FACULTÉ DES SCIENCES
Département d'Informatique
Département de Mathématique

UNIVERSITÉ LIBRE DE BRUXELLES,
UNIVERSITÉ D'EUROPE

ULB

Combinatorial aspects of genome rearrangements and haplotype networks

Anthony Labarre

SCIENTIA VINCERE TENEBRAS
UNIVERSITAS BRUXELLENSIS

Dissertation présentée en vue de l'obtention
du grade de **Docteur en Sciences**

Année académique 2007-2008

PÔLE UNIVERSITAIRE
EUROPÉEN
DE BRUXELLES
WALLONIE

ACADÉMIE UNIVERSITAIRE
WALLONIE-BRUXELLES

This Ph. D. dissertation was written under the supervision of Prof. Dr. Jean-Paul Doignon (Université libre de Bruxelles, Belgium) and defended on September 12, 2008. The Jury consisted of:

- Prof. Dr. Martine Labbé (Université libre de Bruxelles, Belgium, Dean of the Faculty of Sciences, President of the Jury);
- Prof. Dr. Jean Cardinal (Université libre de Bruxelles, Belgium, Département d'Informatique, Secretary of the Jury);
- Prof. Dr. Jean-Paul Doignon (Université libre de Bruxelles, Belgium, Département de Mathématique);
- Dr. Samuel Fiorini (Université libre de Bruxelles, Belgium, Département de Mathématique);
- Dr. Patrick Mardulyn (Université libre de Bruxelles, Belgium, Laboratoire d'Éco-éthologie évolutive);
- Prof. Dr. Marie-France Sagot (Université Claude Bernard, Lyon I, France);
- Prof. Dr. Alain Guénoche (Institut de Mathématiques de Luminy, Marseille, France).

Acknowledgements

First and foremost I would like to thank my advisor Jean-Paul Doignon, without whom the present text would not lie before your eyes. Jean-Paul introduced me to the world of research and taught me a lot, from the day we started working on genome rearrangement problems, which were the subject of my Master's Thesis and which he also supervised. I would like to thank him for being a wonderful advisor and a wonderful professor (I could not possibly overemphasise his ability to explain even the most complex concepts in such a simple and interesting way), and for believing in me enough to supervise me as a Ph. D. student.

Research is fun, especially when you reach results interesting enough to be published, but it also has its moments of loneliness, doubt and pain; I would like to thank Lin, my friends and my family for supporting me, helping me and bearing with me during those hard times, and reminding me that there is more to life than work. Colleagues from both Departments (Mathematics and Computer Science) have also been most helpful, both on a personal and on a professional level. I would like to take this opportunity to thank in particular Jean Cardinal, who has always been able to spare a few minutes for discussing problems and whose "algorithmic lunches", as they are called, have been a nice and refreshing opportunity for informal discussions, presentation of results and exchange of ideas between enthusiastic people on various topics.

I would especially like to thank Éric Tannier and Marie-France Sagot for inviting me to a one-week open problem session in Lyon in April 2006, as well as my other collaborators: Guillaume Fertin, Irena Rusu, and Stéphane Vialette, for believing in and eventually collaborating with Éric and I on a survey which would later become a book. I must credit Éric for that idea.

I would also like to thank Jean-Stéphane Varré for inviting me to Lille in March 2008 and giving me the opportunity to present some of my work to the SEQUOIA bioinformatics team at INRIA.

I am also grateful to the Fonds National de la Recherche Scientifique (F.N.R.S.) for supporting me with a F.R.I.A. (Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture) grant during those four years.

Finally, I would like to thank *you* for the time you will spend on my dissertation, and wish you a pleasant read.

Introduction

We study two problems motivated by computational biology: genome rearrangements, which under some assumptions can be recast as the problem of sorting a permutation (therefore viewed as a linear ordering) using as few allowed moves as possible, and the construction of haplotype networks, which generalise haplotype trees in that they allow multiple paths between species. Our main contributions are:

- new upper bounds and formulae for computing the exact transposition distance of many permutations (a problem of unknown computational complexity); those results do not rely on the classical cycle graph (a variant of the breakpoint graph restricted to unsigned permutations) as most previous results do, but rather on the classical disjoint cycle decomposition of permutations;

- formulae for enumerating permutations whose cycle graph contains a given number of cycles, and for enumerating permutations that belong to a given conjugacy class (not in the classical sense, but rather in terms of the cycle graph); we give simpler formulae for a few particular cases, including 2-permutations and 3-permutations, and show how our results can be used to completely characterise the distribution of the block-interchange distance;

- a new framework for computing and finding lower bounds on all edit distances between permutations; in particular, this framework can be applied to genome rearrangement distances, as we show by using it to recover previous results in a simple way;

- a new lower bound on the prefix transposition distance, which, as we show both theoretically and experimentally, is a substantial improvement over previous results;

- a new lower bound of $\left\lfloor \frac{3n+1}{4} \right\rfloor$ on the prefix transposition diameter, thereby improving on the previously known lower bound of $2n/3$;

- a new model for computing a haplotype network from maximum parsimonious trees, which consists in finding a minimum common supergraph of a given set of partially labelled trees;

- two exact algorithms for computing the minimum common supergraph of two partially labelled graphs: one that runs in exponential time in the general case, and another that runs in polynomial time provided that at least one graph belongs to a particular class.

We will give an overview of the dissertation at the end of the first chapter (more specifically in Section 1.3), which is dedicated to the biological concepts and motivations behind the problems we tackle, and some perspectives for future research after Chapter 6. This thesis is partly based on the following publications:

1. *A New Tight Upper Bound on the Transposition Distance*, 2005 [51]
2. *New Bounds and Tractable Instances for the Transposition Distance*, 2006 [52]
3. *On Hultman Numbers*, 2007 [29], a joint work with my advisor Jean-Paul Doignon
4. *Edit Distances and Factorisations of Even Permutations*, 2008 [53]

During the course of my Ph. D., I also had the privilege to work with Guillaume Fertin, Irena Rusu, Éric Tannier and Stéphane Vialette on a book that surveys the combinatorial and algorithmic aspects of genome rearrangements. This book is quite naturally entitled “Combinatorics of genome rearrangements”, and will be published in 2009 by The MIT Press [36].

Contents

Chapter 1

Biological motivations

The present thesis is concerned with the study of two problems motivated by biology, from a mathematical and computer scientific point of view. The first problem (detailed in Section 1.1.3) is the study of "genome rearrangements", and the second problem (detailed in Section 1.2.2) is the study of "haplotype networks". Both topics belong to the broader category of comparative genomics, which studies relationships between different genomes or species. In this introductory chapter, we give basic biological definitions and present an informal overview of the concepts and problems that will be tackled in this thesis.

1.1 Comparing genomes

Before speaking about the problems we are interested in, we must describe and define a few biological concepts. We refer the reader to Alberts et al. [4] and Ridley [66] for a more detailed and technical exposition of these notions.

1.1.1 A minimalist introduction to molecular evolution

Chromosomes are the support for the genetic information contained in every cell of all living organisms (e.g. bacteria, plants, animals), and the *genome* of an organism is the set of all its chromosomes. Chromosomes are made of *DNA* (which stands for **d**eoxyribo**n**ucleic **a**cid), a double-stranded molecule in which each *strand* is a long *sequence* of *nucleotides*. Nucleotides can be of four types; for our purpose, it will be enough to consider them as letters from the alphabet $\{A, C, G, T\}$. Both DNA strands are said to be *complementary*, because an A at a given position on one strand is always paired with a T at the same position on the other strand, and a C at a given position on one strand is always coupled with a G at the same position on the other strand. Complementarity implies that the sequence formed by the nucleotides on one strand uniquely determines the sequence on the other strand, and it is therefore sufficient to consider only one of those two sequences. Figure 1.1 summarises all these concepts.

A *segment* of DNA is a portion of that molecule made of consecutive nucleotides. A *gene* is a segment of DNA that contains the information needed to construct the other molecules in the cell. DNA is able to *replicate* itself with some inaccuracy: one

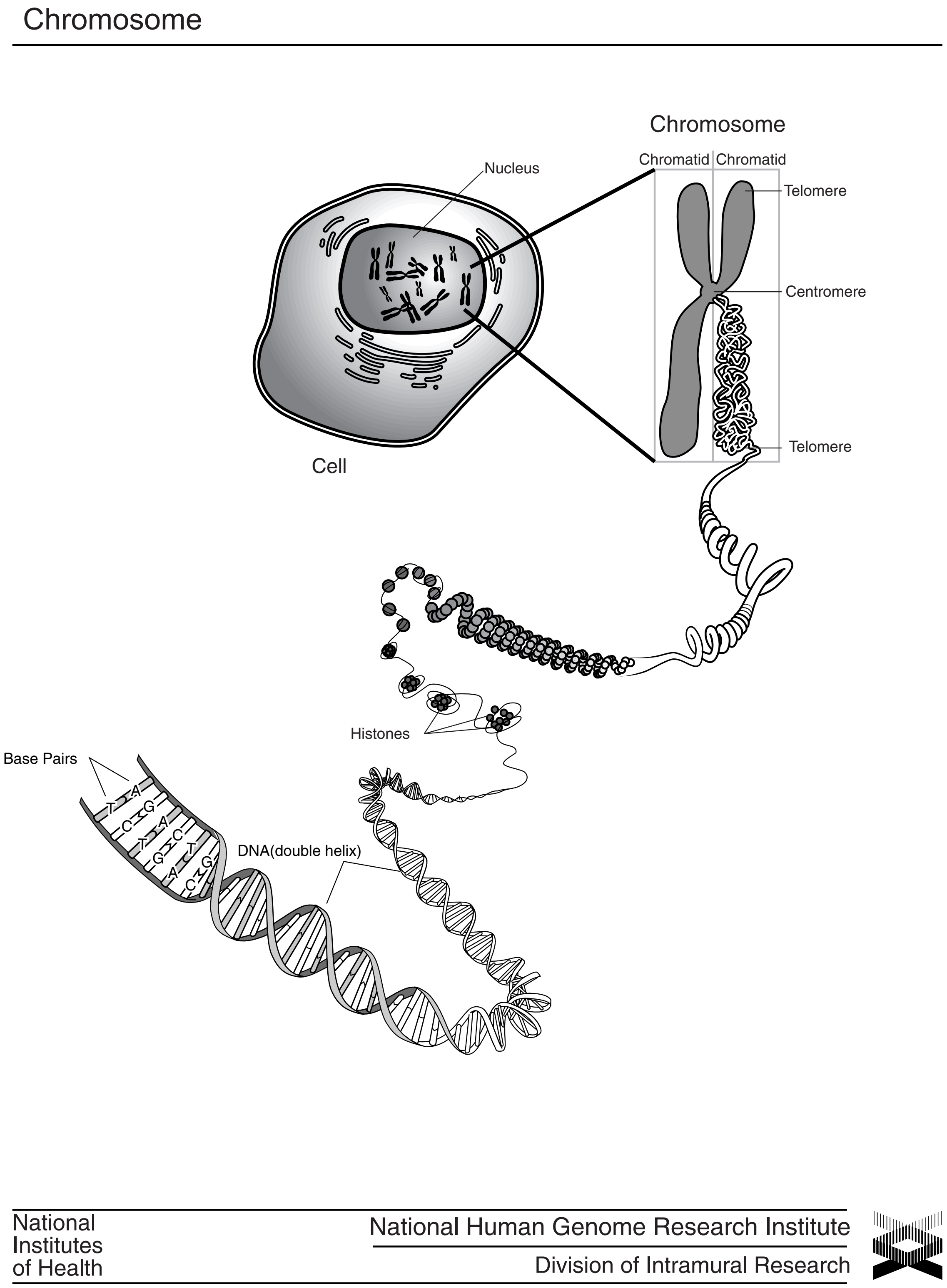


Figure 1.1: A chromosome, and a fragment of a DNA molecule (source: National Human Genome Research Institute [58]).

genome is used to produce another almost identical genome. This inaccuracy is the principle of *molecular evolution*. A DNA molecule may evolve by *mutations* at the nucleotide level or at the gene level, leading to at least two main ways of comparing genomes, which we discuss in more detail in the next two sections.

1.1.2 Evolution at the nucleotide level: sequence alignment

Mutations may occur at the nucleotide level, and are in this case called *point mutations*. These consist in the *insertion* of a new nucleotide, the *deletion* of a nucleotide, and the *substitution* of a nucleotide – that is, the replacement of a nucleotide with a new one. Detecting and explaining these changes and the evolution of a set of sequences by point mutations is the goal of *sequence alignment*.

In the pairwise version of the problem, we are given two sequences of nucleotides, not necessarily of the same length, and we want to *align* them, i.e. to match nucleotides in both sequences so as to minimise an objective function that measures the divergence between both species based on the number of insertions, deletions and substitutions of nucleotides that are needed to transform one sequence into the other. Figure 1.2 shows an example of an alignment of two sequences.

$$\begin{array}{lcccccccccccc} S_1: & \mathbf{T} & \mathbf{C} & \underline{C} & \mathbf{G} & \underline{C} & \mathbf{C} & \underline{A} & - & - & \mathbf{C} & T & \mathbf{A} \\ & | & | & & | & & | & & & & | & & | \\ S_2: & \mathbf{T} & \mathbf{C} & \underline{G} & \mathbf{G} & \underline{A} & \mathbf{C} & \underline{T} & G & G & \mathbf{C} & - & \mathbf{A} \end{array}$$

Figure 1.2: An alignment of two sequences S_1 of length 10 and S_2 of length 11, with six **matches**, three substitutions, two insertions and one deletion.

Multiple sequence alignment, which deals with more than two genomes at the same time, is based on the same principle and aims at aligning all sequences, rather than just two of them, and minimising an objective function that takes all input genomes into account. For more information on (multiple) sequence alignment, see for instance the surveys by Notredame [59, 60].

1.1.3 Evolution at the gene level: genome rearrangements

Mutations occur at the nucleotide level, as seen in the previous section, but can also affect whole segments of DNA which may correspond to genes. Insertions and deletions of genes rather than nucleotides are just an example of the many rearrangement operations that can occur: genes can also be reversed, moved to another segment of the same chromosome, or even exchanged with another gene on a different chromosome (see Figure 1.3 for a few examples). In the presence of such events, sequence alignment is no longer the tool of choice, since a comparison at the nucleotide level may lead to think that the genomes under comparison are very different, whereas they might differ by a very small number of larger scale mutations.

The *genome rearrangement* problem (see for instance Pevzner [64] or Meidanis and Setubal [57] for an introduction, or Li, Wang, and Zhang [55] for a recent survey) can be formulated as that of finding a sequence of evolutionary events that

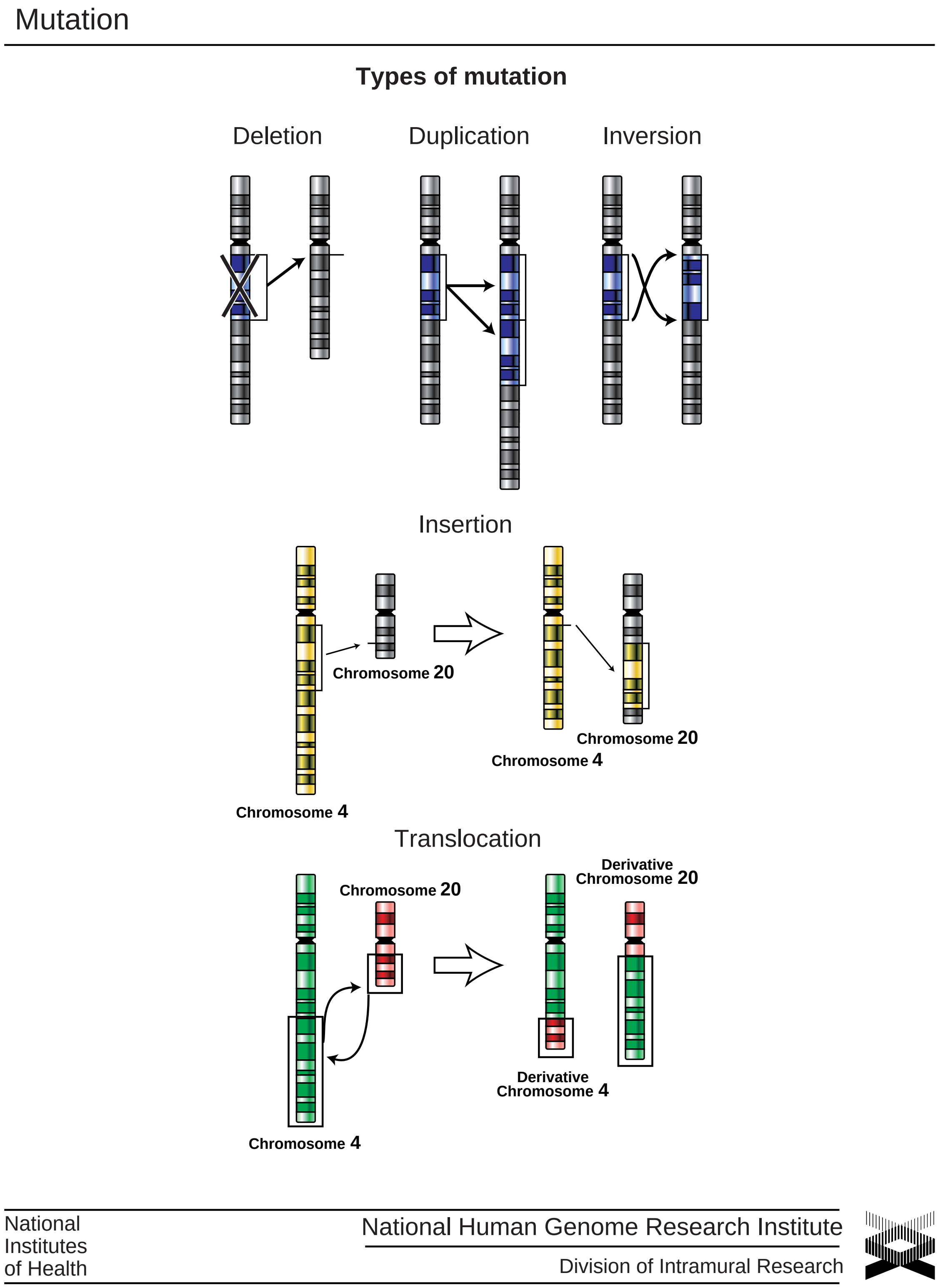


Figure 1.3: Some mutations to which genomes are subject (source: National Human Genome Research Institute [58]).

transforms a given genome into another given one and is of the shortest possible length. The *(evolutionary) distance* between the two genomes is the length of such a sequence. The biological reason for favouring sequences of the shortest possible length is the *parsimony hypothesis*, which states that the most plausible scenario of evolution is the one involving the fewest number of rearrangements, based on the observation that mutations are relatively rare. As shown in Figure 1.3, a lot of mutations can be taken into account, among which the most popular one – and the first one to have been studied from a computational point of view – is certainly the *reversal* (also referred to as *inversion* by biologists, a term that we will refrain from using as it might be confusing in a mathematical context), which reverses a segment of the genome; other operations include *translocations*, which exchange segments that belong to different chromosomes, and *transpositions*, which move genes around in the genome and on which we will focus (more details will be given in the next section).

Although very simply described in its general form, the computational study of genome rearrangements is a very rich field, whose diversity comes from the kind of mutations that are taken into account and how their relative importance is expressed (e.g. one could consider only reversals, or both reversals and transpositions, possibly using different weights according to the probability of occurrence of each kind of mutation in the species under consideration), but also from the models used to represent genomes. Many models can indeed be used, depending on the information that is available about the genomes and what exactly we want to take into account (for example *gene orientation*, i.e. the DNA strand on which a gene is located, duplicated genes, the order of genes along a given chromosome, whether chromosomes are linear or circular, ...). We will not review all these models here, and refer the reader to the aforementioned references [64, 57, 55] for more information.

In this thesis, we will focus on a very simple model, used in the case where the order of genes is known and where all genomes share the same set and number of genes (without duplications); under these assumptions, genomes, as well as mutations, can be represented using *permutations*. This model is certainly not realistic in general, since duplications are known to play a major role in evolution (see e.g. Ohno [61]), but it may be used as a starting point for more complex models; besides, real-world biological examples are known for which permutations are an adequate model (see Kececioglu and Sankoff [49] for a few examples). A very nice property of most rearrangement distances (see Definition 2.24 page 15) will allow us to reduce the genome rearrangement problem to that of *sorting permutations* (by reversals, transpositions, and so on) using as few moves as possible.

1.1.4 Transpositions

During the 1940s and 1950s, McClintock [56] discovered that some sequences of DNA moved around to different positions within the genome of a single cell. Those sequences were called *transposons* (or *transposable elements*, or “jumping genes”) and proved to be responsible for turning physical characteristics on and off. This discovery eventually awarded McClintock the Nobel Prize in Physiology or Medicine in 1983.

Two main reasons motivate the study of transpositions instead of other mutations. From a biological point of view, transpositions are interesting because while they may not occur as frequently as other mutations (like, say, reversals), they are known to occur in *all* species. Moreover, about forty-five percent of the human genome is made of transposons, and while these transposable elements are known to be responsible for many diseases, they also seem to be a promising tool for gene therapies (see Anxolabéhère, Nouaud, Quesneville, and Ronserray [5]).

From a computer-theoretical point of view, not much is known about rearrangement by transpositions, as opposed to the study of other rearrangement operations. For instance, it is known that the problem of sorting by reversals is NP-hard (see Caprara [18]), and that it is NP-hard to approximate it within a factor of 1.0008 (see Berman and Karpinski [12]; however, on the "good news" side, Berman, Hannenhalli, and Karpinski [13] designed an 11/8-approximation, and Caprara, Lancia, and Ng [19] proposed a linear programming formulation that can solve the problem on permutations of up to 200 elements in a few minutes). A variant of sorting by reversals has also been investigated, in which every element has a "+" or a "−" sign and a reversal not only reverses the order of elements, but also flips their signs. This biologically more relevant problem – signs represent the DNA strand on which genes are located – seems more difficult at first glance, but surprisingly turns out to be computationally much easier to solve: indeed, the sorting problem can be solved in subquadratic time (see Tannier, Bergeron, and Sagot [72]), and the corresponding distance can be computed in linear time (see Bader, Moret, and Yan [6]).

However, as far as transpositions are concerned, no exact polynomial-time algorithm for the sorting problem is yet known (the best approximation algorithm to date, by Elias and Hartman [30], has a ratio of 11/8), and its computational complexity remains open. The same is true for the problem of merely computing the associated transposition distance, and even the maximal value that the transposition distance can reach is unknown. Given the lack of progress on the problem of sorting by transpositions, it is certainly justified and interesting to try and contribute to the current state of knowledge about it.

1.2 From genome comparisons to phylogenies

1.2.1 Phylogenetic trees

Biologists are usually interested in comparing more than two genomes. Not only do they want to measure divergence between species, they also want to reconstruct the ancestral genomes that led, after repeated speciations, to the genomes that are available today. Ultimately, the goal is to reconstruct scenarios of evolution between species that are as close as possible to their true evolutionary history.

Phylogenetic trees, whose origins can be traced back to the works of Darwin [25] and which have been in use since then, are the traditional tool for representing evolution. The leaves of those trees are the input genomes, and the internal nodes represent unknown common ancestors of the species at the leaves, reconstructed by the algorithm that has been chosen; branches of various lengths, measuring the divergence between two species, connect these nodes. We note that, although this is not mandatory, most phylogenetic studies focus on trees in which known species are restricted to leaves: these trees model the cases where the species under comparison are all “contemporary”, and the previous level in the tree (i.e. the level of their ancestors) contains species from an earlier period in time. Many methods and algorithms are available for accurately constructing such trees and assessing their quality (see e.g. Felsenstein [34] for an extensive survey). As an example that illustrates the use of genome comparisons that we discussed in the previous section, we mention *distance-based methods*: these methods take as input a distance matrix, which contains distances between all pairs of input genomes, then use an algorithm to reconstruct a tree based on that distance matrix. The process we describe is very general, and encompasses many variants whose diversity stems from how distances are computed (at the nucleotide or gene level, which kinds of mutations are taken into account, how they are weighted, and so on) and how genomes are combined into a tree by the chosen algorithm, based on those distances. Figure 1.4 shows an example of a phylogenetic tree, which represents the relations between the simian and human immunodeficiency viruses.

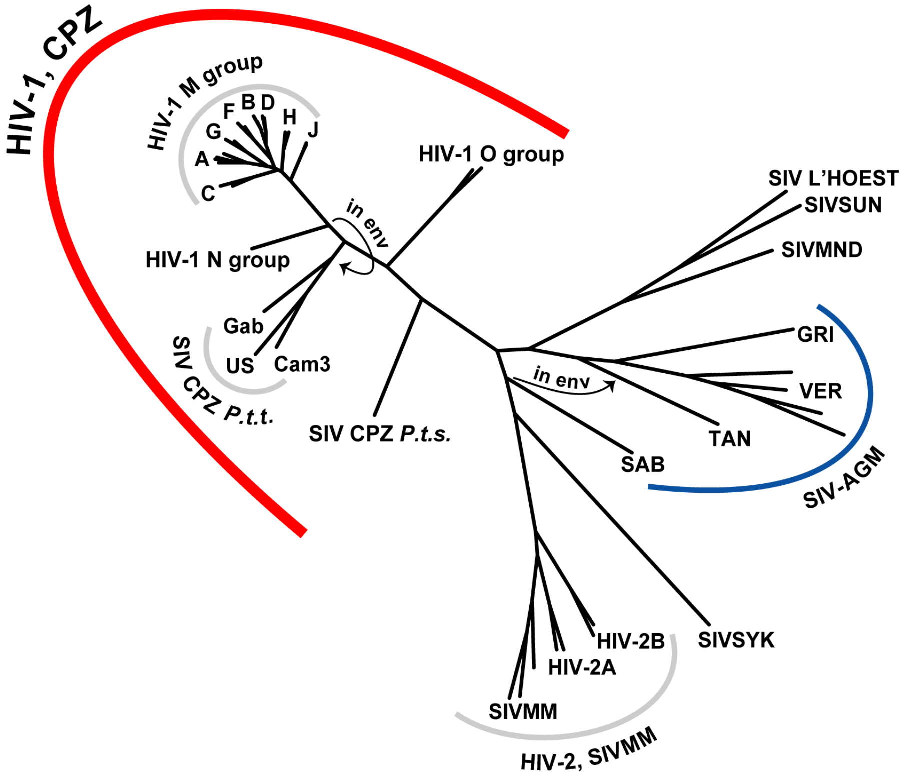


Figure 1.4: Phylogenetic tree of the **s**imian and **h**uman **i**mmunodeficiency **v**iruses (SIV and HIV, respectively), from Kuiken et al. [50].

1.2.2 Phylogenetic networks

The last two decades have witnessed the emergence of a new tool for reconstructing and representing evolution, which has become widespread in phylogenetic studies. This tool is known as a *phylogenetic network*, and generalises phylogenetic trees by allowing multiple paths between species. Figure 1.5 shows an example of a phylogenetic network, depicting the relationships between various *Salmonella* isolates.

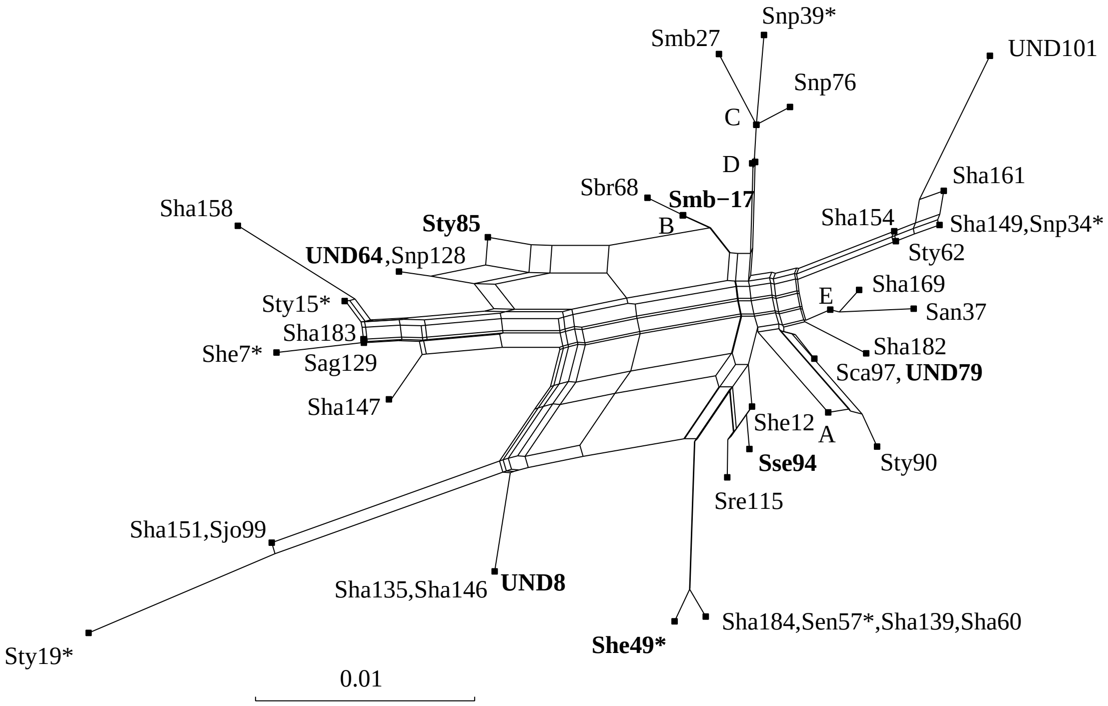


Figure 1.5: Phylogenetic network of *Salmonella* isolates, from Bryant, Moulton, and Spillner [16].

The main reason for using networks rather than trees is that evolution sometimes happens not to be tree-like: genes may be duplicated, transferred or lost, and recombination events (i.e. the breaking of a DNA strand followed by its reinsertion into a different DNA molecule) as well as hybridisation events (i.e. the combination of genetic material from several species) are known to occur. Moreover, even when evolution is indeed tree-like, there exist situations in which a relatively large number of tree topologies might be "equally good", and there is not enough information available to discriminate between those trees. One proposed solution to the latter issue is the use of *consensus trees*, where the idea is to find a tree that represents a compromise between the given topologies; another solution is to build a network that is compatible with all topologies of interest.

Although phylogenetic networks have not – yet – been studied as extensively as trees, a lot of methods and results are already available, and are surveyed for instance by Huber and Moulton [46], Posada and Crandall [65] or Gascuel and Steel [39]. A special category of networks has been used in the context of *intraspecific* studies, and more specifically, in the case where one wants to study relations between genes instead of species. Networks used in that context are known as *haplotype networks* (see e.g. Cassens, Mardulyn, and Milinkovitch [20] for more technical biological definitions). Cassens et al. [20] proposed a new method for reconstructing such networks, based on a set of given trees rather than on the input sequences, contrary to previous methods (see for instance Bandelt, Forster, and Rohl [10], Excoffier and Smouse [33], or Templeton, Crandall, and Sing [73]). The results they obtained on simulated data seem promising, when compared to a few traditional algorithms. However, their algorithm makes a number of arbitrary choices, produces solutions whose quality depends on the order in which the merging process is performed, and is a heuristic with an implicit objective function. We contribute to strengthening their method in a way we will detail in the next section.

1.3 Organisation of the thesis

Basic notation, concepts and terminology concerning graph theory, permutation groups and distances are introduced in Chapter 2; more specific definitions and concepts will be introduced progressively through the rest of the thesis.

In Chapter 3, we consider the problem of *sorting by transpositions* and present nontrivial connections between the *graph of a permutation* (see Definition 2.14 page 13) and a central tool in genome rearrangements, known as the *cycle graph* (see Definition 3.3 page 20). We then use those connections to prove a new tight upper bound on the transposition distance, as well as various formulae and heuristic improvements that allow us, in many cases, to either compute the exact transposition distance or at least to tighten our new bound.

Motivated by an equivalence relation on permutations which preserves the transposition distance as well as the structure of the cycle graph (see Section 3.1.4 page 23), Hultman [47] asked the question of determining the number of permutations with a given number of cycles in their cycle graph. In Chapter 4, we solve the more general problem of determining the number of permutations of a given cycle type (in terms of the cycle graph), and obtain a formula for computing what we call the “Hultman numbers”, thereby answering Hultman’s question. Our proof is based on a bijection between permutations of n elements and related even permutations of $n+1$ elements that can be expressed as the product of two $(n+1)$-cycles. We also obtain simpler formulae in interesting restricted cases.

Since many genome rearrangement distances are based on the cycle graph or on a similar structure, our counting result can be used to estimate the distribution of those distances (this is in particular true for the *block-interchange distance*, as will be explained in Section 4.4.2 page 59). On the other hand, perhaps surprisingly, the bijection we construct in Chapter 4 can also be used to obtain bounds on *any edit distance* between permutations, i.e. distances based on a minimum number of allowed operations required to transform a permutation into another, of which genome rearrangement distances are a particular case. In Chapter 5, we develop a framework based on that bijection which allows us, on the one hand, to encode the cycle graph of a permutation using an even permutation and, on the other hand, to reformulate *any* edit distance problem on permutations in terms of particular factorisations of the latter permutation. We show the power of this approach by providing simple and unified proofs of known results on transpositions and block-interchanges, and then turn to the study of a restricted version of sorting by transpositions, known as sorting by *prefix* transpositions. We use our framework to obtain a new lower bound on the prefix transposition distance (which always outperforms previous results, as we show both theoretically and experimentally), which as a consequence allows us to prove a much better lower bound on the maximal value that distance can reach.

Finally, in Chapter 6, we study the problem of constructing haplotype networks. Most available methods for achieving this goal usually construct a network directly from the sequences, then apply heuristics in order to reduce the size of the network that has been obtained in that way. Another approach, inspired by phylogenetics, is to first build the set of all most parsimonious trees on the data, then to combine them into a single graph that contains all those trees and is “as compact as possible”. Such an approach was proposed by Cassens et al. [20], who obtained competitive results on simulated data with respect to a few traditional algorithms. However, their algorithm makes a number of arbitrary choices, produces solutions whose quality depends on the order in which the merging process is performed, and is a heuristic with an implicit objective function. We propose a formal framework for their method which consists in finding a minimum common supergraph of a set of partially labelled trees, and propose algorithms for solving the problem to optimality on two graphs.

Chapter 2

Reminders

This chapter recalls some general and basic definitions on graph theory (see Berge [11] or Diestel [28] for more information) and permutations (see for instance Wielandt [75] for more information) that will be used throughout the thesis, and with which most readers should be familiar. More specific definitions will be introduced whenever needed.

2.1 Graph theory

Definition 2.1. A *graph* $G = (V, E)$ is an ordered pair that consists of

1. a set $V = \{v_1, v_2, \ldots, v_n\}$,
2. a family $E = \{e_1, e_2, \ldots, e_m\}$ of elements taken in

$$\begin{array}{ll} \{(u,v) \mid u, v \in V\} & \text{if } G \text{ is } \textit{directed}, \\ \{\{u,v\} \mid u, v \in V\} & \text{if } G \text{ is } \textit{undirected}. \end{array}$$

Graphs are graphically represented by dots usually joined by arrows (in the directed case) or by (not necessarily straight) lines (in the undirected case); dots represent the elements of V, which are called *vertices*, and arrows (or line segments) represent the elements of E, which are called *edges* in the undirected case and *arcs* or *directed edges* in the directed case (see Figure 2.1 page 13 for a first example of a directed graph, and Figure 2.2 page 17 for a first example of an undirected graph). A (possibly directed) edge connecting vertices u and v is said to have u and v as *endpoints*; in this case, u and v are said to be *adjacent*, and they are both *incident* to the edge that connects them. We will sometimes make use of the notations $V(G)$ and $E(G)$ to refer to the vertex set and the edge set of a graph G, respectively.

Definition 2.2. The *order* of a graph is the number of vertices it contains.

Definition 2.3. The *size* of a graph is the number of edges it contains.

Definition 2.4. A graph is *simple* if:

1. it has no loop, i.e. an edge whose endpoints coincide, and

2. there is no more than one edge connecting any two vertices.

Definition 2.5. A *path* in a graph G is a sequence of distinct vertices $v_0, v_1, \ldots, v_{k-1}$ such that $\{v_i, v_{i+1}\}$ (or (v_i, v_{i+1}) if G is a directed graph) is in $E(G)$, for $0 \leq i \leq k-2$.

A path in a graph is often denoted by listing its vertices, i.e. $P = (v_0, v_1, \ldots, v_{k-1})$.

Definition 2.6. A *cycle* in a graph G is the union of a path $P = (v_0, v_1, \ldots, v_{k-1})$ and of the edge $\{v_{k-1}, v_0\}$ (or (v_{k-1}, v_0) if G is a directed graph).

The *length* of a path (resp. of a cycle) is the number of vertices it contains; if it has length k, we call it a *k-path* (resp. a *k-cycle*).

Definition 2.7. A *bicoloured* graph $G = (V, E)$ is a graph whose edge set is partitioned into two classes $E_1 \cup E_2$, to each of which a distinct colour is associated.

Definition 2.8. The *neighbourhood* of a vertex v in a graph G, denoted by $N^G(v)$, is the set of all vertices that are adjacent to v in G.

Definition 2.9. A graph G is *bipartite* if its vertex set can be partitioned into two classes V_1 and V_2 in such a way that no two vertices of V_1 (resp. V_2) are adjacent.

Definition 2.10. A *matching* $\mathcal{M}$ in a graph G is a set of pairwise disjoint edges, i.e. no two edges of $\mathcal{M}$ share a vertex. If every vertex of G is incident to an edge in $\mathcal{M}$, then $\mathcal{M}$ is called a *perfect matching*.

2.2 Permutations

2.2.1 The basics

The word "permutation" is used in a variety of close meanings. We adopt the same viewpoint as Bóna [15], by considering them as linear orderings of the elements of a finite set $E = \{e_1, e_2, \ldots, e_n\}$, i.e. a way to list the elements of E in a certain order. Any set can be used, as long as a reference total order on E is specified, but we will mostly work with $E = \{1, 2, \ldots, n\}$ (sometimes also denoted by $[n]$).

Definition 2.11. A *permutation* of $\{1, 2, \ldots, n\}$ is a bijective application of $\{1, 2, \ldots, n\}$ onto itself.

We will denote permutations using lower case Greek letters. A common way of writing permutations is the well-known *two row notation*, i.e.

$$\pi = \left(\begin{array}{cccc} 1 & 2 & \cdots & n \\ \pi_1 & \pi_2 & \cdots & \pi_n \end{array} \right),$$

where $\pi_i = \pi(i)$, for $1 \leq i \leq n$; in this notation, the first line stands for *positions* in the permutation, and the second line stands for its *elements*. Most of the time, we will shorten this notation by keeping only the second row of the above notation enclosed in angular brackets, i.e. $\pi = \langle \pi_1 \ \pi_2 \ \cdots \ \pi_n \rangle$.

As stated in Definition 2.11, permutations are functions, and can therefore be *composed* or *multiplied* as such: the product of two permutations π and σ is denoted by $\pi \circ \sigma$, and is obtained by first applying σ, then π. The *identity permutation* $\langle 1 \ 2 \ \cdots \ n \rangle$ will be denoted by ι.

Definition 2.12. The *symmetric group*, denoted by S_n, is the set of all permutations of $\{1, 2, \ldots, n\}$ with the operation $\circ$ and with ι as neutral element.

Definition 2.13. The permutation π^{-1} is the *inverse* of π, i.e. the permutation such that $\pi \circ \pi^{-1} = \iota = \pi^{-1} \circ \pi$.

π^{-1} is obtained by exchanging positions and elements in π: for $1 \leq i \leq n$, we have $\pi^{-1}_{\pi_i} = i$. For instance, the inverse of $\langle 4\ 8\ 9\ 7\ 6\ 5\ 1\ 3\ 2\ 10\rangle$ is obtained as follows:

$$\begin{aligned}
\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 4 & 8 & 9 & 7 & 6 & 5 & 1 & 3 & 2 & 10 \end{pmatrix}^{-1} &= \begin{pmatrix} 4 & 8 & 9 & 7 & 6 & 5 & 1 & 3 & 2 & 10 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 7 & 9 & 8 & 1 & 6 & 5 & 4 & 2 & 3 & 10 \end{pmatrix}
\end{aligned}$$

and can be written as $\langle 7\ 9\ 8\ 1\ 6\ 5\ 4\ 2\ 3\ 10\rangle$.

2.2.2 The cycle structure of a permutation

Alternatively, permutations can be represented as the product of *disjoint cycles*. This representation is unique, up to ordering of the cycles and of the elements within each cycle, and corresponds to the following graphical representation of a permutation.

Definition 2.14. The *graph of a permutation* π has vertex set $\{1, 2, \ldots, n\}$ and contains an edge (i, j) whenever $\pi_i = j$.

For instance, when $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3\rangle$, the disjoint cycle notation is $\pi = (1, 4, 2)(3, 6, 7)(5)$ (notice the parentheses and the commas). Figure 2.1 shows the graph of this permutation.

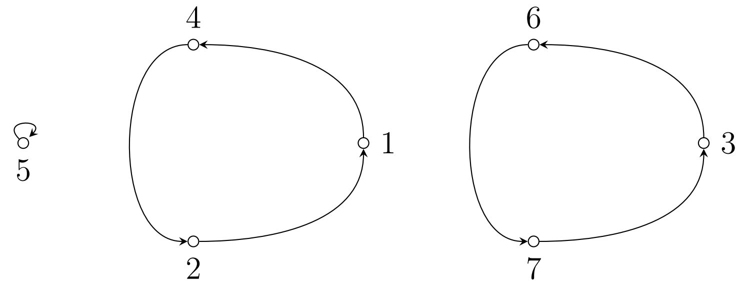


Figure 2.1: The graph of $\pi = \langle 4\ 1\ 6\ 2\ 5\ 7\ 3\rangle = (1, 4, 2)(3, 6, 7)(5)$.

Definition 2.15. A *fixed point* of a permutation π is a position i such that $\pi_i = i$.

Fixed points of permutations correspond to 1-cycles in the disjoint cycle notation or in their graph.

Definition 2.16. A permutation is called a *k-cycle* if it fixes all but $k > 1$ elements, and all nonfixed elements belong to the same cycle.

Definition 2.17. A permutation is *even* if the number of even cycles in its disjoint cycle decomposition is even or, equivalently, if it can be expressed as a product of an even number of 2-cycles. Otherwise, it is *odd*.

Definition 2.18. The *alternating group* A_n is the subgroup of S_n formed by all even permutations.

Definition 2.19. The *conjugate* of a permutation π by a permutation σ (both in S_n) is the permutation

$$\pi^\sigma = \sigma \circ \pi \circ \sigma^{-1}.$$

That permutation has the same disjoint cycle decomposition as π, and can be obtained, if $\pi = (c_{1,1}, c_{1,2}, \ldots, c_{1,\ell_1}) \cdots (c_{m,1}, c_{m,2}, \ldots, c_{m,\ell_m})$, by replacing each element in each cycle of π with the element onto which it is mapped by σ, i.e. $\pi^\sigma = (\sigma_{c_{1,1}}, \sigma_{c_{1,2}}, \ldots, \sigma_{c_{1,\ell_1}}) \cdots (\sigma_{c_{m,1}}, \sigma_{c_{m,2}}, \ldots, \sigma_{c_{m,\ell_m}})$.

Definition 2.20. All permutations that have the same disjoint cycle decomposition form a *conjugacy class*.

For example, $\pi = (1, 2, 3)(4, 5, 6)$ and $\sigma = (1, 3, 5)(2, 4, 6)$ belong to the same conjugacy class. This is the particular case for the symmetric group of the general notion of a conjugacy class in any group.

2.3 Distances on permutation groups

Definition 2.21. A *distance* d on a set S is an application

$$d : S \times S \to \mathbb{R} : (s, t) \mapsto d(s, t)$$

satisfying the following three axioms:

1. $\forall\ s, t \in S : d(s, t) \geq 0$ and $d(s, t) = 0 \Leftrightarrow s = t$ (positivity);
2. $\forall\ s, t \in S : d(s, t) = d(t, s)$ (symmetry);
3. $\forall\ s, t, u \in S : d(s, u) \leq d(s, t) + d(t, u)$ (triangular inequality).

We will be interested, among other properties, in the maximal value a distance can reach.

Definition 2.22. The *diameter* of a set S under a distance d is $\max_{s,t \in S} d(s, t)$.

In this thesis, we will concentrate on distances between permutations that are based on a set $\mathscr{A} \subseteq S_n$ of allowed operations: a distance $d_{\mathscr{A}}(\pi, \sigma)$ between any two permutations π and σ in S_n can then be defined as the minimum number of steps needed to transform π into σ (or σ into π, by symmetry), where a valid step is an operation that belongs to our fixed set $\mathscr{A}$ of allowed operations. When transforming π into σ, each step is modelled by the right-multiplication of the current permutation by some element x of $\mathscr{A}$; therefore, transforming π into σ in t steps will be done as follows:

$$\pi \circ x_1 \circ x_2 \circ \cdots \circ x_t = \sigma, \tag{2.1}$$

where x_1, x_2, …, $x_t \in \mathscr{A}$.

As an example, let $\mathscr{A}$ be the set of all permutations in S_n that fix all but two elements, i.e. $\mathscr{A} = \{\langle 1\ 2\ \cdots\ i-1\ j\ i+1\ \cdots\ j-1\ i\ j+1\ \cdots\ n\rangle \mid 1 \leq i < j \leq n\}$; right-multiplying a permutation by an element of $\mathscr{A}$ therefore consists in exchanging any two elements in that permutation. Therefore, transforming $\pi = \langle 3\ 2\ 5\ 4\ 1\rangle$ into ι using only elements of $\mathscr{A}$ can be done by subsequent exchanges of the boxed elements below:

$$\langle\ \boxed{3}\ 2\ 5\ 4\ \boxed{1}\ \rangle \rightarrow \langle 1\ 2\ \boxed{5}\ 4\ \boxed{3}\ \rangle \rightarrow \langle 1\ 2\ 3\ 4\ 5\rangle,$$

and this transformation can in turn be expressed by

$$\pi \circ \langle 5\ 2\ 3\ 4\ 1\rangle \circ \langle 1\ 2\ 5\ 4\ 3\rangle = \iota.$$

It can be easily seen that π cannot be transformed into ι using only one element of $\mathscr{A}$, and therefore we say that the distance (associated to $\mathscr{A}$) between π and ι is 2, as proved by the above sequence of length 2.

For any two elements of S_n to be a finite distance apart, the set $\mathscr{A}$ of allowed operations must satisfy the following necessary and sufficient condition.

Definition 2.23. A set $\mathscr{A} \subset S_n$ is said to *generate* S_n, or to be a *generating set* of S_n, if every element of S_n can be expressed as the product of a finite number of elements of $\mathscr{A}$. We call the elements of $\mathscr{A}$ *generators* of S_n.

Additionally, all distances we will study satisfy the following important property.

Definition 2.24. A distance d on S_n is *left-invariant* if for all π, σ, τ in S_n:

$$d(\pi, \sigma) = d(\tau \circ \pi, \tau \circ \sigma).$$

Left-invariance is an important underlying concept in genome rearrangements, because it is the reason why many problems considered in that field reduce to a sorting problem: indeed, if d is left-invariant, then computing $d(\pi, \sigma)$ for any π and σ in S_n is equivalent to computing $d(\sigma^{-1} \circ \pi, \iota)$, and we can therefore restrict our attention to the problem of computing the distance between a permutation and the identity permutation. An immediate corollary of this property is that for any π in S_n, we have $d(\pi, \iota) = d(\pi^{-1}, \iota)$ if d is left-invariant. Since we will most of the time be considering the distance between a permutation π and the identity permutation ι, we will often abbreviate $d(\pi, \iota)$ to $d(\pi)$. Intuitively, left-invariance models the fact that, given any two genomes G_1 and G_2 containing the same set and number of genes (without duplications) to be transformed into one another, it does not matter how we choose to number genes in G_1 (resp. G_2), as long as we assign numbers to genes in G_2 (resp. G_1) accordingly.

In this case, it is easily seen that our rearrangement problems based on $\mathscr{A}$ are equivalent to finding a minimum-length factorisation of π that consists only of elements of $\mathscr{A}$: indeed, if we replace σ with ι in (2.1), then

$$\pi \circ x_1 \circ x_2 \circ \cdots \circ x_t = \iota \Leftrightarrow \pi = x_t^{-1} \circ x_{t-1}^{-1} \circ \cdots \circ x_1^{-1},$$

and x_i^{-1} still belongs to $\mathscr{A}$ for $1 \leq i \leq t$ (otherwise symmetry does not hold and $d_{\mathscr{A}}(\pi, \sigma)$ is not a distance).

Problems related to the factorisation of permutations have been studied long before mathematicians and computer scientists became interested in genome rearrangement problems. There are some general complexity results that prevent us from hoping for a general solution to the problem: Even and Goldreich [32] have shown that the following problem, which is the decision version of finding an optimal factorisation, is NP-complete.

MINIMUM GENERATOR SEQUENCE
Instance: a set $\mathscr{A} = \{x_1, x_2, \ldots, x_q\}$ of generators of a permutation group G, a permutation $\pi \in G$ and a natural K.
Problem: is there a generator sequence of length $l \leq K$ such that $\pi = x_{i_1} \circ x_{i_2} \circ \cdots \circ x_{i_l}$?

However, some particular cases are easy to solve: well-known examples include the case where $\mathscr{A}$ is the set of all exchanges of elements (as described on page 15), whether or not these elements are restricted to be adjacent (see Jerrum [48] for more details, as well as other examples). We will show strong connections between these problems on even permutations and some well-studied genome rearrangement problems in Chapter 5.

Finally, we recall the following natural graphical representation of S_n using a given generating set, which is fundamental in group theory.

Definition 2.25. Given a generating set $\mathscr{A}$ of S_n, the *Cayley graph* associated with $(S_n, \mathscr{A})$ is the graph whose vertices are the elements of S_n and whose edges connect two vertices such that the corresponding elements can be transformed into one another using an element of $\mathscr{A}$.

That graph can be either directed or undirected, according to whether or not $\mathscr{A}$ contains the inverse of each generator. Figure 2.2 shows the Cayley graph associated with $(S_4, \mathscr{A})$, where $\mathscr{A}$ is the set of all 2-cycles of the form $(i, i+1)$, for $1 \leq i \leq n-1$. The notion of diameter (Definition 2.22 page 14) has a natural interpretation in that setting: it is the length of the "longest shortest path" between any two vertices of the Cayley graph corresponding to the given group and set of generators.

We conclude this chapter by mentioning that Cayley graphs of permutation groups have received a lot of attention in a field known as the design of interconnection networks, because many of those graphs seem to have properties that are desirable with respect to the criteria in use in that field. We will talk more about that topic in Chapter 5, where we will see that, interestingly enough, genome rearrangements and interconnection network design have common interests. For much more information about the use of Cayley graphs of permutation groups as interconnection networks, see for instance the seminal paper by Akers and Krishnamurthy [2] and the survey by Lakshmivarahan, Jwo, and Dhall [54].

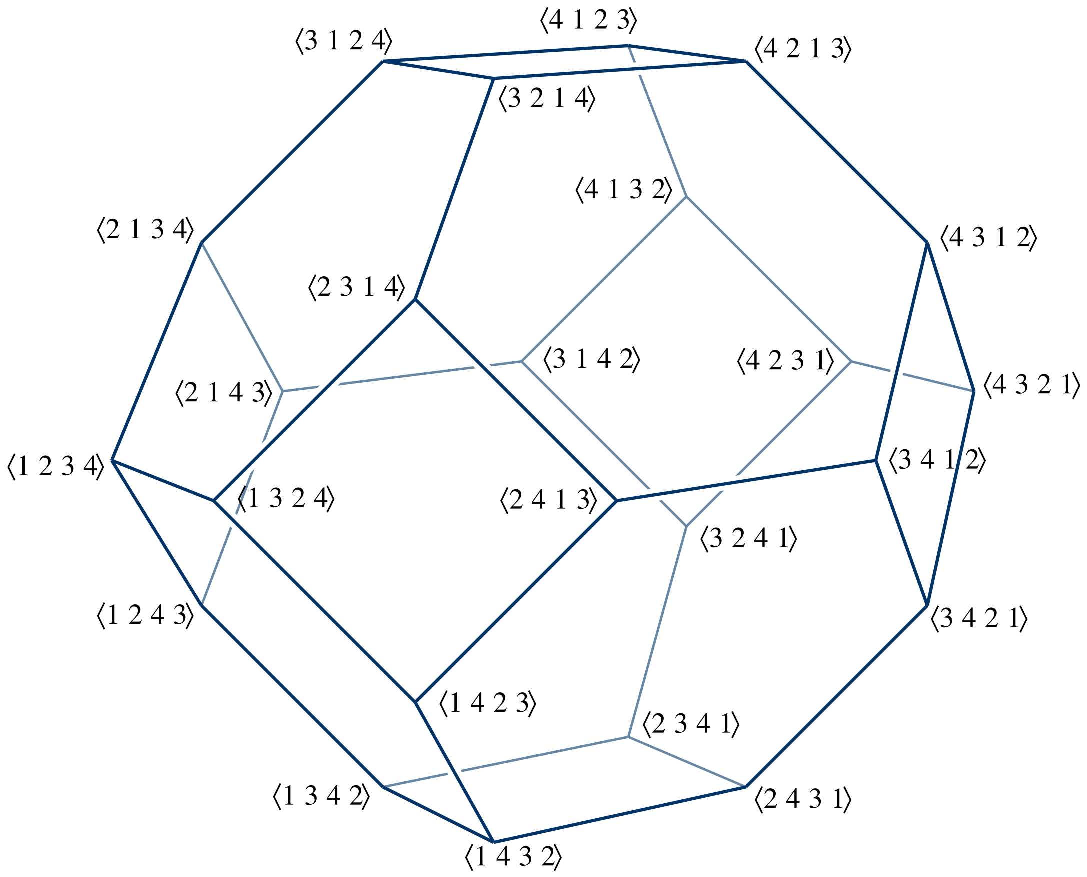


Figure 2.2: The Cayley graph associated with $(S_4, \mathscr{A})$, where $\mathscr{A}$ is the set of all 2-cycles of the form $(i, i+1)$, for $1 \leq i \leq n-1$, also known as the *permutohedron* of order 4 (based on a picture by David Eppstein).

Chapter 3

Sorting by transpositions

In 1995, Bafna and Pevzner [8] initiated the combinatorial and algorithmic study of evolution under transpositions, using the permutation model. Relying on the framework they had introduced in the context of reversals (see Bafna and Pevzner [7]), they succeeded in obtaining various bounds and results, including a 3/2-approximation algorithm for transforming a permutation of n elements into the identity permutation using as few transpositions as possible, running in $O(n^2)$ time.

Thirteen years later, not much progress on the problem has been done, as compared to other similar genome rearrangement problems: the computational complexity of sorting by transpositions, computing the transposition distance, and even determining the maximal value of the transposition distance are still open problems. Building on the results obtained by Bafna and Pevzner [9], simpler and faster algorithms were designed by Christie [23] and Hartman [45], but the approximation guarantee was not improved until 2006, when Elias and Hartman [30] proposed an 11/8-approximation algorithm with time complexity $O(n^2)$. The proof of correctness of that algorithm is heavily computer-driven and based on a huge case analysis, requiring the verification of more than 80 000 configurations. Therefore, it does not seem that designing a simpler algorithm, or one with a better ratio, will be an easy task.

Our main contributions to this problem are the following:

- we establish connections between the common graph of a permutation and the "cycle graph" introduced by Bafna and Pevzner [9], for a particular class of permutations (Proposition 3.4);

- we use those connections to prove formulae for computing, in polynomial time, the transposition distance of a few nontrivial classes of permutations (Theorems 3.10 and 3.14, Corollary 3.2, Proposition 3.9), bypassing the use of any graph structure;

- we prove a new tight upper bound on the transposition distance (Theorems 3.11 and 3.13), and improve that upper bound in some other cases (Proposition 3.12).

Most results presented in this chapter were published in [51] and later in an extended version [52].

3.1 Notation and preliminaries

3.1.1 Transpositions

The word *transposition*, defined below, must be understood in the biological sense: it refers to an evolutionary event that consists, informally, in displacing a segment of the genome, as shown in the following example:

$$\langle 2\ 1\ \boxed{7\ 3\ 5\ 6}\ 4\ 10\ 9\ 8\rangle \quad \text{becomes} \quad \langle 2\ 1\ 4\ 10\ 9\ 7\ 3\ 5\ 6\ 8\rangle.$$

By contrast with *algebraic* transpositions[1], which exchange two not necessarily adjacent elements, *biological* transpositions can be seen as exchanges of adjacent disjoint intervals. Indeed, the transposition considered in the above example exchanges the interval delimited by elements 7 and 6 with the interval delimited by elements 4 and 9.

Definition 3.1. For any π in S_n, the *transposition* $\tau(i, j, k)$ with $1 \leq i < j < k \leq n + 1$ applied to π exchanges the closed intervals determined respectively by i and $j - 1$ and by j and $k - 1$, transforming π into $\pi \circ \tau(i, j, k)$. Therefore, $\tau(i, j, k)$ is the following permutation:

$$\begin{pmatrix} 1 \cdots i-1 & \boxed{i \cdots j-1} & \boxed{j \cdots k-1} & k \cdots n \\ 1 \cdots i-1 & \boxed{j \cdots k-1} & \boxed{i \cdots j-1} & k \cdots n \end{pmatrix}.$$

Notice that the notation $\tau(i, j, k)$ stands for both this element of S_n and the action it induces on S_n (which is called a *right translation* of S_n in algebra).

Definition 3.2. The *transposition distance* between two permutations π and σ in S_n, denoted by $td(\pi, \sigma)$, is the length of a shortest sequence of transpositions that transforms π into σ.

For instance, the transposition distance between $\langle 3\ 1\ 4\ 2\rangle$ and ι is 2; indeed:

$$\langle 3\ \boxed{1}\ \boxed{4}\ 2\rangle \rightarrow \langle\ \boxed{3\ 4}\ \boxed{1\ 2}\ \rangle \rightarrow \langle 1\ 2\ 3\ 4\rangle$$

and $\langle 3\ 1\ 4\ 2\rangle$ cannot be sorted by a single transposition. The transposition distance is indeed a distance on S_n in the mathematical sense (Definition 2.21 page 14), since:

1. for all π, σ in S_n: $td(\pi, \sigma) \geq 0$ and $td(\pi, \sigma) = 0 \Leftrightarrow \pi = \sigma$ (trivial);
2. for all π, σ in S_n: $td(\pi, \sigma) = td(\sigma, \pi)$: indeed, if $td(\pi, \sigma) = k$, then there exist k transpositions τ_1, τ_2, …, τ_k such that $\pi \circ \tau_1 \circ \tau_2 \circ \cdots \circ \tau_k = \sigma$. This implies that $\sigma \circ \tau_k^{-1} \circ \tau_{k-1}^{-1} \circ \cdots \circ \tau_1^{-1} = \pi$. Since the inverse of a transposition is a transposition (indeed: $(\tau(i, j, k))^{-1} = \tau(i, i + k - j, k)$), we get $td(\sigma, \pi) \leq td(\pi, \sigma)$. The opposite inequality is proved in the same way.

[1] Which we will call *exchanges* (see Definition 3.26 page 32) to avoid confusion.

3. for all π, ρ, σ in S_n: $td(\pi, \rho) \leq td(\pi, \sigma) + td(\sigma, \rho)$, otherwise $td(\pi, \rho)$ would not be minimal.

We proceed to prove that the transposition distance is left-invariant (Definition 2.24 page 15), as are many other rearrangement distances.

Proposition 3.1. *The transposition distance is left-invariant.*

Proof. We must show that for all π, σ, ξ in S_n: $td(\pi, \sigma) = td(\xi \circ \pi, \xi \circ \sigma)$. If $td(\pi, \sigma) = m$, then there exists an optimal sequence of m transpositions such that

$$\pi \circ \tau_1 \circ \cdots \circ \tau_m = \sigma,$$

so

$$(\xi \circ \pi) \circ \tau_1 \circ \cdots \circ \tau_m = (\xi \circ \sigma),$$

which implies that $td(\pi, \sigma) = td(\xi \circ \pi, \xi \circ \sigma)$, since the existence of a shorter sequence of transpositions transforming $\xi \circ \pi$ into $\xi \circ \sigma$ would contradict the optimality of $\tau_1 \circ \cdots \circ \tau_m$. □

This property of the transposition distance allows us to restrict our attention to the following problem:

SORTING BY TRANSPOSITIONS
Input: a permutation π in S_n.
Problem: find a minimum-length sequence of transpositions transforming π into ι.

We will therefore abbreviate $td(\pi, \sigma)$ to $td(\pi)$ whenever $\sigma = \iota$, and abuse language by referring to $td(\pi)$ as "the (transposition) distance of π".

3.1.2 The cycle graph

Bafna and Pevzner [9] introduced the following useful graph.

Definition 3.3. The *cycle graph* of π in S_n is the bicoloured directed graph $G(\pi)$, whose vertex set $(\pi_0 = 0, \pi_1, \ldots, \pi_n, \pi_{n+1} = n+1)$ is ordered by positions, and whose arc set consists of:

- *black* arcs (π_i, π_{i-1}) for $1 \leq i \leq n + 1$;
- *grey* arcs $(\pi_i, \pi_i + 1)$ for $0 \leq i \leq n$.

The set of black and grey arcs decomposes in a single way into *alternating cycles*, i.e. cycles that alternate black and grey arcs. Figure 3.1 shows an example of a cycle graph, together with its decomposition.

Definition 3.4. The *length* of an alternating cycle in $G(\pi)$ is the number of black arcs it contains, and a *k-cycle* in $G(\pi)$ is an alternating cycle of length k.

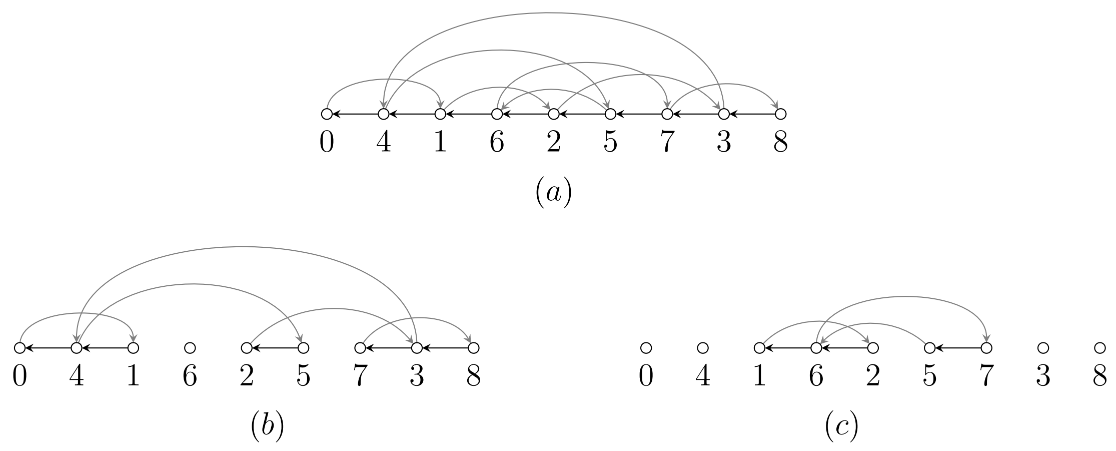


Figure 3.1: (a) The cycle graph of $\langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$, (b) and (c) the two cycles in its decomposition.

Note that these definitions of length and k-cycle disagree with the standard graph-theoretical definitions introduced in the previous chapter (page 12). However, we will see in Chapter 4 that they make sense.

Definition 3.5. A k-cycle in $G(\pi)$ is *odd* (resp. *even*) if k is odd (resp. even).

The number of cycles of $G(\pi)$ will be denoted by $c(G(\pi))$, and the number of odd (resp. even) alternating cycles in $G(\pi)$ will be denoted by $c_{odd}(G(\pi))$ (resp. $c_{even}(G(\pi))$). Bafna and Pevzner [9] proved the following lower bound on the transposition distance.

Theorem 3.1. *[9] For all π in S_n:*

$$td(\pi) \geq \frac{n+1-c_{odd}(G(\pi))}{2} .$$

Theorem 3.1 follows from the facts that the identity permutation is the only permutation whose cycle graph contains $n+1$ odd alternating cycles and that a transposition can create at most two new odd alternating cycles in the cycle graph (see Figure 3.2).

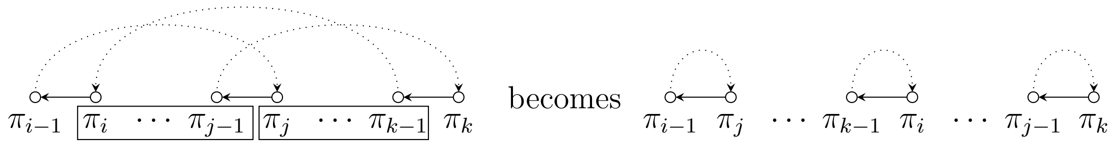


Figure 3.2: A transposition creating two new odd alternating cycles in $G(\pi)$; here, dotted arcs stand for alternating paths in the graph (i.e. paths alternating grey and black arcs).

Definition 3.6. A cycle in $G(\pi)$ is *nonoriented* if it contains exactly one grey arc directed from left to right, and *oriented* otherwise.

For instance, both cycles in the decomposition of the graph of Figure 3.1 are oriented; examples of nonoriented cycles will appear in Figure 3.4 (page 26). A transposition $\tau(i, j, k)$ is said to *act* on black arcs coming out of vertices π_i, π_j and π_k in $G(\pi)$. By extension, a transposition acts on one cycle (resp. on two or three cycles) if all three black arcs on which it acts belong to that cycle (resp. to those two or three cycles).

Definition 3.7. For a permutation π, a *k-transposition* is a transposition τ such that $c(G(\pi \circ \tau)) = c(G(\pi)) + k$.

Lemma 3.1. *[9] A transposition that acts on exactly two cycles in $G(\pi)$ is a* 0*-transposition.*

Figure 3.3 illustrates Lemma 3.1. Two alternating cycles can interact in several different ways, which we define below. To every alternating cycle C in a cycle graph $G(\pi)$, associate an interval I_C defined by the minimum and maximum indices of the vertices that belong to C.

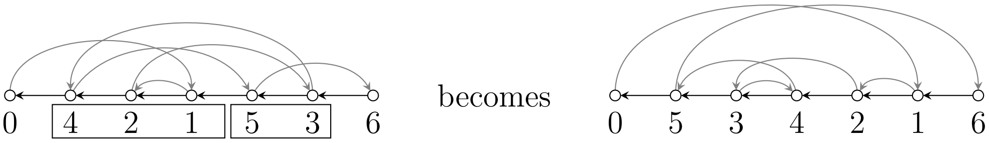


Figure 3.3: Illustration of Lemma 3.1 on $\langle 4\ 2\ 1\ 5\ 3 \rangle$; the cycle graph of the resulting permutation still decomposes into two alternating cycles.

Definition 3.8. A cycle C_1 *contains* a cycle C_2 if $I_{C_1} \supset I_{C_2}$ and no black arc of C_1 belongs to I_{C_2}.

Definition 3.9. Two alternating cycles C_1, C_2 *cross* if they do not contain each other and at least one black arc of C_1 (resp. C_2) belongs to I_{C_2} (resp. I_{C_1}).

Definition 3.10. Two alternating cycles C_1, C_2 *interleave* if when reading the black arcs of C_1 and C_2 from left to right, we alternately get a black arc from either cycle.

For instance, the two alternating cycles in the cycle graph of $\langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$ cross, as can be easily seen in Figure 3.1; the "long" cycle in the cycle graph of $\langle 5\ 3\ 4\ 2\ 1 \rangle$, shown in Figure 3.3, contains the 1-cycle formed by elements 3 and 4.

3.1.3 Reduced permutations

The study of sorting by transpositions can be simplified by restricting oneself to a particular class of permutations, defined below.

Definition 3.11. For a permutation π, an ordered pair (π_i, π_{i+1}) is a *breakpoint* if $\pi_{i+1} \neq \pi_i + 1$, and an *adjacency* otherwise. The number of breakpoints of π is denoted by $b(\pi)$.

There seems to be some confusion in the genome rearrangement literature as to how breakpoints and adjacencies are defined: some authors define them on the actual permutation π, while others base their definitions on what we will call the *extended* permutation $\widetilde{\pi} = \langle 0\ \pi_1\ \pi_2\ \cdots\ \pi_n\ n+1 \rangle$ – the only difference being in the fact that in the second case, the index i in Definition 3.11 takes its values in $\{0, 1, 2, \ldots, n\}$ rather than in $\{1, 2, \ldots, n-1\}$. We will adopt the second point of view because, as can easily be seen in Figure 3.3, this definition allows us to put adjacencies and 1-cycles in $G(\pi)$ in one-to-one correspondence. As an example, all three adjacencies are underlined in the permutation $\langle \underline{0\ 1}\ 3\ 2\ \underline{7\ 8}\ 4\ 9\ \underline{5\ 6}\ 10 \rangle$; all other pair of elements are breakpoints, and there are seven of them.

Definition 3.12. A permutation π in S_n is *reduced* if $\langle 0\ \pi_1\ \cdots\ \pi_n\ n+1 \rangle$ has no adjacency.

Definition 3.13. A *strip* in a permutation π is a maximal interval of π that contains no breakpoint.

Christie [23] shows that every permutation can be uniquely transformed into a reduced permutation without affecting its distance. The transformation of a permutation π into its reduced version $gl(\pi)$ consists in partitioning π into strips, then removing the first strip if it begins with 1, the last strip if it ends with n, replacing every other strip with its minimal element and finally, renumbering the resulting sequence so as to obtain a new permutation of a possibly smaller set. We illustrate this transformation on the following example: $\langle \underline{1}\ \underline{3}\ \underline{2}\ \underline{7\ 8}\ \underline{4}\ \underline{9}\ \underline{5\ 6} \rangle$ decomposes into seven underlined strips, and the first strip begins with 1, so we remove it, thereby obtaining $\langle \underline{3}\ \underline{2}\ \underline{7\ 8}\ \underline{4}\ \underline{9}\ \underline{5\ 6} \rangle$. We then replace each other strip with its minimal element, which yields $\langle 3\ 2\ 7\ 4\ 9\ 5 \rangle$, and finally, we renumber the remaining elements, and obtain the permutation $\langle 2\ 1\ 5\ 3\ 6\ 4 \rangle$. Since an adjacency is a 1-cycle in $G(\pi)$, a reduced permutation can also be defined as one whose cycle graph has no 1-cycles[2].

Definition 3.14. Two permutations π and σ are *equivalent by reduction* if $gl(\pi) = gl(\sigma)$, which we also write as $\pi \equiv_r \sigma$.

Theorem 3.2. *[23] For any two permutations π and σ: if $\pi \equiv_r \sigma$, then $td(\pi) = td(\sigma)$.*

This result confirms the intuition that it never “pays” to break adjacencies when sorting by transpositions. It also allows us to restrict our study of the transposition problems (sorting and computing the associated distance) to reduced permutations, since if we can optimally sort a reduced permutation, then we can easily deduce an optimal sorting sequence for any permutation that reduces to it.

3.1.4 Toric permutations

Eriksson, Eriksson, Karlander, Svensson, and Wästlund [31] introduced an equivalence relation on S_n, whose equivalence classes are called *toric permutations* and which we define using Hultman's notations [47].

[2] Note that $gl(\iota)$ is not defined, because every element would have to be removed. However, we may argue that this is not really a problem, since there is no point in trying to sort ι.

Definition 3.15. The *circular permutation* obtained from a permutation π in S_n is $\pi^\circ = 0\ \pi_1\ \pi_2\ \cdots\ \pi_n$, with indices taken modulo $n+1$ so that $0 = \pi^\circ_0 = \pi^\circ_{n+1}$.

This circular permutation can be read starting from any position, and the original "linear" permutation is reconstructed by taking the element following 0 as π_1 and removing 0. For x in $\{0, 1, 2, \ldots, n\}$, let $\overline{x}^m = (x+m) \pmod{n+1}$, and define the following operation on circular permutations:

$$m + \pi^\circ = \overline{0}^m\ \overline{\pi_1}^m\ \overline{\pi_2}^m\ \cdots\ \overline{\pi_n}^m.$$

We will give examples involving this operation after the next two definitions.

Definition 3.16. For any π in S_n, the *toric permutation* $\pi^\circ_\circ$ is the set of permutations in S_n reconstructed from all circular permutations $m + \pi^\circ$ with $0 \leq m \leq n$.

Definition 3.17. Two permutations π, σ in S_n are *torically equivalent* if $\sigma \in \pi^\circ_\circ$ (or $\pi \in \sigma^\circ_\circ$), which we also write as $\pi \equiv^\circ_\circ \sigma$.

For instance, let $\pi = \langle 3\ 1\ 5\ 2\ 4\ 6 \rangle$; then $\pi^\circ = 0\ 3\ 1\ 5\ 2\ 4\ 6$, and

$$\begin{array}{rcl}
0 + \pi^\circ & = & 0\ 3\ 1\ 5\ 2\ 4\ 6 \\
1 + \pi^\circ & = & 1\ 4\ 2\ 6\ 3\ 5\ 0 \\
2 + \pi^\circ & = & 2\ 5\ 3\ 0\ 4\ 6\ 1 \\
3 + \pi^\circ & = & 3\ 6\ 4\ 1\ 5\ 0\ 2 \\
4 + \pi^\circ & = & 4\ 0\ 5\ 2\ 6\ 1\ 3 \\
5 + \pi^\circ & = & 5\ 1\ 6\ 3\ 0\ 2\ 4 \\
6 + \pi^\circ & = & 6\ 2\ 0\ 4\ 1\ 3\ 5
\end{array}$$

which yields $\pi^\circ_\circ = \{\langle 3\ 1\ 5\ 2\ 4\ 6 \rangle, \langle 1\ 4\ 2\ 6\ 3\ 5 \rangle, \langle 4\ 6\ 1\ 2\ 5\ 3 \rangle, \langle 2\ 3\ 6\ 4\ 1\ 5 \rangle, \langle 5\ 2\ 6\ 1\ 3\ 4 \rangle, \langle 2\ 4\ 5\ 1\ 6\ 3 \rangle, \langle 4\ 1\ 3\ 5\ 6\ 2 \rangle\}$, and all permutations in that set are torically equivalent. The following property is the main reason why toric permutations were introduced.

Lemma 3.2. *[31] For all π, σ in S_n: if $\pi \equiv^\circ_\circ \sigma$, then $td(\pi) = td(\sigma)$.*

Hultman [47] proved another related interesting result.

Lemma 3.3. *[47] For all π in S_n and $0 \leq m \leq n$: every cycle in $G(\pi)$ is mapped onto a cycle in $G(\sigma)$, where σ is the permutation obtained from $\pi^\circ + m$.*

The toric equivalence relation therefore preserves the structure of the cycle graph in a way similar to how conjugation preserves the disjoint cycle decomposition of permutation. However, this analogy must not be carried too far, since all permutations that have the same disjoint cycle decomposition belong to the same conjugacy class, but two permutations whose cycle graph decomposes in a similar way are not necessarily torically equivalent: for instance, the cycle graphs of both $\pi = \langle 5\ 4\ 3\ 2\ 1 \rangle$ and $\sigma = \langle 2\ 1\ 3\ 5\ 4 \rangle$ decompose into two alternating cycles of length three, but it can easily be checked that $\pi \not\equiv^\circ_\circ \sigma$.

3.1.5 Upper bounds on the transposition distance

Since we are going to prove new upper bounds on the transposition distance, it is only fair that we conclude this introductory section with all upper bounds on the transposition distance we are aware of. We will experimentally compare those upper bounds to our new bounds (see Section 3.6) in order to assess the quality of our results.

Theorem 3.3. *[9] For all π in S_n:*

$$td(\pi) \leq n + 1 - c(G(\pi)). \tag{3.1}$$

Theorem 3.4. *[9] For all π in S_n:*

$$td(\pi) \leq \frac{3(n + 1 - c_{odd}(G(\pi)))}{4} \ . \tag{3.2}$$

Theorem 3.5. *[27] For all π in S_n:*

$$td(\pi) \leq 3\ b(\pi)/4. \tag{3.3}$$

Theorem 3.6. *[31] For all π in S_n:*

$$td(\pi) \leq \begin{cases} \lceil 2n/3 \rceil & \text{if } n < 9\ ; \\ \lfloor (2n-2)/3 \rfloor & \text{if } n \geq 9. \end{cases} \tag{3.4}$$

Elias and Hartman [30] proved upper bounds on the distance of three special classes of permutations[3].

Definition 3.18. A permutation π in S_n is *simple* if $G(\pi)$ contains no cycle of length greater than three.

Definition 3.19. A permutation π in S_n is a 2-*permutation* (resp. 3-*permutation*) if all cycles in $G(\pi)$ are of length 2 (resp. 3).

Note that a 2-permutation (resp. 3-permutation) only exists if $n + 1$ can be divided by 4 (resp. 3). This follows directly from the definitions and, in the case of 2-permutations, from the fact that $c_{even}(G(\pi))$ is even for all π in S_n (see Christie [23]). Figure 3.4 shows examples of permutations from each of those classes.

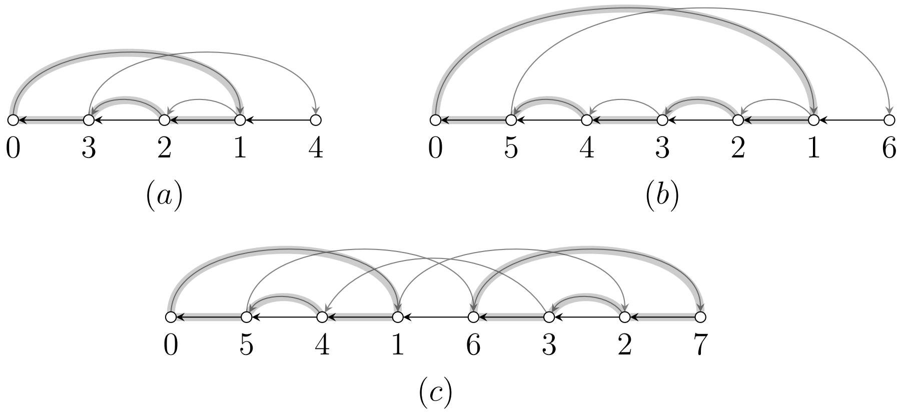
Theorem 3.7. *[30] For every simple permutation π in S_n which is not a 3-permutation:*

$$td(\pi) \leq \lfloor (n + 1)/2 \rfloor . \tag{3.5}$$

Theorem 3.8. *[30] For every 2-permutation π in S_n :*

$$td(\pi) \leq (n + 1)/2. \tag{3.6}$$

[3] The definitions we give here are not the ones introduced by Hannenhalli and Pevzner [44] and Elias and Hartman [30], but we prove the equivalence between our definitions and theirs in Section 3.9.



(a)

(b)

(c)

Figure 3.4: The cycle graphs of (a) a 2-permutation, (b) a 3-permutation, (c) a simple permutation. A few alternating cycles have been highlighted to help the reader see the decompositions.

This upper bound can easily be seen to be the actual transposition distance of 2-permutations: indeed, their cycle graph contains no odd cycle, and as a result their transposition distance is at least $(n+1)/2$, according to Theorem 3.1.

Theorem 3.9. *[30] For every 3-permutation π in S_n :*

$$td(\pi) \leq 11 \left\lfloor \frac{n+1}{24} \right\rfloor + \left\lfloor \frac{3(\frac{n+1}{3} \bmod 8)}{2} \right\rfloor + 1. \tag{3.7}$$

Finally, Guyer, Heath, and Vergara [43] deduced an upper bound based on increasing subsequences.

Definition 3.20. Given a permutation π, a *subsequence* of π is a subset $\pi_{i_1}, \ldots, \pi_{i_k}$ of not necessarily contiguous elements of π, with $i_1 < i_2 < \cdots < i_k$. The subsequence is *increasing* if $\pi_{i_1} < \pi_{i_2} < \cdots < \pi_{i_k}$, and it is a *longest increasing subsequence* if there is no other increasing subsequence in π with more elements.

Guyer et al. [43] observe that a permutation can be sorted by transpositions by "growing" its longest increasing subsequence. That subsequence can always be increased by at least one at each step, which yields the following upper bound.

Observation 3.1. *[43] For all π in S_n :*

$$td(\pi) \leq n - |LIS(\pi)|, \tag{3.8}$$

where $|LIS(\pi)|$ is the length of a longest increasing subsequence of π.

3.2 The distribution of the transposition distance

Table 3.1 shows some experimental values of the number of permutations in S_n with transposition distance equal to k. The distribution was obtained by generating S_n from the identity permutation by repeatedly composing transpositions.

$n\backslash k$	0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	4	1	0	0	0	0	0
4	1	10	12	1	0	0	0	0
5	1	20	68	31	0	0	0	0
6	1	35	259	380	45	0	0	0
7	1	56	770	2 700	1 513	0	0	0
8	1	84	1 932	13 467	22 000	2 836	0	0
9	1	120	4 284	52 512	191 636	114 327	0	0
10	1	165	8 646	170 907	1 183 457	2 010 571	255 053	0
11	1	220	16 203	484 440	5 706 464	21 171 518	12 537 954	0

Table 3.1: Number of permutations in S_n with transposition distance equal to k, for $1 \leq n \leq 11$.

In Sections 3.1.3 and 3.1.4, we have seen two equivalence relations on permutations that preserve the transposition distance. Two other classes can actually be deduced.

Proposition 3.2. *For all π in S_n: $td(\pi) = td(\pi^{-1})$.*

Proof. Straightforward by left-invariance (Proposition 3.1):

$$td(\pi, \iota) = td(\pi^{-1} \circ \pi, \pi^{-1}) = td(\pi^{-1}, \iota).$$

□

Another special permutation can be obtained from π which has the same distance: the conjugate of π by the *reversed permutation* $\chi = \langle n \; n-1 \; n-2 \; \cdots \; 3 \; 2 \; 1\rangle$. To prove this, we first show that if τ is a transposition, then so is its conjugate τ^χ.

Lemma 3.4. *For any transposition $\tau(i, j, k)$ in S_n, we have*

$$(\tau(i, j, k))^\chi = \tau(n - k + 2, n - j + 2, n - i + 2).$$

Proof. Using the definitions of a transposition and of a conjugate, we have:

$$\begin{aligned}
(\tau(i,j,k))^\chi &= \begin{pmatrix} \chi_1 \cdots \chi_{i-1} & \boxed{\chi_i \cdots \chi_{j-1}} & \boxed{\chi_j \cdots \chi_{k-1}} & \chi_k \cdots \chi_n \\ \chi_1 \cdots \chi_{i-1} & \boxed{\chi_j \cdots \chi_{k-1}} & \boxed{\chi_i \cdots \chi_{j-1}} & \chi_k \cdots \chi_n \end{pmatrix} \\
&= \begin{pmatrix} n \cdots n-i+2 & \boxed{n-i+1 \cdots n-j+2} & \boxed{n-j+1 \cdots n-k+2} & n-k+1 \cdots 1 \\ n \cdots n-i+2 & \boxed{n-j+1 \cdots n-k+2} & \boxed{n-i+1 \cdots n-j+2} & n-k+1 \cdots 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 \cdots n-k+1 & \boxed{n-k+2 \cdots n-j+1} & \boxed{n-j+2 \cdots n-i+1} & n-i+2 \cdots n \\ 1 \cdots n-k+1 & \boxed{n-j+2 \cdots n-i+1} & \boxed{n-k+2 \cdots n-j+1} & n-i+2 \cdots n \end{pmatrix} \\
&= \tau(n-k+2, n-j+2, n-i+2).
\end{aligned}$$

□

Proposition 3.3. *For all π in S_n, we have $td(\pi) = td(\pi^\chi)$.*

Proof. Let $\pi = \tau_r \circ \cdots \circ \tau_1$, where $r = td(\pi)$; then

$$\begin{aligned} \chi \circ \pi \circ \chi^{-1} &= \chi \circ \tau_r \circ \cdots \circ \tau_1 \circ \chi^{-1} \\ &= \underbrace{(\chi \circ \tau_r \circ \chi^{-1})}_{\tau'_r} \circ \underbrace{(\chi \circ \tau_{r-1} \circ \chi^{-1})}_{\tau'_{r-1}} \circ \cdots \circ \underbrace{(\chi \circ \tau_1 \circ \chi^{-1})}_{\tau'_1}, \end{aligned}$$

where $\tau'_1, \ldots, \tau'_r$ are again transpositions (Lemma 3.4). Therefore

$$td(\pi) \geq td(\pi^\chi) \geq td((\pi^\chi)^\chi) = td(\pi).$$

□

The following corollary is a consequence of Propositions 3.1, 3.2 and 3.3.

Corollary 3.1. *For all π in S_n: $td(\pi, \iota) = td(\pi \circ \chi, \chi) = td(\pi^{-1} \circ \chi, \chi)$.*

3.3 Another useful graph

We introduce a slight variant of the well-known graph of a permutation (Definition 2.14 page 13), in which vertices are ordered by their position in the permutation. This graph will allow us to connect the disjoint cycle decomposition of some permutations and the alternating cycle decomposition of their cycle graph in Section 3.4.

Definition 3.21. The Γ-*graph* of a permutation π in S_n is the directed graph $\Gamma(\pi)$ with ordered vertex set $(\pi_1, \ldots, \pi_n)$ and arc set $\{(i, \pi_i) \mid i = 1, 2, \ldots, n\}$.

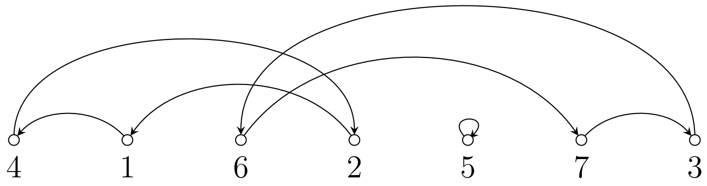


Figure 3.5: The Γ-graph of the permutation $\langle 4\ 1\ 6\ 2\ 5\ 7\ 3 \rangle$.

Figure 3.5 shows an example of a Γ-graph. The disjoint cycle decomposition of π naturally matches the decomposition of the Γ-graph into disjoint cycles, so we will refer to a cycle of length k in $\Gamma(\pi)$ as a *k-cycle* as well.

Definition 3.22. A k-cycle in $\Gamma(\pi)$ is *increasing* (resp. *decreasing*) if $k \geq 3$ and its elements can be cyclically shifted so as to form an increasing (resp. decreasing) sequence, and *nonmonotonic* otherwise. A cycle that is either increasing or decreasing is also referred to as *monotonic*.

For instance, in Figure 3.5, cycle $(4,\ 2,\ 1)$ is decreasing, cycle (5) is nonmonotonic, and cycle $(3,\ 6,\ 7)$ is increasing. In a quite similar fashion to the parity of cycles defined in the context of $G(\pi)$, a k-cycle in $\Gamma(\pi)$ is *odd* (resp. *even*) if k is odd (resp. even). Likewise, $c(\Gamma(\pi))$ denotes the number of cycles in $\Gamma(\pi)$, and $c_{odd}(\Gamma(\pi))$ (resp. $c_{even}(\Gamma(\pi))$) denotes the number of odd (resp. even) cycles in $\Gamma(\pi)$. Finally, note that Definitions 3.8, 3.9 and 3.10 naturally adapt to the Γ-graph.

3.4 An explicit formula for some permutations

We now introduce a special class of reduced permutations, and show (Proposition 3.8 page 34) that their transposition distance can be computed in polynomial time.

Definition 3.23. A *γ-permutation* is a reduced permutation that fixes all even elements.

It follows from Definition 3.23 that γ-permutations only exist for odd values of $n \geq 3$. An example of a γ-permutation is $\langle 3\ 2\ 1\ 4\ 7\ 6\ 9\ 8\ 5\rangle$. From the definition, it is easy to see that the number of γ-permutations in S_n is exactly the number of derangements of $(n+1)/2$ elements, i.e. the number of fixed point free permutations in $S_{(n+1)/2}$. The following simple observation will be of interest later.

Observation 3.2. *Any γ-permutation has a longest increasing subsequence of length $\frac{n-1}{2}$, formed by its even elements.*

The following result characterises an interesting relation between cycles in the Γ-graph and cycles in the cycle graph of γ-permutations.

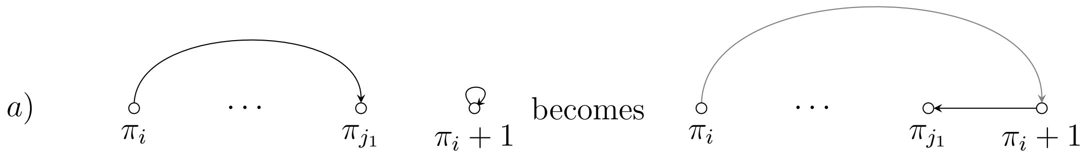
Proposition 3.4. *For every γ-permutation π in S_n:*

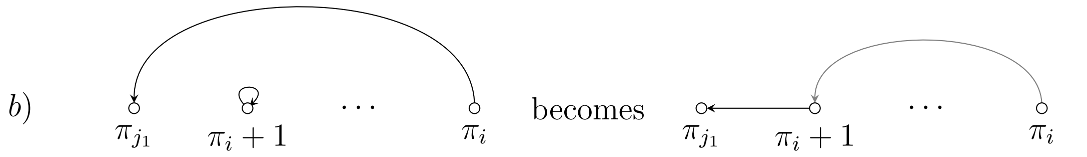
$$\begin{cases} c_{even}(G(\pi)) &= 2\ c_{even}(\Gamma(\pi)); \\ c_{odd}(G(\pi)) &= 2\left(c_{odd}(\Gamma(\pi)) - \frac{n-1}{2}\right). \end{cases}$$

Proof. Each vertex π_i of $\Gamma(\pi)$, with i odd, is both the starting point of an arc (π_i, π_{j_1}) and the ending point of an arc (π_{j_2}, π_i). From our definitions, $\pi_i + 1$ is mapped onto itself, since it is even. In $G(\pi)$, those arcs are each transformed, as explained below, into one sequence of two arcs (grey-black for the first one, black-grey for the second one):

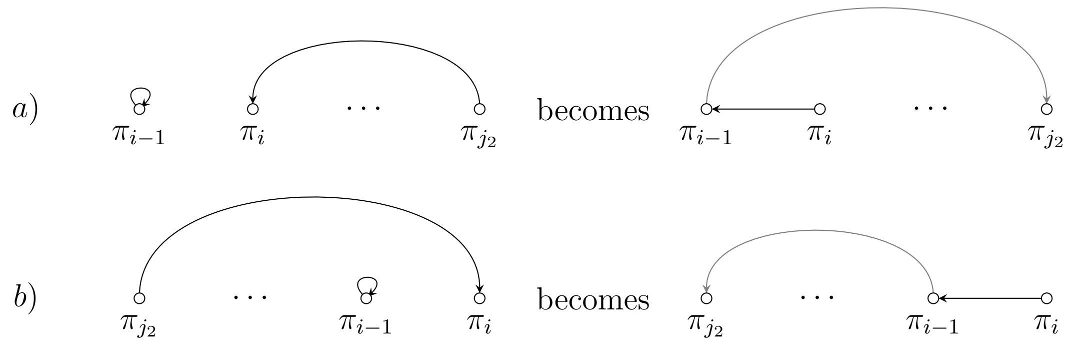
- (π_i, π_{j_1}) becomes $(\pi_i, \pi_i + 1), (\pi_i + 1, \pi_{j_1})$;
- (π_{j_2}, π_i) becomes $(\pi_i, \pi_{i-1}), (\pi_{i-1}, \pi_{j_2})$.

I.e. (π_i, π_{j_1}) is transformed in one of the following ways (depending on the relative positions of π_i and π_{j_1}):





By definition of $\Gamma(\pi)$, we know that $\pi_{j_2} = i$. Since $\pi_{i-1} = i - 1$, the arc (π_{j_2}, π_i) is transformed in one of the following ways (depending on the relative positions of π_i and π_{j_2}):



Therefore each k-cycle ($k \geq 2$) in $\Gamma(\pi)$ provides two alternating k-cycles in $G(\pi)$, one of which actually corresponds to the backwards course of the cycle in $\Gamma(\pi)$. Finally, 1-cycles in $\Gamma(\pi)$ do not produce cycles in $G(\pi)$, and there are $\frac{n-1}{2}$ of them. □

It can be easily seen that the above Proposition does not hold for all permutations: for instance, the cycle graph of $\chi = \langle n\ n-1\ \cdots\ 2\ 1\rangle$ decomposes into $n \pmod 2) + 1$ cycles of length $n+1$, but its Γ-graph contains $(n-1)/2$ 2-cycles and $n \pmod 2$ 1-cycles. The next observation follows naturally from the proof of Proposition 3.4 (recall Definitions 3.6, 3.10 and 3.22 for the concepts of "(non)oriented", "interleaving" and "(non)monotonic" cycles, respectively).

Observation 3.3. *For a γ-permutation π, the two alternating cycles C_1, C_2 in $G(\pi)$ that correspond to a k-cycle C in $\Gamma(\pi)$ interleave. Moreover:*

1. *if $k = 2$, then C_1 and C_2 are nonoriented;*
2. *if C is monotonic, then either C_1 or C_2 is oriented;*
3. *if C is nonmonotonic and $k \geq 4$, then both C_1 and C_2 are oriented.*

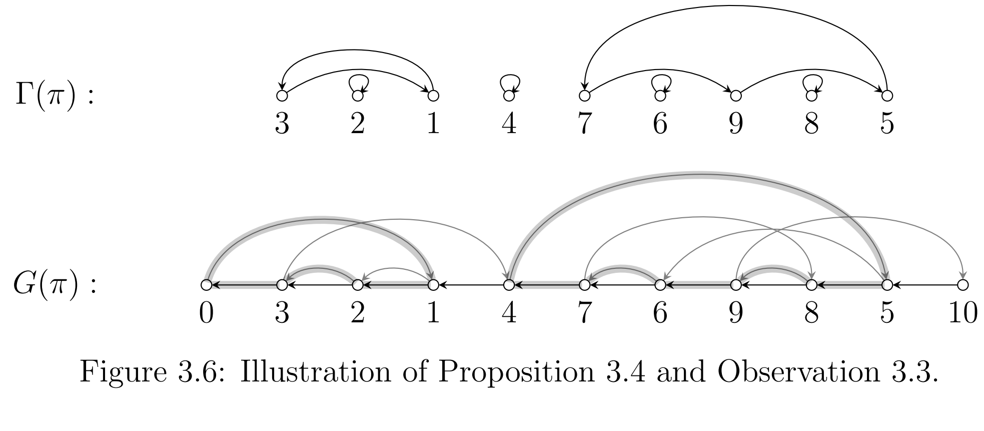


Figure 3.6: Illustration of Proposition 3.4 and Observation 3.3.

Figure 3.6 illustrates Proposition 3.4 and Observation 3.3. We derive the following lower bound from Proposition 3.4 and Theorem 3.1.

Lemma 3.5. *For every γ-permutation π in S_n, we have $td(\pi) \geq n - c_{odd}(\Gamma(\pi))$.*

Proof. Straightforward. □

We will prove that sorting each cycle in $\Gamma(\pi)$ individually is an optimal strategy for γ-permutations, and therefore that the right-hand side of the above quantity gives the actual transposition distance of γ-permutations. In order to do that, we need to be able to compute the number of transpositions required to sort each cycle, and this is why we first study γ-permutations whose Γ-graph has only one "long" k-cycle (i.e. with $k > 1$), distinguishing between monotonic cycles and nonmonotonic ones.

3.4.1 Monotonic cycles

Definition 3.24. An *α-permutation* is a reduced permutation that fixes even elements and whose $\frac{n+1}{2}$ odd elements form one monotonic cycle in the Γ-graph, referred to as its *main cycle*.

An example of an α-permutation, for $n = 7$, is $\langle 3\ 2\ 5\ 4\ 7\ 6\ 1\rangle$. Note that for fixed $n \geq 5$, there are only two α-permutations in S_n: one has an increasing main cycle, and the other has a decreasing main cycle. Therefore, the only other α-permutation, for $n = 7$, is $\langle 7\ 2\ 1\ 4\ 3\ 6\ 5\rangle = \langle 3\ 2\ 5\ 4\ 7\ 6\ 1\rangle^{-1}$.

Proposition 3.5. *For every α-permutation π in S_n, we have*

$$td(\pi) = n - c_{odd}(\Gamma(\pi)) = |C| - (|C| \bmod 2)\,,$$

where $|C| = \frac{n+1}{2}$ is the number of elements in its main cycle C.

Proof. Every α-permutation is a γ-permutation, so $td(\pi) \geq |C| - (|C| \bmod 2)$ (Lemma 3.5). We may assume without loss of generality that C is increasing, since the decreasing case corresponds to π^{-1}, and will distinguish between two cases:

1. if $|C|$ is odd, consider transpositions $\tau_1 = \tau(2, 4, n+1)$ and $\tau_2 = \tau(1, 3, n)$; we prove by induction on $|C|$ that

   $$(\tau_2 \circ \tau_1)^{\frac{|C|-1}{2}}$$

   is an optimal sorting sequence for π. The base case is $\pi = \langle 3\ 2\ 5\ 4\ 1\rangle$; we have $\pi \circ \tau_1 = \langle 3\ 4\ 1\ 2\ 5\rangle$, and $\langle 3\ 4\ 1\ 2\ 5\rangle \circ \tau_2 = \iota$. For the induction, the permutation to sort is $\pi = \langle 3\ 2\ 5\ 4\ 7\ 6\ \cdots\ n-2\ n-3\ n\ n-1\ 1\rangle$, to which we apply transpositions τ_1 and τ_2:

   $$\begin{gathered}\pi = \langle 3\ \boxed{2\ 5}\ \boxed{4\ 7\ 6\ \cdots\ n-2\ n-3\ n\ n-1\ 1}\ \rangle\\ \downarrow\\ \pi \circ \tau_1 = \langle\ \boxed{3\ 4}\ \boxed{7\ 6\ \cdots\ n-2\ n-3\ n\ n-1\ 1\ 2}\ 5\rangle\\ \downarrow\\ (\pi \circ \tau_1) \circ \tau_2 = \langle 7\ 6\ \cdots\ n-2\ n-3\ n\ n-1\ 1\ 2\ 3\ 4\ 5\rangle.\end{gathered}$$

   Reducing the latter permutation merges the last five elements into a new element called 1, and subtracts 4 to every other element. It is then clear that, if σ is the permutation for which our induction hypothesis is true, then $\pi \circ \tau_1 \circ \tau_2$ reduces to σ.

2. if $|C|$ is even, we need not even build an optimal sorting sequence, since the upper bound follows from Observations 3.1 and 3.2.

□

3.4.2 Nonmonotonic cycles

Definition 3.25. A *β-permutation* is a reduced permutation that fixes even elements and whose odd elements form one nonmonotonic cycle in the Γ-graph, referred to as its main cycle.

An example of a β-permutation is $\langle 5\ 2\ 7\ 4\ 3\ 6\ 9\ 8\ 1\rangle$. From the definition, it is easy to see that the number of β-permutations in S_n, for $n \geq 7$, is exactly the number of 1-cycles in $S_{(n+1)/2}$ minus 2 (since two α-permutations must be discarded). Note that if $n = 3$, then the only β-permutation is $\langle 3\ 2\ 1\rangle$, and there are no β-permutations for $n = 5$. We now show that Proposition 3.5 still holds if the main cycle of the Γ-graph is nonmonotonic. We use so-called *exchanges* in order to simplify the proofs, thus bypassing the construction of optimal sequences of transpositions.

Definition 3.26. For any π in S_n, the *exchange* $\varepsilon(i, j)$ with $1 \leq i < j \leq n$ applied to π is the permutation that swaps elements in positions i and j, transforming π into the permutation $\pi \circ \varepsilon(i, j)$. Therefore, $\varepsilon(i, j)$ is the following permutation:

$$\left(\begin{array}{cccccccccccc} 1 & \cdots & i-1 & \boxed{i} & i+1 & \cdots & j-1 & \boxed{j} & j+1 & \cdots & n \\ 1 & \cdots & i-1 & \boxed{j} & i+1 & \cdots & j-1 & \boxed{i} & j+1 & \cdots & n \end{array}\right).$$

In this section, we only use exchanges of the form $\varepsilon(i, i + 2k)$ with $k \geq 1$; such an exchange has the same effect as two transpositions, but determining how many transpositions are needed to achieve the same action as the product of several such exchanges requires a little more care:

Proposition 3.6. *Let $\sigma = \varepsilon(i, i+2) \circ \varepsilon(i, i+4) \circ \cdots \circ \varepsilon(i, i+2t)$; then for any π in S_n, we have:*

$$td(\pi, \pi \circ \sigma) = td(\pi, \pi \circ \sigma^{-1}) = t + (t \bmod 2).$$

Proof. By left-invariance (Proposition 3.1), we may assume without loss of generality that $\pi = \iota$, and Proposition 3.2 allows us to restrict our attention to σ. We note that $\Gamma(\sigma)$ contains one $(t+1)$-cycle and that all elements outside that cycle are fixed points, since they are never affected by any exchange. If $t = 1$, then the long cycle is nonmonotonic, and it is easily seen that $td(\sigma) = 2$; otherwise, the long cycle of $\Gamma(\sigma)$ is increasing, and all elements before position i and after position $i + 2t$ are fixed. Therefore, removing them transforms σ into $gl(\sigma)$, which is an α-permutation whose main cycle has $t + 1$ elements. By Theorem 3.2 and Proposition 3.5, we then have

$$td(\sigma) = t + 1 - ((t + 1) \bmod 2) = t + (t \bmod 2),$$

which completes the proof. □

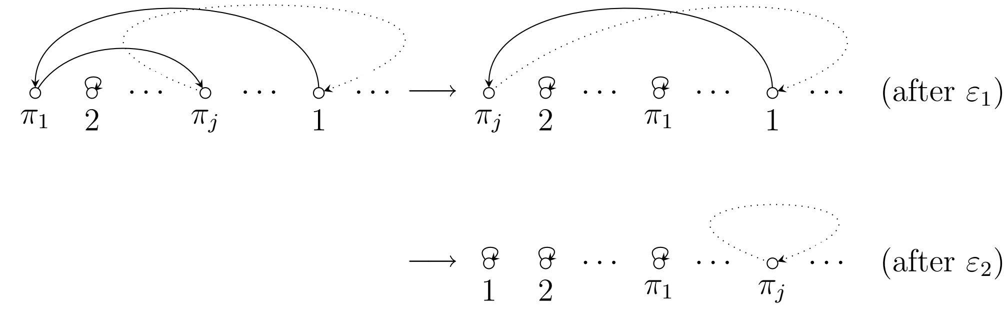
Proposition 3.7. *For every β-permutation π in S_n, we have*

$$td(\pi) = n - c_{odd}(\Gamma(\pi)) = |C| - (|C| \bmod 2)\,,$$

where $|C| = \frac{n+1}{2}$ is the number of elements in its main cycle C.

Proof. Every β-permutation is a γ-permutation, so $td(\pi) \geq |C| - (|C| \bmod 2)$ (Lemma 3.5). Just as it was the case for α-permutations, if $|C|$ is even, then we are done, since the upper bound that follows from Observations 3.1 and 3.2 equals our lower bound. Therefore, we assume that $|C|$ is odd, and we are going to prove that given any β-permutation (with $|C|$ odd), it is always possible to find a sequence of transpositions of even length that transforms it into a permutation that reduces either to a β-permutation or to an α-permutation, with an odd main cycle in both cases.

We apply two subsequent exchanges to π: the first exchange is $\varepsilon_1(1, \pi^{-1}_{\pi_1})$, and the second exchange is $\varepsilon_2(1, \pi_1^{-1})$, which respectively send π_1 and 1 to the positions they must occupy in ι. This process is schematically drawn below:



Note that $\sigma = \pi \circ \varepsilon_1 \circ \varepsilon_2$ has two more fixed points than π, still fixes all even elements and has only one cycle of length at least 2; therefore, it reduces either to an α-permutation or to a β-permutation. Now, recall that $|C| \geq 5$ and is odd; if $|C| = 5$, then the number of elements in the main cycle of σ is 3, so it reduces to an α-permutation with three elements in its main cycle. By Proposition 3.5, we have $td(\sigma) = 2$, which implies $td(\pi) \leq 2 + 2 = 4 = |C| - (|C| \bmod 2)$ and verifies the base case of the induction proof.

If $|C| > 5$, then one just needs to apply the same process to the resulting reduced version of σ as long as one does not obtain an α-permutation. Let t denote the number of pairs of exchanges that we need to use; since at each step, two exchanges that correspond to two transpositions are applied, and two elements are removed from C and turned into fixed points, we have:

$$\begin{aligned} td(\pi) &\leq 2t + (|C| - 2t) - ((|C| - 2t) \bmod 2) \\ &= |C| - ((|C| - 2t) \bmod 2) \\ &= |C| - (|C| \bmod 2), \end{aligned}$$

which completes the proof. □

3.4.3 Transposition distance of γ-permutations

Any permutation π can be sorted (by transpositions) by sorting each cycle of its Γ-graph individually, so that after sorting a cycle, the resulting permutation has the same Γ-graph as π, except that the sorted cycle has been transformed into fixed points. This strategy yields the following upper bound on $td(\pi)$.

Lemma 3.6. *For every permutation π, consider its disjoint cycle decomposition $\Gamma(\pi) = C_1 \cup C_2 \cup \cdots \cup C_{c(\Gamma(\pi))}$. Denote $td(C)$ the minimum number of transpositions required to transform $C = (i_1, i_2, \ldots, i_k)$ into $(i_1), (i_2), \ldots, (i_k)$; then*

$$td(\pi) \leq \sum_{i=1}^{c(\Gamma(\pi))} td(C_i). \tag{3.9}$$

The strategy described above works for any permutation, but is not necessarily optimal; however, we show below that it is optimal for γ-permutations.

Proposition 3.8. *For every γ-permutation π in S_n :*

$$td(\pi) = n - c_{odd}(\Gamma(\pi)). \tag{3.10}$$

Proof. Denote $odd(\Gamma(\pi))$ (resp. $even(\Gamma(\pi))$) the set of odd (resp. even) cycles in $\Gamma(\pi)$; Lemma 3.6 and Propositions 3.5 and 3.7 yield

$$\begin{aligned} td(\pi) &\leq \sum_{i=1}^{c(\Gamma(\pi))} |C_i| - (|C_i| \bmod 2) \\ &= \sum_{C_{i_1} \in\ odd(\Gamma(\pi))} (|C_{i_1}| - 1) + \sum_{C_{i_2} \in\ even(\Gamma(\pi))} |C_{i_2}| \\ &= \sum_{i=1}^{c(\Gamma(\pi))} |C_i| - c_{odd}(\Gamma(\pi)) \\ &= n - c_{odd}(\Gamma(\pi)), \end{aligned}$$

which, together with Lemma 3.5, completes the proof. □

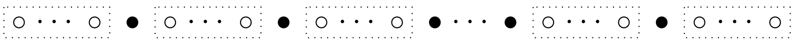
This proposition actually leads to a more general result.

Theorem 3.10. *Every permutation π in S_n that reduces to a γ-permutation has distance*

$$td(\pi) = n - c_{odd}(\Gamma(\pi)). \tag{3.11}$$

Moreover, every permutation σ in S_n with n odd and whose odd elements occupy odd positions and form an increasing subsequence modulo $n+1$ can be transformed into a permutation π such that $td(\sigma) = td(\pi) = n - c_{odd}(\Gamma(\pi))$.

Proof. If π reduces to a γ-permutation, then $\Gamma(\pi)$ can be schematically drawn as follows (labels and arrows are omitted for clarity):



In this representation, ∘ stands for a fixed point, and • stands for an element that belongs to a cycle of length at least 2 in $\Gamma(\pi)$. Every dotted rectangle corresponds to an interval of π that consists only of fixed points, and has nonzero length (except possibly for the leftmost and rightmost intervals). Therefore, reducing π comes down to removing fixed points, and each such removal decreases both n and $c_{odd}(\Gamma(\pi))$ by 1. The quantity $n - c_{odd}(\Gamma(\pi))$ is thus unaffected by reduction, and Theorem 3.2 completes the proof. For the second category discussed in the thesis, note that $\sigma^\circ \pm 1$ fixes all even elements and therefore falls into the category discussed above. The proof then follows from Lemma 3.2. □

3.5 A new upper bound

We now show that the right-hand side of (3.10) is an upper bound on the transposition distance.

Theorem 3.11. *For all π in S_n:*

$$td(\pi) \leq n - c_{odd}(\Gamma(\pi)). \tag{3.12}$$

Proof. We use the sorting strategy described in Lemma 3.6: each cycle C in $\Gamma(\pi)$ is sorted individually using an optimal sequence of exchanges, which is in turn converted into a corresponding sequence of transpositions of length at most $|C| - (|C| \pmod 2)$ (Propositions 3.5 and 3.7). As in the proof of Proposition 3.8, the distances of the cycles in $\Gamma(\pi)$ sum up to $n - c_{odd}(\Gamma(\pi))$, which completes the proof. □

3.6 Tests and heuristic improvements of our upper bound

Table 3.2 shows the number of cases where the value of (3.12) is at most the value of each of the previously known upper bounds given in Section 3.1.5. A first heuristic improvement can be obtained through torism.

Theorem 3.12. *For all π in S_n:*

$$td(\pi) \leq n - \max_{\sigma \in \pi_\circ^\circ} c_{odd}(\Gamma(\sigma)). \tag{3.13}$$

Proof. Straightforward from Theorem 3.11 and Lemma 3.2. □

Experiments show (Table 3.3) that (3.13) is a substantial improvement over (3.12), but it is hard to express or evaluate this improvement because the evolution of the Γ-graph under the toric equivalence relation does not seem easy to predict, whereas that of the cycle graph is well known (Lemma 3.3). By the way, note that the other upper bounds cannot benefit from this improvement, since neither the cycle graph structure nor the number of breakpoints will be affected. A second heuristic improvement of (3.12) can be obtained through reduction.

n	$n!$	$\lvert(3.12)\leq(3.1)\rvert$	$\lvert(3.12)\leq(3.2)\rvert$	$\lvert(3.12)\leq(3.3)\rvert$	$\lvert(3.12)\leq(3.4)\rvert$	$\lvert(3.12)\leq(3.8)\rvert$
1	1	1	1	1	1	1
2	2	2	1	1	1	1
3	6	6	2	1	6	5
4	24	19	8	8	15	15
5	120	101	45	24	31	37
6	720	529	304	49	495	179
7	5 040	3 837	2 055	722	1 611	1 035
8	40 320	28 354	17 879	3 094	4 355	4 257
9	362 880	257 844	104 392	60 871	10 243	18 733
10	3 628 800	2 469 217	430 164	361 659	485 154	124 110

n	|simple permutations|	|2-permutations|	|3-permutations|	$\lvert(3.12)\leq(3.5)\rvert$	$\lvert(3.12)\leq(3.6)\rvert$	$\lvert(3.12)\leq(3.7)\rvert$
1	1	-	-	1	-	-
2	2	-	1	1	-	1
3	6	1	-	6	1	-
4	16	-	-	11	-	-
5	48	-	12	19	-	12
6	204	-	-	32	-	-
7	876	21	-	369	9	-
8	3 636	-	464	749	-	10
9	18 756	-	-	1 433	-	-
10	105 480	-	-	2 678	-	-
11	561 672	1 485	38 720	46 342	87	862

Table 3.2: Comparison of upper bound (3.12) with previous results.

n	$n!$	$\lvert(3.13)\leq(3.1)\rvert$	$\lvert(3.13)\leq(3.2)\rvert$	$\lvert(3.13)\leq(3.3)\rvert$	$\lvert(3.13)\leq(3.4)\rvert$	$\lvert(3.13)\leq(3.8)\rvert$
1	1	1	1	1	1	1
2	2	2	1	1	1	1
3	6	6	2	1	6	5
4	24	24	11	9	21	21
5	120	112	60	36	54	54
6	720	671	451	73	703	231
7	5 040	4 654	3 318	1 336	3 574	1 934
8	40 320	37 209	27 486	5 957	9 864	7 870
9	362 880	336 744	259 195	132 801	21 610	32 001
10	3 628 800	3 280 815	1 244 002	931 584	1 376 134	246 124
n	$\lvert$simple permutations$\rvert$	$\lvert$2-permutations$\rvert$	$\lvert$3-permutations$\rvert$	$\lvert(3.13)\leq(3.5)\rvert$	$\lvert(3.13)\leq(3.6)\rvert$	$\lvert(3.13)\leq(3.7)\rvert$
1	1	-	-	1	-	-
2	2	-	1	2	-	1
3	6	1	-	6	1	-
4	16	-	-	16	-	-
5	48	-	12	48	-	12
6	204	-	-	190	-	-
7	876	21	-	692	21	-
8	3 636	-	464	1 530	-	20
9	18 756	-	-	2 781	-	-
10	105 480	-	-	4 896	-	-
11	561 672	1 485	38 720	112 364	281	2454

Table 3.3: Comparison of upper bound (3.13) with previous results.

Theorem 3.13. *For all $\pi \neq \iota$ in S_n, let $gl(\pi)$ denote its reduced version in S_m, where $m \leq n$; then*

$$td(\pi) \leq m - \max_{\sigma \in (gl(\pi))_\circ^\circ} c_{odd}(\Gamma(\sigma)). \qquad (3.14)$$

All other bounds can take advantage of this reduction as well, except for (3.1), (3.2) and (3.3). This time, we do not compare (3.14) with other bounds; instead, for $1 \leq i \leq 10$, we generate all permutations with their distance, and check how (3.14) overestimates their distance. Table 3.4 shows the results.

n	$n!$	$\Delta = 0$	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$	$\Delta = 6$
1	1	1	0	0	0	0	0	0
2	2	1	1	0	0	0	0	0
3	6	2	4	0	0	0	0	0
4	24	11	11	2	0	0	0	0
5	120	48	51	21	0	0	0	0
6	720	197	401	108	14	0	0	0
7	5 040	1 318	2 460	966	296	0	0	0
8	40 320	8 775	13 875	15 150	2 512	8	0	0
9	362 880	45 415	132 257	145 394	34 702	5 112	0	0
10	3 628 800	231 738	1 208 900	1 124 842	972 323	90 511	420	66

Table 3.4: Number of cases where (3.14) overestimates $td(\pi)$ by Δ.

3.7 Perforations of α-permutations

We now know how to compute the transposition distance of γ-permutations, which fix all even elements; what about other permutations? One possible way to attack this problem is to study how removing fixed points from γ-permutations affects the distance of the resulting permutation. Since that question seems difficult, we restrict our study to the particular case of α-permutations, for which we are able to give an answer to the above question. Building on that result, we obtain a formula for computing the distance of permutations obtained from α-permutations in that way (Corollary 3.2), and succeed in improving (3.12) in many other cases, sometimes even reaching the actual distance.

Note that removing a 1-cycle from the Γ-graph of a γ-permutation π, which will be located in an even position i, can be done by moving $\pi_i + 1$ right after π_i using a transposition, then removing the adjacency obtained in that way and renumbering the other elements appropriately.

Definition 3.27. A *k-perforation* π in S_n of an α-permutation σ in S_{n+k} is a permutation obtained by removing $k \geq 1$ 1-cycles from $\Gamma(\sigma)$ and renumbering the remaining elements.

For instance, a 3-perforation of the α-permutation $\langle 3\ 2\ 5\ 4\ 7\ 6\ 9\ 8\ 11\ 10\ 1\rangle$ is $\langle 3\ \boxed{2}\ 5\ 4\ 7\ \boxed{6}\ 9\ \boxed{8}\ 11\ 10\ 1\rangle = \langle 2\ 4\ 3\ 5\ 6\ 8\ 7\ 1\rangle$. The following result, which will be followed by an example in Figure 3.7 (page 40), describes how the structure of the cycle graph evolves when perforating an α-permutation.

Lemma 3.7. *For every k-perforation π of an α-permutation σ in S_{n+k}:*

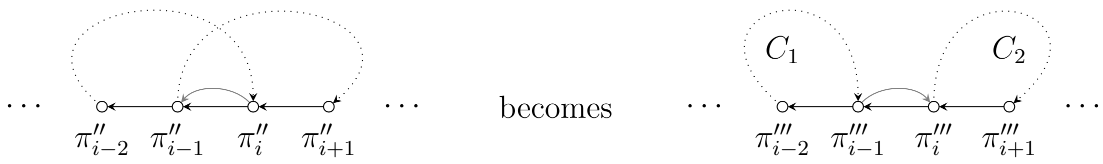
$$c(G(\pi)) = c_{odd}(G(\pi)) = k$$

and $G(\pi)$ contains only noncrossing cycles, not containing each other, except for a large one containing all others.

Proof. Induction on k. The main cycle of $\Gamma(\sigma)$ is again assumed to be increasing, the decreasing case corresponding to σ^{-1} whose cycle graph has the same structure (see Hultman [47]). Therefore, for every even i, we have $\sigma_{i-1} = \sigma_i + 1$. Recall that $n + k$ is odd, by definition of σ.

If $k = 1$, let us remove some fixed element $\sigma_i = i$ (i is therefore even). We first apply the transposition $\tau(i-1, i, i+1)$, which acts on two interleaving cycles of same parity in G (Observation 3.3), and is therefore a 0-transposition (Lemma 3.1) that transforms those two cycles into a 1-cycle and an $(n+k)$-cycle, both odd. Removing the 1-cycle and renumbering all elements as required, we obtain π, and $c(G(\pi)) = c_{odd}(G(\pi)) = 1$.

For the induction, we again remove 1-cycles from $\Gamma(\sigma)$ in two steps, by first applying all our transpositions, then removing k adjacencies. Since the thesis is assumed to hold for $k-1$ perforations, we start with the corresponding $(k-1)$-perforation π', and put back the $k-1$ adjacencies that needed to be deleted, thus obtaining a permutation π'' with $c(G(\pi'')) = c_{odd}(G(\pi'')) = 2(k-1)$. By our induction hypothesis, none of these cycles cross, and one of them contains all others. Let us now select some fixed point in an even position i that we wish to remove, and apply the adequate transposition $\tau(i-1, i, i+1)$ to create an adjacency. The odd alternating cycle to which this element belongs will be split into three cycles: an adjacency (1-cycle) "framed" by two cycles C_1 and C_2 of the same parity.



We need to prove that both C_1 and C_2 are odd, which comes down to showing that one of them is since they have the same parity. By induction, there is an adjacency on the right-hand side of C_2 or on the left-hand side of C_1, and without loss of generality we will assume that we are in the first case. This adjacency, that we denote (π_j, π_{j+1}), contains an even element in an odd position, namely, π_j; since the leftmost element of C_2 occupies an even position, the number $j - i$ of black arcs between position i and position j, which all belong to C_2, is odd. Therefore, the three new cycles are odd, and we get the permutation $\pi''' = \pi'' \circ \tau$ with $c(G(\pi''')) = c_{odd}(G(\pi''')) = 2k$. We now remove k 1-cycles from $G(\pi''')$, and the proof follows. □

Figure 3.7 illustrates the claim of Lemma 3.7. This result leads to a formula for computing the distance of perforations of α-permutations.

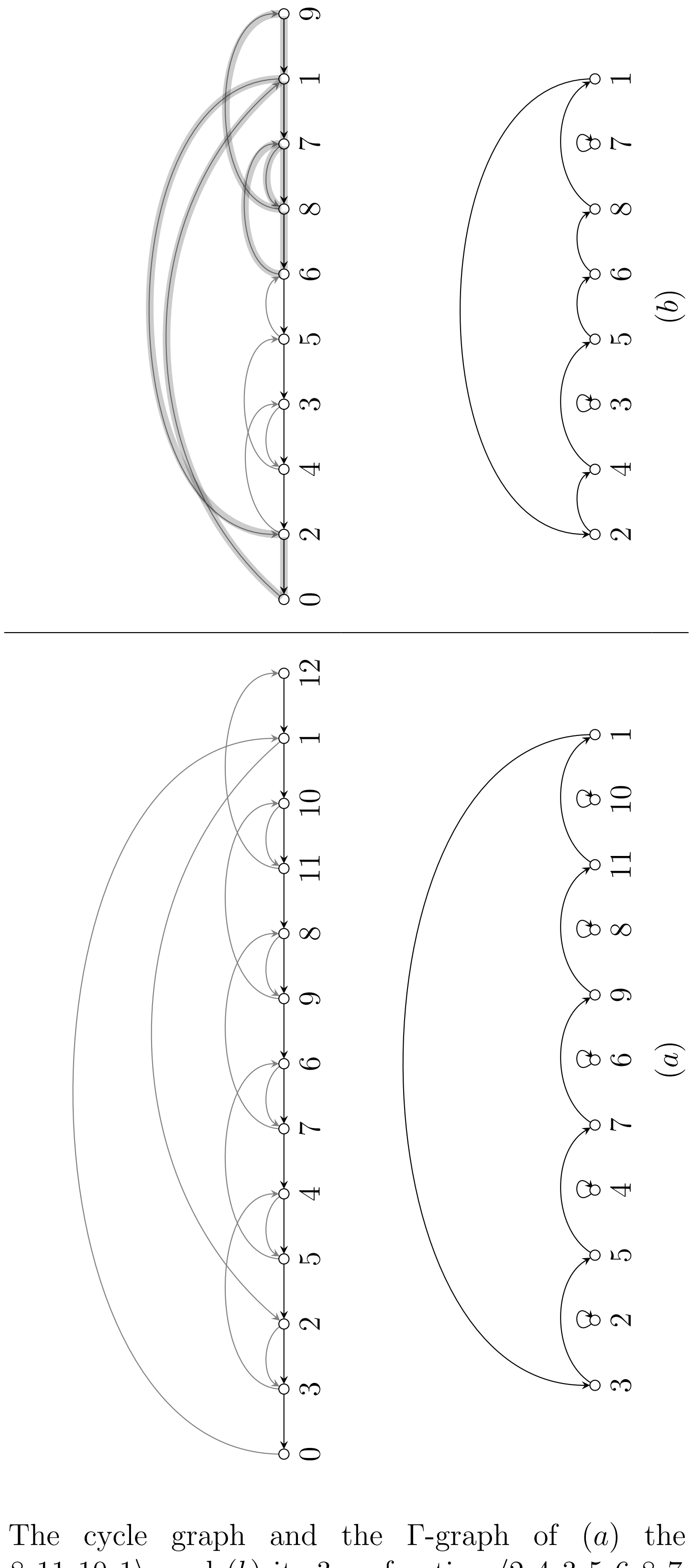


Figure 3.7: The cycle graph and the Γ-graph of (a) the α-permutation $\langle 3\ 2\ 5\ 4\ 7\ 6\ 9\ 8\ 11\ 10\ 1\rangle$, and ($b$) its 3-perforation $\langle 2\ 4\ 3\ 5\ 6\ 8\ 7\ 1\rangle$. The "wrapping" cycle in the cycle graph of the perforation has been highlighted, and contains a 3-cycle and a 1-cycle.

Corollary 3.2. *Let π in S_n be a k-perforation of an α-permutation; then*

$$td(\pi) = n - c_{odd}(\Gamma(\pi)) - k + (|C| \bmod 2)\,,$$

where $|C| = \frac{n+k+1}{2}$ is the number of elements in the main cycle C of π.

Proof. Again, assume without loss of generality that the main cycle is increasing. A lower bound of $|C| - k$ is given by Lemma 3.7 and Theorem 3.1. On the other hand, the elements of the main cycle of σ (the α-permutation in S_{n+k} from which π is obtained), without 1, form a longest increasing subsequence of length $\frac{n-1}{2} = |C| - 1$. Moreover, perforating an α-permutation does not affect this sequence, so Observation 3.1 yields an upper bound of $n - |C| + 1$, which equals $|C| - k$ since $|C| = \frac{n+k+1}{2}$. Therefore:

$$\begin{aligned} td(\pi) &= |C| - k \\ &= |C| - k - (|C| \bmod 2) + (|C| \bmod 2) \\ &= td(\sigma) - k + (|C| \bmod 2) \\ &= n + k - c_{odd}(\Gamma(\sigma)) - k + (|C| \bmod 2) \\ &= n - c_{odd}(\Gamma(\sigma)) + (|C| \bmod 2) \\ &= n - c_{odd}(\Gamma(\pi)) - k + (|C| \bmod 2)\,. \end{aligned}$$

□

The next logical move, as in our analysis of γ-permutations, would be to consider perforations of β-permutations. However, counter-examples have been found that prevent us from proving an equivalent of Lemma 3.7 for such permutations; for instance, consider the β-permutation $\langle 7\ 2\ 13\ 4\ 3\ 6\ 5\ 8\ 15\ 10\ 9\ 12\ 11\ 14\ 1\rangle$. Then the cycle graph of the 4-perforation $\langle 7\ 2\ 13\ \boxed{4}\ 3\ \boxed{6}\ 5\ \boxed{8}\ 15\ 10\ 9\ 12\ 11\ \boxed{14}\ 1\rangle = \langle 5\ 2\ 10\ 3\ 4\ 11\ 7\ 6\ 9\ 8\ 1\rangle$ has only two cycles, both odd.

We can nevertheless still prove results on permutations whose Γ-graph contains noncrossing cycles only: fortunately, the exact distance of some subcases in that family can be computed exactly; if not, we are nonetheless still able to improve (3.12). Before tackling this general problem in the next section, we conclude the current one with the particular case where all noncrossing long cycles are perforations of α-permutations, starting with the case shown in Figure 3.8, where we do not allow containment of long cycles. In such a configuration, the 1-cycles between every pair of long cycles are referred to as the *separating* 1-cycles or, more concisely, the *separators*.

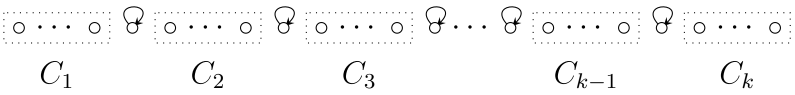


Figure 3.8: A Γ-graph formed by sub-permutations separated by 1-cycles. Those sub-permutations contain one long cycle and a collection of fixed points, and no two long cycles in the whole permutation cross, interleave or contain each other.

Proposition 3.9. *Let π in S_n be a permutation with $\Gamma(\pi)$ of the form shown in Figure 3.8, where C_i ($1 \leq i \leq k$) is a k_i-perforation of an α-permutation (up to appropriate renumbering); then*

$$td(\pi) \quad = \quad n - c_{odd}(\Gamma(\pi)) - K + \sum_{i=1}^{k} (|C_i| \bmod 2),$$

where $K = \sum_{i=1}^{k} k_i$ and $|C_i|$ is the number of elements in the main cycle of each perforation.

Proof. Lemma 3.7 and Theorem 3.1 yield

$$td(\pi) \geq \frac{n+1-\sum_{i=1}^{k} k_i}{2} = \frac{n+1-K}{2}.$$

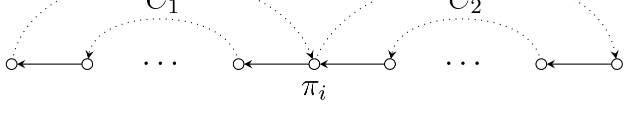
We have $n = k-1+\sum_{i=1}^{k} n_i$, where n_i is the number of elements of each perforation, and Lemma 3.6 yields

$$\begin{aligned} td(\pi) \leq \sum_{i=1}^{k} td(C_i) &= \sum_{i=1}^{k} |C_i| - k_i \\ &= \sum_{i=1}^{k} \frac{n_i + k_i + 1 - 2k_i}{2} \\ &= \frac{1}{2} \sum_{i=1}^{k} (n_i + 1 - k_i) \\ &= \frac{n+1-K}{2}. \end{aligned}$$

The expression given in the thesis is obtained by replacing $td(C_i)$ with the expression provided by Corollary 3.2. □

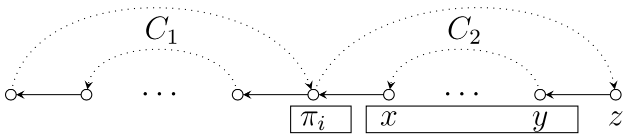
We now show that removing any subset of the separators in the case we just examined does not affect the distance. For any transposition τ and any permutation π, let $\Delta c_{odd}(\tau, G(\pi)) = c_{odd}(G(\pi \circ \tau)) - c_{odd}(G(\pi))$. The following lemma will be useful.

Lemma 3.8. *Let $\tau = \tau(i, i+1, (\pi^{-1})_{\pi_i+1})$, and let C_1, C_2 be two noncrossing cycles in $G(\pi)$ which share vertex π_i as shown below:*

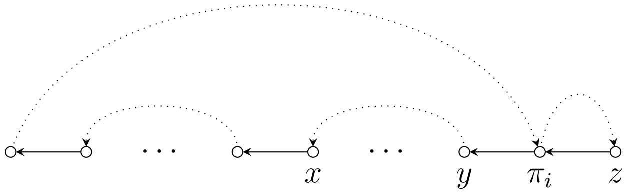


Then $\Delta c_{odd}(\tau, G(\pi)) = 2$ if both C_1 and C_2 are even, and 0 otherwise.

Proof. The transposition $\tau(i, i+1, (\pi^{-1})_{\pi_i+1})$ acts on two cycles, and is therefore a 0-transposition (Lemma 3.1) transforming C_1 and C_2 into a 1-cycle and a $(|C_1| + |C_2| - 1)$-cycle as shown below:



becomes



The proof follows from parity arguments. □

Corollary 3.3. *Let π be a permutation that satisfies the conditions of Proposition 3.9; then removing j ($1 \leq j \leq k-1$) separators from $\Gamma(\pi)$ yields a permutation with the same distance.*

Proof. By Lemma 3.7, each C_i in $\Gamma(\pi)$ corresponds to a collection of alternating cycles in $G(\pi)$ wrapped in a large one, and all of them are odd. Every pair of consecutive "wrapping cycles" in $G(\pi)$ shares a vertex, which is the 1-cycle separating the corresponding long cycles in $\Gamma(\pi)$. By Lemma 3.8, deleting that separating cycle does not change the bounds obtained in Proposition 3.9, and the proof follows. □

Similar arguments can be used to handle the case of cycles in the Γ-graph that contain other ones, so we have the following result.

Theorem 3.14. *For every π in S_n that reduces to a concatenation of 1-cycles and of k perforations of α-permutations (up to appropriate renumbering):*

$$td(\pi) \quad = \quad n - c_{odd}(\Gamma(\pi)) - K + \sum_{i=1}^{k} \left(|C_i| \bmod 2\right),$$

where K is the number of arcs of length 1 in $\Gamma(\pi)$.

Proof. Perforating an α-permutation creates an arc of length 1 in $\Gamma(\pi)$. The formula follows from Proposition 3.9 and previous observations. □

It is less clear how exactly a perforation would be defined in the case of crossing cycles. Even less clear is the evolution of cycles in the cycle graph when deleting fixed points in this situation: this seems to depend both on how cycles cross and on their monotonicity. We can however prove some further results on permutations whose Γ-graph has no crossing cycles, which we do in the next section.

3.8 Noncrossing cycles in the Γ-graph

We consider permutations with a Γ-graph of the form shown in Figure 3.8 (page 41), and have a look at what happens in the cycle graph and in the Γ-graph when deleting separators (this is more general than the case examined in Proposition 3.9, since we allow each C_i to be something other than a perforation of an α-permutation). Depending on the parity of each long cycle, the deletion of separators can have various effects.

Proposition 3.10. *Let π in S_n be a permutation with $\Gamma(\pi)$ of the form shown in Figure 3.8, where C_i $(1 \leq i \leq k)$ is one of the following:*

- *an α-permutation with an odd main cycle;*
- *a β-permutation with an odd main cycle;*
- *a perforation of an α-permutation.*

Then deleting j separators $(1 \leq j \leq k-1)$ transforms π into a permutation with the same distance.

Proof. By Propositions 3.4 and 3.8, we have $td(\pi) = \frac{n+1-c_{odd}(G(\pi))}{2}$. Each pair (C_i, C_{i+1}) yields a pair of alternating cycles (Observation 3.3 and Lemma 3.7) that share the separator as described in Lemma 3.8. This Lemma also implies that deleting the separator does not change the lower bound of Theorem 3.1, which is tight for π, because it will decrease both n and the number of odd alternating cycles by 1. So $td(\pi)$ is a lower bound on the distance of the resulting permutation, and since $td(\pi)$ is also an upper bound on that distance (Lemma 3.6), the proof follows. □

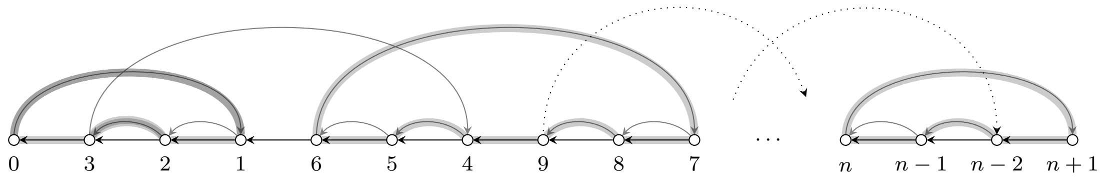
Although we are unable to compute the exact distance when all large cycles are even (and are not perforations of α-permutations), we can still lower (3.12) in that case. In order to express this improved bound formally, we need to introduce the following graph.

Definition 3.28. Given a permutation π with $\Gamma(\pi)$ of the form shown in Figure 3.8, the *contact graph* $H(\pi)$ is the undirected graph whose vertices are the long cycles in $\Gamma(\pi)$ and whose edges are $\{C_i, C_{i+1}\}$ if C_i and C_{i+1} are even and not separated by a 1-cycle in $\Gamma(\pi)$.

The graph $H(\pi)$ uniquely decomposes into p connected components, which we denote $\mathcal{C}_1, \ldots, \mathcal{C}_p$. The following lemma will be useful.

Lemma 3.9. *Let $\xi_k = \langle\underbrace{3\ 2\ 1}_{1}\ \underbrace{6\ 5\ 4}_{2}\ \cdots\ \underbrace{n\ n-1\ n-2}_{k}\rangle$; then $td(\xi_k) \leq \left\lceil \frac{3k}{2} \right\rceil = \left\lceil \frac{n}{2} \right\rceil$.*

Proof. As shown below, the cycle graph of ξ_k contains only two cycles of length two and $k+1$ cycles of length three, a few of which are highlighted:



ξ_k is therefore a simple permutation, and the proof follows from Theorem 3.7. □

One way to sort ξ_k is to handle 2-cycles of $\Gamma(\xi_k)$ pairwise, i.e. partition ξ_k into $\left\lfloor \frac{k}{2} \right\rfloor$ sub-permutations of the form of ξ_2. Those can each be sorted optimally using three transpositions (indeed: $\langle\ \boxed{3\ 2}\,\boxed{1\ 6}\ 5\ 4\rangle \to \langle 1\ \boxed{6\ 3}\,\boxed{2\ 5}\ 4\rangle \to \langle 1\ 2\ \boxed{5\ 6}\,\boxed{3\ 4}\ \rangle \to \iota$), and possibly, one last sub-permutation of the form of ξ_1 will require two transpositions. Note that ξ_k generalises an example given by Christie [23] that shows the nonoptimality of his improved lower bound on the transposition distance (meaning that even though it gives a larger value than the lower bound of Theorem 3.1, it still underestimates the true distance). Branch-and-bound seems however to indicate that the upper bound of Lemma 3.9 is the actual distance of ξ_k.

Proposition 3.11. *Let π be a γ-permutation with $\Gamma(\pi)$ of the form shown in Figure 3.8, where C_i $(1 \leq i \leq k)$ is either an α-permutation or a β-permutation with an even main cycle; then deleting j separators $(1 \leq j \leq k-1)$ transforms π into a permutation σ such that*

$$td(\sigma) \leq td(\pi) - 2k + \sum_{i=1}^{p} \left\lceil \frac{3|\mathcal{C}_i|}{2} \right\rceil,$$

where $\mathcal{C}_i$ $(1 \leq i \leq p)$ is a connected component of $H(\sigma)$.

Proof. Instead of removing separators directly, we first apply some transpositions to π. Each sub-α-permutation and each sub-β-permutation can be sorted "incompletely" using the process described in the proof of Proposition 3.7, i.e. keep applying transpositions until the resulting sub-permutation reduces to $\langle 3\ 2\ 1\rangle$. By reduction, the resulting permutation has a Γ-graph of the form shown in Figure 3.8, where each C'_i is now of the form of ξ_1. Let us now remove a subset of j separators $(1 \leq j \leq k-1)$ from that permutation; this will diminish the number of components in its contact graph, thus creating sub-permutations of the form of ξ_k. The following upper bound is obtained from Lemmas 3.6 and 3.9:

$$\begin{aligned} td(\sigma) \leq \sum_{i=1}^{p} td(\mathcal{C}_i) \quad &\leq \quad \sum_{i=1}^{p} \left(\sum_{C_j \in \mathcal{C}_i} (td(C_j) - 2) + td(\xi_{|\mathcal{C}_i|}) \right) \\ &= \quad \sum_{i=1}^{p} \sum_{C_j \in \mathcal{C}_i} (td(C_j) - 2) + \sum_{i=1}^{p} td(\xi_{|\mathcal{C}_i|}) \\ &\leq \quad td(\pi) - 2k + \sum_{i=1}^{p} \left\lceil \frac{3|\mathcal{C}_i|}{2} \right\rceil. \end{aligned}$$

□

An easy particular case of this proposition is when all separators are deleted; in that case, $td(\sigma) \leq td(\pi) - \left\lceil \frac{k}{2} \right\rceil$. There remains one case to deal with, which encompasses both previous Propositions.

Proposition 3.12. *Let π be a γ-permutation with $\Gamma(\pi)$ of the form shown in Figure 3.8, where C_i $(1 \leq i \leq k)$ is one of the following:*

- *an α-permutation or a β-permutation with an even or an odd main cycle;*
- *a perforation of an α-permutation.*

Then deleting j separators $(1 \leq j \leq k-1)$ transforms π into a permutation σ such that

$$td(\sigma) \leq td(\pi) - 2k + \sum_{i=1}^{p} \left\lceil \frac{3|\mathcal{C}_i|}{2} \right\rceil,$$

where $\mathcal{C}_i$ $(1 \leq i \leq p)$ is a connected component of $H(\sigma)$.

Proof. As hinted by Lemma 3.8 and confirmed by subsequent results, the only case in which deleting a separator affects the distance of the resulting permutation is when that deletion occurs between two even cycles. This means that Proposition 3.11 naturally generalises to the case where some cycles are allowed to be odd, because deleting separators adjacent to at least one long odd cycle will not modify the distance of the resulting permutation. By the same arguments as those used in that Proposition's proof, we obtain the same upper bound on the distance of the resulting permutation, and α-permutations, β-permutations as well as perforations of the former kind can be handled individually in σ as was already done in π. □

We conclude with the case where we allow containment of cycles and perforations of α-permutations.

Theorem 3.15. *For all π in S_n with $\Gamma(\pi)$ containing only noncrossing monotonic cycles and 1-cycles, we have*

$$td(\pi) \quad = \quad n - c_{odd}(\Gamma(\pi)) - K + \sum_{i=1}^{k} (|C_i| \bmod 2),$$

where C_i $(1 \leq i \leq k)$ are the long cycles in $\Gamma(\pi)$ and K is the number of arcs of length 1.

Proof. Assume that every pair of consecutive long cycles in $\Gamma(\pi)$ is separated by a 1-cycle; since each long cycle is odd or spans a perforation of an α-permutation, the alternating cycles in $G(\pi)$ that correspond to this sub-permutation are all odd (Proposition 3.4 and Lemma 3.7). Therefore, removing any subset of the separators cannot affect the distance (Lemma 3.8), so the strategy of Lemma 3.6 remains optimal and the proof follows from Theorem 3.14. □

3.9 Equivalence of the cycle graph and of the breakpoint graph

A reader familiar with the problem of sorting by reversals will have noted that the cycle graph is very similar to a structure known as the *breakpoint graph*. In this section, we prove that those two graphs are indeed equivalent, at least for unsigned permutations. This equivalence allows the use of either structure for the study of sorting by transpositions (other researchers, e.g. Elias and Hartman [30], have already been using the breakpoint graph instead of the cycle graph, but without proving the equivalence between both graphs), and may allow the extension of results obtained in this chapter to other problems.

A *signed permutation* is a permutation whose elements can be either positive or negative. Denote $S_n^{\pm}$ the group of permutations of $\{\pm 1, \pm 2, \ldots, \pm n\}$. It is not mandatory for a signed permutation to have negative elements, so $S_n \subset S_n^{\pm}$ (since each permutation in S_n can be viewed as a signed permutation without negative elements in $S_n^{\pm}$). The following graph was introduced by Bafna and Pevzner [7] in the context of sorting permutations by reversals.

Definition 3.29. Given a signed permutation π in $S_n^{\pm}$, transform it into an unsigned permutation π' in S_{2n} by replacing π_i with the sequence $(2\pi_i - 1, 2\pi_i)$ if $\pi_i > 0$, or $(2|\pi_i|, 2|\pi_i| - 1)$ if $\pi_i < 0$, for $1 \leq i \leq n$. The *breakpoint graph* of π' is the undirected bicoloured graph $BG(\pi')$ with ordered vertex set $(\pi'_0 = 0, \pi'_1, \pi'_2, \ldots, \pi'_{2n}, \pi'_{2n+1} = 2n+1)$ and whose edge set consists of:

- black edges $\{\pi'_{2i}, \pi'_{2i+1}\}$ for $0 \leq i \leq n$;
- grey edges $\{\pi'_{2i}, \pi'_{2i} + 1\}$ for $0 \leq i \leq n$.

We show that for every signed permutation π with no negative element, the cycle graph $G(\pi)$ is equivalent to the breakpoint graph $BG(\pi')$. By equivalent, we mean that every alternating cycle in $G(\pi)$ is an alternating cycle in $BG(\pi')$, and that the "topological" relations between the cycles are the same; for instance, if two cycles cross in either graph, then they also cross in the other one.

Theorem 3.16. *For all π in S_n, we have $G(\pi) \equiv BG(\pi')$.*

Proof. We show that either graph can be constructed by transforming the other one without affecting its features. Intuitively, transforming $G(\pi)$ into $BG(\pi')$ is done by spacing black arcs in $G(\pi)$ (i.e. splitting each vertex into two nonadjacent vertices) and removing the orientation; conversely, transforming $BG(\pi')$ into $G(\pi)$ is done by orienting edges in $BG(\pi')$, then merging every consecutive pair of vertices that are not connected by a black arc.

1. starting with $G(\pi)$: split each vertex π_i ($1 \leq i \leq n$) into two unconnected vertices $(\pi_i)_l$, $(\pi_i)_r$ (one to the left and one to the right), and rename π_0 (resp. π_{n+1}) into $(\pi_0)_r$ (resp. $(\pi_{n+1})_l$). Black arc (π_i, π_{i-1}) is mapped onto a new black arc $((\pi_i)_l, (\pi_{i-1})_r)$, as shown in Figure 3.9. Similarly, grey arc $(\pi_i, \pi_i + 1)$ is mapped onto a new grey arc $((\pi_i)_r, (\pi_i + 1)_l)$, as shown in Figure 3.10. This

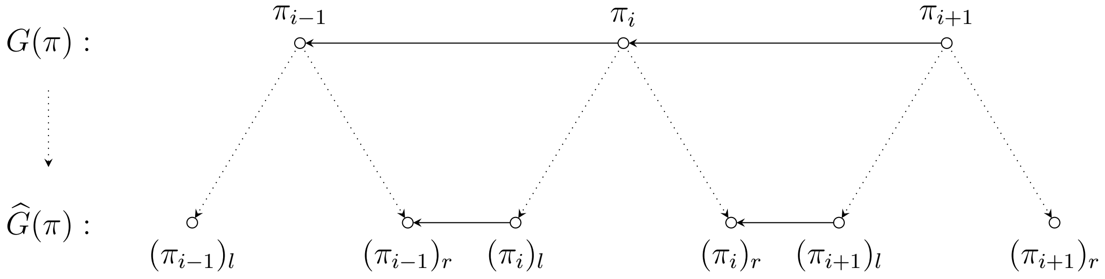


Figure 3.9: Mapping of the black arcs in the transformation of $G(\pi)$ into $BG(\pi')$; here, $\widehat{G}(\pi)$ is a graph that will be isomorphic to $BG(\pi')$ once the orientation of arcs is removed.

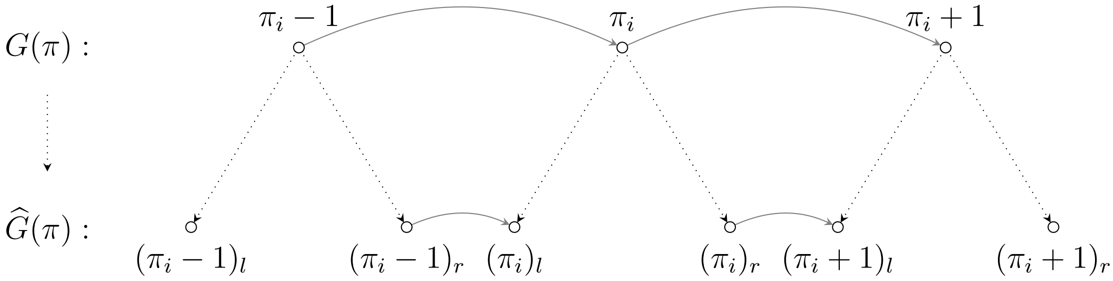


Figure 3.10: Mapping of the grey arcs in the transformation of $G(\pi)$ into $BG(\pi')$.

transformation yields $\widehat{G}(\pi)$, a graph that will be isomorphic to $BG(\pi')$ once the orientation of arcs is removed; indeed, let us then rename $(\pi_i)_l$ (resp. $(\pi_i)_r$) into $2\pi_i - 1$ (resp. $2\pi_i$) and remove the orientation of arcs: we obtain $BG(\pi')$, since:

(a) each black arc (π_i, π_{i-1}) is mapped onto a black edge $\{(\pi_i)_l, (\pi_{i-1})_r\} = \{2\pi_i - 1, 2\pi_{i-1}\}$;

(b) each grey arc $(\pi_i, \pi_i + 1)$ is mapped onto a grey edge $\{(\pi_i)_r, (\pi_i + 1)_l\} = \{2\pi_i, 2\pi_i + 1\}$.

2. starting with $BG(\pi')$: since π' comes from some permutation π with no negative element, for all $1 \leq i \leq n$, we have $\pi'_{2i} = 2\pi_i$ and $\pi'_{2i-1} = 2\pi_i - 1$. This implies that alternating cycles in $BG(\pi')$ can be followed starting from the leftmost vertex of a black edge, then following a grey edge that will take us to the rightmost vertex of the next black edge. Therefore, adding an orientation to all edges that corresponds to this course will result in a collection of directed alternating cycles that can be followed using the direction of the arrows, and this orientation is obtained by transforming grey edge $\{\pi'_{2i}, \pi'_{2i} + 1\}$ into $(\pi'_{2i}, \pi'_{2i} + 1)$, and black edge $\{\pi'_{2i}, \pi'_{2i+1}\}$ into (π'_{2i+1}, π'_{2i}).

   Next, for $1 \leq i \leq n$, merge vertices π'_{2i-1} and π'_{2i} into vertex π'_{2i}, and rename vertex π'_{2n+1} into π'_{2n+2}; black arc (π'_{2i+1}, π'_{2i}) is mapped onto a new black arc (π'_{2i+2}, π'_{2i}).

   Finally, replace π'_{2i} with π_i, for $0 \leq i \leq n + 1$. This results in $G(\pi)$, since:

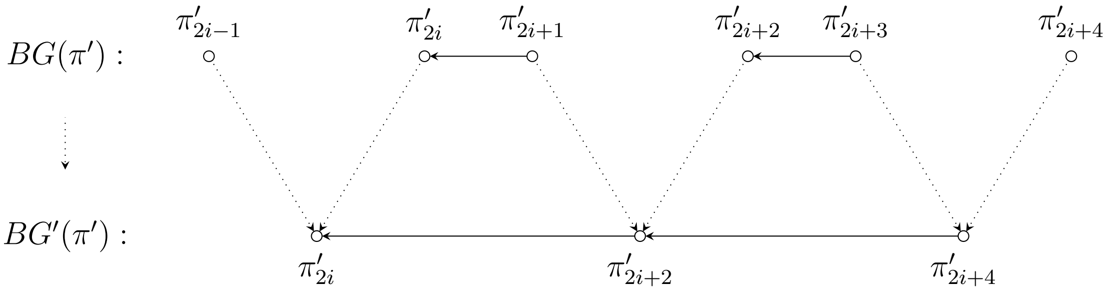


Figure 3.11: Mapping of the black edges in the transformation of $BG(\pi')$ into $G(\pi)$; here, $BG'(\pi')$ is a graph isomorphic to $G(\pi)$.

(a) each black edge $\{\pi'_{2i}, \pi'_{2i+1}\}$ is mapped onto a black arc $(\pi'_{2(i+1)}, \pi'_{2i}) = (\pi_{i+1}, \pi_i)$;

(b) each grey edge $\{\pi'_{2i}, \pi'_{2i} + 1\}$ is mapped onto a grey arc $(\pi_i, \pi_i + 1)$.

□

As in the case of the cycle graph, the *length* of a cycle in a breakpoint graph is the number of black edges it contains.

Definition 3.30. [44] A permutation π in $S_n^{\pm}$ is *simple* if $BG(\pi')$ does not contain a cycle of length greater than three.

Definition 3.31. [44] A permutation π in $S_n^{\pm}$ is a 2-*permutation* (resp. 3-*permutation*) if all cycles in $BG(\pi')$ are of length 2 (resp. 3).

Corollary 3.4. *For every π in S_n, Definition 3.18 (resp. Definition 3.19) and Definition 3.30 (resp. Definition 3.31) are equivalent.*

Proof. Straightforward from Theorem 3.16. □

Chapter 4

Hultman numbers

As we have seen in Chapter 3, the toric equivalence relation has proved useful for sorting permutations by transpositions and computing, or at least approximating the transposition distance. Hultman [47] enumerated the corresponding equivalence classes (sequence A002619 of The On-line Encyclopedia of Integer sequences [69] – hereafter abbreviated to OEIS). He also noticed that the structure of the cycle graph (Definition 3.3 page 20) is preserved under this equivalence relation (Lemma 3.3 page 24), whereas the classical disjoint cycle decomposition (page 13) is in general not preserved. This led him to propose an analogue of the Stirling number of the first kind based on the cycle graph, and to ask for a determination of the resulting number, which we will refer to as the "Hultman number". Hultman was not able to characterise this number, and stated this question as an open problem in his work. Our main contributions in this chapter are the following:

- we present a bijection between the set of permutations of n elements whose cycle graph contains k cycles, on the one hand, and, on the other hand, the set of factorisations of a given $(n+1)$-cycle into the product of an $(n+1)$-cycle and a permutation whose disjoint cycle decomposition contains k cycles (Theorem 4.1);

- we indicate how to construct a similar bijection for permutations with a fixed conjugacy class (in the sense of the cycle graph), based on arguments used for establishing the previous bijection (Theorem 4.2);

- we deduce a general formula from these relations and the work of Goupil and Schaeffer [42] for computing Hultman numbers (Theorem 4.4) and the number of permutations with a fixed conjugacy class, in the sense of the cycle graph (Theorem 4.5);

- we provide simpler formulae for enumerating a few particular classes of permutations (Section 4.4).

Most results presented in this chapter have been published in [29], a joint work with Jean-Paul Doignon.

4.1 Notation and definitions

4.1.1 Stirling numbers and the disjoint cycle decomposition

We will again rely on the Γ-graph of a permutation (Definition 3.21 page 28) to describe its disjoint cycle structure (page 13). We also work with $\mathrm{Sym}(\{0, 1, 2, \ldots, n\})$, the symmetric group on $\{0, 1, 2, \ldots, n\}$, abbreviated to $S(1 + n)$.

Definition 4.1. The *Stirling number of the first kind* $\mathcal{S}(n, k)$ counts the number of permutations in S_n whose disjoint cycle decomposition consists of k cycles:

$$\mathcal{S}(n, k) = |\{\pi \in S_n \mid c(\Gamma(\pi)) = k\}| .$$

As an example, consider S_3, which has six elements; then $\mathcal{S}(3, 2) = 3$, because the only permutations in S_3 that decompose into exactly two cycles are

$$\begin{aligned} \langle 1\ 3\ 2\rangle &= (1)(2, 3), \\ \langle 2\ 1\ 3\rangle &= (1, 2)(3), \text{and} \\ \langle 3\ 2\ 1\rangle &= (1, 3)(2). \end{aligned}$$

A more general counting situation occurs when the conjugacy class of the relevant permutations is completely specified.

Definition 4.2. A *partition* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l)$ is a finite sequence of integers called *parts* such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_l > 0$. Its *length* is l, and we write $\lambda \vdash n$ if $\sum_{i=1}^{l} \lambda_i = n$, in which case we say that λ is a *partition of* n. A permutation π in S_n is of *class* λ if λ is obtained by writing in nonincreasing order the various lengths of the disjoint cycles in $\Gamma(\pi)$; we then set $\lambda = C(\pi)$.

For instance, all permutations in the example preceding Definition 4.2 have conjugacy class $(2, 1)$. The number of permutations in S_n of a given class λ is easily derived (see for instance Proposition 4.1 in Stanley's book [71]): let α_i denote the number of λ_j's that are equal to i (for $1 \leq i \leq n$), and set $z_\lambda = \prod_i \alpha_i!\, i^{\alpha_i}$; then the number of permutations of class λ equals $n! \,/\, z_\lambda$.

4.1.2 Hultman numbers and the cycle graph

Recall the cycle graph of a permutation (Definition 3.3 page 20); in the present chapter, we slightly modify its structure, by identifying vertices 0 and $n + 1$, which transforms $G(\pi)$ into a *circular cycle graph* that we denote $G'(\pi)$. This does not alter the decomposition into alternating cycles, as illustrated by Figure 4.1, so referring to cycles of $G(\pi)$ or of $G'(\pi)$ is equivalent. Our motivation for considering this circular version of the cycle graph, which will become clearer in Section 4.2, is the following: we want to build a new permutation whose disjoint cycle decomposition will correspond to the decomposition of $G'(\pi)$ (and hence of $G(\pi)$) into alternating cycles. That permutation will be the product of the cycles formed respectively by the grey and the black arcs, which will also be viewed as permutations.

Hultman [47] proposed an analogue of the Stirling number of the first kind based on the cycle graph.

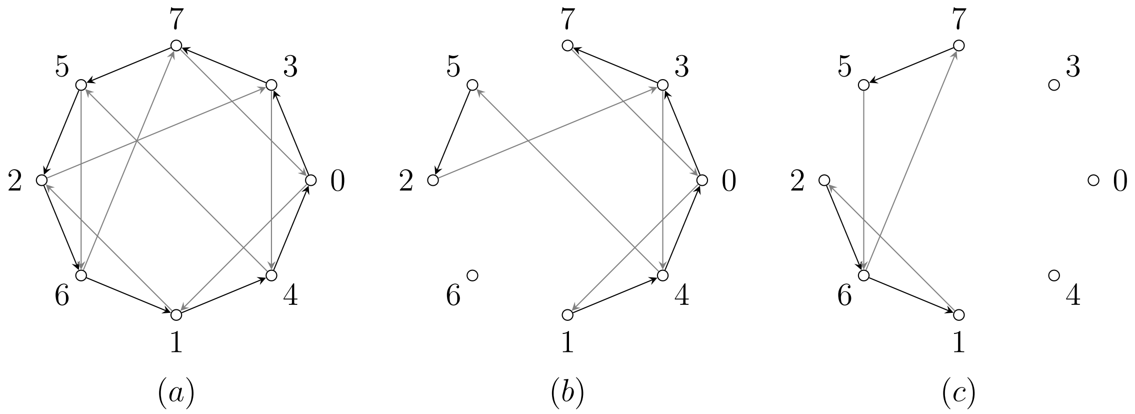


Figure 4.1: (a) The circular cycle graph of $\langle 4\ 1\ 6\ 2\ 5\ 7\ 3\rangle$; $(b), (c)$ its decomposition into two alternating cycles, which matches that shown in Figure 3.1 (page 21).

Definition 4.3. The *Hultman number* $\mathcal{S}_H(n,k)$ counts the number of permutations in S_n whose cycle graph decomposes into k alternating cycles:

$$\mathcal{S}_H(n,k) = |\{\pi \in S_n \mid c(G(\pi)) = c(G'(\pi)) = k\}|.$$

Hultman does not obtain a full characterisation of these numbers, but notes that $\mathcal{S}_H(n, n+1) = 1$ and $\mathcal{S}_H(n, n-1) = \binom{n+2}{4}$. It is also obvious that $\mathcal{S}_H(n,k) = 0$ for all $k \notin [1, n+1]$, since $G'(\pi)$ always contains at least one alternating cycle and at most $n+1$ alternating cycles.

Lemma 4.1. *[47] For all π in S_n, we have $c(G(\pi)) \equiv n+1 \pmod 2$.*

Lemma 4.1 immediately yields the following corollary.

Corollary 4.1. *For all $k \equiv n \pmod 2$, we have $\mathcal{S}_H(n,k) = 0$.*

In the course of characterising Hultman numbers, we will also solve the more general problem of counting permutations in S_n whose cycle graph has a given structure. We define the following notion, which is similar in nature to the concept of conjugacy classes of S_n, but which is based on the cycle graph rather than on the disjoint cycle decomposition of permutations.

Definition 4.4. A permutation π in S_n has *Hultman class* λ if λ is obtained by writing in nonincreasing order the various lengths of the arc-disjoint alternating cycles in $G'(\pi)$. We then set $C_H(\pi) = \lambda$.

For instance, if π is the permutation whose circular cycle graph is shown in Figure 4.1, then we have $C_H(\pi) = (5,3)$.

4.2 The bijection

We now construct a bijection between the set of permutations π in S_n with $c(G'(\pi)) = k$ and the set of factorisations in $S(1+n)$ of a given $(n+1)$-cycle into some $(n+1)$-cycle and some permutation σ with $c(\Gamma(\sigma)) = k$.

Let first π be any permutation in S_n, and consider the arcs of its circular cycle graph $G'(\pi)$. The grey arcs are the pairs $(i, (i+1) \bmod (n+1))$, for $0 \leq i \leq n$, which form the following $(n+1)$-cycle (independent of π) in $S(1+n)$:

$$\alpha = (0,\, 1,\, \dots,\, n).$$

Similarly, the black arcs $(\pi_i, \pi_{(i-1) \bmod (n+1)})$, for $0 \leq i \leq n$, form the following $(n+1)$-cycle in $S(1+n)$:

$$\dot{\pi} = (0, \pi_n,\, \pi_{n-1},\, \dots, \pi_1).$$

Notice that the mapping $S_n \to S(1+n) : \pi \mapsto \dot{\pi}$ is injective. Moreover, its codomain is the set of all $(n+1)$-cycles in $S(1+n)$. By Definition 3.3, the alternating cycles in the graph $G'(\pi)$ build up a permutation $\overline{\pi}$ in $S(1+n)$ that maps i onto j when there is a black arc from $(i+1) \bmod (n+1)$ to $j \bmod (n+1)$ (remember that $(i, (i+1) \bmod (n+1))$ is always a grey arc). Therefore, we have $\overline{\pi} = \dot{\pi} \circ \alpha$, that is

$$\alpha = \dot{\pi}^{-1} \circ \overline{\pi}. \tag{4.1}$$

As an example, Figure 4.2 shows the "isolated" cycles formed respectively by the black arcs and by the grey arcs of the circular cycle graph shown in Figure 4.1; the corresponding alternating cycle decomposition is encoded by the permutation

$$\begin{aligned} \overline{\pi} &= (0,4,2,7,3)(1,6,5) \\ &= (0,3,7,5,2,6,1,4) \circ (0,1,2,3,4,5,6,7), \end{aligned}$$

whose disjoint cycle decomposition corresponds to the alternating cycle decomposition of the (circular) cycle graph of π and can be reconstructed as explained just before Equation (4.1).

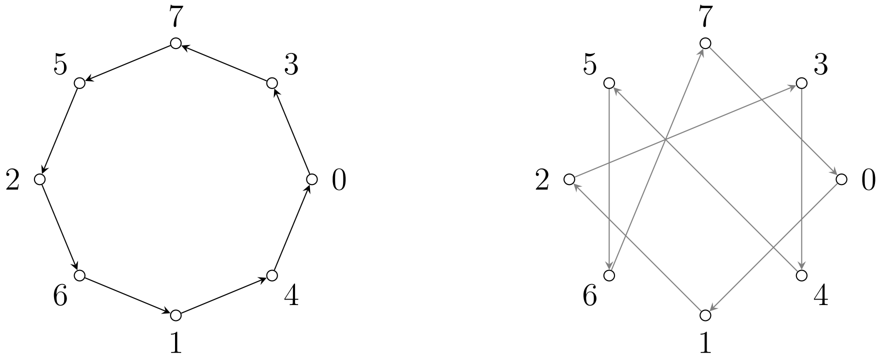


Figure 4.2: The cycles formed by the black arcs and by the grey arcs, respectively, of the graph shown in Figure 4.1.

Theorem 4.1. *Let* $\alpha = (0,\, 1,\, \dots,\, n)$*. The mapping*

$$\begin{aligned} F \;:\; & \{\pi \in S_n \mid c(G'(\pi)) = k\} \to \\ & \{\tau \circ \alpha \mid \tau \in S(1+n), c(\Gamma(\tau)) = 1 \text{ and } c(\Gamma(\tau \circ \alpha)) = k\} \\ :\; & \pi \mapsto \overline{\pi} = \dot{\pi} \circ \alpha \end{aligned}$$

is bijective.

Proof. First note that F is well defined. If $\pi \in S_n$, then $\overline{\pi}$ satisfies $\alpha = \tau \circ \overline{\pi}$, where $\tau = \dot{\pi}^{-1}$ is an $(n+1)$-cycle (cf. Equation (4.1)). Moreover, we have $c(G'(\pi)) = c(\Gamma(\overline{\pi}))$. Since the mapping $S_n \to S(1+n) : \pi \mapsto \dot{\pi}$ is injective, the injectivity of F follows from Equation (4.1).

On the other hand, every $(n+1)$-cycle τ in $S(1+n)$ is of the form $\dot{\pi}$ for some π in S_n; Equation (4.1) then implies $\tau \circ \alpha = \overline{\pi}$, and therefore F is also surjective (remember $c(G'(\pi)) = c(\Gamma(\overline{\pi}))$). □

Given any $(n+1)$-cycle β in $S(1+n)$, define $D(n+1,k)$ as the number of factorisations of β into the product $\rho \circ \sigma$, where ρ is some $(n+1)$-cycle and σ some permutation in $S(1+n)$ with $c(\Gamma(\sigma)) = k$. We get the following result.

Corollary 4.2. *For all n, k in $\mathbb{N}$, we have $\mathcal{S}_H(n,k) = D(n+1,k)$.*

The bijection F in Theorem 4.1 is induced by a much more general bijection, which is established along the same arguments.

Theorem 4.2. *Let λ be a partition of $n+1$. The mapping*

$$\begin{aligned} F \ :\ & \{\pi \in S_n \mid C_H(\pi) = \lambda\} \to \\ & \{\tau \circ \alpha \mid \tau \in S(1+n), c(\Gamma(\tau)) = 1 \text{ and } C(\Gamma(\tau \circ \alpha)) = \lambda\} \\ :\ & \pi \mapsto \overline{\pi} = \dot{\pi} \circ \alpha \end{aligned}$$

is bijective.

We conclude this section by noting that using a very similar approach, although working with another conjugacy class of $S(1+n)$, Hultman [47] characterises what he calls "valid decompositions" of the cycle graph.

4.3 An explicit formula for the Hultman numbers

Several authors give formulae for the number $D(n,k)$ of factorisations of an n-cycle into the product $\rho \circ \sigma$, where $\rho, \sigma \in S_n$, ρ is an n-cycle, and $c(\Gamma(\sigma)) = k$ (see e.g. Goupil [41] or Stanley [70]). Goupil and Schaeffer [42] give a general formula for the number of factorisations of an n-cycle into two permutations of given classes, expressed as a summation of only nonnegative terms. We will use their result and Theorem 4.1 to derive a formula for computing $\mathcal{S}_H(n,k)$.

Definition 4.5. A *composition* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l)$ is a finite sequence of nonnegative integers. Its *length* is l. If $\sum_{i=1}^{l} \lambda_i = n$, we say that λ is a *composition of* n and we write $\lambda \models n$.

Compositions are very similar to partitions (Definition 4.2 page 51), except that the parts of a composition can be equal to 0 and that they are not written in decreasing order. As a consequence, compositions $(0,1,0,1)$ and $(1,0,1,0)$ of 2 are different objects. Denote $c_{\lambda\mu}^{(n)}$ the number of ways to express a given n-cycle as the product of two permutations whose classes are given respectively by partitions λ and μ.

Theorem 4.3. *[42] Let $\lambda = (\lambda_1, \dots, \lambda_l)$ and $\mu = (\mu_1, \dots, \mu_k)$ be any two partitions of n, and set $g = \frac{n+1-l-k}{2}$. Then*

$$c_{\lambda\mu}^{(n)} = \frac{n}{z_\lambda z_\mu 2^{2g}} \sum_{(g_1,g_2)\models g} (l+2g_1-1)!\,(k+2g_2-1)! \times \sum_{\substack{(i_1,\dots,i_l)\models g_1\\(j_1,\dots,j_k)\models g_2}} \prod_{h_1=1}^{l} \binom{\lambda_{h_1}}{2i_{h_1}+1} \prod_{h_2=1}^{k} \binom{\mu_{h_2}}{2j_{h_2}+1}.$$

From the previous section, we need to count the number of factorisations of a given n-cycle into an n-cycle and a permutation with k disjoint cycles. In the present notation, this number is the sum of $c_{(n)\mu}^{(n)}$ over all partitions μ of n of length k. As we show below, it is possible to obtain a slightly simpler expression for $c_{(n)\mu}^{(n)}$.

Lemma 4.2. *The number of ways to express a given n-cycle as the product of an n-cycle and a permutation of class given by $\mu = (\mu_1, \dots, \mu_k)$ is*

$$c_{(n)\mu}^{(n)} = \frac{n!}{z_\mu 2^{n-k}} \sum_{i=0}^{\frac{n-k}{2}} \frac{1}{2i+1} \sum_{(j_1,\dots,j_k)\models \frac{n-k}{2}-i} \prod_{h=1}^{k} \binom{\mu_h}{2j_h+1}.$$

Proof. Simplification of the formula of Theorem 4.3 in the case where $\lambda = (n)$. Since $l = 1$, there is a unique composition $(g_1) \models g_1$. Therefore

$$\sum_{\substack{(i_1,\dots,i_l)\models g_1\\(j_1,\dots,j_k)\models g_2}} \prod_{h_1=1}^{l} \binom{\lambda_{h_1}}{2i_{h_1}+1} \prod_{h_2=1}^{k} \binom{\mu_{h_2}}{2j_{h_2}+1} = \binom{n}{2g_1+1} \sum_{(j_1,\dots,j_k)\models g_2} \prod_{h_2=1}^{k} \binom{\mu_{h_2}}{2j_{h_2}+1}.$$

On the other hand, the length of μ is k, so $g = \frac{n-k}{2}$, and $g_2 = g - g_1$; we have

$$(l+2g_1-1)!\,(k+2g_2-1)! = (2g_1)!\,(n-2g_1-1)!,$$

which simplifies further when both sides are multiplied by $\binom{n}{2g_1+1}$:

$$(l+2g_1-1)!\,(k+2g_2-1)! \binom{n}{2g_1+1} = \frac{(2g_1)!\,(n-2g_1-1)!\,n!}{(2g_1+1)!\,(n-2g_1-1)!} = \frac{n!}{2g_1+1}.$$

Finally, note that $z_\lambda = \prod_i \alpha_i!\, i^{\alpha_i} = 1!\, n^1 = n$. Hence

$$\frac{n}{z_\lambda z_\mu 2^{2g}} = \frac{n}{n z_\mu 2^{n-k}} = \frac{1}{z_\mu 2^{n-k}}.$$

All these operations lead to the following simplification of the formula in Theorem 4.3:

$$c_{(n)\mu}^{(n)} = \frac{n!}{z_\mu 2^{n-k}} \sum_{(g_1,g_2)\models g} \frac{1}{2g_1+1} \sum_{(j_1,\dots,j_k)\models g_2} \prod_{h_2=1}^{k} \binom{\mu_{h_2}}{2j_{h_2}+1}.$$

Setting $g_1 = i$ and $h = h_2$, we obtain $g_2 = \frac{n-k}{2} - i$ and then the required equality. □

We can now give an explicit formula for computing the Hultman number $\mathcal{S}_H(n,k)$.

Theorem 4.4. *For all n, k in $\mathbb{N}$:*

$$\mathcal{S}_H(n,k) = \frac{(n+1)!}{2^{n+1-k}} \sum_{(\mu_1,\ldots,\mu_k)\vdash(n+1)} \frac{1}{z_\mu} \sum_{i=0}^{\frac{n+1-k}{2}} \frac{1}{2i+1} \sum_{(j_1,\ldots,j_k)\models\frac{n+1-k}{2}-i} \prod_{h=1}^{k} \binom{\mu_h}{2j_h+1} \quad (4.2)$$

Proof. Clearly, $D(n+1,k) = \sum_{(\mu_1,\ldots,\mu_k)\vdash(n+1)} c^{(n+1)}_{(n+1)\mu}$. Equation (4.2) follows from Corollary 4.2 and Lemma 4.2. □

Using this time Theorem 4.2 together with Lemma 4.2, we similarly derive:

Theorem 4.5. *The number of permutations π in S_n having Hultman class $\mu = (\mu_1,\ldots,\mu_k)$ equals*

$$c^{(n+1)}_{(n+1)\mu} = \frac{(n+1)!}{z_\mu 2^{n+1-k}} \sum_{i=0}^{\frac{n+1-k}{2}} \frac{1}{2i+1} \sum_{(j_1,\ldots,j_k)\models\frac{n+1-k}{2}-i} \prod_{h=1}^{k} \binom{\mu_h}{2j_h+1}.$$

Table 4.1 shows a few numerical values of $\mathcal{S}_H(n,k)$. Note that the successive values of $\mathcal{S}_H(n,1)$ ($n = 2,4,\ldots$) form sequence A060593 in OEIS [69] (in the context of cycle factorisations), whereas the other values do not appear in OEIS (except of course for $\binom{n+2}{4}$).

4.4 Applications

4.4.1 Counting results for restricted cases

A first consequence of Theorem 4.1 is the derivation of two simple formulae for particular values of $\mathcal{S}_H(n,k)$ and $D(n+1,k)$:

1. for all even n, we have $\mathcal{S}_H(n,1) = 2\frac{n!}{n+2}$, by computing the number $D(n+1,1)$ of ways of expressing an $(n+1)$-cycle as the product of two $(n+1)$-cycles (see e.g. Boccara [14]);

2. for all n in $\mathbb{N}$, we have $D(n+1,n-1) = \binom{n+2}{4}$, by computing the number $\mathcal{S}_H(n,n-1)$ of permutations of S_n whose cycle graph decomposes into $n-1$ alternating cycles (see Hultman [47]).

On the other hand, Theorem 4.5 paves the way for some other counting formulae. The number of simple permutations (Definition 3.18 page 25) can be obtained by restricting partitions $(\mu_1,\ldots,\mu_k)$ of $n+1$ in Theorem 4.5 to those made up only of 1's, 2's and 3's. More compact formulae for enumerating 2-permutations and 3-permutations (Definition 3.19 page 25) are derived as follows.

Proposition 4.1. *The number of 2-permutations in S_n is*

$$\frac{(n+1)!}{\left(\frac{n+1}{2}+1\right)!\,2^{\frac{n+1}{2}}}.$$

$n\backslash k$	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1										
2	1	0	1									
3	0	5	0	1								
4	8	0	15	0	1							
5	0	84	0	35	0	1						
6	180	0	469	0	70	0	1					
7	0	3 044	0	1 869	0	126	0	1				
8	8 064	0	26 060	0	5 985	0	210	0	1			
9	0	193 248	0	152 900	0	16 401	0	330	0	1		
10	604 800	0	2 286 636	0	696 905	0	39 963	0	495	0	1	
11	0	19 056 960	0	18 128 396	0	2 641 925	0	88 803	0	715	0	1

Table 4.1: A few values of $\mathcal{S}_H(n,k)$.

Proof. In Theorem 4.5, let $\mu = (\underbrace{2, 2, \ldots, 2}_{\frac{n+1}{2}})$. Thus $k = \frac{n+1}{2}$ and the number of 2-permutations in S_n is therefore

$$\frac{(n+1)!}{\left(\frac{n+1}{2}\right)!\, 2^{\frac{n+1}{2}} 2^{n+1-\frac{n+1}{2}}} \sum_{i=0}^{\frac{n+1-\frac{n+1}{2}}{2}} \frac{1}{2i+1} \sum_{(j_1,\ldots,j_{\frac{n+1}{2}}) \models \left(\frac{n+1-\frac{n+1}{2}}{2}-i\right)} \prod_{h=1}^{\frac{n+1}{2}} \binom{2}{2j_h+1}$$

$$= \frac{(n+1)!}{\left(\frac{n+1}{2}\right)!\; 2^{n+1}} \sum_{i=0}^{\frac{n+1}{4}} \frac{1}{2i+1} \sum_{(j_1,\ldots,j_{\frac{n+1}{2}}) \models \left(\frac{n+1}{4}-i\right)} \prod_{h=1}^{\frac{n+1}{2}} \binom{2}{2j_h+1}.$$

Since $\frac{n+1}{4} \geq i$, we have two cases:

1. if $\frac{n+1}{4} > i$, then there exists $1 \leq h_0 \leq h$ such that $j_{h_0} \geq 1$, which implies $\binom{2}{2j_{h_0}+1} = 0 = \prod_{h=1}^{\frac{n+1}{2}} \binom{2}{2j_h+1}$.
2. if $\frac{n+1}{4} = i$, then the only composition of 0 being obviously $(0, \ldots, 0)$ we derive $\prod_{h=1}^{\frac{n+1}{2}} \binom{2}{2j_h+1} = 2^{\frac{n+1}{2}}$.

Consequently, the number of 2-permutations in S_n equals

$$\frac{(n+1)!}{\left(\frac{n+1}{2}\right)!\, 2^{n+1}} \frac{1}{2\frac{n+1}{4}+1} 2^{\frac{n+1}{2}},$$

which gives the wanted expression. □

Removing all 0's from the resulting sequence yields sequence A035319 in OEIS, which counts certain maps on orientable surfaces. A bijection relating those maps to special factorisations of cycles is derived from Proposition 4.1 in [42]. In turn, our Theorem 4.5 explains why sequence A035319 also counts 2-permutations.

Proposition 4.2. *The number of* 3*-permutations in* S_n *is*

$$\frac{(n+1)!}{\left(\frac{n+1}{3}\right)!\, 12^{\frac{n+1}{3}}} \sum_{i=0}^{\frac{n+1}{3}} \binom{\frac{n+1}{3}}{i} \frac{3^i}{2i+1}.$$

Proof. In Theorem 4.5, let this time $\mu = (\underbrace{3, 3, \ldots, 3}_{\frac{n+1}{3}})$. Thus $k = \frac{n+1}{3}$ and the number of 3-permutations in S_n is therefore

$$\frac{(n+1)!}{\left(\frac{n+1}{3}\right)!\, 3^{\frac{n+1}{3}}\, 2^{n+1-\frac{n+1}{3}}} \sum_{i=0}^{\frac{n+1-\frac{n+1}{3}}{2}} \frac{1}{2i+1} \sum_{(j_1,\ldots,j_{\frac{n+1}{3}}) \models \frac{n+1-\frac{n+1}{3}}{2}-i} \prod_{h=1}^{\frac{n+1}{3}} \binom{3}{2j_h+1}$$

$$= \frac{(n+1)!}{\left(\frac{n+1}{3}\right)!\, 12^{\frac{n+1}{3}}} \sum_{i=0}^{\frac{n+1}{3}} \frac{1}{2i+1} \sum_{(j_1,\ldots,j_{\frac{n+1}{3}}) \models \frac{n+1}{3}-i} \prod_{h=1}^{\frac{n+1}{3}} \binom{3}{2j_h+1}.$$

The binomial coefficient at the end of the last expression takes value 1 when $j_h = 1$, value 3 when $j_h = 0$, and value 0 otherwise. In the last summation, we may thus assume $j_h \in \{0, 1\}$ for all h. Then the number of j_h equal to 1 in every composition of $\frac{n+1}{3} - i$ is $\frac{n+1}{3} - i$, the number of j_h equal to 0 is i, and there are exactly $\binom{\frac{n+1}{3}}{i}$ such compositions. Therefore the number of 3-permutations in S_n is

$$\frac{(n+1)!}{\left(\frac{n+1}{3}\right)!\,12^{\frac{n+1}{3}}}\sum_{i=0}^{\frac{n+1}{3}}\binom{\frac{n+1}{3}}{i}\frac{3^i}{2i+1}.$$

□

Tables 4.2, 4.3 and 4.4 give a few values of the number of simple permutations, 2-permutations, and 3-permutations, respectively.

n	1	2	3	4	5	6	7	8	9	10	11
simple	1	2	6	16	48	204	876	3 636	18 756	105 480	561 672

Table 4.2: A few values of the number of simple permutations in S_n, for $1 \leq n \leq 11$.

n	2-permutations
3	1
7	21
11	1 485
15	225 225
19	59 520 825
23	24 325 703 325
27	14 230 536 445 125
31	11 288 163 762 500 625
35	11 665 426 077 721 040 625
39	15 230 046 989 184 655 753 125
43	24 515 740 420 894 935 215 128 125
47	47 702 727 710 977 364 941 596 305 625
51	110 378 811 620 122 624 989 860 340 515 625
55	299 564 288 571 094 868 959 550 279 320 078 125
59	942 438 915 208 811 912 419 937 422 298 363 203 125
63	3 402 290 160 168 829 986 737 103 179 717 300 105 390 625

Table 4.3: A few values of the number of 2-permutations in S_n.

4.4.2 Inferring parameters of various distances

Our counting results may also be used to infer the distribution of those rearrangement distances that can be computed using parameters based on the cycle graph. We illustrate this on a generalisation of transpositions, introduced by Christie [22] and known as a *block-interchange*.

n	3-permutations
2	1
5	12
8	464
11	38 720
14	5 678 400
17	1 294 720 000
20	423 809 075 200
23	188 422 340 198 400
26	109 244 157 102 080 000
29	80 068 011 114 291 200 000
32	72 384 558 633 074 688 000 000
35	79 125 533 869 852 634 644 480 000
38	102 879 028 406 438 808 699 535 360 000
41	156 917 389 218 035 568 246 207 283 200 000
44	277 479 100 225 377 558 605 912 342 528 000 000
47	563 104 506 388 148 477 458 573 873 381 376 000 000
50	1 299 869 094 832 702 209 261 806 910 827 397 120 000 000
53	3 386 720 940 743 333 065 762 421 673 456 346 071 040 000 000
56	9 890 793 789 905 402 493 205 694 039 349 656 131 993 600 000 000
59	32 179 606 708 070 074 004 398 610 275 324 468 031 127 552 000 000 000
62	115 991 897 893 712 179 934 231 392 287 334 384 679 952 318 464 000 000 000

Table 4.4: A few values of the number of 3-permutations in S_n.

Definition 4.6. For any π in S_n, the *block-interchange* $\beta(i,j,k,l)$ with $1 \leq i < j \leq k < l \leq n+1$ applied to π exchanges the closed intervals determined respectively by i and $j-1$ and by k and $l-1$, transforming π into $\pi \circ \beta(i,j,k,l)$. Therefore, $\beta(i,j,k,l)$ is the following permutation:

$$\left(\begin{array}{cccccc} 1 \cdots i-1 & \boxed{i \cdots j-1} & j\; j+1 \cdots k-1 & \boxed{k \cdots l-1} & l\; l+1 \cdots n \\ 1 \cdots i-1 & \boxed{k \cdots l-1} & j\; j+1 \cdots k-1 & \boxed{i \cdots j-1} & l\; l+1 \cdots n \end{array}\right).$$

By contrast with transpositions, the block-interchange distance, which we denote $bid(\pi)$, can be computed in linear time [22], and finding an optimal sorting sequence of block-interchanges can be done in $O(n \log n)$ time (see Christie [22] and Feng and Zhu [35]). Christie proved the following formula for computing the block-interchange distance.

Theorem 4.6. *[22] For all π in S_n, we have $bid(\pi) = \frac{n+1-c(G(\pi))}{2}$.*

Therefore, the number of permutations in S_n whose block-interchange distance is k is exactly $\mathcal{S}_H(n, n+1-2k)$, which can be computed as in Theorem 4.4. Other distances, such as the transposition distance, are bounded by functions of the number of cycles in the cycle graph (as illustrated by Theorem 3.1 page 21); therefore, even though their distribution cannot be directly computed using Hultman numbers, these can still be used to approximate the distribution, or some related functions such as the expected value of the distance.

4.4.3 Obtaining bounds on various distances

The mapping we presented, which maps a given permutation π in S_n onto a permutation $\overline{\pi} = (0, \pi_n, \pi_{n-1}, \ldots, 1) \circ (0, 1, \ldots, n)$ in A_{n+1}, can be applied to any permutation; in particular, rearrangements we have seen so far, i.e. transpositions (page 19), exchanges (page 32) or block-interchanges (page 61), can be viewed as permutations. Therefore, given a set $\mathscr{A}$ of transformations that are revertible (in the sense that $\mathscr{A}$ contains both the transformations and their inverses) and can be represented as permutations, we can reformulate *any* rearrangement problem on π using operations from $\mathscr{A}$ as a factorisation problem on $\overline{\pi}$ using operations from $\mathscr{A}'$ (which is the image of $\mathscr{A}$ by our mapping). We will show in the next chapter how this new framework can be used to obtain bounds on rearrangement distances.

Chapter 5

A general framework for edit distances

Genome rearrangement distances are but a particular case of *edit distances* between permutations, i.e. distances based on the minimum number of allowed edit operations needed to transform a permutation into another. Another area in which edit distances have applications, besides computational biology, is the field of *interconnection network design*, where the goal is to find how to connect devices so that the resulting network is efficient and reliable (see Lakshmivarahan et al. [54] for more information on what these adjectives mean in that context, and for other desirable properties). In graph-theoretic terms, one wants to find out how to add edges to a graph whose vertex set consists of the devices we wish to connect.

A growing and successful trend in that field, starting with the seminal work of Akers and Krishnamurthy [2], has been to use Cayley graphs of permutation groups as interconnection networks, using an adequate generating set. Many generators studied in that context act on the first element or on the initial segment of the permutation, which is also called its *prefix*. Incidentally, a restriction of the transpositions we studied in Chapter 3 was introduced by Dias and Meidanis [26], which are called *prefix transpositions* and displace the initial segment of the permutation. This variant was introduced in the hope that it would shed light on the seemingly challenging problem of sorting by transpositions. Our main contributions in this chapter include the following:

- we present a framework that allows us to reformulate *any* edit distance problem on permutations in terms of particular factorisations of related even permutations (Lemma 5.3 and Theorem 5.3), using the mapping we introduced in Chapter 4;

- using this general framework, we show how to recover, in a simpler way, results previously obtained by other authors (Lemma 5.5 and Theorem 5.4) and prove a new lower bound on the prefix transposition distance;

- we prove that our new lower bound on the prefix transposition distance always outperforms that of Dias and Meidanis [26] (Theorem 5.6), and show experimentally that it is a significant improvement over all previously known results (see Section 5.8);

- we use our expression to improve the lower bound on the prefix transposition diameter of the symmetric group from $2n/3$ to $\left\lfloor \frac{3n+1}{4} \right\rfloor$ (Theorem 5.7);
- we give the prefix transposition distance of 2-permutations (Proposition 5.2).

Most results presented in this chapter have been published in [53].

5.1 A word on prefix sorting problems

We have already explained the relevance of edit distances and sorting problems on permutations in the context of computational biology (see Chapter 1). However, genome rearrangements are but one of the many possible applications: another field in which these concepts have been studied for several decades is the field of *interconnection network design*.

An *interconnection network* can be abstracted as a graph (directed or not) whose vertex set represents the devices we wish to connect (e.g. processors) and whose edges correspond to actual physical connections between two devices. Several parameters are used to assess the "quality" of the network with respect to miscellaneous criteria: we mention, among other parameters, the fact that the graph representing the network should have a small degree and a small diameter (see Lakshmivarahan, Jwo, and Dhall [54] for a thorough survey of the field and other relevant parameters).

A very large part of the interconnection network design literature is concerned with Cayley graphs of permutation groups, whose use as interconnection networks was first proposed by Akers and Krishnamurthy [2]. The idea is that each permutation stands for one of the physical devices, and each pair of devices is connected according to whether or not the corresponding permutations can be transformed into each other using one of the generators. The following two examples have received a lot of attention:

1. the *pancake network* is the Cayley graph of S_n with *prefix reversals* as generators, where a prefix reversal reverses the order of the first k elements of a permutation (an operation first studied as a game by Gates and Papadimitriou [40]); see Figure 5.1 for the pancake network of order 4;
2. the *star graph* is the Cayley graph of S_n with *prefix exchanges* as generators, where a prefix exchange swaps the first element of the permutation with any other element; see Figure 5.2 for the star graph of order 4.

Incidentally, the idea of restricting operations to the prefix (or initial segment) of permutations was reused by Dias and Meidanis [26], who initiated the study of sorting by prefix transpositions in the hope that this restricted problem would shed light on the seemingly challenging problem of sorting by transpositions. We find it interesting to note that genome rearrangement problems and interconnection network design problems are concerned with similar issues: sorting algorithms in genome rearrangements are meant to explain evolution between species, whereas in interconnection networks, they correspond to routing algorithms on the network. Moreover, as we have already mentioned, the diameter of the Cayley graph under consideration is also of interest in the field of interconnection networks, since those networks should preferably have a small diameter.

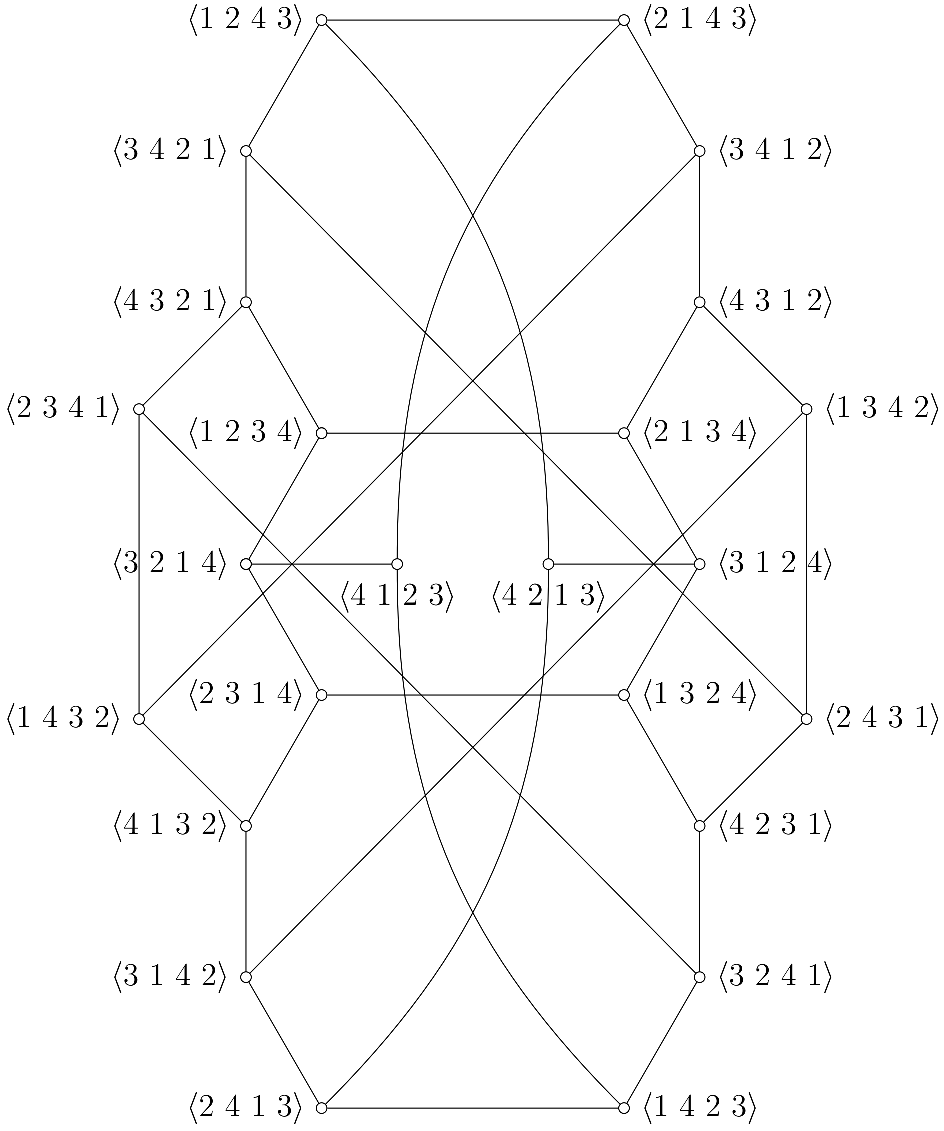


Figure 5.1: The pancake network of order 4.

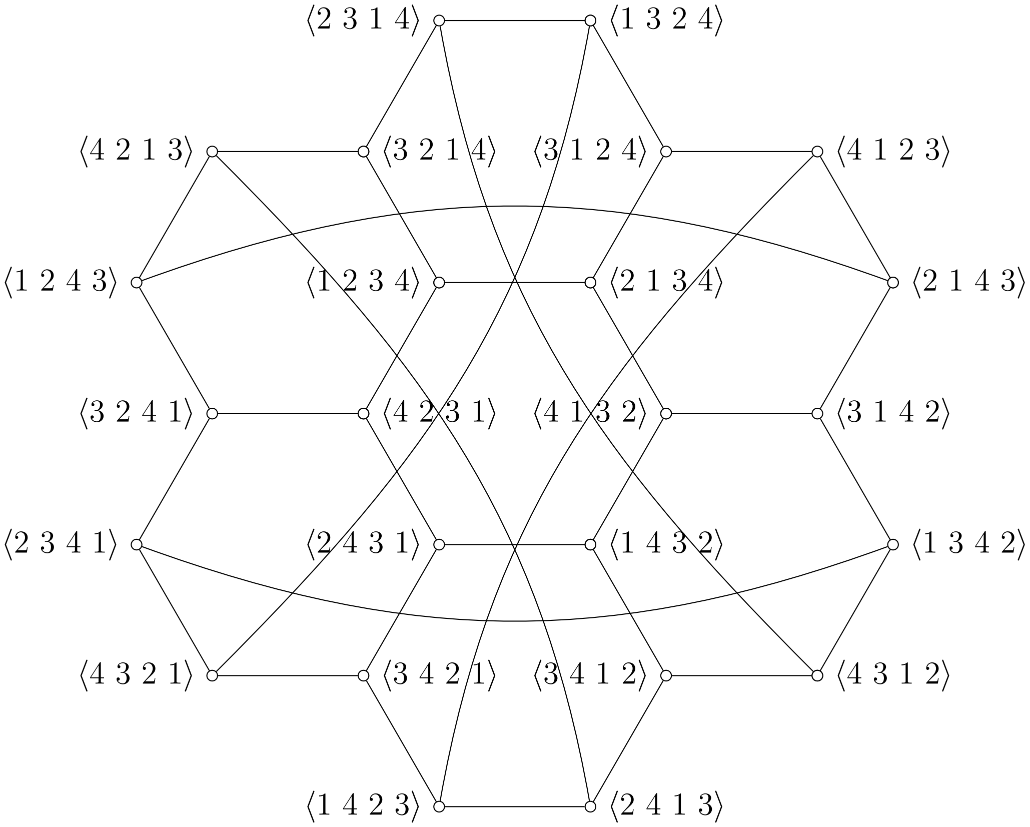


Figure 5.2: The star graph of order 4.

5.2 Background

In the previous chapters, we discussed three rearrangement operations on permutations: transpositions (Definition 3.1 page 19), exchanges (Definition 3.26 page 32) and block-interchanges (Definition 4.6 page 61), which were respectively denoted by $\tau(i,j,k)$, $\varepsilon(i,j)$ and $\beta(i,j,k,l)$. Setting $i = 1$ in those rearrangement operations turns them into "prefix rearrangements", namely, *prefix transpositions*, *prefix exchanges* and *prefix block-interchanges*. The corresponding sorting problems and "prefix distances" are defined as before.

Definition 5.1. The *prefix transposition distance* between two permutations π and σ in S_n, denoted by $ptd(\pi,\sigma)$, is the length of a shortest sequence of prefix transpositions that transforms π into σ.

Definition 5.2. The *prefix exchange distance* between two permutations π and σ in S_n, denoted by $pexc(\pi,\sigma)$, is the length of a shortest sequence of prefix exchanges that transforms π into σ.

The proof of Proposition 3.1 page 20 can be adapted to prefix transpositions and prefix exchanges in order to prove that the corresponding distances are also left-invariant, thereby allowing us to restrict our attention to sorting by prefix transpositions or by prefix exchanges and to computing the distance of a permutation with respect to the identity. As we did with the transposition distance, we therefore abbreviate $ptd(\pi,\sigma)$ (resp. $pexc(\pi,\sigma)$) to $ptd(\pi)$ (resp. to $pexc(\pi)$) whenever $\sigma = \iota$. While the computational complexity of sorting by transpositions or by prefix transpositions is unknown, a polynomial time algorithm for sorting by prefix exchanges is known, as well as a formula for computing the associated distance, both due to Akers, Krishnamurthy, and Harel [3]. The formula relies on the disjoint cycle decomposition of permutations, which we express in terms of the Γ-graph (Definition 3.21 page 28).

Theorem 5.1. *[3] For any π in S_n, we have*

$$pexc(\pi) = n + c(\Gamma(\pi)) - 2c_1(\Gamma(\pi)) - \begin{cases} 0 & \text{if } \pi_1 = 1, \\ 2 & \text{otherwise}, \end{cases}$$

where $c_1(\Gamma(\pi))$ denotes the number of 1-cycles in $\Gamma(\pi)$, or equivalently the number of fixed points of π.

Dias and Meidanis [26] initiated the study of sorting by prefix transpositions, and derived a lower bound on the corresponding distance using the following concepts.

Definition 5.3. Given a permutation π in S_n, build the permutation $\widetilde{\pi} = \langle 0\ \pi_1\ \cdots\ \pi_n\ n+1\rangle$; a pair $(\widetilde{\pi}_i, \widetilde{\pi}_{i+1})$ with $0 \leq i \leq n$ is a *prefix transposition breakpoint* if $\widetilde{\pi}_{i+1} \neq \widetilde{\pi}_i + 1$ or if $i = 0$. The number of prefix transposition breakpoints of π is denoted by $ptb(\pi)$.

Noting that a prefix transposition can remove at most two prefix transposition breakpoints and that ι is the only permutation with one prefix transposition breakpoint, they obtained the following lower bound.

Lemma 5.1. *[26] For any π in S_n:*

$$ptd(\pi) \geq \left\lceil \frac{ptb(\pi) - 1}{2} \right\rceil. \tag{5.1}$$

Chitturi and Sudborough [21] then obtained other bounds on the prefix transposition distance using the following concepts, based on permutations of $\{0, 1, 2, \ldots, n-1\}$ rather than $\{1, 2, \ldots, n\}$.

Definition 5.4. For a permutation π of $\{0, 1, 2, \ldots, n\}$, an ordered pair (π_i, π_{i+1}) is an *anti-adjacency* if $\pi_{i+1} = \pi_i - 1 \pmod n$.

Definition 5.5. A *clan* is a maximal interval of π that contains only anti-adjacencies.

Recall (Definition 3.13 page 23) that a strip in a permutation is a maximal interval that contains no breakpoint. Chitturi and Sudborough [21] prove the following lower bound.

Lemma 5.2. *[21] For any π in S_n, let $\Upsilon(\pi)$ denote the set of all clans of π of length at least* 3*, and $s(\pi)$ denote the number of strips of π. Then*

$$ptd(\pi) \geq \frac{s(\pi) + \frac{\sum_{C \in \Upsilon(\pi)}(|C|-2)}{3}}{2}. \tag{5.2}$$

Using Lemma 5.2, Chitturi and Sudborough prove a lower bound of $2n/3$ on the prefix transposition distance of $\chi = \langle n\ n-1\ \cdots\ 2\ 1\rangle$, and therefore on the prefix transposition diameter. They also prove the following upper bound on the prefix transposition diameter.

Theorem 5.2. *[21] For all π in S_n, we have $ptd(\pi) \leq n - \log_8 n$.*

5.3 The distribution of the prefix transposition distance

As we did with the transposition distance (Table 3.1 page 27), we generated S_n from the identity permutation by repeatedly composing prefix transpositions. Table 5.1 shows some experimental values of the resulting number of permutations in S_n with prefix transposition distance equal to k.

It can be seen that for $n = 4$, 7 and 11, there are only three permutations with maximal prefix transposition distance; these are:

- for $n = 4$: $\langle 1\ 4\ 3\ 2\rangle$, $\langle 2\ 1\ 4\ 3\rangle$ and $\langle 4\ 3\ 2\ 1\rangle$;
- for $n = 7$: $\langle 2\ 1\ 7\ 6\ 5\ 4\ 3\rangle$, $\langle 4\ 3\ 2\ 1\ 7\ 6\ 5\rangle$ and $\langle 7\ 6\ 5\ 4\ 3\ 2\ 1\rangle$;
- for $n = 11$: $\langle 7\ 6\ 5\ 4\ 3\ 2\ 1\ 8\ 11\ 10\ 9\rangle$, $\langle 8\ 2\ 7\ 6\ 5\ 4\ 3\ 1\ 11\ 10\ 9\rangle$ and $\langle 11\ 10\ 9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1\rangle$.

$n \setminus k$	0	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	1	3	2	0	0	0	0	0	0	0
4	1	6	14	3	0	0	0	0	0	0
5	1	10	50	55	4	0	0	0	0	0
6	1	15	130	375	194	5	0	0	0	0
7	1	21	280	1 575	2 598	562	3	0	0	0
8	1	28	532	4 970	18 096	15 532	1 161	0	0	0
9	1	36	924	12 978	85 128	188 386	74 183	1 244	0	0
10	1	45	1 500	29 610	308 988	1 364 710	1 679 189	244 430	327	0
11	1	55	2 310	61 050	933 108	7 030 210	19 713 542	11 759 676	416 845	3

Table 5.1: The number of permutations π in S_n with $ptd(\pi) = k$, for $1 \leq n \leq 11$.

Unfortunately, these permutations do not seem to share any obvious common feature (except for the reversed permutation χ, which appears for all three values of n) that would reveal why they are harder to sort.

We also report on some "bad news" regarding sorting by prefix transpositions: recall the three equivalence relations we have seen in Chapter 3, namely:

- the reduction equivalence relation (Section 3.1.3 page 22),
- the toric equivalence relation (Section 3.1.4 page 23), and
- the conjugacy relation based on χ (Section 3.2 page 26).

All three operations have been shown to preserve the transposition distance; unfortunately, none of them preserves the prefix transposition distance, as the following counter-examples show:

1. we have $\langle 1\ 4\ 3\ 2\rangle \equiv_r \langle 3\ 4\ 2\ 1\rangle$, but $ptd(\langle 1\ 4\ 3\ 2\rangle) = 3$ and $ptd(\langle 3\ 4\ 2\ 1\rangle) = 2$;
2. the prefix transposition distance of $\langle 2\ 1\ 4\ 3\rangle$ is 3, but the prefix transposition distance of $\langle 4\ 1\ 3\ 2\rangle$, which is torically equivalent to $\langle 2\ 1\ 4\ 3\rangle$, is 2;
3. finally, conjugation by χ does not preserve the prefix transposition distance either: indeed, the prefix transposition distance of $\langle 1\ 4\ 3\ 2\rangle$ is 3, but the prefix transposition distance of $\langle 3\ 2\ 1\ 4\rangle = \langle 1\ 4\ 3\ 2\rangle^{\chi}$ is 2.

5.4 A general lower bounding technique

Recall the mapping we introduced in Chapter 4, which allowed us to encode any permutation in S_n using an even permutation:

$$F : S_n \rightarrow A_{n+1} : \pi \mapsto \overline{\pi} = (0, \pi_n, \pi_{n-1}, \ldots, \pi_1) \circ (0, 1, 2, \ldots, n).$$

It can be easily checked that $\overline{\iota} = \langle 0\ 1\ 2\ \cdots\ n\rangle$. As we have seen in Section 4.2, speaking about cycles of $\overline{\pi}$, of $\Gamma(\overline{\pi})$ or of $G(\pi)$ is equivalent. We will now demonstrate how F can be used to obtain bounds on sorting problems. The following result expresses how the action of *any* rearrangement operation σ on π is translated on $\overline{\pi}$. In the following, we identify a permutation π in S_n with the permutation $\langle 0\ \pi_1\ \pi_2\ \cdots\ \pi_n\rangle$ in S_{n+1}.

Lemma 5.3. *For all π, σ in S_n: $\overline{\pi \circ \sigma} = \overline{\sigma}^{\pi} \circ \overline{\pi}$.*

Proof. The following relation will be useful:

$$\pi = (0, \pi_n, \pi_{n-1}, \ldots, \pi_1) \circ \pi \circ (0, 1, \ldots, n). \tag{5.3}$$

By definition, we have:

$$\begin{aligned}
\overline{\pi \circ \sigma} &= (0, (\pi \circ \sigma)_n, (\pi \circ \sigma)_{n-1}, \ldots, (\pi \circ \sigma)_1) \circ (0, 1, \ldots, n) \\
&= (0, \pi_{\sigma_n}, \pi_{\sigma_{n-1}}, \ldots, \pi_{\sigma_1}) \circ (0, 1, \ldots, n) \\
&= \pi \circ (0, \sigma_n, \sigma_{n-1}, \ldots, \sigma_1) \circ \pi^{-1} \circ (0, 1, \ldots, n) \\
&= \pi \circ (0, \sigma_n, \sigma_{n-1}, \ldots, \sigma_1) \circ (0, 1, \ldots, n) \circ (0, 1, \ldots, n)^{-1} \circ \pi^{-1} \\
&\quad \circ (0, 1, \ldots, n) \\
&= \pi \circ \overline{\sigma} \circ (\pi \circ (0, 1, \ldots, n))^{-1} \circ (0, 1, \ldots, n) \\
&= \pi \circ \overline{\sigma} \circ \left((0, \pi_n, \ldots, \pi_1)^{-1} \circ \pi\right)^{-1} \circ (0, 1, \ldots, n) \qquad \text{(using (5.3))} \\
&= \pi \circ \overline{\sigma} \circ \pi^{-1} \circ (0, \pi_n, \ldots, \pi_1) \circ (0, 1, \ldots, n) \\
&= \pi \circ \overline{\sigma} \circ \pi^{-1} \circ \overline{\pi}.
\end{aligned}$$

□

We are now ready to prove our main result.

Theorem 5.3. *Let X be a subset of S_n whose elements are mapped by F onto $X' \subseteq A_{n+1}$. Moreover, let $\mathscr{C}$ be the union of the conjugacy classes (of S_{n+1}) that intersect with X'; then for any π in S_n, any factorisation of π into t elements of X yields a factorisation of $\overline{\pi}$ into t elements of $\mathscr{C}$. More explicitly, if*

$$\pi = g_t \circ g_{t-1} \circ \cdots \circ g_1,$$

where $g_i \in X$ for $1 \leq i \leq t$, then

$$\overline{\pi} = \overline{g_1}^{(g_t \circ g_{t-1} \circ \cdots \circ g_2)} \circ \cdots \circ \overline{g_{t-2}}^{(g_t \circ g_{t-1})} \circ \overline{g_{t-1}}^{g_t} \circ \overline{g_t},$$

where all terms in this factorisation of $\overline{\pi}$ belong to $\mathscr{C}$.

Proof. Induction on t. The base case is $\pi \in X$, and clearly $\overline{\pi} \in X' \subseteq \mathscr{C}$. For the induction, let $\pi = g_t \circ g_{t-1} \circ \cdots \circ g_1$, where $g_i \in X$ for $1 \leq i \leq t$, and let $\sigma = g_{t-1} \circ \cdots \circ g_2 \circ g_1$; by Lemma 5.3, we have:

$$\overline{\pi} = \overline{g_t \circ g_{t-1} \circ \cdots \circ g_2 \circ g_1} = \overline{g_t \circ \sigma} = g_t \circ \overline{\sigma} \circ g_t^{-1} \circ \overline{g_t}.$$

By induction, $\overline{\sigma} = g'_{t-1} \circ g'_{t-2} \circ \cdots \circ g'_1$, where $g'_i \in \mathscr{C}$ for $1 \leq i \leq t$; therefore:

$$\begin{aligned}
g_t \circ \overline{\sigma} \circ g_t^{-1} &= g_t \circ g'_{t-1} \circ g'_{t-2} \circ \cdots \circ g'_1 \circ g_t^{-1} \\
&= \underbrace{g_t \circ g'_{t-1} \circ g_t^{-1}}_{h_t} \circ \underbrace{g_t \circ g'_{t-2} \circ g_t^{-1}}_{h_{t-1}} \circ g_t \circ \cdots \circ g_t^{-1} \circ \underbrace{g_t \circ g'_1 \circ g_t^{-1}}_{h_1},
\end{aligned}$$

and $h_1, \ldots, h_{t-1} \in \mathscr{C}$, which completes the proof. □

As we briefly explain before applying our method in the next section, Theorem 5.3 allows us to prove lower bounds on our sorting problems: indeed, as we explained in Section 2.3, any sorting sequence of length t for π made of elements of X yields a factorisation of π into the product of t elements (of X, provided X contains both the transformations and their inverses, which is easily shown to be the case for all operations considered here). The resulting factorisation can in turn be converted, using Theorem 5.3, into a factorisation of $\overline{\pi}$ into the product of t elements of $\mathscr{C}$. The length of a shortest such factorisation of $\overline{\pi}$ into the product of elements of $\mathscr{C}$ is therefore a lower bound on the length of a factorisation of π into the product of elements of X, which in turn yields a lower bound on the edit distance of interest.

5.5 Recovering previous results

We illustrate how to use Theorem 5.3 to recover two previous results on the block-interchange and transposition distances. First, we need to characterise the image of a block-interchange by our mapping.

Lemma 5.4. *For any block-interchange $\beta(i,j,k,l)$, we have*

$$\overline{\beta(i,j,k,l)} = (j-1,l-1)\circ(i-1,k-1).$$

Proof. Using our mapping and the definition of a block-interchange, we have

$$\begin{aligned} & (0,n,n-1,\ldots,l,j-1,j-2,\ldots,i,k-1,k-2,\ldots,j,l-1,l-2,\ldots,\\ & k,i-1,i-2,\ldots,1)\circ(0,1,2,\ldots,n)\\ = \; & (0)(1)\cdots(i-2)(i-1,k-1)(i)(i+1)\cdots(j-2)(j-1,l-1)(j)\\ & (j+1)\cdots(k-2)(k)(k+1)\cdots(l-2)(l)(l+1)\cdots(n)\\ = \; & (j-1,l-1)\circ(i-1,k-1). \end{aligned}$$

□

Note that the two cycles $(j-1,l-1)$ and $(i-1,k-1)$ might not be disjoint, since by definition of $\beta(i,j,k,l)$ we may have $j=k$ (hence the use of $\circ$ in the expression of $\overline{\beta(i,j,k,l)}$). We can now recover a known lower bound on the block-interchange distance, which is actually the exact distance (recall Theorem 4.6 page 61).

Lemma 5.5. *[22] For all π in S_n, we have $bid(\pi) \geq \frac{n+1-c(\Gamma(\overline{\pi}))}{2}$.*

Proof. By Theorem 5.3 and Lemma 5.4, a lower bound on $bid(\pi)$ is given by the length of a minimum factorisation of $\overline{\pi}$ into pairs of exchanges. Since this length equals $(n+1-c(\Gamma(\overline{\pi})))/2$ (see e.g. Jerrum [48]), the proof follows. □

Let us now characterise the image of a transposition by our mapping.

Lemma 5.6. *For any transposition $\tau(i,j,l)$, we have*

$$\overline{\tau(i,j,l)} = (i-1,l-1,j-1).$$

Proof. Note that a transposition is a particular case of a block-interchange; more precisely, we have $\tau(i,j,l)=\beta(i,j,j,l)$, and Lemma 5.4 yields:

$$\overline{\tau(i,j,l)} = \overline{\beta(i,j,j,l)} = (j-1,l-1)\circ(i-1,j-1) = (i-1,l-1,j-1).$$

□

We recover the lower bound of Bafna and Pevzner [9] on the transposition distance (Theorem 3.1 page 21). For convenience, we restate the result below.

Theorem 5.4. *[9] For all π in S_n, we have $td(\pi) \geq \frac{n+1-c_{odd}(\Gamma(\overline{\pi}))}{2}$.*

Proof. By Theorem 5.3 and Lemma 5.6, a lower bound on $td(\pi)$ is given by the length of a minimum factorisation of $\overline{\pi}$ into 3-cycles. Since this length equals $(n+1-c_{odd}(\Gamma(\overline{\pi})))/2$ (see e.g. Jerrum [48]), the proof follows. □

5.6 An improved lower bound on the prefix transposition distance

Using our theory, we prove a *new* lower bound on $ptd(\pi)$ and show that it always outperforms lower bound (5.1), previously obtained by Dias and Meidanis [26]. We will find it convenient to express $ptb(\pi)$ (defined after Theorem 5.1 page 66) as follows.

Lemma 5.7. *For any π in S_n, we have*

$$ptb(\pi) = n + 1 - c_1(\Gamma(\overline{\pi})) + \begin{cases} 1 & \textit{if } \pi_1 = 1, \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. Recall, as we already mentioned when we defined breakpoints and adjacencies (Definition 3.11 page 22), that among the $n+1$ pairs of adjacent elements in $\widetilde{\pi} = \langle 0\ \pi_1\ \cdots\ \pi_n\ n+1\rangle$, each adjacency in $\widetilde{\pi}$ gives rise to a 1-cycle in $\Gamma(\overline{\pi})$, so $b(\pi) = n+1-c_1(\Gamma(\overline{\pi}))$. If $\pi_1 \neq 1$ then $b(\pi) = ptb(\pi)$; otherwise, the 1-cycle that corresponds to $(0, \pi_1) = (0, 1)$ is a prefix transposition breakpoint, and we must therefore add 1 to $b(\pi)$ in order to obtain $ptb(\pi)$. □

Let $d_3^1(\pi)$ denote the length of a minimum factorisation of π in S_n into a product of 3-cycles, where each 3-cycle in the factorisation is further required to contain the first element.

Proposition 5.1. *For any π in S_n, we have $ptd(\pi) \geq d_3^1(\overline{\pi})$.*

Proof. Replace i with 1 in Lemma 5.6, and mimic the proof of Theorem 5.4. □

Next, we show how to compute $d_3^1(\pi)$ for π in A_n. The following simple observation will be useful.

Observation 5.1. *For any π in A_n, we have $n \equiv c(\Gamma(\pi)) \pmod 2$.*

Lemma 5.8. *For any π in A_n, we have*

$$d_3^1(\pi) = \frac{n + c(\Gamma(\pi))}{2} - c_1(\Gamma(\pi)) - \begin{cases} 0 & \textit{if } \pi_1 = 1, \\ 1 & \textit{otherwise.} \end{cases}$$

Proof. Given a minimum factorisation of length ℓ of an even permutation π into prefix exchanges, we can construct a sequence of $\ell/2$ 3-cycles containing 1 by noting that $(1, j) \circ (1, i) = (1, i, j)$. Therefore $d_3^1(\pi) \leq \ell/2$. On the other hand, assume there exists a shorter sequence of 3-cycles acting on the first element whose product is π; then one can split each of these 3-cycles into two prefix exchanges using the relation above and find a shorter expression for π as a product of prefix exchanges, a contradiction. The result follows from Theorem 5.1. □

As a corollary, we obtain the following lower bound on the prefix transposition distance:

Theorem 5.5. *For any π in S_n, we have*

$$ptd(\pi) \geq \frac{n+1+c(\Gamma(\overline{\pi}))}{2} - c_1(\Gamma(\overline{\pi})) - \left\{ \begin{array}{ll} 0 & \textit{if } \pi_1 = 1, \\ 1 & \textit{otherwise.} \end{array} \right. \tag{5.4}$$

Proof. Follows from Proposition 5.1 and Lemma 5.8. □

We conclude this section by proving that our lower bound always outperforms Dias and Meidanis' (Lemma 5.1).

Theorem 5.6. *Lower bound* (5.4) *is always at least as large as lower bound* (5.1).

Proof. Assume $\pi \neq \iota$ (otherwise the result trivially holds); this implies that $\Gamma(\overline{\pi})$ has at least one cycle of length at least 2, which means that $c(\Gamma(\overline{\pi})) - c_1(\Gamma(\overline{\pi})) \geq 1$. There are two cases to prove: if $\pi_1 = 1$, then lower bound (5.1) becomes

$$\left\lceil \frac{(n+1-c_1(\Gamma(\overline{\pi}))+1)-1}{2} \right\rceil = \left\lceil \frac{n+1-c_1(\Gamma(\overline{\pi}))}{2} \right\rceil ,$$

and lower bound (5.4) satisfies

$$\frac{n+1+c(\Gamma(\overline{\pi}))-2c_1(\Gamma(\overline{\pi}))}{2} \geq \frac{n+2-c_1(\Gamma(\overline{\pi}))}{2} \geq \left\lceil \frac{n+1-c_1(\Gamma(\overline{\pi}))}{2} \right\rceil .$$

On the other hand, if $\pi_1 \neq 1$, then lower bound (5.1) becomes

$$\left\lceil \frac{(n+1-c_1(\Gamma(\overline{\pi})))-1}{2} \right\rceil = \left\lceil \frac{n-c_1(\Gamma(\overline{\pi}))}{2} \right\rceil ,$$

and by Observation 5.1, lower bound (5.4) becomes

$$\begin{aligned} \frac{n+1+c(\Gamma(\overline{\pi}))}{2} - c_1(\Gamma(\overline{\pi})) - 1 &= \left\lceil \frac{n+1+c(\Gamma(\overline{\pi}))-2c_1(\Gamma(\overline{\pi}))-2}{2} \right\rceil \\ &\geq \left\lceil \frac{n-c_1(\Gamma(\overline{\pi}))}{2} \right\rceil . \end{aligned}$$

□

5.7 A tighter lower bound on the prefix transposition diameter

Dias and Meidanis [26] observed that the prefix transposition diameter lies between $n/2$ and $n-1$, and conjectured that it is equal to $n - \left\lfloor \frac{n}{4} \right\rfloor$. Recently, Chitturi and Sudborough [21] improved those bounds to $2n/3$ and $n - \log_8 n$, respectively. Using our new lower bound, we further improve the lower bound on the prefix transposition diameter. We prove our result in a constructive way, by building families of permutations whose prefix transposition distance is at least $\left\lfloor \frac{3n+1}{4} \right\rfloor$ (Figure 5.3, which follows our result, shows examples of such permutations for each case described in the proof).

Theorem 5.7. *For $n \geq 2$, the prefix transposition diameter of S_n is at least* $\left\lfloor \frac{3n+1}{4} \right\rfloor$.

Proof. We construct a family of permutations whose prefix transposition distance is at least $\left\lfloor \frac{3n+1}{4} \right\rfloor$. Let $\pi = \langle 3\ 2\ 1\ 4\ 7\ 6\ 5\ \cdots\ n-4\ n\ n-2\ n-3 \rangle$, or any other 2-*permutation*, i.e. a permutation such that $\Gamma(\overline{\pi})$ contains only cycles of length 2 (this requires that $n \equiv 3 \pmod 4$). There are four cases to examine, each of which relies on Theorem 5.5:

1. if $n \equiv 3 \pmod 4$, we have $ptd(\pi) \geq (n+1+(n+1)/2)/2 - 0 - 1 = \frac{3n-1}{4}$ (see Figure 5.3(a) for an example).
2. if $n \equiv 0 \pmod 4$, let σ be a permutation such that $\Gamma(\overline{\sigma})$ is obtained by inserting a fixed point at the beginning of $\Gamma(\overline{\pi})$; since $\overline{\sigma}$ fixes 0 and has $n/2$ 2-cycles, we have $ptd(\sigma) \geq (n+1+n/2+1)/2 - 1 - 0 = \frac{3n}{4}$ (see Figure 5.3(b) for an example).
3. if $n \equiv 1 \pmod 4$, let σ' be a permutation such that $\Gamma(\overline{\sigma'})$ is obtained by inserting a fixed point anywhere in $\Gamma(\overline{\sigma})$; we have $ptd(\sigma') \geq (n+1+\frac{n-2+1}{2}+2)/2 - 2 = \frac{3n+1}{4}$ (see Figure 5.3(c) for an example).
4. if $n \equiv 2 \pmod 4$, let σ'' be a permutation such that $\Gamma(\overline{\sigma''})$ is obtained by inserting a 3-cycle (a, c, b) with $a < b < c$ anywhere in $\Gamma(\overline{\pi})$. Since $\overline{\sigma''}$ has $(n+1-3)/2+1$ cycles of length at least 2, we have $ptd(\sigma'') \geq (n+1+\frac{n+1-3}{2}+1)/2 - 0 - 1 = \frac{3n-2}{4}$ (see Figure 5.3(d) for an example).

□

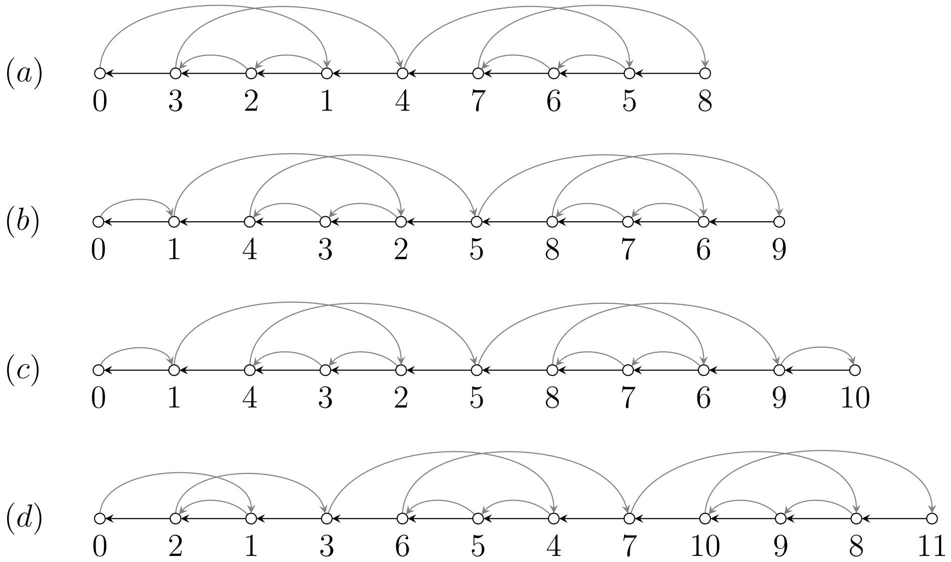


Figure 5.3: Examples of permutations π in S_n with $ptd(\pi) \geq \left\lfloor \frac{3n+1}{4} \right\rfloor$, for all values of n mod 4.

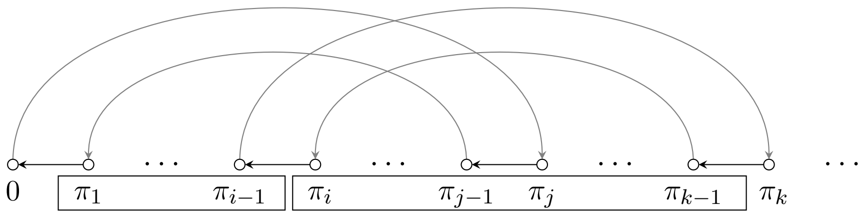
We can actually show that the lower bound on the prefix transposition distance of 2-permutations is tight. In order to do that, we will need the following result on nonoriented cycles (recall Definition 3.6 page 21).

Lemma 5.9. *[9] For any π in S_n, let C_1 be a nonoriented cycle in $G(\pi)$ and a, b be two arbitrary black arcs of C_1; then there exists another cycle C_2 in $G(\pi)$ containing two black arcs c and d such that (a, b) and (c, d) interleave.*

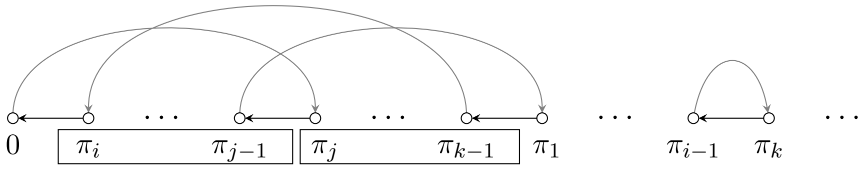
In particular, this result implies that in the cycle graph of a 2-permutation, any 2-cycle crosses another 2-cycle. We are now ready to prove the following result.

Proposition 5.2. *For any 2-permutation π in S_n, we have $ptd(\pi) = (3n - 1)/4$.*

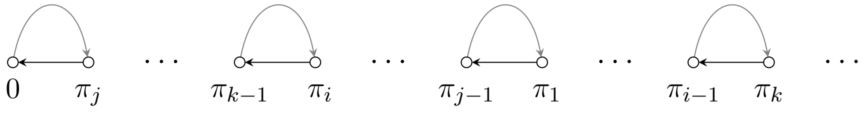
Proof. The lower bound has already been observed in Theorem 5.7. To show that it is also an upper bound, we give an algorithm that sorts π in exactly that number of steps. By Lemma 5.9, every 2-cycle crosses another one, and as observed by Bafna and Pevzner [9], a sequence of two transpositions on any two crossing 2-cycles will transform them into four adjacencies:



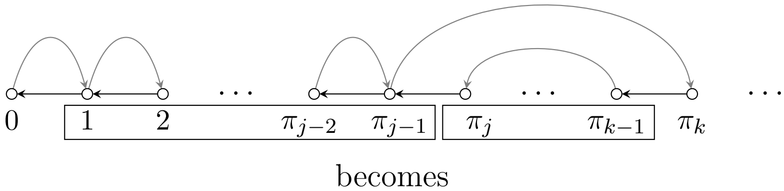
becomes



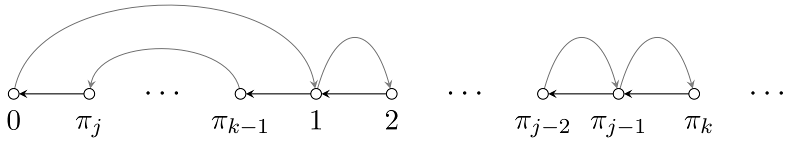
which becomes



We transform the leftmost 2-cycle and any 2-cycle it crosses into four adjacencies using two prefix transpositions, which transforms π into a permutation σ that contains $\frac{n+1}{2} - 2$ 2-cycles and fixes the first element. Then, we carry out again this process until σ is sorted, but we need three prefix transpositions at each step, since one move must be wasted to move the first element out of the way, for instance as follows:



becomes



The proof follows from the fact that the number of prefix transpositions used by this algorithm is

$$2 + \frac{3}{2}\left(\frac{n+1}{2} - 2\right) = \frac{8 + 3n - 9}{4} = \frac{3n - 1}{4}.$$

□

5.8 Experimental results

We generated all permutations in S_n, for $1 \leq n \leq 10$, along with their prefix transposition distance, and compared lower bounds (5.1), (5.2) and (5.4) to the actual distance. Table 5.2 shows the results. It can be observed that many more permutations are tight with respect to our lower bound (column 5) than with respect to Dias and Meidanis' (column 3) or Chitturi and Sudborough's (column 4).

n	$n!$	tight w.r.t. (5.1)	tight w.r.t. (5.2)	tight w.r.t. (5.4)
1	1	1	1	1
2	2	2	2	2
3	6	4	4	6
4	24	13	15	22
5	120	41	48	106
6	720	196	255	574
7	5 040	862	1 144	3 782
8	40 320	5 489	7 737	27 471
9	362 880	31 033	44 187	229 167
10	3 628 800	247 006	369 979	2 103 510

Table 5.2: Experimental results; column 3 lists the number of cases where (5.1) is tight [37], column 4 lists the number of cases where (5.2) is tight, and column 5 lists the number of cases where (5.4) is tight.

Figure 5.4 shows a perhaps more intuitive presentation of those experimental results, exhibiting a plot with lines corresponding to the percentage of permutations in S_n that are tight with respect to each of the lower bounds.

We also examined how large the gap between our lower bound and the actual prefix transposition distance can get. Table 5.3 counts permutations whose prefix transposition distance equals our lower bound plus Δ. We note that, for $n \leq 9$, all permutations have a prefix transposition distance that is at most our lower bound plus 2 (plus 3 for $n = 10$).

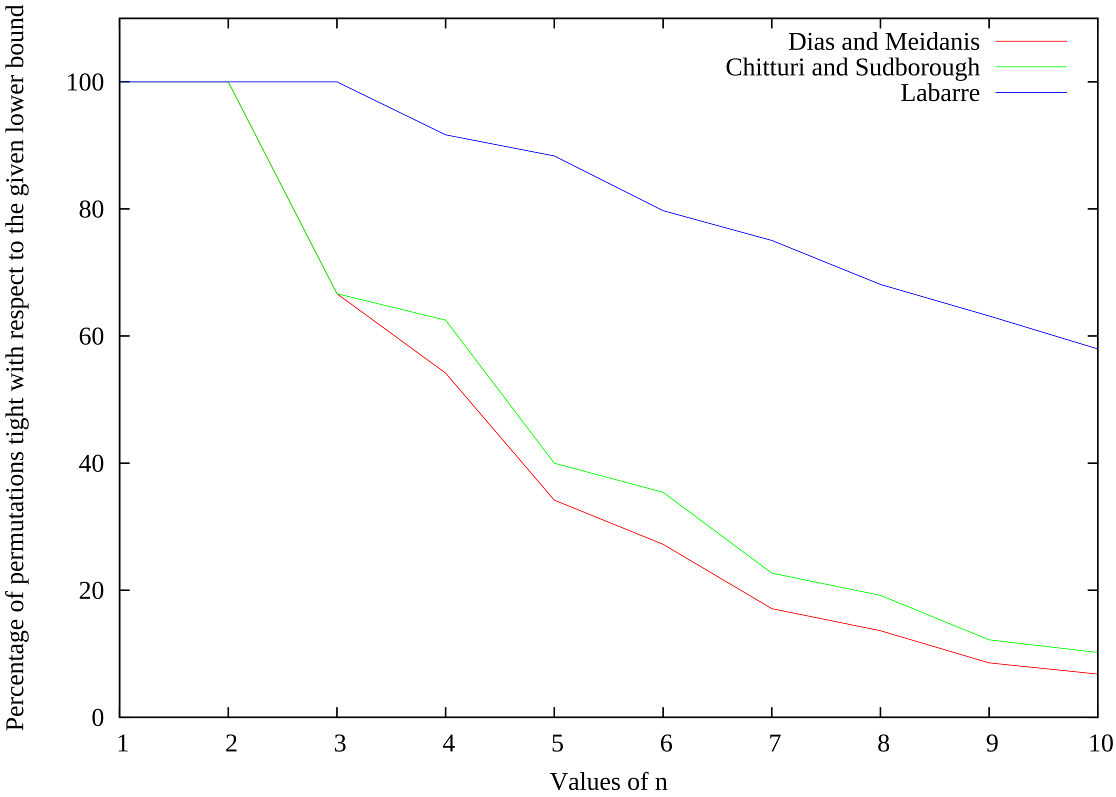


Figure 5.4: Percentage of permutations in S_n that are tight with respect to each of the lower bounds.

n	$n!$	$\Delta = 0$	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$
1	1	1	0	0	0
2	2	2	0	0	0
3	6	6	0	0	0
4	24	22	2	0	0
5	120	106	14	0	0
6	720	574	143	3	0
7	5 040	3 782	1 234	24	0
8	40 320	27 471	12 310	539	0
9	362 880	229 167	128 576	5 137	0
10	3 628 800	2 103 510	1 427 966	97 321	3

Table 5.3: Number of cases where (5.4) underestimates $ptd(\pi)$ by Δ.

Chapter 6

Haplotype networks

As discussed in more detail in Chapter 1, a lot of methods have been proposed for reconstructing and analysing phylogenetic networks (reviewed by Huber and Moulton [46], Posada and Crandall [65], or more recently in a book edited by Gascuel and Steel [39]). In the case of intraspecific studies, these networks are referred to as “haplotype networks” (see e.g. Cassens, Mardulyn, and Milinkovitch [20] for more technical definitions concerning biological concepts). Most available methods reconstruct a network directly from the sequences (as defined on page 1), and then apply local heuristics in order to reduce the size or order of the network obtained in that way. The method proposed in the case of intraspecific studies by Cassens et al. [20] is inspired by phylogenetic methods and consists in combining a given set of most parsimonious trees[1] into a graph. They tested their method, which they call “Union of Maximum Parsimonious trees” (or UMP for short), on simulated datasets against other widely-used methods, and noted that UMP provides a good estimate of the true genealogy, sometimes performing better than the other methods. Cassens et al. [20] also provide an algorithm for merging their trees into a graph. However, their algorithm makes a number of arbitrary choices, produces solutions whose quality depends on the order in which the merging process is performed, and is a heuristic with an implicit objective function. Our main contributions in this chapter are the following:

- we propose a possible formal model for UMP in terms of finding a minimum common supergraph of a set of partially labelled trees;
- we give two algorithms for finding an optimal solution to this problem on two graphs: one that runs in polynomial time, provided that at least one of the graphs belongs to a particular class (Proposition 6.3), and another that runs in exponential time in the case where both graphs are arbitrary (Theorem 6.1).

[1] Those trees can be obtained using one of the many methods described e.g. by Felsenstein [34]; in the study by Cassens et al. [20], trees are inferred by a program called PAUP. However, we will only focus on combining a given set of trees into a graph, and not on generating those trees.

6.1 Notation and preliminaries

All graphs considered in this chapter are connected, simple and undirected. We begin with a few definitions and a formal statement of the problem we are going to study, then illustrate all these concepts in Figure 6.1 (page 81).

6.1.1 Partially labelled graphs

Definition 6.1. An (n, k)-*graph* $G = (V, E, \mathcal{L})$, where $V = V_l \cup V_u$ is a graph on $n = |V|$ vertices, k of which are labelled. We distinguish between the set $V_l(G)$ of labelled vertices and the set $V_u(G)$ of unlabelled vertices. The *labelling* $\mathcal{L}$ assigns distinct labels to each vertex in $V_l(G)$; it is called a *partial* labelling if $k < n$ (in which case we say that G is *partially labelled*), and a *complete* labelling if $k = n$ (in which case we say that G is *completely labelled*).

Unless explicitly stated, all (n, k)-graphs will use the label set $\{1, 2, \ldots, k\}$ for labelled vertices.

Definition 6.2. An (n, k)-*tree* is a connected (n, k)-graph with $n - 1$ edges and whose labelled vertex set includes all vertices of degree 1.

For convenience, we will also use the following function, which (possibly) returns the label of vertex v in the (n, k)-graph G:

$$lab : V(G) \to \{1, 2, \ldots, k\} \cup \{\varnothing\} : v \mapsto lab(v) = \begin{cases} i & \text{if } v \text{ has label } i, \\ \varnothing & \text{otherwise.} \end{cases}$$

This is not to be confused with the labellings introduced in Definition 6.1: labelling $\mathcal{L}$ *assigns* labels to vertices, while function *lab* (possibly) *returns* labels. We will also use *lab* on edges, in order to obtain the pairs of labels that correspond to the endpoints of interest: if v, $w \in V_l$, then $lab(\{v, w\}) = \{lab(v), lab(w)\}$. Therefore, we have:

$$\begin{aligned} lab(E(G)) \;=\; & \{\{i, j\} \mid i, j \in \{1, 2, \ldots, k\} \cup \{\varnothing\} \text{ and} \\ & \exists\, v, w \in V(G) : lab(v) = i, lab(w) = j\}. \end{aligned}$$

We intentionally refrain from providing an explicit definition of the image of $lab(\{v, w\})$ for unlabelled vertices, because we will only use that function on pairs of labelled vertices.

Definition 6.3. Given an (n, k)-graph G, we partition the neighbourhood $N^G(v)$ of any vertex v in $V(G)$ into

- the *labelled* neighbourhood $N_l^G(v) = \{w \in N^G(v) \mid lab(w) \neq \varnothing\}$ of v, and
- the *unlabelled* neighbourhood $N_u^G(v) = \{w \in N^G(v) \mid lab(w) = \varnothing\}$ of v.

6.1.2 Subgraphs and supergraphs

We will find it convenient to define the following well-known concepts using labellings. By *completing a labelling*, we mean transforming a partial labelling into a complete labelling.

Definition 6.4. Two (n,k)-graphs G and H are *isomorphic* if their labellings can be completed in such a way that the resulting (n,n)-graphs G' and H' satisfy $lab(E(G')) = lab(E(H'))$. In that case, we write $G \cong H$.

Definition 6.5. An (n,k)-graph G is a *subgraph* of an (n,k)-graph H if the labellings of G and H can be completed in such a way that the resulting (n,n)-graphs G' and H' satisfy $lab(E(G')) \subseteq lab(E(H'))$. In that case, we also say that H is a *supergraph* of G.

Informally speaking, G is a subgraph of H (and H is a supergraph of G) if one can obtain an (n,k)-graph isomorphic to G by removing an appropriate set of edges from H. The problem of finding a subgraph of a given graph that is isomorphic to another given graph is well-known to be NP-complete in the case where no vertex is labelled (see e.g. Garey and Johnson [38]). We adapt the following definition from Bunke, Jiang, and Kandel [17] to our purposes.

Definition 6.6. A *common supergraph* of a set $\{G_1, G_2, \ldots, G_t\}$ of (n,k)-graphs is an (n,k)-graph G that is a supergraph of each G_i (for $1 \leq i \leq t$). It is *minimum* if there is no other graph G' with $|E(G')| < |E(G)|$ that shares this property.

The optimisation problem we want to solve is formally stated below.

MINIMUM COMMON SUPERGRAPH (I)
Instance: (n,k)-trees T_1, T_2, …, T_t on the same label set.
Problem: find a minimum common supergraph of $\{T_1, T_2, \ldots, T_t\}$.

Figure 6.1 shows two (n,k)-trees, along with two other (n,k)-graphs G_1 and G_2: G_2 is a common supergraph of T_1 and T_2, but it is not a minimum one: indeed, G_1 is also a common supergraph of T_1 and T_2, but it has fewer edges than G_2; and G_1 is indeed minimum, since it has only one more edge than T_1 or T_2, which are not isomorphic.

Alternatively, the minimum common supergraph problem on partially labelled trees can be conveniently reformulated using (complete) labellings.

Definition 6.7. The *union* of two (n,n)-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $\mathscr{G} = (V, E)$ whose vertices are labelled by $\{1, 2, \ldots, n\}$ and whose edge set is defined by

$$E = \{\{v,w\} \mid v, w \in V \text{ and } \{lab(v), lab(w)\} \in lab(E_1) \cup lab(E_2)\}.$$

Figure 6.2 shows an example of the union of two completely labelled graphs on eight vertices. Note that we defined and illustrated this concept on two graphs, but it straightforwardly generalises to any number of input graphs. We can now use the concepts of labellings and unions to restate the minimum common supergraph problem on partially labelled trees as follows:

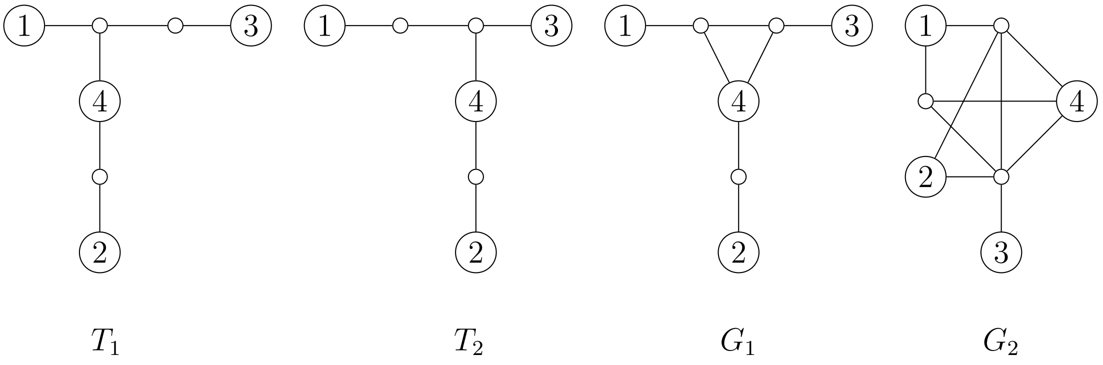


Figure 6.1: Two $(7, 4)$-trees T_1 and T_2, a minimum common supergraph G_1 of T_1 and T_2, and a common supergraph G_2 of T_1 and T_2 which is not minimum. For conciseness, we refrain from displaying the vertices' names (i.e. v_1, v_2, …), which are irrelevant, and only write the labels of labelled vertices.

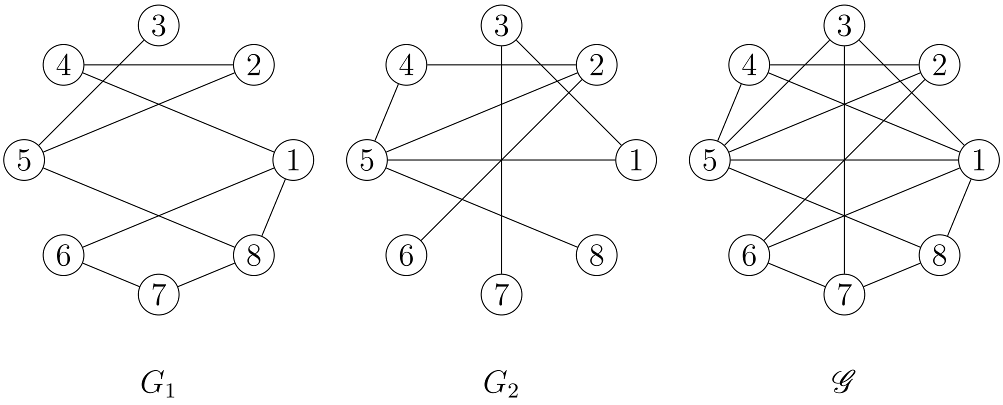


Figure 6.2: The union $\mathscr{G}$ of two completely labelled graphs G_1 and G_2.

MINIMUM COMMON SUPERGRAPH (II)
Instance: (n, k)-trees T_1, T_2, …, T_t on the same label set.
Problem: completely label T_1, T_2, …, T_t in such a way that the union of the resulting completely labelled trees has minimal size.

Recall (Definition 2.3 page 11) that the size of a graph is the number of edges it contains. Labellings that minimise the size of the union of our input trees will be referred to as *optimal*. The first formulation of the minimum common supergraph problem is closer to the actual biological motivation, but the second formulation will be proved more attractive to work with in Section 6.3. Note that a solution to the second formulation trivially provides a solution to the first formulation, obtained by removing the new labels (i.e. labels in $\{k + 1, k + 2, \ldots, n\}$) from the union of our completely labelled trees.

6.2 Previous and related work on minimum common supergraphs

As far as we know, very few results exist on minimum common supergraphs. Moreover, the definition of the problem seems to vary from paper to paper. Bunke et al. [17] define the problem on graphs whose vertices *and* edges can be labelled, and try to minimise a function that takes both the number of vertices and edges into account. They observe that an edit distance can be defined between two graphs, in terms of the minimum number of edge or vertex deletions, insertions or substitutions that need to be applied to transform one graph into the other, and point out a connection with the maximum common subgraph of the two graphs under consideration (i.e. the graph of largest order and size that is a subgraph of both input graphs).

Parker and Lee [63] study the problem of finding a minimum common supergraph of two completely labelled, connected and directed acyclic graphs. They prove that finding a minimum common supergraph in that case is NP-hard, if the function to optimise is the quantity $|V(G)|+c|E(G)|$, where c is a small constant. The problem is however solvable in polynomial time if either input graph is a path.

Finally, a research topic that seems closely related to our concern is the problem of determining *universal graphs for spanning trees*, i.e. graphs on n vertices that contain all nonisomorphic spanning trees on n vertices, with as few edges as possible (in this case, all vertices and edges are unlabelled, and graphs are undirected). This problem was first investigated by Chung and Graham [24], who proved bounds on the number of edges of an optimal solution; there does not seem to be any computational complexity result on the topic.

6.3 The ISOMORPHIC (n, k)-TREE problem

An approach that comes to mind for building a minimum common supergraph of t (n, k)-trees is to remove edges from the complete (n, k)-graph as long as the resulting graph contains all our trees. However, one must make sure that the resulting graph contains each input tree, and checking this is difficult, as we show by a simple transformation from ISOMORPHIC SPANNING TREE (defined below). This shows that the approach described above is not likely to be computationally efficient, and further motivates our liking for formulation (II) of the minimum common supergraph problem, since labellings provide a quickly verifiable proof that each tree is indeed a subgraph of the proposed solution.

ISOMORPHIC SPANNING TREE
Instance: graph $G = (V, E)$, tree $T = (V_T, E_T)$.
Question: does G contain a spanning tree isomorphic to T?

ISOMORPHIC SPANNING TREE is known to be NP-complete (see Garey and Johnson [38], problem ND8 page 207), even in the case where T is a path, in which case the problem comes down to determining whether G is Hamiltonian. We prove, by a simple transformation, that it remains NP-complete in the partially labelled case, as defined below:

ISOMORPHIC (n, k)-TREE
Instance: (n, k)-graph G, (n, k)-tree T on the same label set.
Question: does G contain an (n, k)-tree isomorphic to T?

Proposition 6.1. Isomorphic (n, k)-tree *is NP-complete.*

Proof. That isomorphic (n, k)-tree is in NP is straightforward: a solution to isomorphic (n, k)-tree is a pair of complete labellings for T and G, and we only need to check that $lab(E(T)) \subseteq lab(E(G))$, which is clearly doable in polynomial time.

We now show how isomorphic (n, k)-tree can be used to solve isomorphic spanning tree. Let $\{G = (V, E), T = (V_T, E_T)\}$ be an instance of isomorphic spanning tree, and let l denote the number of leaves of T; we add a set V' of l new vertices to both $V(G)$ and $V(T)$, together with the corresponding labellings $\mathcal{L}_{G'}$ and $\mathcal{L}_{T'}$ (i.e. the labels 1, 2, . . ., l are assigned to vertices in V' in an arbitrary way). Each vertex in V' will be connected to exactly one leaf of T, and to every (unlabelled) vertex of G, thus yielding an instance $\{G' = (V \cup V', E_{G'}, \mathcal{L}_{G'}), T' = (V \cup V', E_{T'}, \mathcal{L}_{T'})\}$ of isomorphic $(|V| + l, l)$-tree. Figure 6.3 illustrates the transformation on a simple example.

Clearly, solutions to isomorphic $(|V| + l, l)$-tree on $\{G', T'\}$ and solutions to isomorphic spanning tree on G and T are in one-to-one correspondence; indeed:

1. if the answer to isomorphic spanning tree on G and T is “yes” and can be verified using labellings $\mathcal{L}_1$ and $\mathcal{L}_2$, then clearly the same labellings can be used to answer isomorphic $(|V| + l, l)$-tree on G' and T' positively, by completing $\mathcal{L}_{G'}$ and $\mathcal{L}_{T'}$ using $\mathcal{L}_1$ and $\mathcal{L}_2$;

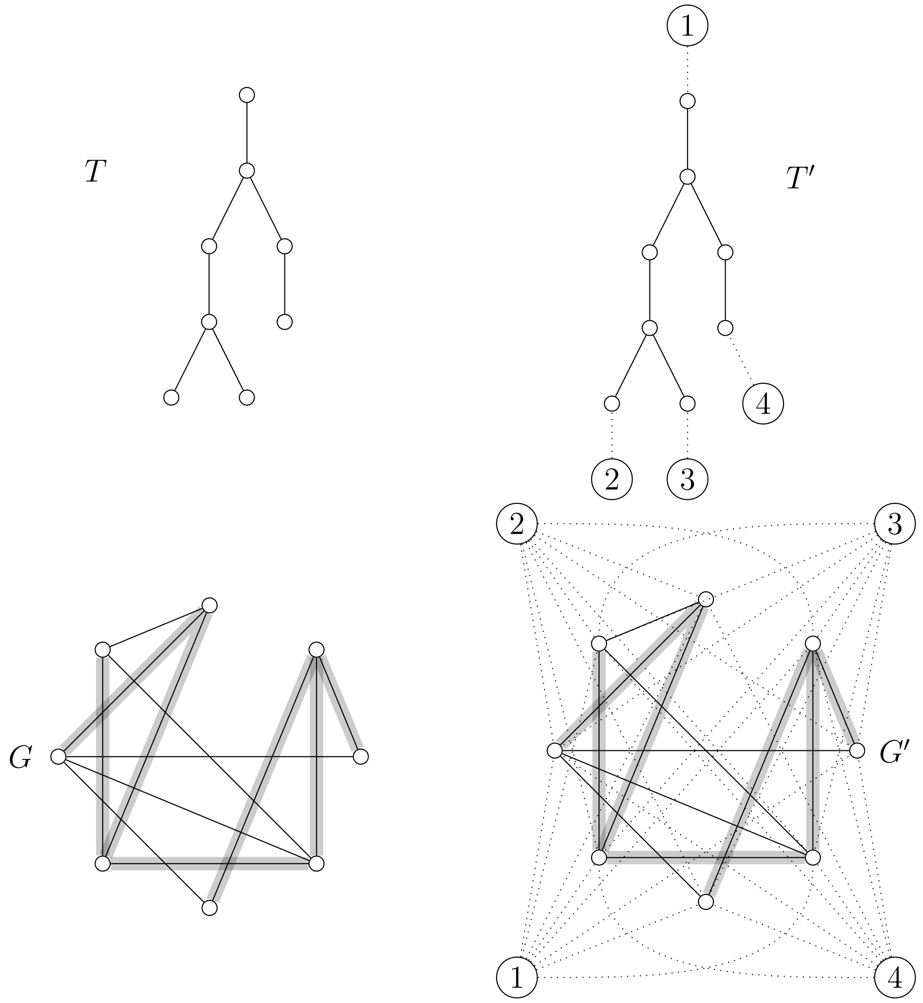
2. on the other hand, if the answer to isomorphic $(|V| + l, l)$-tree on G' and T' is “yes”, then removing all vertices with labels in $\{1, 2, \ldots, l\}$ and all edges incident to those vertices lets us recover G and T as well as a solution to isomorphic spanning tree on G and T, since labellings that certify the positive answer to isomorphic $(|V| + l, l)$-tree on G' and T' are not affected by this transformation.

The transformation is clearly achieved in polynomial time, and this completes the proof. □

As a side remark, we mention that Papadimitriou and Yannakakis [62] prove further computational complexity results on isomorphic spanning tree, and show some cases in which the problem can be solved in polynomial time. Those results might be of interest in the case of isomorphic (n, k)-tree as well, but we will now return to our topic, i.e. the study of the minimum common supergraph problem.

6.4 Polynomial-time solvable cases

We do not know the computational complexity of the minimum common supergraph problem, but are nevertheless able to characterise a few cases in which the problem can be solved in polynomial time. We describe such cases in this section, and give the corresponding algorithms.



instance of
isomorphic spanning tree

instance of
isomorphic $(12, 4)$-tree

Figure 6.3: An example of how to transform an instance of isomorphic spanning tree into an instance of isomorphic (n, k)-tree. Here, the answer to isomorphic spanning tree is "yes", and an occurrence of the input tree in the input graph is highlighted; clearly, the transformation preserves that occurrence. Edges added by the transformation have been dotted for clarity.

6.4.1 Isomorphism of (n, k)-trees

A natural question to answer before trying to find a minimum common supergraph of two (n, k)-trees is whether or not they are isomorphic (this is the special case of ISOMORPHIC (n, k)-TREE where G is also an (n, k)-tree). While the complexity of the general graph isomorphism problem remains open, the problem is known to be solvable in polynomial time for various classes of graphs, including trees (see e.g. Valiente [74], or Aho, Hopcroft, and Ullman [1] for a linear time algorithm). Using ideas similar to those presented by Cassens et al. [20], we show here that it is also the case for (n, k)-trees.

For any two labelled vertices v, w in an (n, k)-graph G, we write $v \overset{G}{\leftrightsquigarrow} w$ if $\{v, w\} \in E(G)$ or if G contains a path between v and w that consists only of unlabelled vertices (except of course for v and w). In the following, T_1 and T_2 will denote two (n, k)-trees; moreover, v, w in T_1 and v', w' in T_2 will denote pairs of labelled vertices with the same labels in T_1 and in T_2, i.e. $\{lab(v), lab(w)\} = \{lab(v'), lab(w')\}$.

Definition 6.8. Let v, w in T_1 and v', w' in T_2 be four labelled vertices such that $v \overset{T_1}{\leftrightsquigarrow} w$ and $v' \overset{T_2}{\leftrightsquigarrow} w'$. We say that the paths between v and w in T_1 and between v' and w' in T_2 *correspond* if:

1. they are of the same length, and
2. the degrees of the unlabelled vertices encountered when traversing the path from v to w (or from w to v) in both trees are equal.

For example, the paths from labelled vertex 2 to labelled vertex 4 in both trees shown in Figure 6.1 correspond, while the paths from labelled vertex 1 to labelled vertex 3 in both trees do not, since the degree sequence (i.e. the size of the neighbourhood of each vertex in the path) of the first path is $(1, 3, 2, 1)$ whereas that of the second path is $(1, 2, 3, 1)$.

Definition 6.9. Let v, w in T_1 and v', w' in T_2 be four labelled vertices such that $v \overset{T_1}{\leftrightsquigarrow} w$ and $v' \overset{T_2}{\leftrightsquigarrow} w'$. The *identification* of the path between v and w in T_1 and between v' and w' in T_2 consists in assigning the same label to unlabelled vertices progressively encountered by traversing both paths in the same direction.

Labels added by the identification process will start with the smallest natural number not already used (i.e. $k + 1$ if T_1 and T_2 are (n, k)-trees). For the above definition to make sense, the paths between vertices v and w in both trees must have the same length (although they need not correspond, in the sense of Definition 6.8).

Observation 6.1. *Let v, w in T_1 and v', w' in T_2 be four labelled vertices such that $v \overset{T_1}{\leftrightsquigarrow} w$ and $v' \overset{T_2}{\leftrightsquigarrow} w'$. If the path between v' and w' in T_2 does not correspond to the path between v and w in T_1, then $T_1 \ncong T_2$.*

Proof. By contradiction, let us assume that either the lengths of the paths differ or the degrees of the unlabelled vertices in both paths do not correspond. Since by definition of a tree, there is a unique path between any two vertices, we have $T_1 \ncong T_2$. □

Observation 6.2. *If the paths between labelled vertices v, w in T_1 and v', w' in T_2 correspond, let T_1' and T_2' be the trees obtained after identifying those paths; then $T_1 \cong T_2 \Leftrightarrow T_1' \cong T_2'$.*

Proof. If $T_1 \cong T_2$, then any isomorphism between T_1 and T_2 must identify the paths between v and w, and therefore $T_1 \cong T_2 \Rightarrow T_1' \cong T_2'$. On the other hand, if $T_1 \ncong T_2$, then identifying the paths between v and w in both trees will not change that property, and we will have $T_1' \ncong T_2'$ as well. □

Observations 6.1 and 6.2 prove the correctness of Algorithm 6.4.1, which tests the isomorphism of two (n, k)-trees. The **while** loop is executed at most $n-k$ times, and the identification of paths takes time at most $n-1$ since a path in an (n, k)-tree cannot exceed that length; therefore, Algorithm 6.4.1 runs in $O(n^2)$ time.

Algorithm 6.4.1 TREE-ISOMORPHISM(T_1, T_2)

Input: two (n, k)-trees T_1 and T_2 on the same label set
Output: **true** if $T_1 \cong T_2$, **false** otherwise

```
while $|V_u(T_1)| > 0$ do
    pick any $v$ in $V_l(T_1)$ with $|N_u^{T_1}(v)| \geq 1$;
    pick any $w$ in $V_l(T_1)$ such that $v \overset{T_1}{\leftrightsquigarrow} w$ and $\{v, w\} \notin E(T_1)$;
    let $v'$ and $w'$ in $V_l(T_2)$ such that $\{lab(v), lab(w)\} = \{lab(v'), lab(w')\}$;
    if $v' \overset{T_2}{\leftrightsquigarrow} w'$ and the paths between $v$ and $w$ in $T_1$ and between $v'$ and $w'$ in $T_2$
    correspond then
        identify those paths {according to Observation 6.2};
    else
        return false {according to Observation 6.1};
    end if
end while
if $lab(E(T_1)) = lab(E(T_2))$ then
    return true;
else
    return false;
end if
```

6.4.2 Restricted graphs

In this section, we show that the minimum common supergraph problem can be solved in polynomial time on two (n, k)-*graphs* that belong to a particular class, which we define below. The reader may be surprised to see those results expressed on graphs rather than trees, since the input of MINIMUM COMMON SUPERGRAPH as we have defined it is a set of trees; however, we see no reason to restrict the presentation of our results to trees, since the strategy we will present solves the problem on graphs with exactly the same time complexity.

Definition 6.10. An (n, k)-graph is *restricted* if it contains no edge between unlabelled vertices.

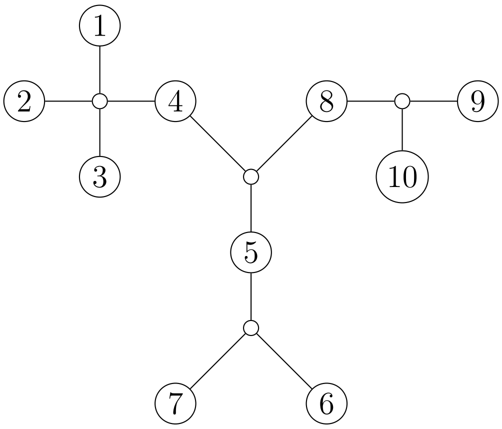


Figure 6.4: A restricted (n,k)-tree.

Figure 6.4 shows an example of a restricted (n,k)-tree. We prove that the minimum common supergraph problem is solvable in polynomial time on two restricted graphs. Figure 6.5, which follows the next result, will illustrate the strategy described in the proof on two restricted $(8,5)$-trees.

Proposition 6.2. *A minimum common supergraph of two restricted (n,k)-graphs G_1 and G_2 can be found in $O((n-k)^3)$ time.*

Proof. Let B be the complete bipartite graph with vertex classes $V_1 = V_u(G_1)$, $V_2 = V_u(G_2)$ and whose edges are weighted by the following function:

$$f(v_1,v_2) = \left|lab(N_l^{G_1}(v_1)) \cup lab(N_l^{G_2}(v_2))\right|, \text{ for } v_1 \in V_u(G_1) \text{ and } v_2 \in V_u(G_2).$$

Every perfect matching $\mathcal{M}$ of B naturally yields two labellings $\mathcal{L}_1^{\mathcal{M}}$ and $\mathcal{L}_2^{\mathcal{M}}$ for G_1 and G_2: if $e_1 = \{v_1, w_1\}$, $e_2 = \{v_2, w_2\}$, …, $e_{n-k} = \{v_{n-k}, w_{n-k}\}$ are the edges of $\mathcal{M}$, then we assign label $i+k$ to $v_i \in V_u(G_1)$ and $w_i \in V_u(G_2)$.

We claim that the minimum common supergraph problem on G_1 and G_2 can be solved by finding a perfect matching of minimum weight in B. For a perfect matching $\mathcal{M}$, we set

$$\begin{aligned} ||\mathcal{M}|| = \sum_{\{v,w\}\in\mathcal{M}} f(v,w) &= \sum_{\{v,w\}\in\mathcal{M}} \left|lab(N_l^{G_1}(v)) \cup lab(N_l^{G_2}(w))\right| \\ &= \left|lab(E(G_1^{\mathcal{M}})) \cup lab(E(G_2^{\mathcal{M}}))\right| - C, \end{aligned}$$

where $G_1^{\mathcal{M}}$ and $G_2^{\mathcal{M}}$ are the completely labelled graphs obtained by applying the labellings given by $\mathcal{M}$, and C is a constant that counts the number of edges in the resulting graph whose both endpoints belong to $V_l(G_1)$ (or $V_l(G_2)$). The last equality follows from Definition 6.10. By definition, if $\mathcal{M}^*$ is a minimum weight perfect matching, then:

$$\begin{aligned} ||\mathcal{M}^*|| &= \min_{\text{all matchings } \mathcal{M}} ||\mathcal{M}|| \\ &= \min_{\text{all matchings } \mathcal{M}} \left|lab(E(G_1^{\mathcal{M}})) \cup lab(E(G_2^{\mathcal{M}}))\right| - C. \end{aligned}$$

Since the latter function is the objective function of MINIMUM COMMON SUPERGRAPH (II), and since an optimal matching can be found in time cubic in the number of vertices of B (see e.g. Schrijver [67]), the proof is complete. □

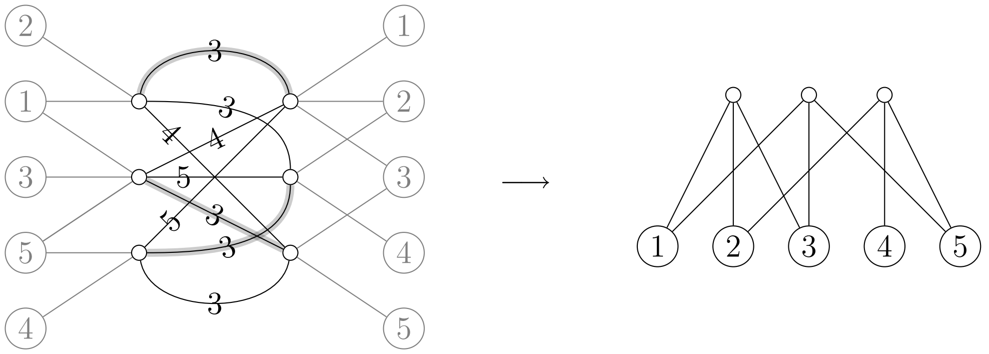


Figure 6.5: Optimally merging two restricted $(8, 5)$-trees; a minimum weight perfect matching of weight 9 yields a minimum common supergraph of those trees that contains 9 edges.

6.4.3 The minimum common supergraph problem on a restricted (n, k)-graph and an arbitrary (n, k)-graph

The special case solved in Section 6.4.2 may seem of little practical interest; but it turns out that the strategy used in Proposition 6.2 can be extended to handle the case where only one of the two graphs to merge is restricted. Furthermore, restricted graphs will prove crucial in the design of an exact algorithm for finding a minimum common supergraph of two arbitrary (n, k)-graphs (see Section 6.5).

Proposition 6.3. *A minimum common supergraph of two (n, k)-graphs G_1 and G_2 can be found in $O((n-k)^3)$ time, provided that G_1 or G_2 is restricted.*

Proof. We have already handled the case where both input graphs are restricted (Proposition 6.2), so we assume that only G_1 is. In that case, edges connecting unlabelled vertices in G_2 can actually be ignored: indeed, there are no such edges in G_1, so whatever assignment is chosen, the number of such edges in the resulting common supergraph will be equal to their number in G_2, a constant. We can therefore only reduce the number of edges connecting labelled vertices to unlabelled ones, which we do using again the matching strategy of Proposition 6.2 with the very same weight function. □

Algorithm 6.4.2 optimally merges a restricted (n, k)-graph and an arbitrary (n, k)-graph in $O((n-k)^3)$ time.

```
Algorithm 6.4.2 MERGE-RESTRICTED-AND-ARBITRARY-GRAPHS($G_1, G_2$)
Input: a restricted $(n, k)$-graph $G_1$ and an arbitrary $(n, k)$-graph $G_2$.
Output: two optimal labellings for $G_1$ and $G_2$.
build the complete bipartite graph $B$ with vertex sets $V_1 = V_u(G_1)$, $V_2 = V_u(G_2)$
and whose edges are weighted by the following function:
    $f(v_1, v_2) = \left|lab(N_l^{G_1}(v_1)) \cup lab(N_l^{G_2}(v_2))\right|$, for $v_1 \in V_u(G_1)$ and $v_2 \in V_u(G_2)$.
find a minimum weight perfect matching in $B$;
for each edge $e_i = \{v_1, v_2\}$ of the matching do
    assign label $i + k$ to $v_1 \in V_u(G_1)$ and $v_2 \in V_u(G_2)$;
end for
return both complete labellings;
```

6.5 An exact algorithm for two graphs

As illustrated in Figure 6.6, the matching strategy on which Algorithm 6.4.2 is based no longer works for merging two arbitrary trees. In this section, we propose an exact exponential time algorithm for solving the minimum common supergraph problem on two arbitrary partially labelled graphs. For the sake of clarity, we defer its formal exposition to the end of this section, and begin with a brief top-down sketch of its design.

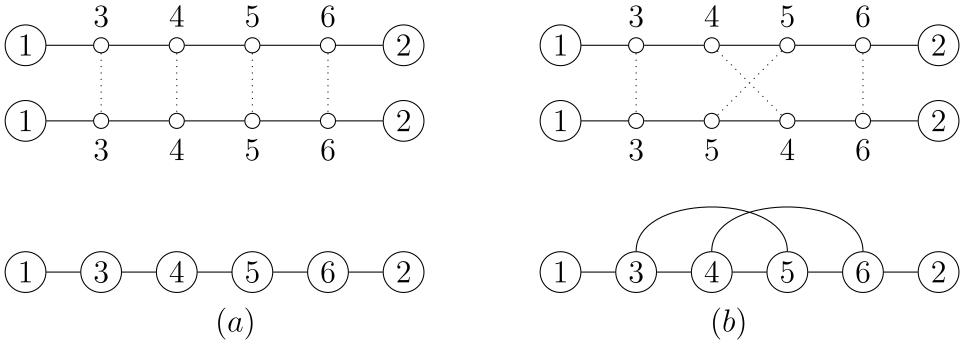


Figure 6.6: Example on which the matching strategy for finding a minimum common supergraph fails: labellings in situations (a) and (b) correspond to two matchings of the same weight (sketched using dotted edges), but solution (a) is optimal while solution (b) is not.

6.5.1 Outline and computational complexity

Let p denote the smallest integer such that G_1 or G_2 (possibly both) can be transformed into a restricted $(n, k+p)$-graph by labelling a subset of p unlabelled vertices (we will give an algorithm for achieving this task in Section 6.5.2). Without loss of generality, we assume that we have been able to pick p vertices in G_1 in such a way that after arbitrarily labelling those vertices using labels in $\{k + 1, k + 2, \ldots,$

$k + p\}$, we obtain a restricted $(n, k + p)$-graph G'_1. Then, we explore all possible ways to select and label p unlabelled vertices in G_2, so as to obtain an (arbitrary) $(n, k+p)$-graph G'_2. Algorithm 6.4.2 is then used to compute in polynomial time, for each such pair of graphs, a minimum common supergraph of G'_1 and G'_2, and we keep the common supergraph with fewest edges over all minimum common supergraphs generated in that way.

6.5.2 Restricting (n, k)-graphs

The algorithm we depicted in Section 6.5.1 relies on our ability to compute p and find the corresponding subset of vertices in an (n, k)-graph. Lemma 6.1 below explains how to achieve this on (n, k)-graphs in general, and relies on the following well-known concept.

Definition 6.11. A *vertex cover* of a graph G is a subset of vertices $U \subseteq V(G)$ such that for every edge $\{v, w\}$ in $E(G)$, we have $v \in U$ or $w \in U$.

Given an (n, k)-graph G, denote G^u the graph obtained from G by removing all edges incident to vertices of $V_l(G)$.

Lemma 6.1. *Given an (n, k)-graph G, let $S \subseteq V_u(G)$; labelling all elements of S transforms G into a restricted $(n, k + |S|)$-graph if and only if S is a vertex cover of G^u.*

Proof. Let C be a vertex cover of G^u. Then labelling all elements of C transforms G^u into a restricted graph, a property that is obviously preserved when reinserting all edges incident to the vertices of $V_l(G)$. Conversely, let $X \subseteq V_u(G)$ be such that labelling all elements of X yields a restricted $(n, k+|X|)$-graph H; by definition of a restricted graph, every edge of $E(H)$ has at least a labelled endpoint, which is also the case for every edge of $E(H^u)$, and X is therefore a vertex cover of G^u. □

Restricting an (n, k)-graph G by labelling as few vertices as possible is therefore equivalent to finding a minimum vertex cover, i.e. a vertex cover of minimum cardinality, on G^u. Finding a minimum vertex cover is well-known to be NP-hard (see Garey and Johnson [38], problem GT1 page 190), but it is solvable in linear time on trees, among other particular cases (see e.g. Skiena [68]). Figure 6.7 shows how a tree is transformed into a restricted tree.

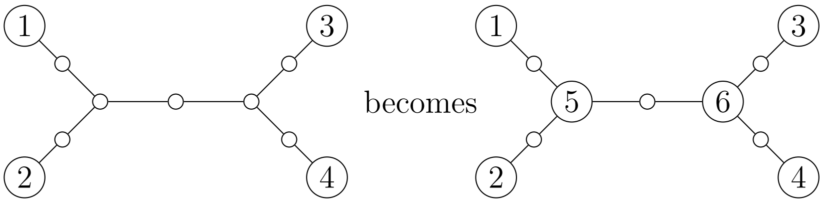


Figure 6.7: Transforming an $(11, 4)$-tree T into a restricted $(11, 6)$-tree T'. The labelled vertices of T' do not constitute a minimum vertex cover of T', but vertices with labels 5 and 6 do constitute a minimum vertex cover of T^u.

6.5.3 Pruning the search tree

There are two levels at which one wishes to prune the search tree explored by our algorithm (for simplicity, we assume that G_1 is the graph that has been transformed into a restricted graph):

1. when selecting a particular subset of vertices of G_2 to which labels are to be assigned, and

2. when permuting the labels assigned to a given subset of vertices of G_2.

The following two observations provide lower bounds that can be used in those two situations. Let us partition the edge set of an (n, k)-graph G into $E(G) = E_0(G) \cup E_1(G) \cup E_2(G)$, where for $0 \leq i \leq 2$, $E_i(G)$ stands for the subset of edges of $E(G)$ that have i labelled endpoints (note that if G is restricted, then $E_0(G)$ is empty). We have the following.

Observation 6.3. *For any common supergraph H of two (n, k)-graphs G_1 and G_2, where G_1 is restricted and G_2 is arbitrary, we have*

$$|E(H)| \geq |E(G_1)| + |E_0(G_2)| + |lab(E_2(G_2)) \setminus lab(E_2(G_1))|. \tag{6.1}$$

Proof. We start building a common supergraph of our two graphs by adding edges to the edges of G_1. Since G_1 is restricted, $E_0(G_1)$ is empty, so we will have to add at least $|E_0(G_2)|$ edges to G_1. Moreover, edges between two labelled vertices in G_2 which do not already appear in G_1 will also have to be added, and there are $|lab(E_2(G_2)) \setminus lab(E_2(G_1))|$ of them. □

Figure 6.8 illustrates the computation of the above lower bound on a simple example. Lower bound (6.1) will be used in the second situation described above, i.e. when assigning labels to a fixed subset of previously unlabelled vertices in G_2. Similar arguments can be used to obtain another lower bound which can be used *before* assigning labels to the given subset, as we explain below.

Observation 6.4. *Let G_1 be a restricted (n, k_1)-graph and G_2 be an arbitrary (n, k_2)-graph, with $k_2 \leq k_1$, and let $S \subseteq V_u(G_2)$ with $|S| = k_1 - k_2$; then for any common supergraph H of G_1 and G_2, we have*

$$|E(H)| \geq |E(G_1)| + |E'_0(G_2)| + |lab(E_2(G_2)) \setminus lab(E_2(G_1))|, \tag{6.2}$$

where $|E'_0(G_2)| = |E_0(G_2)| - |\{\{v, w\} \in E_0(G_2) \mid v \text{ or } w \in S\}|$.

Proof. As in the proof of Observation 6.3, we will be adding edges to the $n - 1$ edges of G_1. Since all vertices in S are going to be labelled, we will be adding only $|E'_0(G_2)| = |E_0(G_2)| - |\{\{v, w\} \in E_0(G_2) \mid v \text{ or } w \in S\}|$ edges to G_1. □

Algorithm 6.5.1 combines all previous results to solve the minimum common supergraph problem on two arbitrary (n, k)-graphs.

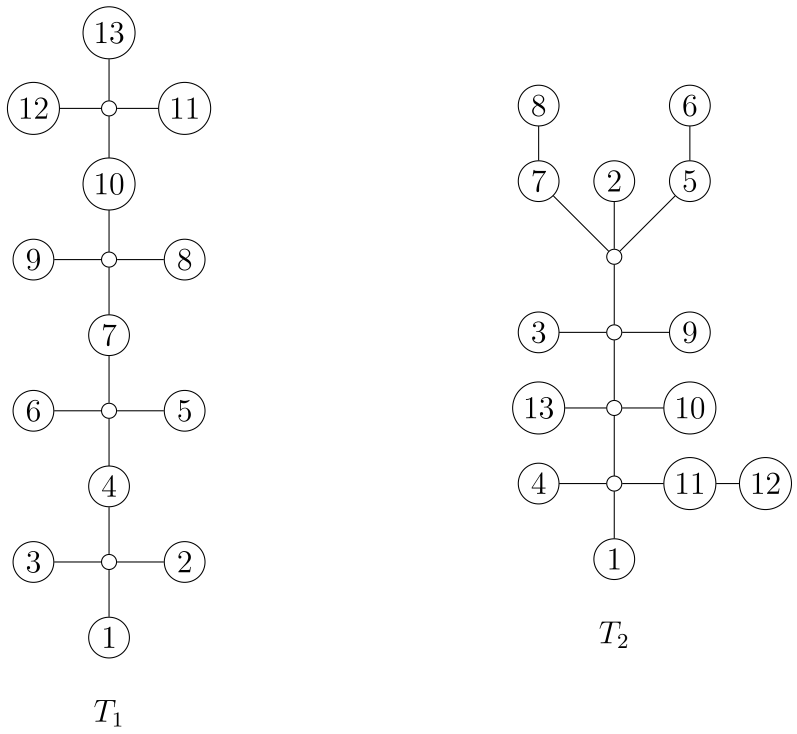


Figure 6.8: Two $(17, 13)$-trees T_1 and T_2 (T_1 is restricted). We have $|E(T_1)| = 16$, $|E_0(T_2)| = 3$, and none of the 3 edges of $E_2(T_2)$ appears in T_1; therefore, any common supergraph of T_1 and T_2 will have at least $16 + 3 + 3 = 22$ edges.

Algorithm 6.5.1 MERGE-TWO-GRAPHS(G_1, G_2)

Input: two (n, k)-graphs G_1 and G_2 on the same label set
Output: a minimum common supergraph of G_1 and G_2

```
G ← ∅;
b ← ∞;
if G_1 or G_2 is restricted then
    G ← MERGE-RESTRICTED-AND-ARBITRARY-GRAPHS(G_1, G_2);
else
    p_1 ← minimum vertex cover on G_1^u;
    p_2 ← minimum vertex cover on G_2^u;
    if |p_1| > |p_2| then
        swap G_1 and G_2;
        swap p_1 and p_2;
    end if
    G'_1 ← the graph obtained from G_1 by labelling the vertices of p_1;
    for all subset S ⊆ V_u(G_2) of size |p_1| do
        G'_2 ← G_2;
        if lower bound (6.2) on G'_1, G'_2, S is strictly less than b then
            arbitrarily label the elements of S;
            for all permutations of the labels assigned to the current subset do
                G'_2 ← the graph obtained by labelling the elements of S using the
                    current permutation;
                if lower bound (6.1) on G'_1, G'_2 is strictly less than b then
                    H ← MERGE-RESTRICTED-AND-ARBITRARY-GRAPHS (G'_1,
                        G'_2);
                    if |E(H)| < b then
                        G ← H;
                        b ← |E(G)|;
                    end if
                end if
            end for
        end if
    end for
end if
return G;
```

6.5.4 Complexity analysis

Algorithm 6.5.1 on two (n, k)-trees has the following time complexity.

Theorem 6.1. *A minimum common supergraph of two arbitrary (n, k)-trees can be computed in $O\left((n-k)^{p+3}\right)$ time, where p is the smallest integer such that either tree, after arbitrarily labelling p of its unlabelled vertices, becomes a restricted $(n, k+p)$-tree.*

Proof. Let us assume, without loss of generality, that T_1 becomes restricted after labelling p of its vertices; the number of ways in which we can select and label p vertices out of $n-k$ in T_2 as candidates to be identified with our p selected vertices in T_1 is the number of injective mappings from the set of p vertices in T_1 to the set of $n-k$ unlabelled vertices in T_2, which is $(n-k)(n-k-1)\cdots(n-k-p+1)$. Since we are, at each step, comparing $(n, k+p)$-graphs, and since an optimal solution to the minimum common supergraph problem on a restricted (n, k)-graph and an arbitrary (n, k)-graph can be obtained in $O((n-k)^3)$ time (Proposition 6.3), the proof follows from the fact that

$$\underbrace{(n-k)(n-k-1)\cdots(n-k-p+1)}_{p\ terms}(n-k-p)^3 \leq (n-k)^{p+3}.$$

□

Note that the computation of the lower bounds presented in Section 6.5.3 can be achieved in linear time. As we have previously stressed, Algorithm 6.5.1 finds an optimal solution on *graphs* rather than trees; however, the computational complexity of the algorithm may increase beyond $O((n-k)^{p+3})$, since restricting an (n, k)-graph is likely to take much more time than restricting a tree, because of the equivalence of this problem with that of finding a minimum vertex cover (Lemma 6.1).

Conclusions

A few problems we have discussed in this dissertation remain open, and some of the results we have obtained might be extendable in a few ways; we outline and suggest below a few possible directions for further research.

Sorting by transpositions. Despite tremendous efforts by many researchers, determining the complexity of sorting by transpositions and computing the transposition distance or its maximal value remain open problems. We have shown how the classical disjoint cycle decomposition of permutations can be used to compute in polynomial time the transposition distance of permutations belonging to a few nontrivial classes and to obtain upper bounds on the transposition distance of other permutations. Nevertheless, there are still permutations for which there is a gap between our bounds and the actual distance: it would be interesting to know whether or not our approach has reached its limits, and to get more information about how efficient it is (we conducted experimental tests, but were unable to prove an approximation guarantee). Can our approach be applied to other genome rearrangement problems, and are there other alternatives to the breakpoint graph approach that would be of interest? A careful analysis of our improved upper bounds may also lead to some progress on determining the value of the transposition diameter. Finally, an intriguing question, which has been open since the first paper on the subject by Bafna and Pevzner [8], is whether -2-transpositions, i.e. transpositions which *remove* two cycles from the cycle graph, are useless. It seems intuitive that they could be safely disregarded, since they seem to take us farther away from the identity permutation, but there is no proof that every permutation admits an optimal sorting sequence using no -2-transposition.

Hultman numbers. In Chapter 4, we enumerated permutations that belong to a particular "Hultman class", and those whose cycle graph contains a given number of alternating cycles. It would be interesting to obtain analogous results for *signed* permutations, since this would immediately characterise, among others, the distribution of the *double cut-and-join distance* (see Yancopoulos, Attie, and Friedberg [76] for definitions), which has become increasingly popular in genome comparisons. Obtaining closed, or at least simpler expressions for the formulae we have obtained, in particular those that count simple permutations and 3-permutations (Proposition 4.2), is also an open problem.

Edit distances in general. We provided a new general framework in Chapter 5, which connects edit distance problems on permutations with factorisations of related even permutations. This framework proved useful in recovering known lower bounds and proving new ones (see Sections 5.5 and 5.6), and there are at least two interesting directions for new research. First, is it possible to obtain *upper bounds* using this approach? Second, can this framework be extended to other objects than permutations (e.g. signed permutations, strings, or other completely different combinatorial structures)? In particular, the breakpoint graph of *signed* permutations admits a unique decomposition into disjoint alternating cycles, which suggests that it might be possible to use the disjoint cycle decomposition of some (signed) permutation to express the structure of this graph.

Interconnection networks. As we have seen in Chapter 5, Cayley graphs of permutation groups have received a lot of attention from researchers over the last two decades in the context of interconnection networks. However, Cayley graphs based on generating sets used in genome rearrangement problems do not seem to have received any attention. This is quite surprising since those two fields seem to have a few goals in common, even though the motivations are different. The study of Cayley graphs from the genome rearrangement literature might be useful in many aspects: it may yield new information and results on genome rearrangement problems, or reveal that these graphs are good candidates for use as interconnection networks, or even allow to solve open problems in the latter field.

Minimum common supergraphs. A lot of work remains to be done on the MINIMUM COMMON SUPERGRAPH problem, starting with the determination of its complexity (on two and on more than two trees). It also seems challenging to come up with exact, approximation or heuristic algorithms that would be competitive in practice (i.e. both fast and accurate).

Glossary

Notation	**Description**	
$\mathcal{M}$	A matching	12
$(i_1, i_2, \ldots, i_k)$	A k-cycle in the disjoint cycle decomposition of a permutation	13
A_n	The alternating group on $\{1, 2, \ldots, n\}$	13
$BG(\pi)$	The breakpoint graph of permutation π	47
$C(\pi)$	Conjugacy class of permutation π	51
$C_H(\pi)$	Hultman class of permutation π	52
$\mathcal{C}_i$	A connected component of the contact graph of a permutation	44
$D(n, k)$	Number of factorisations of an n-cycle into the product of an n-cycle and a permutation with k cycles	54
$E(G)$	The edge set of graph G	11
$G'(\pi)$	The circular cycle graph of permutation π	51
$G(\pi)$	The cycle graph of permutation π	20
$G = (V, E)$	A graph G with vertex set V and edge set E	11
$G = (V, E, \mathcal{L})$	A partially labelled graph	79
G^u	The (n, k)-graph obtained from (n, k)-graph G by removing all edges incident to its labelled vertices	90
$H(\pi)$	The contact graph of permutation π	44
$\mathcal{L}$	A labelling	79
$N^G(v)$	The neighbourhood of vertex v in graph G	12
$N^G_l(v)$	The labelled neighbourhood of vertex v in (n, k)-graph G	79
$N^G_u(v)$	The unlabelled neighbourhood of vertex v in (n, k)-graph G	79
$S(1 + n)$	Shorthand notation for $\mathrm{Sym}(\{0, 1, 2, \ldots, n\})$	50
$\mathcal{S}(n, k)$	Stirling number of the first kind	51
S_n	The symmetric group on $\{1, 2, \ldots, n\}$	12
$\mathcal{S}_H(n, k)$	Hultman number	51
$V(G)$	The vertex set of graph G	11
$V_l(G)$	The set of labelled vertices of partially labelled graph G	79
$V_u(G)$	The set of unlabelled vertices of partially labelled graph G	79
$[n]$	The set $\{1, 2, \ldots, n\}$	12
$\Gamma(\pi)$	The Γ-graph of permutation π	28
$\Upsilon(\pi)$	The set of all clans of permutation π of length at least 3	67
$\beta(i, j, k, l)$	A block-interchange	59
χ	The reversed permutation $\langle n\ n-1\ n-2\ \cdots\ 3\ 2\ 1\rangle$	27

Notation	**Description**	
$\circ$	Composition (or multiplication) operator for permutations, applied from right to left	12
$\cong$	Isomorphism of (n,k)-graphs	79
$\equiv_{\circ}^{\circ}$	Equivalence by torism	24
$\equiv_r$	Equivalence by reduction	23
ι	The identity permutation	12
$\langle \pi_1\ \pi_2\ \cdots\ \pi_n \rangle$	A permutation of n elements, written as a sequence	12
$\lvert\ LIS(\pi)\ \rvert$	The length of a longest increasing subsequence of π	26
$\models$	A composition	54
$\overline{\pi}$	The permutation $(0, \pi_n, \pi_{n-1}, \ldots, \pi_1) \circ (0, 1, 2, \ldots, n)$	53
$\overline{x}^m$	Shorthand notation for $(x+m) \pmod{n+1}$	24
$\pi_{\circ}^{\circ}$	A toric permutation	24
π°	A circular permutation	23
π^{-1}	The inverse of permutation π	13
π^{σ}	The conjugate of permutation π by permutation σ	14
$\dot{\pi}$	Shorthand notation for $(0, \pi_n, \pi_{n-1}, \ldots, \pi_1)$	53
π	A permutation	12
$\tau(i,j,k)$	A (biological) transposition	19
$\varepsilon(i,j)$	An exchange (or algebraic transposition)	32
$\vdash$	Partition of a natural number	51
$\widetilde{\pi}$	The extended version of permutation π	22
ξ_k	The permutation $\langle \underbrace{3\ 2\ 1}_{1}\ \underbrace{6\ 5\ 4}_{2}\ \cdots\ \underbrace{n\ n-1\ n-2}_{k} \rangle$	44
$b(\pi)$	The number of breakpoints of permutation π	22
$bid(\pi)$	The block-interchange distance of permutation π	61
$c(G(\pi))$	The number of alternating cycles in $G(\pi)$	20
$c(\Gamma(\pi))$	The number of cycles in $\Gamma(\pi)$	28
$c_{\lambda\mu}^{(n)}$	The number of ways to express a given n-cycle as the product of two permutations whose classes are given respectively by partitions λ and μ	54
$c_1(\Gamma(\pi))$	The number of 1-cycles in $\Gamma(\pi)$, or equivalently the number of fixed points of π	66
$c_{even}(G(\pi))$	The number of even alternating cycles in $G(\pi)$	20
$c_{even}(\Gamma(\pi))$	The number of even cycles in $\Gamma(\pi)$	28
$c_{odd}(G(\pi))$	The number of odd alternating cycles in $G(\pi)$	20
$c_{odd}(\Gamma(\pi))$	The number of odd cycles in $\Gamma(\pi)$	28
$d_3^1(\pi)$	The length of a minimum factorisation of permutation π into a product of 3-cycles, each of which contains the first element	72
$even(\Gamma(\pi))$	The set of even cycles of $\Gamma(\pi)$	34
$gl(\pi)$	The reduced permutation obtained from permutation π	23
$lab(v)$	The label of vertex v in a given (n,k)-graph	79
$odd(\Gamma(\pi))$	The set of odd cycles of $\Gamma(\pi)$	34
$pexc(\pi)$	The prefix exchange distance of permutation π (with respect to ι)	66

Notation	**Description**	
$pexc(\pi, \sigma)$	The prefix exchange distance between permutations π and σ	66
$ptb(\pi)$	The number of prefix transposition breakpoints of permutation π	66
$ptd(\pi)$	The prefix transposition distance of permutation π (with respect to ι)	66
$ptd(\pi, \sigma)$	The prefix transposition distance between permutations π and σ	66
$s(\pi)$	The number of strips of permutation π	67
$td(\pi)$	The transposition distance of permutation π (with respect to ι)	20
$td(\pi, \sigma)$	The transposition distance between permutations π and σ	19
$v \overset{G}{\leftrightsquigarrow} w$	Labelled vertices v and w in (n, k)-graph G are either adjacent or connected by a path that consists only of unlabelled vertices.	85
z_λ	Shorthand notation for $\prod_i \alpha_i! i^{\alpha_i}$	51

Index

Bibliography

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] S. B. Akers and B. Krishnamurthy, *A group-theoretic model for symmetric interconnection networks*, IEEE Transactions on Computers, 38 (1989), pp. 555–566.

[3] S. B. Akers, B. Krishnamurthy, and D. Harel, *The star graph: An attractive alternative to the n-cube*, in Proceedings of the Fourth International Conference on Parallel Processing (ICPP), Pennsylvania State University Press, Aug. 1987, pp. 393–400.

[4] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, Garland Publishing, 4th ed., 2002.

[5] D. Anxolabéhère, D. Nouaud, H. Quesneville, and S. Ronserray, *Transposons: des gènes anarchistes?*, Pour la Science, 351 (2007), pp. 82–89.

[6] D. A. Bader, B. M. E. Moret, and M. Yan, *A linear-time algorithm for computing inversion distance between signed permutations with an experimental study*, Journal of Computational Biology, 8 (2001), pp. 483–491.

[7] V. Bafna and P. A. Pevzner, *Genome rearrangements and sorting by reversals*, in Proceedings of the Thirty-Fourth Annual Symposium on Foundations of Computer Science (FOCS), Palo Alto, Los Alamitos, CA, 1993, ACM/SIAM, pp. 148–157.

[8] V. Bafna and P. A. Pevzner, *Sorting permutations by transpositions*, in Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, Jan. 1995, ACM/SIAM, pp. 614–623.

[9] V. Bafna and P. A. Pevzner, *Sorting by transpositions*, SIAM Journal on Discrete Mathematics, 11 (1998), pp. 224–240 (electronic).

[10] H.-J. Bandelt, P. Forster, and A. Rohl, *Median-joining networks for inferring intraspecific phylogenies*, Molecular Biology and Evolution, 16 (1999), pp. 37–48.

[11] C. Berge, *Graphes et hypergraphes*, Dunod, Paris, 1970. Monographies Universitaires de Mathématiques, No. 37.

[12] P. Berman and M. Karpinski, *On some tighter inapproximability results (extended abstract)*, in Proceedings of the Twenty-Sixth International Colloquium on Automata, Languages and Programming (ICALP), J. Wiedermann, P. van Emde Boas, and M. Nielsen, eds., vol. 1644 of Lecture Notes in Computer Science, Berlin, July 1999, Springer-Verlag, pp. 200–209.

[13] P. Berman, S. Hannenhalli, and M. Karpinski, 1.375*-approximation algorithm for sorting by reversals*, in Proceedings of the Tenth Annual European Symposium on Algorithms (ESA), R. H. Möhring and R. Raman, eds., vol. 2461 of Lecture Notes in Computer Science, Rome, Italy, Sept. 2002, Springer-Verlag, pp. 200–210.

[14] G. Boccara, *Nombre de représentations d'une permutation comme produit de deux cycles de longueurs données*, Discrete Mathematics, 29 (1980), pp. 105–134.

[15] M. Bóna, *Combinatorics of permutations*, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2004. With a foreword by Richard Stanley.

[16] D. Bryant, V. Moulton, and A. Spillner, *Consistency of the neighbor-net algorithm*, Algorithms for Molecular Biology, 2 (2007), p. 8.

[17] H. Bunke, X. Jiang, and A. Kandel, *On the minimum common supergraph of two graphs*, Computing, 65 (2000), pp. 13–25.

[18] A. Caprara, *Sorting permutations by reversals and eulerian cycle decompositions*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 91–110 (electronic).

[19] A. Caprara, G. Lancia, and S.-K. Ng, *Sorting permutations by reversals through branch-and-price*, INFORMS Journal on Computing, 13 (2001), pp. 224–244.

[20] I. Cassens, P. Mardulyn, and M. C. Milinkovitch, *Evaluating intraspecific "network" construction methods using simulated sequence data: Do existing algorithms outperform the global maximum parsimony approach?*, Systematic Biology, 54 (2005), pp. 363–372.

[21] B. Chitturi and I. H. Sudborough, *Bounding prefix transposition distance for strings and permutations*, in Proceedings of the Forty-First Annual Hawaii International Conference on System Sciences (HICSS), Los Alamitos, CA, USA, Jan. 2008, IEEE Computer Society Press, p. 468.

[22] D. A. Christie, *Sorting permutations by block-interchanges*, Information Processing Letters, 60 (1996), pp. 165–169.

[23] D. A. Christie, *Genome Rearrangement Problems*, PhD thesis, University of Glasgow, Scotland, Aug. 1998.

[24] F. R. K. Chung and R. L. Graham, *On universal graphs*, Annals of the New York Academy of Sciences, 319 (1979), pp. 136–140.

[25] C. Darwin, *On the Origin of Species*, John Murray, 1859.

[26] Z. Dias and J. Meidanis, *Sorting by prefix transpositions*, in Proceedings of the Ninth International Symposium on String Processing and Information Retrieval (SPIRE), A. H. F. Laender and A. L. Oliveira, eds., vol. 2476 of Lecture Notes in Computer Science, Lisbon, Portugal, Sept. 2002, Springer-Verlag, pp. 65–76.

[27] Z. Dias, J. Meidanis, and M. E. M. T. Walter, *A new approach for approximating the transposition distance*, in Proceedings of the Seventh International Symposium on String Processing and Information Retrieval (SPIRE), A Coruña, Spain, Sept. 2000, IEEE Computer Society Press, pp. 199–208.

[28] R. Diestel, *Graph theory*, vol. 173 of Graduate Texts in Mathematics, Springer-Verlag, Berlin, 3rd ed., 2005.

[29] J.-P. Doignon and A. Labarre, *On Hultman numbers*, Journal of Integer Sequences, 10 (2007). Article 07.6.2, 13 pages.

[30] I. Elias and T. Hartman, *A* 1.375*-approximation algorithm for sorting by transpositions*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 3 (2006), pp. 369–379.

[31] H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, and J. Wästlund, *Sorting a bridge hand*, Discrete Mathematics, 241 (2001), pp. 289–300. Selected papers in honor of Helge Tverberg.

[32] S. Even and O. Goldreich, *The minimum-length generator sequence problem is NP-hard*, Journal of Algorithms, 2 (1981), pp. 311–313.

[33] L. Excoffier and P. E. Smouse, *Using allele frequencies and geographic subdivision to reconstruct gene trees within a species: Molecular variance parsimony*, Genetics, 136 (1994), pp. 343–359.

[34] J. Felsenstein, *Inferring Phylogenies*, Sinauer Associates, Sunderland, MA, 2004.

[35] J. Feng and D. Zhu, *Faster algorithms for sorting by transpositions and sorting by block interchanges*, ACM Transactions on Algorithms, 3 (2007), pp. 1–14.

[36] G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette, *Combinatorics of Genome Rearrangements*, Computational Molecular Biology, MIT Press, 2009. To appear.

[37] V. J. Fortuna, *Distâncias de transposição entre genomas*, Master's thesis, Universidade Estadual de Campinas, São Paulo, Brazil, Mar. 2005.

[38] M. R. Garey and D. S. Johnson, *Computers and intractability*, W. H. Freeman and Co., San Francisco, California, 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[39] O. Gascuel and M. Steel, eds., *Reconstructing Evolution: New Mathematical and Computational Advances*, Oxford University Press, Jan. 2007.

[40] W. H. Gates and C. H. Papadimitriou, *Bounds for sorting by prefix reversal*, Discrete Mathematics, 27 (1979), pp. 47–57.

[41] A. Goupil, *On products of conjugacy classes of the symmetric group*, Discrete Mathematics, 79 (1989/90), pp. 49–57.

[42] A. Goupil and G. Schaeffer, *Factoring n-cycles and counting maps of given genus*, European Journal of Combinatorics, 19 (1998), pp. 819–834.

[43] S. A. Guyer, L. S. Heath, and J. P. Vergara, *Subsequence and run heuristics for sorting by transpositions*, in Fourth DIMACS Algorithm Implementation Challenge, Rutgers University, Aug. 1995.

[44] S. Hannenhalli and P. A. Pevzner, *Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals*, Journal of the ACM, 46 (1999), pp. 1–27.

[45] T. Hartman, *A simpler* 1.5*-approximation algorithm for sorting by transpositions*, in Proceedings of the Fourteenth Annual Symposium on Combinatorial Pattern Matching (CPM), R. A. Baeza-Yates, E. Chávez, and M. Crochemore, eds., vol. 2676 of Lecture Notes in Computer Science, Berlin, June 2003, Springer-Verlag, pp. 156–169.

[46] K. T. Huber and V. Moulton, *Mathematics Of Evolution and Phylogeny*, Oxford University Press, New York, May 2005, ch. Phylogenetic Networks, pp. 178–204.

[47] A. Hultman, *Toric permutations*, Master's thesis, Department of Mathematics, KTH, Stockholm, Sweden, 1999.

[48] M. R. Jerrum, *The complexity of finding minimum-length generator sequences*, Theoretical Computer Science, 36 (1985), pp. 265–289.

[49] J. Kececioglu and D. Sankoff, *Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement*, Algorithmica, 13 (1995), pp. 180–210.

[50] C. L. Kuiken, B. Foley, B. Hahn, P. A. Marx, F. McCutchan, J. W. Mellors, J. I. Mullins, S. Wolinsky, and B. Korber, eds., *A Compilation and Analysis of Nucleic Acid and Amino Acid Sequences*, Human Retroviruses and AIDS, Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, Los Alamos, NM, 1999.

[51] A. Labarre, *A new tight upper bound on the transposition distance*, in Proceedings of the Fifth Workshop on Algorithms in Bioinformatics (WABI), R. Casadio and G. Myers, eds., vol. 3692 of Lecture Notes in Computer Science, Mallorca, Spain, Oct. 2005, Springer-Verlag, pp. 216–227.

[52] A. Labarre, *New bounds and tractable instances for the transposition distance*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 3 (2006), pp. 380–394.

[53] A. Labarre, *Edit distances and factorisations of even permutations*, in Proceedings of the Sixteenth Annual European Symposium on Algorithms (ESA), D. Halperin and K. Mehlhorn, eds., vol. 5193 of Lecture Notes in Computer Science, Karlsruhe, Germany, Sept. 2008, Springer-Verlag, pp. 635–646.

[54] S. Lakshmivarahan, J.-S. Jwo, and S. K. Dhall, *Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey*, Parallel Computing, 19 (1993), pp. 361–407.

[55] Z. Li, L. Wang, and K. Zhang, *Algorithmic approaches for genome rearrangement: a review*, IEEE Transactions on Systems, Man and Cybernetics, Part C, 36 (2006), pp. 636–648.

[56] B. McClintock, *The discovery and characterization of transposable elements: the collected papers of Barbara McClintock*, vol. 17 of Genes, cells, and organisms, Garland Publishing, New York, 1987.

[57] J. Meidanis and J. Setubal, *Introduction to Computational Molecular Biology*, Brooks-Cole, 1997.

[58] National Human Genome Research Institute, *The Talking Glossary of Genetics*. Published electronically at `http://www.genome.gov/`. All of the illustrations in the Talking Glossary of Genetics are freely available and may be used without special permission.

[59] C. Notredame, *Recent progress in multiple sequence alignment: a survey*, Pharmacogenomics, 3 (2002), pp. 131–144.

[60] C. Notredame, *Recent evolutions of multiple sequence alignment algorithms*, PLoS Computational Biology, 3 (2007), pp. 1405–1408.

[61] S. Ohno, *Evolution by gene duplication*, Springer-Verlag, 1970.

[62] C. H. Papadimitriou and M. Yannakakis, *The complexity of restricted spanning tree problems*, Journal of the ACM, 29 (1982), pp. 285–309.

[63] D. S. Parker and C. J. Lee, *Pairwise partial order alignment as a supergraph problem – aligning alignments revisited*. Submitted, Sept. 2003.

[64] P. A. Pevzner, *Computational molecular biology*, MIT Press, Cambridge, MA, 2000.

[65] D. Posada and K. A. Crandall, *Intraspecific phylogenetics: trees grafting into networks*, Trends in Ecology and Evolution, 16 (2001), pp. 37–45.

[66] M. Ridley, *Evolution*, Wiley-Blackwell, 4th ed., Sept. 2003.

[67] A. Schrijver, *Combinatorial optimization. Polyhedra and efficiency. Vol. A*, vol. 24 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 2003, ch. 17: Weighted bipartite matching and the assignment problem, pp. 285–300.

[68] S. S. Skiena, *The algorithm design manual*, Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[69] N. J. A. Sloane, *The on-line encyclopedia of integer sequences.* Published electronically at `http://www.research.att.com/~njas/sequences/`.

[70] R. P. Stanley, *Factorization of permutations into n-cycles*, Discrete Mathematics, 37 (1981), pp. 255–262.

[71] R. P. Stanley, *Enumerative Combinatorics*, vol. 1, Cambridge University Press, Cambridge, Great Britain, 1999.

[72] É. Tannier, A. Bergeron, and M.-F. Sagot, *Advances on sorting by reversals*, Discrete Applied Mathematics, 155 (2007), pp. 881–888.

[73] A. R. Templeton, K. A. Crandall, and C. F. Sing, *A cladistic analysis of phenotypic associations with haplotypes inferred from restriction endonuclease mapping and DNA sequence data. III. Cladogram estimation*, Genetics, 132 (1992), pp. 619–633.

[74] G. Valiente, *Algorithms on Trees and Graphs*, Springer-Verlag, Berlin, 2002.

[75] H. Wielandt, *Finite permutation groups*, Translated from German by R. Bercov, Academic Press, New York, 1964.

[76] S. Yancopoulos, O. Attie, and R. Friedberg, *Efficient sorting of genomic permutations by translocation, inversion and block interchange*, Bioinformatics, 21 (2005), pp. 3340–3346.