Proof:**

If \( i \) is moved by \( g \), then \( g(i) = \pi_2 \pi_1(i) \).

**Case 1.** \( \pi_2 \pi_1 \) is a product of two cycles.

If \( \pi_2 \pi_1 \) is a product of two cycles then as proved in Lemma 4.5.3, \( i \) and \( \pi_1(i) \) are in different cycles of \( \pi_2 \pi_1 \). Since \( g \) is precisely one of the two cycles of \( \pi_2 \pi_1 \), it moves exactly one of \( i \) and \( \pi_1(i) \).
Case 2. $\pi_2 \pi_1$ is a single cycle.

If $\pi_2 \pi_1$ is a single cycle as in Eq (4.6.3), then suppose $\pi_1(1) = 1$ (if instead $\pi_2(1) = 1$, a similar argument would apply). Then,

$$1 = (\pi_2 \pi_1)^{2u+1} (1)$$
$$= \pi_2 (\pi_1 \pi_2)^{2u} (1).$$

This implies that $(\pi_1 \pi_2)^{2u} (1) = \pi_2(1)$. Now suppose

$$i = (\pi_2 \pi_1)^b (1),$$

for some $b$. Then

$$\pi_1(i) = (\pi_2 \pi_1)^a (i),$$

for some $a$, since $i$ and $\pi_1(i)$ are in the same cycle. Also,

$$\pi_1(i) = \pi_1 (\pi_2 \pi_1)^b (1)$$
$$= (\pi_1 \pi_2)^b (1)$$
$$= (\pi_1 \pi_2)^{-2u+b} (\pi_1 \pi_2)^{2u} (1)$$
$$= (\pi_1 \pi_2)^{-2u+b} \pi_2(1) = (\pi_1 \pi_2)^{-2u+b-1} (1)$$
$$= (\pi_2 \pi_1)^{2u-b+1} (1).$$

If $i$ is moved by $g$, that is if $i$ is in the cycle $(1, \pi_2 \pi_1(1), \ldots, (\pi_2 \pi_1)^u(1))$, then $b \leq u$ which means that $a = 2u - b + 1 > u$ and $\pi_1(i)$ is not in this cycle and hence not moved by $g$.

If $\pi_1(i)$ is moved by $g$, that is if $\pi_1(i)$ is in the cycle

$$(1, \pi_2 \pi_1(1), \ldots, (\pi_2 \pi_1)^u(1)),$$
then \( a = 2u - b + 1 \leq u \) which means that \( u + 1 \leq b \) and \( i \) is not in this cycle and hence not moved by \( g \).

Thus we have established that whether the product \( \pi_2 \pi_1 \) is given by Eq (4.6.2) or Eq (4.6.3), for any \( i \in C_s, g \) moves either \( i \) or \( \pi_1(i) \) but not both. This proves our claim.

So if \( i \) is moved by \( g \), then \( g(i) = \pi_2 \pi_1(i) \), and since \( \pi_1(i) \) is then not moved by \( g \), \( g(\pi_1(i)) = \pi_1(i) \). Consider the product \( g \pi_1 g^{-1} \). For any 2-cycle \((i, \pi_1(i))\) in \( \pi_1 \) suppose \( i \) is moved by \( g \). We have

\[
g(i, \pi_1(i)) g^{-1} = (g(i), g(\pi_1(i))) = (\pi_2(\pi_1(i)), \pi_1(i))
\]

which is the 2-cycle in \( \pi_2 \) containing \( \pi_1(i) \). Thus

\[
g \pi_1 g^{-1} = \pi_2.
\]

Since the transposition length of \( g \) is \( u \) (that is, we require \( u \) transpositions to express \( g \) as a product), we require \( u \) DCJ conjugation operations to sort \( \pi_1 \) into \( \pi_2 \). Thus \( d^{c}_{\text{DCJ}}(\pi_1, \pi_2) \leq u = \frac{1}{2} \ell_t(\pi_2 \pi_1) \).

As per the remarks made at the beginning of this proof, we have established that

\[
d^{c}_{\text{DCJ}}(\pi_1, \pi_2) = \frac{1}{2} \ell_t(\pi_2 \pi_1).
\]

\( \square \)

As there is nothing special about the choice of 1 in construction of \( g \), the cycle

\[
(\pi_1(1), \pi_2(1), \pi_2 \pi_1 \pi_2(1), \ldots, (\pi_2 \pi_1)^{u-1} \pi_2(1))
\]

is also an optimal sorting element.

The construction in Theorem 4.6.7 might be better understood through an example. Let \( \pi_1 \) and \( \pi_2 \) be the following genomic permutations on 6 regions:
\[ \pi_1 = (1, 6)(2, 3)(4, 5), \quad \pi_2 = (1, 2)(3, 4)(5, 6). \]

Since \( \pi_1 \) and \( \pi_2 \) have the same cycle structure, we know that they are conjugate in \( S_{2n} \). Consider the product \( \pi_1 \pi_2 \).

\[ \pi_1 \pi_2 = (1, 6)(2, 3)(4, 5)(1, 2)(3, 4)(5, 6) = (1, 3, 5)(2, 6, 4). \]

Let \( g = (1, 3, 5) \). Then \( g^{-1} = (1, 5, 3) \).

\[ g \pi_2 g^{-1} = (1, 3, 5)(1, 2)(3, 4)(5, 6)(1, 5, 3) = (1, 6)(2, 3)(4, 5) = \pi_1. \]

A similar construction has been given in [38], where they construct the sorting element by establishing a correspondence between the connected components of adjacency graph and the permutation product.

Finally, putting together the lower bound for DCJ distance from Lemma 4.6.2 with the upper bound from Theorem 4.6.7, we have the following.

**Corollary 4.6.8.** If \( \pi_1 \) and \( \pi_2 \) are conjugate genomic permutations on \( n \) regions, then

\[ d_D(\pi_1, \pi_2) = \frac{1}{2} \ell_t(\pi_2 \pi_1). \]

Corollary 4.6.8 allows us to summarise the discussion about the element sorting \( \pi_1 \) into \( \pi_2 \).

**Lemma 4.6.9.** Let \( \pi_1, \pi_2 \) and \( u \) be as in Corollary 4.6.6. An element \( g \) such that \( \ell_t(g) = d_D(\pi_1, \pi_2) \) that sorts \( \pi_1 \) into \( \pi_2 \) can be constructed from the product \( \pi_2 \pi_1 \) as

\[ g = \left( 1, \pi_2 \pi_1(1), (\pi_2 \pi_1)^2(1), \ldots, (\pi_2 \pi_1)^u(1) \right). \]
4.6.4 DCJ distance between non-conjugate permutations

We will now consider the case where \( \pi_1 \) and \( \pi_2 \) are not conjugate in \( S_{2n} \).

In Section 4.6.2, we saw that the restriction of a genomic permutation to an equivalence class under the relation \( \sim \) can be sorted independently. Thus, as in the previous section, when we talk about non-conjugate permutations, we mean the permutations induced by a single equivalence class.

Recall that a genomic permutation is a product of disjoint 2-cycles. Every point in \( 2n \) is therefore either in a 2-cycle of \( \pi_1 \) or is fixed by it. The same is true for \( \pi_2 \).

Since \( \pi_1 \) and \( \pi_2 \) are assumed to be non-conjugate, they have different cycle structures. If \( F_{\pi_1} \cup F_{\pi_2} = \emptyset \), both \( \pi_1 \) and \( \pi_2 \) would have the same number of 2-cycles and hence the same cycle structure. Therefore, there must be at least one point fixed by \( \pi_1 \) or \( \pi_2 \). From Lemma 4.5.3, we know that the product \( \pi_2 \pi_1 \) is a cycle containing 2 points from \( F_{\pi_1} \cup F_{\pi_2} \) and hence there must be two points in \( F_{\pi_1} \cup F_{\pi_2} \) (and no more since we are dealing with sub-permutations). Once again, the requirement that \( \pi_1 \) and \( \pi_2 \) implies that the two fixed points belong to the same permutation for otherwise each permutation would have two fixed points and their cycle structures would be the same, making them conjugate.

We make use of the above facts in proving Theorem 4.6.10.

**Theorem 4.6.10.** Let \( \pi_1 \) and \( \pi_2 \) be non-conjugate genomic permutations on \( n \) regions. Then

\[
d_D(\pi_1, \pi_2) = \frac{1}{2} (\ell_1(\pi_2 \pi_1) + 1).
\]

**Proof.** The preceding discussion establishes that \( \pi_1 \) and \( \pi_2 \) have two fixed points between them and both the fixed points belong to the same permutation.
Suppose the fixed points are \( i_1 \) and \( i_2 \) and that they belong to \( \pi_1 \). Recall that if \( i, j \) are both fixed in \( \pi \) then \( d_{i,j}(\pi) = (i, j)\pi \), and hence

\[
d_{i_1,i_2}(\pi_1) = (i_1, i_2)\pi_1.
\]

The number of 2-cycles in \( \pi_1 \) is one less than the number of 2-cycles of \( \pi_2 \), since it has two fixed points. Therefore \( d_{i_1,i_2}(\pi_1) \) is conjugate to \( \pi_2 \). Similarly, if the two fixed points belong to \( \pi_2 \) then \( d_{i_1,i_2}(\pi_2) \) is conjugate to \( \pi_1 \).

Let \( \pi' = d_{i_1,i_2}(\pi_1) \). From Theorem 4.6.7 we have that

\[
d_D(\pi', \pi_2) = \frac{1}{2} \ell_t(\pi_2 \pi') = \frac{1}{2} \ell_t(\pi' \pi_2) = \frac{1}{2} \ell_t((i_1, i_2)\pi_1 \pi_2).
\]

Since \( i_1 \) and \( i_2 \) are in the same cycle of \( \pi_1 \pi_2 \), multiplication by \((i_1, i_2)\) will split this cycle into two cycles, reducing the transposition length of the product by 1. Hence

\[
d_D(\pi', \pi_2) = \frac{1}{2} \ell_t((i_1, i_2)\pi_1 \pi_2) = \frac{1}{2} (\ell_t(\pi_1 \pi_2) - 1).
\]

The DCJ distance between \( \pi_1 \) and \( \pi' \) is 1. Thus the triangle inequality gives

\[
d_D(\pi_1, \pi_2) \leq d_D(\pi_1, \pi') + d_D(\pi', \pi_2)
\]

\[
= \frac{1}{2} (\ell_t(\pi_1 \pi_2) - 1) + 1
\]

\[
= \frac{1}{2} (\ell_t(\pi_1 \pi_2) + 1).
\]

At the same time we have a lower bound on the distance (Lemma 4.6.2), so that

\[
\frac{1}{2} \ell_t(\pi_1 \pi_2) \leq d_D(\pi_1, \pi_2) \leq \frac{1}{2} (\ell_t(\pi_1 \pi_2) + 1).
\]
Since the DCJ distance is an integer (the number of DCJ operations), and the transposition length of $\pi_1\pi_2$ is odd, we have

$$d_D(\pi_1, \pi_2) = \frac{1}{2} (\ell_t(\pi_1\pi_2) + 1)$$

as required. \hfill \Box

4.6.5 DCJ distance as the sum of distances between sub-permutations

As we have already seen, the sub-permutations induced by an equivalence class $C_i$ under $\sim$ can be sorted independently of the sub-permutations induced by the other equivalence classes. Therefore

$$d_D(\pi_1, \pi_2) \leq \sum_s d_D\left(\pi_1^{(s)}, \pi_2^{(s)}\right).$$

We claim that a sequence of DCJ operations sorting the induced sub-permutations that involves a DCJ operation $d_{ij}$, where $i, j$ are in different equivalence classes under $\sim$ cannot be shorter than a sequence that sorts each partition independently.

Let $C_r$ and $C_s$ be distinct equivalence classes of $2n$ under $\sim$. We write $d_D(C_r)$ to denote the DCJ distance of the sub-permutation induced by $C_r$ from the identity. Then we have the following result.

**Theorem 4.6.11.** Let $\pi_1$ and $\pi_2$ be genomic permutations on the set $2n$. Let $\pi_1^{(s)}$ and $\pi_2^{(s)}$ be sub-permutations induced by an equivalence class $C_s$ under $\sim$. Then

$$d_D(\pi_1, \pi_2) = \sum_s d_D\left(\pi_1^{(s)}, \pi_2^{(s)}\right).$$
Proof. Let \( \ell_t\left(\pi_1^{(r)}\pi_2^{(r)}\right) \) be \( l_r \) and \( \ell_t\left(\pi_1^{(s)}\pi_2^{(s)}\right) \) be \( l_s \). Then from the DCJ distances of conjugate and non-conjugate sub-permutation, we know that

\[
\frac{1}{2} (l_r) \leq d_D\left(\pi_1^{(r)}\pi_2^{(r)}\right) \leq \frac{1}{2} (l_r + 1).
\]

Similarly

\[
\frac{1}{2} (l_s) \leq d_D\left(\pi_1^{(s)}\pi_2^{(s)}\right) \leq \frac{1}{2} (l_s + 1).
\]

Combining these, we have

\[
\frac{1}{2} (l_r + l_s) \leq d_D(C_r) + d_D(C_s).
\]

Suppose \( i \in C_r \) and \( j \in C_s \). The action of \( d_{i,j} \) on \( \pi_1 \) may combine the two partitions \( C_r \) and \( C_s \) into a single partition \( C_t \) or it may change each of the partitions \( C_r \) and \( C_s \). In the latter case, by an abuse of notation we use \( C_t \) to denote the union of the partitions changed by the action of \( d_{i,j} \). We wish to determine the transposition length of \( \pi_1^{(t)}\pi_2^{(t)} \) in order to find the DCJ distance.

The action of \( d_{i,j} \) on \( \pi_1 \) may be left multiplication by \((i,j)\) or conjugation by \((i,j)\). If it acts by left multiplication, so that \( d_{i,j} (\pi_1) = (i,j)\pi_1 \), then since \( i \) and \( j \) are in different equivalence classes and hence in different cycles of \( \pi_1^{(r)}\pi_2^{(r)}\pi_1^{(s)}\pi_2^{(s)} \), multiplication by \((i,j)\) will combine the two cycles that contain \( i \) and \( j \). The transposition length of the product will therefore increase by 1. That is, in this case,

\[
\ell_t\left(\pi_1^{(t)}\pi_2^{(t)}\right) = \ell_t\left(\pi_1^{(r)}\pi_2^{(r)}\pi_1^{(s)}\pi_2^{(s)}\right) + 1.
\]

If \( d_{i,j} \) acts by conjugation then \( d_{i,j} (\pi_1) = (i,j)\pi_1 (i,j) \), and we have that

\[(i,j)\pi_1 (i,j) = (i,j) (\pi_1 (i), \pi_1 (j)) \pi_1.
\]
The images $\pi_1(i)$ and $\pi_1(j)$ are in different cycles of $\pi_r^{(r)} \pi_2^{(r)} \pi_1^{(s)} \pi_2^{(s)}$ since $i$ and $j$ are in different equivalence classes. In $x = (\pi_1(i), \pi_1(j)) \pi_1 \pi_2$, the cycles containing $\pi_1(i)$ and $\pi_1(j)$ will combine into a single cycle, increasing the length of the product by 1. Then the cycle of $x$ containing $\pi_1(i)$ and $\pi_1(j)$ with contain either both, one, or neither of $i$ and $j$.

Accordingly, multiplication by $(i, j)$ will either split this cycle into two cycles or combine two different cycles into one cycle. Thus the transposition length may increase or decrease by 1 (from the previous step). Hence

$$\ell_t \left( \pi_1^{(t)} \pi_2^{(t)} \right) = \ell_t \left( \pi_1^{(r)} \pi_2^{(r)} \pi_1^{(s)} \pi_2^{(s)} \right),$$

or,

$$\ell_t \left( \pi_1^{(t)} \pi_2^{(t)} \right) = \ell_t \left( \pi_1^{(r)} \pi_2^{(r)} \pi_1^{(s)} \pi_2^{(s)} \right) + 2.$$

In both cases $l_r + l_s \leq \ell_t(C_t)$ and hence

$$\frac{1}{2} (l_r + l_s) \leq d_D \left( \pi_1^{(t)} \pi_2^{(t)} \right).$$

Since one DCJ operation was needed to change the partition $C_r$ and $C_s$ into $C_t$, and the DCJ distance of the sub-permutations induced by $C_t$ is at least $\frac{1}{2} (l_r + l_s)$, any sorting sequence for $C_r$ and $C_s$ that steps through $C_t$ is of length at least $\frac{1}{2} (l_r + l_s) + 1$. At the same time, the sum of the distances of the partitions $C_r$ and $C_s$ is bounded above by:

$$d_{C_r} + d_{C_s} \leq \frac{1}{2} (l_r + l_s) + 1.$$

Therefore we conclude that no sequence of DCJ operations sorting $\pi_1$ into $\pi_2$ can be shorter than a sequence that sorts the sub-permutations independently. □
Theorems 4.6.11, 4.6.7 and 4.6.10 immediately lead to the following conclusion about the total DCJ distance.

**Theorem 4.6.12.** Let $\pi_1$ and $\pi_2$ be genomic permutations on $n$ regions. The DCJ distance between $\pi_1$ and $\pi_2$ is given by

$$d_D(\pi_1, \pi_2) = \frac{1}{2}(\ell_1(\pi_2\pi_1) + n_c)$$

where $n_c$ is the number of cycles in the product $\pi_2\pi_1$ which contain two fixed points of $F_{\pi_1}$ or $F_{\pi_2}$.

4.7 **Counting the optimal sorting scenarios**

To sort a permutation $\pi_a$ into $\pi_b$ means to transform $\pi_a$ into $\pi_b$ through a sequence of allowed operations (in this case the DCJ operation). A sorting scenario is defined as follows.

**Definition 4.7.1.** A sorting scenario of length $k$ that sorts genomic permutation $\pi_a$ into genomic permutation $\pi_b$ is a sequence of genomic permutations

$$\{(\pi_a =)\pi_0, \pi_1, \pi_2, \ldots, \pi_{k-1}, \pi_k(= \pi_b)\},$$

such that each element of the sequence is obtained from the previous element through a single DCJ operation.

That is, a sorting scenario is the sequence of permutations we step through as $\pi_a$ is sorted into $\pi_b$ through DCJ operations. If the DCJ distance is $d$, then an optimal sorting scenario (scenario of minimal length) is a sequence of length $d + 1$. Two optimal sorting scenarios are equal if they are equal as sequences i.e., corresponding terms are equal.
The total number of optimal sorting scenarios between a pair of genomes is an interesting and important question. In constructing a phylogenetic history, the minimal distance with respect to some mutational operation is used. However such a minimal path is seldom unique. Hence as Miklós and Darling [70] and Siepel [90] point out, it would be more appropriate to account for and average over all possible evolutionary paths to draw meaningful statistical inferences.

Braga and Stoye [21] extended their earlier work [20] to give a closed formula for the number of optimal DCJ sorting scenarios for certain instances of the problem. Ouangraoua and Bergeron [74] also present similar results for the number of optimal sorting scenarios and establish connections between the number of sorting scenarios and other combinatorial objects such as parking functions.

Considering genomes as permutations and the DCJ as an action on a permutation allows us to count the number of optimal sorting scenarios in some special cases in a straightforward manner. Our results are equivalent to the results obtained by the previous papers.

4.7.1 Sorting scenarios between conjugate permutations

We first consider genomic permutations that are conjugate in \( S_{2n} \). Once again, we emphasise that sub-permutations induced by the equivalence classes under \( \sim \) can be sorted independently (4.6.2). Therefore, throughout this section, when we talk about conjugate permutations, we are talking of conjugate sub-permutations.

Before proving our main theorem about the number of optimal sorting scenarios, we establish some other observations that will be used to prove the result about this count.
Our first result characterises the DCJ operations that have the same effect on a permutation $\pi$.

Lemma 4.7.2. Let $\pi$ be a genomic permutation on $n$ regions. Let $d_{i,j}$ and $d_{k,l}$ act on $\pi$ via conjugation. If

$$d_{i,j}(\pi) = d_{k,l}(\pi),$$

then either $(i, j) = (k, l)$ or $(i, j)$ and $(k, l)$ are disjoint transpositions such that $(k, l) = (\pi(i), \pi(j))$.

Proof. The restriction of $\pi$ to the cycles containing $i$ and $j$ is $(i, \pi(i)) (j, \pi(j))$. As $d_{i,j}$ acting on $\pi$ only affects the cycles containing $i$ and $j$, we write the product of the other cycles in $\pi$ as $w$ and note the result of the action of $d_{i,j}$ on $\pi$:

$$d_{i,j}(\pi) = (i, j) (i, \pi(i)) (j, \pi(j)) w(i, j)$$

Now we know that $g(i_1, i_2, \ldots i_k) g^{-1} = (g(i_1), g(i_2), \ldots g(i_k))$. Therefore,

$$g(i, \pi(i)) (j, \pi(j)) g^{-1} = (g(i), g(\pi(i))) (g(j), g(\pi(j))).$$

We want that the RHS should be equal to $(i, \pi(j)) (j, \pi(i))$. We can choose the action of $g$ on $i, \pi(i), j$ and $\pi(j)$ to make this happen. The table below lists all the possible choices for the images of these elements under $g$ and the resulting element.
<table>
<thead>
<tr>
<th></th>
<th>$g(i)$</th>
<th>$g(\pi(i))$</th>
<th>$g(j)$</th>
<th>$g(\pi(j))$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j$</td>
<td>$\pi(i)$</td>
<td>$\pi(j)$</td>
<td>$i$</td>
<td>$(i,j,\pi(j))$</td>
<td></td>
</tr>
<tr>
<td>$\pi(i)$</td>
<td>$j$</td>
<td>$\pi(j)$</td>
<td>$i$</td>
<td>$(i,\pi(i),j,\pi(j))$</td>
<td></td>
</tr>
<tr>
<td>$\pi(j)$</td>
<td>$i$</td>
<td>$\pi(i)$</td>
<td>$j$</td>
<td>$(i,\pi(j),j,\pi(i))$</td>
<td></td>
</tr>
<tr>
<td>$i$</td>
<td>$\pi(j)$</td>
<td>$\pi(i)$</td>
<td>$j$</td>
<td>$(\pi(i),\pi(j),j)$</td>
<td></td>
</tr>
<tr>
<td>$i$</td>
<td>$\pi(j)$</td>
<td>$j$</td>
<td>$\pi(i)$</td>
<td>$(\pi(i),\pi(j))$</td>
<td></td>
</tr>
<tr>
<td>$\pi(j)$</td>
<td>$i$</td>
<td>$j$</td>
<td>$\pi(i)$</td>
<td>$(i,\pi(j),\pi(i))$</td>
<td></td>
</tr>
<tr>
<td>$j$</td>
<td>$\pi(i)$</td>
<td>$i$</td>
<td>$\pi(j)$</td>
<td>$(i,j)$</td>
<td></td>
</tr>
<tr>
<td>$\pi(i)$</td>
<td>$j$</td>
<td>$i$</td>
<td>$\pi(j)$</td>
<td>$(i,\pi(i),j)$</td>
<td></td>
</tr>
</tbody>
</table>

Only two of the eight rows give us a 2-cycle. Thus the DCJ operation of conjugation by these 2-cycles gives us the desired result. If $\pi$ is of the form $(i)(j,\pi(j))$ then a similar argument shows that there is no DCJ operation other than $d_{i,j}$ itself that takes $\pi$ to $d_{i,j}(\pi)$.

If $\pi_1$ and $\pi_2$ are conjugate, then $\pi_1$ can be sorted into $\pi_2$ through a sequence of conjugation DCJ operations alone. We prove below that an optimal sequence of DCJ operations sorting $\pi_1$ into $\pi_2$ cannot contain any non-conjugate operations.

**Lemma 4.7.3.** Let $\pi_1$ and $\pi_2$ be genomic permutations on $n$ regions that are conjugate in $S_{2n}$. A minimal sequence of DCJ operations for sorting $\pi_1$ into $\pi_2$ does not contain any non-conjugate DCJ operations.

**Proof.** Since $\pi_1$ and $\pi_2$ are conjugate in $S_{2n}$, from Corollary 4.6.6 we know that either $F_{\pi_1} \cup F_{\pi_2} = \emptyset$ or $F_{\pi_1}$ and $F_{\pi_2}$ both contain 1 point each.

The only non-conjugation DCJ operation possible on $\pi_1$ is therefore splitting one of its 2-cycles, say $(i,j)$. Let us assume that this is the first DCJ operation in the sequence as the argument does not lose generality by this assumption.
\[ \pi' = d_{ij}(\pi_1) = (i, j)\pi_1. \]

The transposition length of \(\pi_2\pi'\) is equal to the transposition length of \(\pi'\pi_2\). We know that multiplication by a transposition can change the transposition length by at most 1. Therefore

\[ \ell_t(\pi_2\pi') = \ell_t(\pi'\pi_2) \geq d - 1. \]

Lemma 4.6.2 then gives us a lower bound on the DCJ distance between \(\pi'\) and \(\pi_2\).

\[
d_D(\pi', \pi_2) \geq \frac{1}{2}\ell_t(\pi'\pi_2)
\geq \frac{1}{2}(d - 1).
\]

A single DCJ operation is required to go from \(\pi_1\) to \(\pi'\). Therefore, if a non-conjugation DCJ operation is used, then the number of DCJ operations needed to sort \(\pi_1\) to \(\pi_2\) is at least \(1 + (d - 1)/2\), which is greater than the minimal number of DCJ operations needed \((d/2)\).

We are now ready to prove the main result that gives the number of optimal sorting scenarios between two conjugate genomic permutations. Let \(\pi_a\) and \(\pi_b\) be genomic permutations on \(n\) regions such that \(\pi_a\) and \(\pi_b\) are conjugate in \(S_{2n}\). Let \(N_{opt}\) be the number of optimal DCJ sorting scenarios sorting \(\pi_a\) into \(\pi_b\) and let the DCJ distance \(d_D(\pi_a, \pi_b) = d\). Then,

\[ N_{opt} = (d + 1)^{d-1}. \]

We will prove this result in two steps. In the first step, we establish a lower bound for the number of optimal sorting scenarios and in the second step, we
show that the number of sorting scenarios is bounded above by this number. The boxed text below lays out the strategy of the proof. The assertions made in this outline will be proved in Theorems 4.7.4 and 4.7.5.

There exists an element \( g \) of transposition length \( d \) such that \( g\pi_ag^{-1} = \pi_b \). Let \( S(g) \) be the set of all factorisations of \( g \). The cardinality of \( S(g) \) is \((d + 1)^{d-1}\). Each element of \( S(g) \) represents a distinct sorting scenario. Furthermore, if there is \( h \in S_{2n} \) such that \( \ell_1(h) = d \), then a sorting scenario produced by a factorisation of \( h \) into transpositions is the same as the sorting scenario produced by an element of \( S(g) \). Therefore, the total number of optimal sorting scenarios is \(|S(g)|\).

**Theorem 4.7.4.** Let \( \pi_a \) and \( \pi_b \) be genomic permutations on \( n \) regions such that \( \pi_a \) and \( \pi_b \) are conjugate in \( S_{2n} \). Let \( N_{opt} \) be the number of optimal DCJ sorting scenarios sorting \( \pi_a \) into \( \pi_b \) and let the DCJ distance \( d_D(\pi_a, \pi_b) = d \). Then,

\[
N_{opt} \geq (d + 1)^{d-1}.
\]

**Proof.** Lemma 4.6.9 gives the construction of a cycle \( g \in S_{2n} \) of length \( d + 1 \) (and consequently of transposition length \( d \)), and satisfying

\[
g\pi_ag^{-1} = \pi_b.
\]

Let \( g \) be as in the statement of Lemma 4.6.9 i.e.,

\[
g = \left(1, \pi_b\pi_a(1), (\pi_b\pi_a)^2(1), \ldots , (\pi_b\pi_a)^d(1)\right).
\]

The number of ways to represent a cycle of length \( n \) as the product of \( n - 1 \) transpositions (i.e., as a minimal product) is \( n^{n-2} \) [28]. Hence \( g \) can be written as a product of transpositions in \((d + 1)^{d-1}\) ways.
Let \( S(g) \) be the set of all expressions for \( g \) as a minimal product of transpositions. For example, if \( g = (1, 3, 5) \) then \( S(g) = \{(1, 5)(1, 3), (1, 3)(3, 5), (3, 5)(1, 5)\} \).

We claim that each expression of \( g \) as a minimal product of transpositions corresponds to a distinct sorting scenario.

From the construction of \( g \) in the proof of Theorem 4.6.7, we know that for any \( i \in 2n \), \( g \) moves either \( i \) or \( \pi_a(i) \) but not both. Suppose \( g \) moves \( i \). Then since \( g \) fixes \( \pi_a(i) \), in a minimal factorization of \( g \) as a product of transpositions, no transposition moves \( \pi_a(i) \).

To observe this, note that the number of trees on \( d + 1 \) labeled vertices is \((d + 1)^{d-1}\) (given by Cayley’s formula). Thus there is a bijection between the \( S(g) \) and the set of trees on \( d + 1 \) vertices. A minimal factorization of \( g \) into transpositions can be associated with a tree by considering a transposition \((i, j)\) to correspond to the edge between vertices \( i \) and \( j \). If a point \( \pi_a(i) \) fixed by \( g \) is moved by some transposition in a minimal factorization of \( g \), then the factorization must contain a cycle that would move \( \pi_a(i) \) back to itself. But such a cycle would correspond to a loop in the graph corresponding to the factorization, which cannot be as the graph is a tree. Hence in a minimal factorization of \( g \) as a product of transpositions, no transposition moves \( \pi_a(i) \).

Suppose \( u_d u_{d-1} \ldots u_1 \) and \( w_d w_{d-1} \ldots w_1 \) are distinct elements of \( S(g) \) that produce the same sorting scenarios. This means that the sequences,

\[
\{\pi_a, u_1(\pi_a)u_1, u_2u_1(\pi_a)u_1u_2, \ldots, u_d u_{d-1} \ldots u_1(\pi_a)u_1 \ldots u_{d-1}u_d\}
\]

and

\[
\{\pi_a, w_1(\pi_a)w_1, w_2w_1(\pi_a)w_1w_2, \ldots, w_d w_{d-1} \ldots w_1(\pi_a)w_1 \ldots w_{d-1}w_d\}
\]

are equal while \( u_d u_{d-1} \ldots u_1 \neq w_d w_{d-1} \ldots w_1 \). We will derive a contradiction.
Let $k$ be the lowest index such that $u_k \neq w_k$. Let $w_{k-1} \cdots w_1 = u_{k-1} \cdots u_1 = g'$ and let $g' \pi_a g'^{-1} = \pi_{k-1}$.

Let $u_k = (i, j)$. Since we have assumed that conjugating $\pi_{k-1}$ by $u_k$ or by $w_k$ produces the same result, we can use Lemma 4.7.2 to infer that $u_k$ and $w_k$ are disjoint and $w_k = (\pi_{k-1}(i), \pi_{k-1}(j))$.

Now, $u_k$ is a transposition in the minimal expression for $g$. Since $u_k$ moves $i$, $i$ is in the support of $g$, and $\pi_a(i)$ is not in the support of $g$. The support of $g'$ is a subset of the support of $g$, hence $\pi_a(i)$ is not in the support of $g'$.

If $u_k$ is disjoint from $g'$, then

$$\pi_{k-1}(i) = g' \pi_a g'^{-1}(i) = \pi_a(i).$$

Hence $\pi_{k-1}(i)$ is not in the support of $g$.

If, on the other hand, $u_k$ is not disjoint from $g'$, then let $i$ be in the support of $g'$.

Clearly, $g'^{-1}(i)$ is in the support of $g'$ (it gets mapped to $i$ by $g'$) and hence in the support of $g$. Therefore, $\pi_a((g')^{-1}(i))$ is not in the support of $g$ (and hence not in the support of $g'$) since for any $i \in 2n$, $g$ moves either $i$ or $\pi_a(i)$. Hence

$$\pi_{k-1}(i) = g' \pi_a g'^{-1}(i) = \pi_a \left( g'^{-1}(i) \right).$$

That is, $\pi_{k-1}(i)$ is not in the support of $g$.

Thus, in both cases (whether $u_k$ is disjoint from $g'$ or not), $w_k$ moves an element that is not in the support of $g$. This contradicts the assertion that $w_d w_{d-1} \cdots w_1 = g$. Thus either $w_d w_{d-1} \cdots w_1 \notin S(g)$ or the sorting scenarios produced by distinct elements are distinct. Each expression of $g$ as a minimal product of transpositions therefore gives a unique sorting scenario and

$$N_{opt} \geq |S(g)| = (d + 1)^{d-1}.$$
Theorem 4.7.5. Let $\pi_a$ and $\pi_b$ be genomic permutations on $n$ regions such that $\pi_a$ and $\pi_b$ are conjugate in $S_{2n}$. Let $N_{opt}$ be the number of optimal DCJ sorting scenarios sorting $\pi_a$ into $\pi_b$ and let the DCJ distance $d_D(\pi_a, \pi_b) = d$. Then,

$$N_{opt} = (d + 1)^{d-1}.$$  

Proof. As before, let $g$ be an element of transposition length $d$ such that $g\pi_a g^{-1} = \pi_b$ and let $S(g)$ be the set of all expressions for $g$ as a minimal product of transpositions.

From Theorem 4.7.4, we know that

$$N_{opt} \geq (d + 1)^{d-1}.$$  

Therefore to prove that $N_{opt} = (d + 1)^{d-1}$, we only need to show that $N_{opt} \leq (d + 1)^{d-1}$. Let $h \in S_{2n}, h \neq g$ such that $\ell_t(h) = \ell_t(g) = d$ and $h\pi_a h^{-1} = \pi_b$. Let $w_d w_{d-1} \ldots w_1$ be a minimal factorization of $h$ into transpositions so that

$$\{\pi_a, w_1 \pi_a w_1, \ldots, (w_d \ldots w_1) \pi_a (w_1 \ldots w_d)\}$$

constitutes an optimal sorting scenario. We claim that there exists some element $u_d u_{d-1} \ldots u_1 \in S(g)$ that produces the same sorting scenario as produced by $w_d w_{d-1} \ldots w_1$.

As the first step towards proving this assertion, we prove that there is some element $u_d \ldots u_1 \in S(g)$ that produces a sorting scenario with its second term the same as that of the scenario produced by $w_d w_{d-1} \ldots w_1$. That is, $S(g)$ contains some element $u_d \ldots u_1$ such that

$$u_1 \pi_a u_1 = w_1 \pi_a w_1.$$
Suppose that this is not the case. That is, no element in $S(g)$ produces a sorting scenario that has $w_1 \pi_a w_1$ as the second term (the first term in all sorting scenarios is $\pi_a$).

Consider the element $gw_1$. Since $\ell_t(g) = d$ and $w_1$ is a transposition, $\ell_t(gw_1)$ is either $d + 1$ or $d - 1$. If $\ell_t(gw_1) = d - 1$ we can write $gw_1$ as a product of $d - 1$ transpositions, say $gw_1 = v_{d-1} \ldots v_1$. Then $g = v_{d-1} \ldots v_1 w_1$. This is an expression of length $d$ equal to $g$ that is not in $S(g)$ because we have assumed that there is no element in $S(g)$ such that $u_1 \pi_a u_1 = w_1 \pi_a w_1$. This is a contradiction since $S(g)$ by definition contains all factorizations of $g$ of length $d$.

Now suppose that $\ell_t(gw_1)$ is $d + 1$. This happens if $w_1$ moves elements that are in different cycles of $g$. As $g$ consists of a single cycle, $w_1$ moves elements that are fixed by $g$. Suppose $w_1 = (i, j)$. Since $g$ fixes both $i$ and $j$, $g$ must move $\pi_a(i)$ and $\pi_a(j)$. Therefore there is at least one factorisation, say $y_d y_{d-1} \ldots y_1$, of $g$ that has $y_1 = (\pi_a(i), \pi_a(j))$ as the first factor.

$\ell_t(h) = d$ and $w_1$ is part of a minimal factorisation of $h$. Therefore, from Lemma 4.7.3 we know that $d_{i,j}$ must be a conjugation operation. Lemma 4.7.2 then tells us that $d_{i,j}(\pi_a) = d_{\pi_a(i), \pi_a(j)}(\pi_a)$.

Now we have the following facts: (1) there is at least one factorisation of $g$ that has $(\pi_1(i), \pi_1(j))$ as the first factor, and (2) $d_{i,j}(\pi_1) = d_{\pi_1(i), \pi_1(j)}(\pi_1)$. From these two statements, we can conclude that there is at least one element $y_d y_{d-1} \ldots y_1$ in $S(g)$ that produces a sorting scenario with $w_1 \pi_a w_1$ as its second term.

Let $S_1(g) = \{u_d \ldots u_1 \in S(g) \mid u_1 \pi_a u_1 = w_1 \pi_a w_1\}$.

That is, $S_1(g)$ consists of all the factorisations of $g$ that have $w_1 \pi_a w_1$ as the second term in the sorting scenario. Once again we argue that $S_1(g)$ contains some element $u_d \ldots u_1$ such that $u_2 (u_1 \pi_a u_1) u_2 = w_2 (u_1 \pi_a u_1) w_2$. 

110
In this way we continue to form nested subsets $S(g) \supseteq S_1(g) \cdots \supseteq S_k(g) \cdots \supseteq S_d(g)$ where for any $k$,

$$S_k(g) = \{ u_d \ldots u_1 \mid u_d \ldots u_1 \in S_{k-1}(g), \, u_k(u_{k-1} \ldots u_1 \pi_a u_1 \ldots u_{k-1})u_k = w_k(u_{k-1} \ldots u_1 \pi_a u_1 \ldots u_{k-1})w_k \}$$

Suppose that there does not exist any element in $S_k(g)$ such that

$$u_{k+1}(u_k \ldots u_1 \pi_a u_1 \ldots u_k)u_{k+1} = w_{k+1}(u_k \ldots u_1 \pi_a u_1 \ldots u_k)w_{k+1}.$$

We will repeat the argument we have made above to derive a contradiction.

Let $u_d \ldots u_1 \in S_k(g)$. Consider the element $u_d \ldots u_{k+1}w_{k+1}$. Let $u_d \ldots u_{k+1} = g'$. The transposition length of $g'$ is $d - k$. Therefore $\ell_t(u_d \ldots u_{k+1}w_{k+1})$ is either $d - k - 1$ or $d - k + 1$.

If $\ell_t(u_d \ldots u_{k+1}w_{k+1}) = d - k - 1$, then we can rewrite $u_d \ldots u_{k+1}w_{k+1}$ as a product of $d - k - 1$ transpositions say $v_{d-k} \ldots v_1$. That is,

$$u_d \ldots u_{k+1}w_{k+1} = v_{d-k} \ldots v_1.$$

This means that $u_d \ldots u_{k+1} = v_{d-k} \ldots v_1w_{k+1}$ and therefore,

$$u_d \ldots u_{k+1}u_k \ldots u_1 = v_{d-k} \ldots v_1w_{k+1}u_k \ldots u_1.$$

Now $v_{d-k} \ldots v_1w_{k+1}u_k \ldots u_1$ is an expression of length $d$ equal to $g$ and prefix $u_k \ldots u_1$ that is not in $S_k(g)$ because of the assumption that there does not exist any element in $S_k(g)$ such that

$$u_{k+1}(u_k \ldots u_1 \pi_a u_1 \ldots u_k)u_{k+1} = w_{k+1}(u_k \ldots u_1 \pi_a u_1 \ldots u_k)w_{k+1}.$$
This contradicts the definition of $S_k(g)$.

If $\ell_t(u_d \ldots u_{k+1} w_{k+1}) = d - k + 1$, we know that $w_{k+1}$ moves elements that are in different cycles of $g' = u_d \ldots u_{k+1}$. Let $w_{k+1} = (i, j)$ and let $(u_k u_{k-1} \ldots u_1) \pi_a (u_1 \ldots u_{k-1} u_k) = \pi'_a$.

Since $g'$ fixes both $i$ and $j$, $g'$ must move $\pi'_a(i)$ and $\pi'_a(j)$. Therefore there is at least one factorisation of $g'$, say $y_d y_{d-1} \ldots y_1$, that has $y_1 = (\pi'_a(i), \pi'_a(j))$ as the first factor.

From Lemma 4.7.3 we know that $d_{i,j}$ must be a conjugation operation as $w_{k+1}$ is a part of a minimal sorting element and Lemma 4.7.2 then tells us that $d_{i,j}(\pi'_a) = d_{\pi'_a(i), \pi'_a(j)}(\pi'_a)$.

As earlier, we have at least one factorisation $y_d y_{d-1} \ldots y_1$ of $g'$ into transpositions that has $(\pi'_a(i), \pi'_a(j))$ as the first factor. We can use this factorisation to construct a element $y_d y_{d-1} \ldots y_1 u_k \ldots u_1$ of $S_k(g)$ such that

$$u_{k+1}(u_k \ldots u_1 \pi_a u_1 \ldots u_k) u_{k+1} = w_{k+1}(u_k \ldots u_1 \pi_a u_1 \ldots u_k) w_{k+1}.$$

By repeating this argument, we can conclude that there exists some element in $S(g)$ that gives the same sorting scenario as that produced by the factorisation of $h$. Therefore,

$$N_{opt} \leq |S(g)|.$$

The above observation coupled with the lower bound on $N_{opt}$ proved in Theorem 4.7.4 establishes the result.

### 4.7.2 Sorting scenarios between non-conjugate permutations

In the previous section, we established that if the DCJ distance between two conjugate genomic permutations is $d$, then the number of optimal sorting scenarios is $(d+1)^{d-1}$. In this section, we will show that the number of
optimal sorting scenarios for non-conjugate permutations is $d^{d - 2}$ if their DCJ distance is $d$.

Let $\pi_1$ and $\pi_2$ be non-conjugate genomic permutations on $n$ regions. As we saw in the proof of Theorem 4.6.10, if the permutations $\pi_2$ and $\pi_1$ are not identical, their product contains exactly two fixed points of $\pi_1$ or $\pi_2$. Since $\pi_1$ and $\pi_2$ are not conjugate, both the fixed points belong to the same permutation. Suppose the fixed points are $i_1$ and $i_2$. Without loss of generality, we can assume that they belong to $\pi_1$.

Let the DCJ distance between $\pi_1$ and $\pi_2$ be $d$. There are three choices for the first operation in a sequence of DCJ operations sorting $\pi_1$ into $\pi_2$.

1. The first DCJ operation is $d_{i_1,i_2}$.

2. The first DCJ operation is $d_{i,j}$ where $i,j$ are not fixed in $\pi_1$ and $i,j$ are not in the same 2-cycle of $\pi_1$.

3. The first DCJ operation is $d_{i,j}$ where $i,j$ are not fixed in $\pi_1$ and $\pi_1(i) = j$.

Let $d_{i,j}(\pi_1) = \pi'$. We will consider the effect of each of these choices on the DCJ distance between $\pi'$ and $\pi_2$. Once we have characterised an optimal sequence of DCJ operations, we can then count the number of optimal sorting scenarios.

Case 1. The first DCJ operation is $d_{i_1,i_2}$.

As seen in the proof of Theorem 4.6.10, $\pi'_1 = d_{i_1,i_2}(\pi_1)$ is conjugate to $\pi_2$. The DCJ distance between $\pi'_1$ and $\pi_2$ is $d - 1$.

Case 2. The first DCJ operation is $d_{i,j}$ where $i,j$ are not fixed in $\pi_1$ and $i,j$ are not in the same 2-cycle of $\pi_1$.

If $i,j$ are not fixed in $\pi_1$ and $i,j$ are not in the same 2-cycle of $\pi_1$, then

$$
\pi'_1 = d_{i,j}(\pi_1) = (i,j)\pi_1(i,j).
$$
A conjugation operation on $\pi_1$ does not change the cycle type of $\pi_1$. Therefore, as before, $\pi'_1$ has two fixed points and $\pi_2$ has none. We can once again use Lemma 4.5.3 to determine the cycle structure of the product $\pi'_1 \pi_2$. $\pi'_1 \pi_2$ would consist of a single cycle that contains all the elements moved by $\pi'_1$ and $\pi_2$. Utilising the language of sub-permutations, we can say that partition is not changed by the conjugation operation. Therefore, the transposition length of $\pi'_1 \pi_2$ is the same as the transposition length of $\pi_1 \pi_2$.

The conjugation operation does not change the number of fixed points of $\pi_1$. Since the DCJ distance between two non-conjugate genomic permutations depends on the transposition length of $\pi_1 \pi_2$, the DCJ distance between $\pi'_1$ and $\pi_2$ is $d$.

Case 3. The first DCJ operation is $d_{i,j}$ where $i, j$ are not fixed in $\pi_1$ and $\pi_1(i) = j$

In this case $\pi'_1 = d_{i,j}(\pi_1) = (i, j) \pi_1$.

Recall from Lemma 4.5.3 in Section 4.5 that the product of $\pi_1$ and $\pi_2$ is a single cycle. Left multiplication by the 2-cycle $(i, j)$ splits this cycle into two cycles and reduces the transposition length by 1 in this case. Hence $\ell_t(\pi'_1 \pi_2) = \ell_t(\pi_1 \pi_2) - 1$.

The product $\pi'_1 \pi_2$ consists of two disjoint cycles each of which contains two fixed points. This is because $\pi_1$ had two fixed points and since these were not changed by the DCJ operation, they are also fixed in $\pi'_1$. In addition to these two fixed points, the DCJ operation has created two new fixed points, $i$ and $j$. Recall from Theorem 4.6.12 that the DCJ distance between $\pi'_1$ and $\pi_2$ is

$$d_D(\pi'_1, \pi_2) = \frac{1}{2} \left( \ell_t(\pi_2 \pi'_1) + n_c \right)$$

where $n_c$ is the number of cycles in the product $\pi_2 \pi'_1$ that contain two fixed points of $F_{\pi'_1}$ or $F_{\pi_2}$.
Therefore,
\[
d_D(\pi'_1, \pi_2) = \frac{1}{2} (\ell_{\ell}(\pi'_1 \pi_2) + 2) \\
= \frac{1}{2} (\ell_{\ell}(\pi_1 \pi_2) - 1 + 2) \\
= \frac{1}{2} (\ell_{\ell}(\pi_1 \pi_2) + 1) \\
= d_D(\pi_1, \pi_2).
\]

Thus we see that of the three choices available for the first DCJ operation, the only one that reduces the DCJ distance by 1 is \(d_{i_1, i_2}\). Hence an optimal sequence of DCJ operations sorting \(\pi_1\) into \(\pi_2\) must begin with \(d_{i_1, i_2}\).

Since the first DCJ operation in an optimal sorting sequence does not involve any arbitrary choices, the number of optimal sorting scenarios between \(\pi_1\) and \(\pi_2\) is the number of optimal sorting scenarios between \(d_{i_1, i_2}(\pi_1)\) and \(\pi_2\). As we have already seen, \(d_{i_1, i_2}(\pi_1)\) and \(\pi_2\) are conjugate in \(S_{2n}\). Since the DCJ distance between \(d_{i_1, i_2}(\pi_1)\) and \(\pi_2\) is \(d - 1\), from Theorem 4.7.5, we know that the number of optimal sorting scenarios is \(d^{d-2}\).

**Theorem 4.7.6.** Let \(\pi_a\) and \(\pi_b\) be genomic permutations on \(n\) regions such that \(\pi_a\) and \(\pi_b\) are not conjugate in \(S_{2n}\). If the DCJ distance \(d_D(\pi_a, \pi_b) = d\) then the number of optimal DCJ sorting scenarios sorting \(\pi_a\) into \(\pi_b\) is \(d^{d-2}\).

### 4.8 Summary and Conclusions

In this chapter we presented a rigorous algebraic treatment of the DCJ theory and provided new proofs for some of the results related to the DCJ operator. The algebraic treatment suggests a new way of looking at some of the problems arising from the DCJ literature. One example of such a problem is determining the distance between the sorting sequences. Braga and Stoye [21]
show that any optimal DCJ sorting sequence can be transformed into another optimal DCJ sorting sequence through a limited number of operations and raise the question of finding the minimum number of operations needed to achieve this transformation. Another very interesting and useful question is that of finding a weighted DCJ distance. As we have seen, the DCJ operator is able to generate a large number of rearrangement operators. A useful extension to the DCJ distance would be weight the various rearrangements as a sequence is sorted into another. For instance, the weights could be based on the observed frequencies of the various rearrangements, or on the length of the genome that is affected by the rearrangement. This could be a challenging problem as the DCJ operator acts on the genome content rather than genome positions and its action on a genome is determined by the structure of the genome.

In the next chapter, we deal with a simpler variant of the problem of finding the weighted rearrangement distance. Utilising the language we have developed in Chapter 3, in our formulation of the problem the genome is represented as a map from positions to regions. Hence the rearrangement operators are maps from positions to positions and it is possible to assign weights to the operators that do not vary as the genome is sorted.
“But if you don’t know anything about Cathy, I don’t see how any theory will help.”

Vadez sighed, not unpleasantly . . . “My friend, if you know all about Cathy, do you think you would be able to find her?”

“I’m sure I could,” Marvin said.

“Even without knowing the Theory of Searches?”

“Yes.”

“Well then, apply the same reasoning to the reverse condition. I know all there is to know about the Theory of Searches, and therefore I need to know nothing about Cathy.”

Mindswap, Robert Sheckley

5.1 Introduction

As we have seen so far in this thesis, a number of methods are used by researchers to determine the distance between genome arrangements in terms of a single rearrangement operator or a combination of rearrangement operators. In addition to inversions, translocations of chromosomal fragments
[7, 8, 101], fission/fusion of chromosomes, duplication of sequences [22], deletion/insertion and a combination of these different operators [100] have all been considered as rearrangement operators.

The rearrangement distance between a pair of genomes is usually defined as the minimal number of events from the set of allowed operations required to transform one genome into another. For instance, in determining inversion distance between two genomes, the set of legal operations consists of all possible inversions on a gene sequence. Initial solutions in the case of inversions involve finding the smallest number of inversion events between two genomes and the distance was the count of the events. Thus, each inversion event carries the same weight. If the weight assigned to an inversion event represents the probability of that event, then a model where all events have the same weight can be thought of as finding distances under the uniform distribution. In Chapter 3 (see Section 3.1), we briefly introduced some on these methods. One of the most popular of inversion distance algorithms is the Hannenhali-Pevzner [51] approach draws a graph based on the genomes and calculates the minimal distance as a function of features of the graph (for example the number of cycles and paths).

These methods are simple and fast and have been implemented in software for use by the research community [85, 89, 95]. As pointed out above, an implicit assumption underlying most of these methods is that all rearrangement operators included in the model are equally probable and thus are given the same weight in the rearrangement distance. For example, in determining the reversal distance, all reversals are considered equally likely irrespective of the number of regions they reverse. An inversion model that lies at the other extreme is one that allows only very short inversions. The well-known bubble sort uses only adjacent inversions for sorting. A group-theoretic model for sorting circular permutations using inversions acting on two adjacent regions
was described by [35]. In a similar vein, [43] presents an approximation algorithm for sorting signed permutations by only length 2 reversals while [23] gives a characterisation of linear and circular permutations that can be sorted by only length $k$ reversals for a fixed $k$.

The biological evidence however points somewhere between these two extremes. For example, focusing only on the evidence related to inversions, several studies have suggested that inversions of a short chromosomal fragment are more frequent than that of longer fragments [25, 36, 65, 88]. Similarly, [88] found a high prevalence of short inversions in the yeast genome. They observe that the conservation of a small neighborhood of genes, without absolute conservation of order or orientation, suggests that small DNA inversions have contributed significantly to the evolution of ascomycete genomes. In an analysis of four pairs of related bacterial genomes, [65] report an over-representation of short inversions, especially those involving a single gene, in comparison with a random inversion model. Analysis of the genome of Y. Pestis has also found that all inversions were shorter than expected under a neutral model [25].

In view of this information accruing from biology, a natural extension to the definition of rearrangement distance as the minimal number of rearrangement operators suggests itself. Although any scientific model is at best a partial representation of reality, a model that allows for assigning weights derived from empirical information to the operators, and calculates the minimal weighted distance between genome arrangements, might be a better approximation of the underlying biology. Thus it would be useful to have methods of determining inversion distances where the use of an inversion operator can be penalised based on the number of genes/regions it affects, or the different operators in a model may be weighted based on type.
One of the first algorithms for determining rearrangement distance, proposed in [83, 86], is in principle capable of assigning weights for inversions and transpositions. An approximation algorithm for sorting a permutation under a particular class of length sensitive cost models, where the cost function is additive i.e., $f(x) + f(y) = f(x + y)$, was presented in [78]. This approach has been generalized to a wider class of cost functions [10]. This work also improved the bounds on the cost for sorting using an additive cost function. Further pursuing this line of inquiry, [93] extend the results for signed permutations as well as circular permutations.

In this chapter, we present a flexible group-theoretic framework that can be used to determine the weighted rearrangement distance for any model of genome rearrangement in which the rearrangements allowed are invertible. Thus the framework we propose is applicable to models involving inversions and translocations, but not, for instance, insertions and deletions. The present work is based on the group theoretic approach of [35] and [40]. Throughout the chapter, we will focus on determining the minimal weighted reversal distance.

ABOUT THIS WORK

The work presented in this chapter has been carried out in collaboration with Prof Andrew Francis (Western Sydney University), Dr. Attila Egri-Nagy, Prof Cheryl E. Praeger (The University of Western Australia) and A/Prof. Volker Gebhardt (Western Sydney University). The project was conceptualised in discussions between Prof Andrew Francis and Prof Cheryl Praeger. The initial idea was to construct a library of rules to enable rewriting words in a group. Dr. Volker Gebhardt pointed out the connections between the initial idea and Knuth-Bendix algorithm and Dr Attila Egri-Nagy suggested that we explore
the KBMAG software. I carried out the background research, implemented
the GAP scripts using KBMAG to develop the examples and played a major
role in writing the results.

5.2 Path Deformation

The central idea of the method we propose in this chapter is ‘path deformation’. The genome space is the collection of all possible genomes. A path in
the genome space is a sequence of genomes where the consecutive elements
are connected through a single rearrangement operator. To find the minimal
weighted distance between two genomes $A$ and $B$, we start by constructing a
path between them. At the same time, we have also constructed a library of
rules in this space. These rules consist of alternate paths, or shortcuts, for a
number of small paths. We scan the existing path for any subpaths that could
be replaced by a shortcut from our library generating a new, shorter path. In
this way, the existing path is deformed into a new path which is more opti-
mal than the original path although it might still not be the least weighted
path. Successive iterations of the deformation step should ideally lead us to
an optimal path.

The “rewriting rules” in themselves are easy to generate, given a group
defined by generators and relations (defined in Section 5.4). The relations,
together with the weighting functions, can be transformed to give a set of
rewriting rules. It is also not too difficult to construct an initial path between
the two genomes which can be edited using the rules in the library, at least for
some models of genome rearrangement. However it is not clear at the outset
in what sequence to apply the rules in such a way that one is guaranteed to
end with a minimal weight path from one genome to another. This is where
the theory of rewriting systems is used.
A rewriting system that is guaranteed to produce a minimal expression, regardless of the sequence in which the rewriting rules are applied, is called a \textit{confluent} rewriting system. We use the \textit{Knuth-Bendix algorithm} to transform our rewriting system into a confluent system and use it to construct a minimal weighted path between two genomes given an existing path between them.

5.3 \textsc{Group theoretic inversion systems}

The notion of an \textit{inversion system} was formalised in [35]. Since our work uses much of the language, we briefly summarise the key concepts in this section followed by an extension to a weighted inversion system.

\textit{Genomes as permutations and inversion as an action}

As we have seen in Chapter 3, a chromosome is represented as a map from a set of positions \( n = \{1, 2, \ldots, n\} \) to a set of regions \( X \), usually also labeled with the integers \( n = \{1, 2, \ldots, n\} \). If we denote the chromosome map by \( \pi \), we can write the arrangement in two-line notation as:

\[
\pi = \begin{pmatrix}
1 & 2 & \cdots & n \\
\pi_1 & \pi_2 & \cdots & \pi_n 
\end{pmatrix},
\]

where \( \pi_i = \pi(i) \). Recall from the discussion on the genome representation paradigms in Chapter 3, that the top row in this view represents the \( n \) positions on the chromosome and the bottom row represents the set of regions.

An unsigned inversion operator \( t_{i,j} \) in this paradigm is a map from positions to positions. Recall from Chapter 3 that when the genome is modeled as a map from positions to regions and a rearrangement operator is a bijection
on the set of positions, we require that the rearrangement operator act first on a position and then we map the new position to a region using the genome map. The function composition therefore is from left to right. For a detailed discussion of right and left actions see Chapter 3 or [16]. The inversion operator \( t_{i,j} \) maps \( \pi \) as follows:

\[
t_{i,j} \begin{pmatrix} \cdots & i & i+1 & \cdots & j & \cdots \\ \cdots & \pi_i & \pi_{i+1} & \cdots & \pi_j & \cdots \end{pmatrix} = \begin{pmatrix} \cdots & i & i+1 & \cdots & j & \cdots \\ \cdots & \pi_j & \pi_{j-1} & \cdots & \pi_i & \cdots \end{pmatrix}
\]

Thus the inversion operator \( t_{i,j} \) flipping regions in positions \( i \) to \( j \) can be written in cycle notation as follows:

\[
t_{i,j} := \begin{cases} (i,j)(i+1,j-1)\ldots(\frac{i+j}{2}-1, \frac{i+j}{2}+1) & \text{if } j-i \text{ is even,} \\ (i,j)(i+1,j-1)\ldots(\frac{i+j-1}{2}, \frac{i+j+1}{2}) & \text{if } j-i \text{ is odd.} \end{cases}
\]

For example, \( t_{1,6} = (1,6)(2,5)(3,4) \) and \( t_{1,5} = (1,5)(2,4) \). For genomes \( \pi_1 \) and \( \pi_2 \), if there is a sequence of \( k \) inversion operations such that

\[
t_{i_k,j_k} \ldots t_{i_1,j_1} \pi_1 = \pi_2,
\]

then since \( \pi_1 \) is a bijective map from the set of positions to regions, \( \pi_1^{-1} \) is well-defined and we can compose with \( \pi_1^{-1} \).

\[
t_{i_k,j_k} \ldots t_{i_1,j_1} e = \pi_2 \pi_1^{-1},
\]

where \( e \) is the function that maps position \( i \) to region \( i \). \( \pi_2 \pi_1^{-1} \) is a function from positions to positions and therefore an element of \( S_n \). Thus, as discussed in Section 3.4, the problem of determining a sequence of inversion operations that transforms \( \pi_1 \) into \( \pi_2 \) is equivalent to the problem of expressing the
group element \( \pi_2 \pi_1^{-1} \) as a product of the group elements corresponding to the rearrangement operators.

An inversion system is defined as a tuple \((G, I)\) where \(G\) is a group of permutations and \(I\) is a set of inversions such that \(G = \langle I \rangle\). That is, every permutation in \(G\) is expressible as a product of elements of \(I\). In group theoretic language, we say that \(I\) generates \(G\).

In general, if we have a subset \(S \subset G\) of elements from \(G\), then a word over \(S\) is a sequence of elements of \(S\). We use \(S^*\) to represent the set of all words over \(S\). Notice that \(S^*\) is an infinite set. In the discussion in this chapter, we will assume that \(S\) is closed under the operation of taking inverses i.e., for all \(s \in S, s^{-1} \in S\).

If \(S\) generates \(G\), then there is a natural map \(\Gamma: S^* \to G\) that sends a word \(w = [s_1, s_2, \ldots, s_k]\) to the group element \(g = s_1 s_2 \ldots s_k\). The brackets in \(w\) are used to emphasise that a word is an ordered sequence of elements of \(S\) and to distinguish the sequence from the product \(s_1 s_2 \ldots s_k\). \(S^*\) also contains the empty sequence which maps to the identity element of \(G\). The length of a group element \(g\) with respect to the generating set \(S\) is the smallest \(r \in \mathbb{N}\) such that there is some element \(w \in S^*, w = [s_1, \ldots, s_r]\) and \(\Gamma(w) = g\). That is,

\[
s_1 s_2 \ldots s_r = g.
\]

The inversion distance between permutations \(\pi_1\) and \(\pi_2\) is the length of the group element \(\pi_2 \pi_1^{-1}\) in the inversion system \((G, I)\). For details of inversion system, the reader is referred to [35].

The notion of the length of a group element can be extended to the weighted length of a group element. Suppose the elements of \(S\) are assigned (positive) weights. The weighted length of a word \(w = [s_1, s_2, \ldots, s_k]\) in \(S^*\) is the sum of the weights of the \(s_i\) where \(i\) runs from 1 through \(k\). The weighted length of
a group element \( g \) is obtained by taking an infimum over the set of all words in \( S^* \) that map to \( g \).

**Definition 5.3.1** (Weighted word length). Let \( S \) be a set of generators of a group \( G \). Let \( \omega \) be a bounded function \( \omega : S \to \mathbb{R}^+ \). The weighted word length of a group element \( g \in G \) is defined as

\[
\ell_{S,\omega}(g) := \inf \left\{ \sum_{i=1}^{t} \omega(s_i) \mid s_1s_2\ldots s_t = g, s_i \in S \right\}.
\]

The weighted word length of the identity element \( e \) of \( G \) is 0.

Similar to an inversion system, we can define a weighted inversion system to be a 3-tuple \((G, I, \omega)\) where \( G = \langle I \rangle \) as before and \( \omega : I \to \mathbb{R}^+ \).

### 5.4 Group Presentations

In order to develop our approach to the problem of determining weighted distance, we will make use of a powerful tool from group theory: group presentations. A group presentation is an abstract description of a group \( G \) in terms of a generating set \( S \) and set of relations \( R \) among the generators. Following [24, Chapter 1], we give a formal definition.

**Definition 5.4.1** (Group Presentation). Let \( G \) be a group and let \( e \) be the identity element of \( G \). A presentation \( \langle S \mid R \rangle \) for \( G \) consists of a generating set \( S \subseteq G \) and a set \( R \subseteq S^* \) such that

\[ \Gamma(R_i) = e \quad \text{for all } R_i \in R, \]

and for \( w \in S^* \), if \( \Gamma(w) = e \) then \( w \) is an algebraic consequence of the words in \( R \) and the group axioms.

That is, \( w \) is the same as the word we get by one or more of the following algebraic transformations: replace any occurrence of \( R_i, R_i \in R \) in \( w \) by the
empty word; and replace any occurrence of $gg^{-1}$ or $g^{-1}g$ in $w$ by the empty word for any $g \in S$

The elements of $\mathcal{R}$ are called relators. A group presentation may also be written as $\langle S \mid u_i = v_i, \quad i \in I \rangle$ where $u_i, v_i \in S^*$ as before and $I$ is an indexing set. An equation of the form $u = v$ in $S^*$ is referred to as a relation. The relation $u = v$ is equivalent to the relator $uv^{-1}$ as $u = v \iff uv_i^{-1} = e$ where $v_i^{-1}$ is the inverse of $v_i$ in $S^*$. It is worth noting at this point that both a relator and a relation can be thought of as an element of $S^* \times S^*$ as $(R_i, \emptyset)$ and $(u, v)$ respectively. We make use of this formulation later in Section 5.6.

A group $G$ can have many different generating sets and consequently many presentations.

For example, a presentation for the symmetric group $S_n$ with the generating set $S = \{s_i \mid s_i = (i, i + 1), 1 \leq i < n\}$ consists of the relations:

\[
\begin{align*}
    s_i^2 &= e \quad \forall 1 \leq i < n \\
    s_i s_j &= s_j s_i & \text{if } |i - j| > 1 \\
    s_i s_{i+1} s_i &= s_{i+1} s_i s_{i+1} & 1 \leq i < n - 1.
\end{align*}
\]

In particular for $S_4$, with the generating set $\{s_1, s_2, s_3\}$, we have the following set of relations

\[
\begin{align*}
    (R1) \quad s_1^2 &= e & (R4) \quad s_1 s_2 s_1 = s_2 s_1 s_2 \\
    (R2) \quad s_2^2 &= e & (R5) \quad s_2 s_3 s_2 = s_3 s_2 s_3 \\
    (R3) \quad s_3^2 &= e & (R6) \quad s_1 s_3 = s_3 s_1
\end{align*}
\]

The word $w = [s_2, s_3, s_2, s_1, s_3, s_1, s_2, s_3]$ satisfies $s_2 s_3 s_2 s_1 s_3 s_1 s_2 s_3 = e$ in $S_4$. As asserted in the definition of a group presentation, $w = e$ is an algebraic
consequence of the group axioms and the relations in the presentation of $S_4$. This can be seen by rewriting $w$ using the relations in the presentation and the group axioms.

\[
\begin{align*}
  s_2s_3s_1s_3s_2s_3 &= s_2s_3s_2s_1s_3s_1(s_3s_3)s_2s_3 \\
  &= s_2s_3s_2s_1s_3s_1s_3s_2s_3 \\
  &= s_2s_3s_2s_3s_2s_3 \\
  &= e.
\end{align*}
\]

The above example suggests how the relations might be developed into a set of rewriting rules and the process of rewriting carried out in a systematic manner. In Section 5.6, we will formalise the notion of such a rewriting system and discuss the properties that make a rewriting system effective.

### 5.5 Words on a Cayley Graph

Another useful way to understand relations and rewriting of words in groups is through a Cayley graph. For a group $G$ and a subset $S \subseteq G$, the Cayley graph $\mathcal{C}(G, S)$ of $G$ with respect to $S$ is a directed graph that has a vertex for each element of $G$. There is an edge from vertex $g$ to vertex $h$ if $gh^{-1} \in S$. That is, there is an edge labeled $s$ from $g$ to $h$ if there is some $s \in S$ such that $sh = g$. If $S$ generates $G$, then the Cayley graph $\mathcal{C}(G, S)$ is connected. In the rest of this discussion, we will talk about the Cayley graph of $G$ with respect to a generating set. It is also customary to assume that $e \notin S$ when constructing $\mathcal{C}(G, S)$.

The labels on the edges in a path from vertex $h$ to $g$ in $\mathcal{C}(G, S)$ give a word $w$ in $S^*$ such that $\Gamma(w) = gh^{-1}$. Recall that $\Gamma$ maps a word in $S^*$ to an
element in $G$. The length of a group element $g$ is the length of a shortest path between the identity vertex $e$ and $g$. If the edges of this graph are assigned weights, we can talk about the weighted path length between an ordered pair of vertices. In particular, a path in $C(G, S)$ from a vertex $g$ to itself gives a word $w$ that represents $e$. Since the Cayley graph of a group is vertex-transitive, any node in the graph may be fixed as the identity vertex. In the context of genome rearrangement models, a permutation is a genome arrangement. The generating set is the set of the allowed operators under this model. For instance, when inversions are considered to the only allowed rearrangement, the generating set is $I$. The set of all genome arrangements is the genome space which corresponds to the vertex set of the Cayley graph $C(G, I)$.

The process of rewriting words using relators is equivalent to deforming a path in a Cayley graph using loops as ‘shortcuts’. As we have seen, a word in $S^*$ that equals $e$ can be rewritten using the relators in the group presentation and the group axioms. On a Cayley graph, this can be understood as a closed path being constructed using the closed paths in the presentation as lego blocks.
Figure 16: A word in $S^*$ that equals $e$ can be rewritten using the relators in the group presentation and the group axioms. For instance, the word $s_1s_2s_3s_2s_5s_1s_3 = e$ in $S_4$. Walking along an edge is equivalent to multiplying by a generator (edge label). Starting in the top left corner and tracing the word clockwise, we get the word $s_1s_2s_3s_2s_5s_1s_3$. The closed path $s_1s_2s_3s_2s_5s_1s_3$ is constructed using the relators $s_1s_3s_1s_3$ and $s_2s_3s_2s_5s_2s_3$.

5.6 REWRITING SYSTEMS

Let $A$ be a set of symbols (also called alphabets) and let $A^*$ be the set of all words on $A$ including the empty word. A rewriting system $A$ on $A$ is a subset of $A^* \times A^*$. If $(l, r) \in A$, then for any word $w$ in $A^*$, an occurrence of $l$ in $w$ can be replaced by $r$.

In particular, we extend this idea to a set $S$ that generates a group $G$. Let $S$ and $S^*$ be as before. Recall that a presentation for $G$ over $S$ consists of a set of relators $\mathcal{R}$. Each relator in the set $\mathcal{R}$ is an element of $S^* \times S^*$. Thus $\mathcal{R}$ can be thought of as a rewriting system over $S$. If a word $w \in S^*$ is written as $w'$ using an element of $\mathcal{R}$, then $w$ and $w'$ are also equal as elements of $G$. Formally, $\Gamma(w) = \Gamma(w')$. Recall that $\Gamma$ is a map that sends a word in $S^*$ to the equivalent group element $g$. For $(l, r) \in \mathcal{R}$, we will write $l \rightarrow r$ and refer to $l$ as the left side of the rule and $r$ as its right side.
If we impose the constraint that $R$ be antisymmetric, (i.e., $(l, r) \in R \implies (r, l) \notin R$), then the rewriting process becomes directional. We write $R^*$ for the reflexive transitive closure of $R$. (The reflexive transitive closure of a binary relation $R$ on a set $X$ is the smallest reflexive transitive relation on $X$ that contains $R$.) That is, if $(x, y) \in R^*$, this means that $x$ can be reduced to (rewritten as) $y$ using the rules in $R$. We will write this as $x \xrightarrow{R^*} y$.

A word $w \in S^*$ is said to be reducible with respect to $R$ if there is some $z \in S^*$ such that $w \xrightarrow{R} z$. If no such $z$ exists, then $w$ is said to be irreducible with respect to $R$.

In applying the rewriting rules to rewrite a word, one may have to make choices at each step. For instance, a word may contain the left sides of more than one rule in $R$. For the process of rewriting to be effective, we need to ensure that a given word can be reduced to a unique irreducible word. In addition to this, an essential requirement is that this irreducible representative can be obtained by the application of rewriting rules in a finite number of steps. Formally, we talk about confluence and termination of a rewriting system.

**Definition 5.6.1** (Terminating rewriting system). A rewriting system $R$ over $S$ is said to be terminating if there is no infinite sequence of words $w_i \in S^*$ such that $w_0 \xrightarrow{R^*} w_1 \xrightarrow{R^*} \ldots w_k \ldots$.

**Definition 5.6.2** (Confluence). A rewriting system $R$ over $S$ is said to be confluent (Figure 17) if

for all $u, v, w \in S^*$, if $u \xrightarrow{R^*} v, u \xrightarrow{R^*} w$ then there exists $x \in S^*$ such that $v \xrightarrow{R^*} x$ and $w \xrightarrow{R^*} x$.

A set of defining relations (or relators) in a presentation can be turned into a rewriting system. To ensure that the rewriting system thus created is terminating and confluent, we will need to do some more work.
Figure 17: In a confluent rewriting system, if a word $u$ can be reduced to the words $v$ and $w$, then $v$ and $w$ can be reduced to some word $x$.

5.6.1 Termination

The termination of a rewriting system $\mathcal{R}$ can be established by imposing a reduction order on the set $S^*$. A reduction order on $S^*$ is a transitive relation $>$ such that for any $s, t \in S^*$,

- exactly one of the following holds: $s > t$, $t > s$ or $s = t$, and

- $s > t \implies asb > atb$, for all $a, b \in S^*$, and

- there is no infinite sequence of elements $s_0, s_1, \ldots, s_i, \ldots$ such that $s_i \geq s_{i+1}$.

The idea behind imposing $>$ on $S^*$ is that if $u \rightarrow v \implies u > v$, then an infinite sequence of words $w_i$ such that $w_i \rightarrow w_{i+1}$ induces an infinite decreasing sequence under $>$. Since the latter is not possible, $\mathcal{R}$ must terminate.

Recall that if a group $G$ is generated by a set $S$, then there is a natural map from $S^*$ to $G$ that maps words in $S^*$ to their product in $S$. The length of a word $s$ in $S^*$ is the number of letters in $s$ while the length of a group element $g$ represented by $s$ is the infimum over the subset of words in $S^*$ that map to $g$. This distinction between the length of an element of $G$ and that of a word in $S^*$ is worth bearing in mind as the relation $>$ is imposed on $S^*$ and not on a group generated by $S$. Thus, the constraints imposed by $>$ may not hold in $G$. For instance, let $S = \{(1,2), (3,4)\}$ and for $s, t \in S^*$, let $s > t$ if the length
of $s$ is greater than the length of $t$ i.e., $\ell(s) > \ell(t)$. $(1,2)(3,4)$ and $(1,2)$ are words in $S^*$ and,

$$(1,2)(3,4) > (1,2).$$

Multiplying on the left by $(1,2)$ and on the right by $(3,4)$ we get the words $(1,2)(1,2)(3,4)(3,4)$ and $(1,2)(1,2)(3,4)$ and,

$$(1,2)(1,2)(3,4)(3,4) > (1,2)(1,2)(3,4)$$

as required by the second condition in the definition of a reduction order. However, the word $(1,2)(1,2)(3,4)(3,4)$ equals $(\cdot)$ in the group $G$ generated by $S$ while $(1,2)(1,2)(3,4)$ is the group element $(3,4)$ with length 1. Since the length of $(\cdot)$ is 0, the relation $>$ on $S^*$ is clearly not a reduction order on $G$ but it is a reduction order on $S^*$.

**Definition 5.6.3 (Weighted Lexicographic Order).** Let $S$ be a non-empty finite set. Fix any ordering $\succ$ on the elements of $S$. Let $\omega : S \rightarrow \mathbb{R}^+$ be a function that assigns a positive weight to each element of $S$. Let $u = s_1s_2...s_k$ and $v = t_1t_2...t_l$ be in $S^*$. Define $u > v$ if either

(i) $\sum_{i=1}^{k} \omega(s_i) > \sum_{j=1}^{l} \omega(t_j)$

(ii) or if, $\sum_{i=1}^{k} \omega(s_i) = \sum_{j=1}^{l} \omega(t_j)$ and $s_i \succ t_i$ where $i = \min\{p : s_p \neq t_p\}$.

It is easy to see that the weighted lexicographic order is a reduction order.

**Proposition 5.6.4.** For any finite set $S$, weighted lexicographic order is a reduction order.

5.6.2 Confluence

If certain mathematical conditions are satisfied, a rewriting system can be transformed into a confluent rewriting system through a procedure due to
Knuth and Bendix [62]. We will discuss the Knuth-Bendix algorithm and the properties necessary for it to return a terminating, confluent rewriting system later in this section after introducing the necessary definitions.

Definition 5.6.5 (Critical Pair). Let $R$ be a rewriting system over $S^*$ as before. Let $u_1a \rightarrow v_1$ and $au_2 \rightarrow v_2$ be two rules in $R$ where $u_i, v_i, a \in S^*, a \neq e$. That is, a non-empty suffix of the left hand side of a rule overlaps a prefix of the left hand side of another rule. Rules $u_1a \rightarrow v_1$ and $au_2 \rightarrow v_2$ are said to overlap. The word $w = u_1au_2$ can be reduced to both $v_1u_2$ and $u_1v_2$. The words $v_1u_2$ and $u_1v_2$ constitute a critical pair.

A critical pair $(v_1u_2, u_1v_2)$ is said to be resolved if there exists $w \in S^*$ such that $v_1u_2 \xrightarrow{R^*} w$ and $u_1v_2 \xrightarrow{R^*} w$.

Theorem 5.6.6 ([2, Lemma 2.7.2]). A terminating rewriting system is confluent if and only if all its critical pairs are resolved.

The power of Theorem 5.6.6 derives from the fact that it allows us to ascertain confluence of a rewriting system by checking for confluence locally. This suggests a simple procedure for making a rewriting system confluent. Resolve each critical pair $(u, v)$ by adding a rule $u \rightarrow v$ if $u > v$ and $v \rightarrow u$ otherwise. This is the gist of the Knuth-Bendix algorithm.

However, it is not clear a priori that this process of resolving critical pairs and adding new rules would stop eventually. In fact, it is an established fact that the Knuth-Bendix algorithm halts with a confluent and terminating rewriting system if each equivalence relation generated by $R$ has finitely many equivalence classes [56, Corollary 12.21].

For a terminating and locally confluent rewriting system, each equivalence class under the equivalence closure of the relation generated by $R$ contains a unique, irreducible element. (The equivalence closure of a binary relation $B$ is the smallest possible equivalence relation that contains $B$.) Since each ele-
ment of $S^*$ maps to a group element, each unique, irreducible element maps to a unique group element. An equivalence class in $S^*$ under the equivalence closure of $\mathcal{R}$ consists of all the elements of $S^*$ that map to the same element in $G$ under $\Gamma$. Thus, the number of equivalence classes in $S^*$ under the equivalence closure of $\mathcal{R}$ must be finite if the group generated by $S$ is finite. Thus for a finite group, the Knuth-Bendix algorithm will give us the requisite set of rewriting rules.

The upshot of this observation is that for a genome rearrangement model, where the rearrangement operators are invertible, the Knuth-Bendix algorithm is guaranteed to generate a finite, confluent, terminating rewriting system since we are dealing with finite groups. The restriction that the operators be invertible is necessary to ensure that the operators generate a group.

In Section 5.7, we construct rewriting systems for two different weighted inversion models and use them to find the weighted distance for genomes.

5.7 IMPLEMENTATION AND BIOLOGICAL EXAMPLES

The first model consists of linear unsigned permutations on 7 regions. The set of inversions $\mathcal{I}$ consists of all inversions $t_{i,j}$, for $1 \leq i < j \leq 7$, where $t_{i,j}$ is (as defined in Section 5.3),

$$t_{i,j} := \begin{cases} (i,j)(i+1,j-1)\ldots\left(\frac{i+j}{2}-1,\frac{i+j}{2}+1\right) & \text{if } j-i \text{ is even,} \\ (i,j)(i+1,j-1)\ldots\left(\frac{i+j-1}{2},\frac{i+j+1}{2}\right) & \text{if } j-i \text{ is odd.} \end{cases}$$

For any $n$, the set of all inversions $t_{i,j}$ contains the transpositions $(i,i+1)$ for all $1 \leq i < n$. Since the set of all transpositions is a generating set for the symmetric group, the set of all inversions $t_{i,j}$ generates $S_n$ for any $n$. In particular, this set generates the symmetric group $S_7$. A simple monotonic weighting function is $\omega$ where $\omega(t_{i,j}) := j - i$. 


The first step in using the model presented in this chapter is to write a presentation for the group with this generating set.

\[ G = \langle I \mid (t_{i_1,j_1}t_{i_2,j_2})^m \rangle, \]

where \( m \) is the smallest integer such that \( (t_{i_1,j_1}t_{i_2,j_2})^m = e. \)

As we have already seen, the relators in this presentation constitute a rewriting system. This rewriting system may not be confluent but can be made confluent by using Knuth-Bendix algorithm. We used the software package KBMAG [55] to run Knuth-Bendix on this presentation. The resulting confluent rewriting system consists of 6220 rules. Note that the number of rules in a confluent rewriting system will usually be much larger than the number of relations in the input group presentation as in each iteration of the algorithm, new rules are added to make the rewriting system locally confluent.

KBMAG can also use the rewriting system to find a minimal representative for a given permutation.

We generated four random permutations in \( S_7 \) and determined the weighted distance matrix which was fed into RPhylip [81] to construct a phylogenetic tree (Figure 18b). For the same permutations, we also constructed phylogenies with the distance matrix from GRIMM [95] and the Coxeter distance matrix (Figure 18c and Figure 18a respectively). GRIMM assigns unit weight to all inversions. Coxeter generators are reversals of adjacent regions \((i, i+1)\). Therefore the inversion model underlying Coxeter distance assigns unit weight to reversals of adjacent regions and infinite weight to all other reversals.

The three topologies presented in Figure 18 differ from each other in either the clustering of nodes or the branch lengths. An important point to be noted is that the weighted distance model results in the clustering \( \text{AC}|\text{BD} \) while the uniform weight model (GRIMM) clusters the nodes as \( \text{AB}|\text{CD} \). This differ-
Figure 18: The topologies produced with the distance matrices from the different distance algorithms as input. The leaves are linear unsigned permutations on 7 regions – $A = (1, 4)(3, 7, 6)$, $B = (1, 3, 7, 5, 2, 6, 4)$, $C = (3, 4, 6)$ and $D = (1, 7, 6, 4, 2, 3, 5)$.

ence is interesting since both the methods have the same set of inversions but different weights assigned to the generators.

Our second example deals with circular rather than linear genomes. This example was intended as a test for the distances determined using rewriting systems. To construct the rewriting system, we used the same set of generators and relations as those in the circular inversion model presented in [35]. The generating set consists of the inversions of adjacent regions $(i, i + 1)$ for $1 \leq i < n$ and the inversion $(1, n)$ that swaps the positions $n$ and 1. Following the notation in [35], we denote these generators by $s_i$. The generating set is thus $\{s_i \mid i = 1, 2, \ldots n\}$ and the relations are:

\[
\begin{align*}
    s_i^2 &= e \\
    s_is_j &= s_js_i & \text{for each } i = 1, \ldots n, \\
    s_is_{i+1}s_i &= s_{i+1}s_is_{i+1} & \text{if } i - j \mod n \neq 1, \\
    s_is_{i+1}s_i &= s_{i+1}s_is_{i+1} & \text{for each } i = 1, \ldots n - 1, \text{and}
\end{align*}
\]
\[ s_n = s_{n-1}s_{n-2} \cdots s_2s_1s_2 \cdots s_{n-2}s_{n-1}. \]

All the generators are assigned unit weight as in the circular inversion model of [35]. We use this presentation as an input to KBMAG. The confluent rewriting system in this case has 6622 rules. Once again, we generated four random permutations in the group and found the distances using KBMAG, GRIMM and the method presented in [35] which we refer to as EGTF. This latter method has been implemented by the authors in the package BioGAP [34] for the software GAP [87]. Both EGTF and the rewriting system method have been set up to take into account the rotational and reflection symmetry of a circular genome. EGTF and our method also have the same generating set and the same weights.

Thus, if the rewriting method is set up correctly, the distance matrices derived from these two methods would be the same, which is exactly what happened. The resulting phylogenies produced using RPhylip are presented in figure 19.
Figure 19: The topologies produced for circular genomes on 8 regions with the distance matrices from the different distance algorithms. The four permutations are $A = (2, 3)(6, 8)$, $B = (1, 7, 6, 8)(4, 5)$, $C = (1, 5, 6, 4, 3, 8, 7)$ and $D = (1, 2, 4)(5, 6, 7)$. 
5.8 Discussion and Future Work

Researchers have recognised the need for methods to determine weighted distances in the field of genome rearrangement right from the start. Beginning with the pioneering work of [83, 86], a number of approaches have been tried. While they differ in the techniques employed, a common feature of the previous studies is that the proposed algorithms are tied to a particular model of rearrangement. The novelty of our work is that the framework presented here can be adapted to a wide variety of models. In addition, to the best of our knowledge, this work presents the first use of the theory of rewriting systems to a problem in comparative phylogenetics.

The current approach however has some limitations, which present opportunities for interesting research. The method presented in this thesis can only be used with invertible rearrangement operators. The use of other algebraic structures such as a semigroup might allow this restriction to be removed allowing more rearrangement models to be used.

Another important limitation is that the method works by distorting an existing path (in terms of the operators in the model) between two genomes into an optimal path. However, for some models, finding a path between two arbitrary genomes may be non trivial.

Even in the case where such a path is known, for instance in the inversion model, the other deficiency at the moment is the lack of a software implementation. We have used KBMAG to derive the confluent rewriting systems. However, the use of KBMAG with finite groups is not recommended by the authors as it is optimised for infinite groups. It is not surprising therefore that for larger values of \( n \), KBMAG cannot return a confluent rewriting system even though we know that a confluent system exists. The size of a confluent rewriting system increases very quickly with \( n \) (see Table 4). Thus an efficient
A software implementation of Knuth-Bendix optimised for finite groups and in particular for models arising from biology would be very useful.

<table>
<thead>
<tr>
<th>n</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>204</td>
</tr>
<tr>
<td>6</td>
<td>1049</td>
</tr>
<tr>
<td>7</td>
<td>6220</td>
</tr>
</tbody>
</table>

Table 4: The size of a confluent rewriting system grows very quickly with $n$. The rewriting system is for a weighted lexicographic order with weight of $t_{i,j}$ is $j - i$.

The application of rewriting systems to a new problem also gives rise to new mathematical questions. For instance, it would be interesting to investigate the effect of the weighting function used on the size and efficiency of the rewriting system.
LOOKING BACK AND LOOKING AHEAD

‘Do you know anything about m-dimensional topography?’
‘Um. No.’
‘Then I shouldn’t aspire to hold any opinions, if I was you,’ said Albert.

Mort, Terry Pratchett

Mathematical models appear to be in an asymptotic relation with reality – forever trying to be closer to the “truth” but incapable by definition of fully realising it. It is frequently said that “all models are bad, some are useful”. It is our hope that the models presented in our thesis belong to the category of useful models. The contributions of this work can be summed up along three lines:

• a clear and concise presentation of the various permutation models used in the field;

• a rigorous algebraic treatment of the Double Cut and Join operator; and

• a flexible framework for determining weighted rearrangement distances.

Like any scientific enterprise, our thesis also suggests a number of directions for future work, some of which we have already mentioned. Employing
an algebraic approach offers a fresh look at the familiar problem of determining genome rearrangement distance and new ways of approaching the related problems. For instance, an interesting problem raised in [21] is to find the distance between sorting sequences in terms of a set of operations that transform one sorting sequence into another. Modeling a sorting scenario as a sequence of genomic permutations as we have done in our work, one could identify a sorting scenario with the set of group elements that generate it. The problem of determining a distance between two scenarios is then translated into a problem of finding a distance between two subsets of the group.

With respect to the problem of determining weighted rearrangement distances, our work presents the first proof of concepts for using the very well developed theory of rewriting systems. There is a considerable potential to extend this work through an efficient software implementation as well as investigating the mathematical questions arising out of the use of rewriting systems in phylogenetics.

A key advantage of using algebraic models is that it becomes easier to incorporate biological constraints, for example by using a more restrictive algebraic structure. As briefly mentioned in Chapter 5, it would be worthwhile to investigate the use of semigroups rather than group based rewriting systems to allow the use of non-invertible rearrangement operators.

The merit of a scientific project is often judged by its applicability to “real world” problems. But there is another important criterion, one that scientists themselves employ – “was this problem interesting?” In his book The Log from the Sea of Cortez, John Steinbeck writes about his expedition to the Gulf of California to collect marine specimens: “The real picture of how it had been there and how we had been there was in our minds, bright with sun and wet with sea water and blue or burned, and the whole crusted over with exploring thought. Here
was no service to science, no naming of unknown animals, but rather – we simply liked it. We liked it very much."

Since he says it so well, I find it prudent to borrow his words and say of my 4 year journey, “We liked it very much.”


[85] David Sankoff, G Leduc, and D Rand. DERANGE – minimum weight
generation of oriented permutation by block inversions and block move-
ments. *Macintosh Application, Centre of Recherches Mathematiques, Universi-

[86] David Sankoff, Guillame Leduc, Natalie Antoine, Bruno Paquin,
B Franz Lang, and Robert Cedergren. Gene order comparisons for phy-
logenetic inference: evolution of the mitochondrial genome. *Proceedings


[88] Cathal Seoighe, Nancy Federspiel, Ted Jones, Nancy Hansen, Vesna
Bivolarovic, Ray Surzycki, Raquel Tamse, Caridad Komp, Lucas Huizar,
Ronald W Davis, et al. Prevalence of small inversions in yeast gene
order evolution. *Proceedings of the National Academy of Sciences*, 97(26):

[89] Mingfu Shao, Yu Lin, and Bernard ME Moret. An exact algorithm to
compute the double-cut-and-join distance for genomes with duplicate

[90] Adam C Siepel. An algorithm to enumerate all sorting reversals. In
Proceedings of the Sixth Annual International Conference on Computational
Biology, 2002.

[91] Andrew Solomon, Paul Sutcliffe, and Raymond Lister. Sorting circular
permutations by reversal. In *Workshop on Algorithms and Data Structures*,


