Subtree Transfer Operations on Phylogenetic Networks



Ross Atkins

University College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2016

Abstract

Leaf-labelled, rooted binary trees can be used to represent a wide range of data. One way to measure the discrepancy between a pair of such trees, is to measure how many subtree transfer operations are required to transform one into the other. Four standard subtree transfer operations for binary trees, used in particular for leaf-labelled binary trees, are: tree bisection and reconnection ($\mathcal{TBR}$), subtree prune and regraft ($\mathcal{SPR}$), rooted subtree prune and regraft ($r\mathcal{SPR}$) and nearest neighbour interchange ($\mathcal{NNI}$). Each of these operations induces a metric over the class, $\mathcal{B}(n)$, of binary trees with leaves labelled from 0 to $n-1$. For a pair of trees in $\mathcal{B}(n)$, we investigate the maximum possible distance between them, as well as the typical distance if they were chosen randomly from some exchangeable distribution.

The TBR (and rSPR) distances between two trees are equivalent to the size of a minimal unrooted agreement forest (and minimal rooted agreement forest respectively) of the pair of trees. The notion of an agreement forest generalises to arbitrarily large sets of trees. For a fixed integer, k, we determine quite tight bounds on the worst-case size of an agreement forest for a set of k trees in $\mathcal{B}(n)$. We also estimate the expected size of a minimal agreement forest of k randomly chosen trees, each independently chosen from their own exchangeable distribution.

We then construct two network operations: the phylogeny subtree prune and regraft ($p\mathcal{SPR}$) and the phylogeny nearest neighbour interchange ($p\mathcal{NNI}$), which are generalisations of the rSPR and NNI operations, respectively. The pSPR and pNNI, which act over the class of phylogenies (phylogenetic networks with recombination), are both special cases of the edge-exchange ($\mathcal{EEX}$) operation. We investigate the conditions under which it is possible to transform any phylogeny into any other phylogeny, and present some bounds on the number of operations required.

We also consider the extreme phylogenetic triple problem: For a given integer, $n \geq 3$, what is the minimum size of a set of rooted binary trees, with leaves labelled 1 to n, such that every phylogenetic triplet, $(a|bc)$, is satisfied by at least one tree in the set? We show that the answer to this question is $\Theta(\log \log n)$, by relating it to the nonbetweenness problem.

The final two chapters focus on two problems in communication network theory: the multiple unicast problem (the guessing game) and the information flow problem. For a fixed number of colours, s, we determine the guessing number of all large enough cycles, which answers a long-standing open question. We also provide a detailed constructive solution to the information flow problem on the full clock digraph, and on clock digraphs in general. These results solve two open problems, both had been conjectured 8 years ago.

Contents

1 Introduction and definitions **5**

1.1 Leaf-labelled binary trees 5

1.2 Subtree transfer operations 12

1.3 Tree metrics 17

1.4 Agreement forests 21

1.5 Agreement subtrees 26

2 Subtree transfer operations on binary trees **30**

2.1 Dividing a tree into almost-equal sized parts 31

2.2 Upper bound for the rSPR diameter 33

2.3 Upper bound for the expectation 35

2.4 Lower bound for the TBR radius 38

2.5 Lower bound for the expectation 42

2.6 Concentration results 43

3 The NNI metric and the RF metric **45**

3.1 The NNI metric 45

3.2 Robinson Foulds metric 49

4 Subtree transfer operations on networks with recombination **55**

4.1 Basic definitions 55

4.2 Edge exchanges 58

4.3 SPR moves on phylogenies with recombination 67

4.4 Nearest neighbour interchange operations on phylogenies 77

5 Betweenness and nonbetweenness **82**

5.1 Notation and definitions 82

5.2 Nonbetweenness 85

5.3 Betweeness 89

5.4 The extreme constraint problem 92
5.5 The extreme phylogenetic triplet problem 95

6 The guessing game on undirected cycles 98
6.1 Introduction 98
6.2 Backround material and notation 101
6.3 The fractional-clique-partition protocol 105
6.4 Entropy results 109
6.5 Guessing numbers of large odd cycles 114
6.6 Two colours 117
6.7 Index coding with side information 123

7 The information flow problem 126
7.1 The information flow problem 126
7.2 Full clock networks 131
7.3 Analysis of a specific case 135
7.4 General clock networks 136

8 Future research 141

A Subtree transfer operation distance calculations 144

Bibliography 156

In phylogenetics, leaf-labelled, rooted binary trees can be used to represent a wide range of phylogenetic data, for example the process of speciation. One way to measure the discrepancy between a pair of such trees, is to measure how many tree-rearrangement operations are required to transform one tree into the other. To put such measures into context, one might naturally ask what is the maximum number of operations required to transform one tree into the other? If the trees were generated randomly and independently, and X was the random variable equal to the difference between the two random trees (the number of tree rearrangement operations required to transform one into the other), then what can we say about the distribution of X?

In chapter 1, we introduce the basic objects and concepts which are to be investigated throughout this thesis. We define, $\mathcal{B}(n)$, the class of binary trees with n leaves labelled $\{0, 1, 2, \ldots, n-1\}$, and we introduce two common distributions over $\mathcal{B}(n)$: the uniform distribution and the Yule-Harding distribution. We also define the four standard subtree transfer (tree rearrangement) operations: tree bisection and reconnection ($\mathcal{TBR}$), subtree prune and regraft ($\mathcal{SPR}$), rooted subtree prune and regraft ($r\mathcal{SPR}$) and nearest neighbour interchange ($\mathcal{NNI}$). We also introduce other standard concepts including: splits, phylogenetic-triplets, the Robinson-Foulds metric and agreement forests. Chapter 1 concludes with a brief section regarding the maximum agreement subtree (MAST) problem.

Each subtree transfer operation induces a metric over the class, $\mathcal{B}(n)$. In chapter 2, we investigate these metric spaces. Specifically, for any $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}\}$, if X is the χ-distance between a pair of trees chosen randomly and independently from $\mathcal{B}(n)$ using any exchangeable distribution, then the maximum value of X is $n - \Theta(\sqrt{n})$, the expected value of X is $n - \Theta(n^{2/3})$ and the distribution of X is highly concentrated about its mean. More generally, for any fixed integer $k \geq 2$, if X were the size of a minimal agreement forest of a set of k random trees chosen from $\mathcal{B}(n)$, independently with an exchangeable distribution, then the maximum value of X is $n - \Theta(n^{1/k})$ and the expected value of X is $n - \Theta(n^{2/(k+1)})$. By the agreement forest lemmas, the metric results can be seen to be a special case (the $k = 2$ case) of the agreement forest results. The content of chapter 2 is joint work with McDiarmid [6], and has been accepted to be published in *Annals of Combinatorics*.

In chapter 3, we explore the $\mathcal{NNI}$-distance between a pair of random trees chosen uniformly and independently from $\mathcal{B}(n)$. The typical and extreme values of this distance are known to be $\Theta(n \log n)$. The results in chapter 3 are all known results, however, we establish the upper bound using a novel construction. For the Robinson-Foulds metric, can find the limiting distribution exactly. Explicitly for a pair of

trees $A, B \in \mathcal{B}(n)$ chosen independently with the uniform distribution, then the distribution of the random variable, $n - 3 - d_{RF}(A, B)$, tends (as $n \to \infty$) to a Poisson distribution with mean $1/8$. If instead A and B are chosen with the Yule Harding distribution, then the limiting distribution is still Poisson, but has mean $2/9$.

Sometimes, phylogenetic data cannot be adequately represented by trees; sometimes it is more accurate to represent the data by a phylogenetic network with recombination. In chapter 4, we attempt to generalise some of the subtree transfer operations to act on the class, $\mathcal{P}(l, r)$, of directed binary networks with recombination. Specifically, the *phylogeny subtree prune and regraft* ($p\mathcal{SPR}$) and the *phylogeny nearest neighbour interchange* ($p\mathcal{NNI}$) are natural generalisations of the $r\mathcal{SPR}$ operation and $\mathcal{NNI}$ operation respectively. To help introduce these network rearrangement operations, we define a more elementary operation; the *edge exchange of order two* ($\mathcal{EEX}_2$), which is the operation in which the endpoints of two edges are permuted. As it turns out, the class, $\mathcal{P}(l, r)$ is not traversable by $\mathcal{EEX}_2$ moves. *i.e.* In general, it is not possible to transform any phylogeny in $\mathcal{P}(l, r)$ into any other using only $\mathcal{EEX}_2$ moves, such that all intermediate networks are acyclic. We then construct an invariant, which we call the *signature*, which characterises exactly which trees can be obtained from one another by $\mathcal{EEX}_2$ moves (such that all intermediate trees are phylogenies in $\mathcal{P}(l, r)$).

In the second half of chapter 4, we show that the class, $\mathcal{P}(l, r)$, is traversable by $p\mathcal{SPR}$ operations, and in fact at most $l + \mathcal{O}(r)$ $p\mathcal{SPR}$ operations are required, where l is the number of leaves and r is the number of recombination vertices. This result is actually quite long and tedious to prove, because we have to be certain that each operation in the construction, does not create any double edges or cycles. In the final section of chapter 4, we show that the class $\mathcal{P}(l, r)$ is also traversable by $p\mathcal{NNI}$ operations. Recalling that any $\mathcal{NNI}$ operation is a special case of the $r\mathcal{SPR}$ operation ($\mathcal{NNI} \subseteq r\mathcal{SPR}$), one might imagine that the $p\mathcal{NNI}$ is a special case of the $p\mathcal{SPR}$. However, this is not the case; certain $p\mathcal{NNI}$s cannot be achieved by a single $p\mathcal{SPR}$. So proving a class traversable by $p\mathcal{NNI}$s is not strictly stronger than proving the class traversable by $p\mathcal{SPR}$s. The results in chapter 4 are all original work, as far as I am aware.

In chapter 5, we define the following problems called betweenness and nonbetweenness. The betweenness function ($\text{bet}(n)$) is the minimum size of a set of orderings of n objects, such that for every ordered triple (a, b, c) of distinct objects, there is at least one ordering in which b is between a and c. The nonbetweenness function ($\text{nbet}(n)$) is the minimum size of a set of orderings of n objects, such that for every

ordered triple (a, b, c) of distinct objects, there is at least one ordering in which b is not between a and c. We show that:

$$\text{nbet}(n) = \lceil \log_2 \log_2 n \rceil \quad \text{and} \quad \text{bet}(n) = \Theta(\log n).$$

Both betweenness and nonbetweenness are special cases of a more general problem called the extreme constraint problem ($\text{ecp}(n)$). We use the asymptotics of $\text{bet}(n)$ and $\text{nbet}(n)$ to find the asymptotics of $\text{ecp}(n)$ in general. The extreme phylogenetic triplet function ($\text{ept}(n)$) is the size of a minimum set of binary trees, each with a root and n other leaves labelled $1, 2, 3, \ldots, n$, such that every phylogenetic triplet, is consistent with at least one of the trees. We use the asymptotics of $\text{ecp}(n)$ to show that

$$\text{ept}(n) = \Theta(\log \log n).$$

Chapter 6 focuses on a cooperative n-player games, called the guessing game, which is played on a graph G with n vertices (each vertex represents a player). Each vertex of G is assigned a colour, uniformly and independently at random, from a set of $s \geq 2$ different colours. Informally, the players are required to 'guess' the colour assigned to their own vertex, knowing only the colours assigned to each neighbouring vertex. Determining the guessing number of a graph G (*i.e.* finding an optimal protocol for the guessing game), is equivalent to solving the multiple unicast problem, which is an important problem in network information theory. For all perfect graphs, the guessing number is known. However, the guessing number of even the simplest non-perfect graphs, the odd cycles, remains tantalisingly unsolved. For any constant s, we solve the guessing game on all large enough odd cycles - answering an open problem conjectured by Chrisfides and Markström [17]. Specifically we show that, for any large enough odd n, the guessing number of C_n is

$$\text{gn}(C_n, s) = \frac{n-1}{2} + \log_s(a),$$

where a is the greatest divisor of s that is at most $\sqrt{s}$. The proof of this Theorem is long and dense, but this problem has remained open for quite a long time, so perhaps there is no significantly easier way to prove it. The content of Chapter 6 is joint work with Rombach and Skerman [7].

Chapter 7 focuses on another problem in network information theory, the information flow problem. It was thought that the information flow problem on the full clock network was solved in 2007 [43], however their proof is incomplete. The solution to the information flow problem on clock digraphs in general, has been only conjectured,

until now. I present a detailed constructive solution to the information flow problem on every full clock network. I also present a constructive solution to the information flow problem on arbitrary clock networks of length n and width r such that $n \geq 3r^3$. These two main results are original work. Intuitively, I expect that it is possible to replace the cubic bound of $3r^3$ with something quadratic in r. The chapter concludes with an example that demonstrates that this bound cannot be replaced (in general) by anything less than $r^2 - r$. The content of chapter 6 is original work, and has been written up into a paper [5].

Chapter 1

Introduction and definitions

1.1 Leaf-labelled binary trees

Basic definitions

In this document we will typically follow the standard notations as in [45]. Recall that a *finite tree* is any finite connected graph which contains no cycles.

Definition 1.1.1. Let S be a non-empty, finite set. If $|S| > 1$, then a *leaf-labelled binary tree* with leaf-label set S is a finite tree which has 2 types of vertex:

- *tree* vertices of degree 3, and
- *leaf* vertices of degree 1, labelled with the set S (there is a bijection between the leaves and S).

If $0 \in S$, the leaf labelled 0 is called the *root* of the tree. In the trivial case, when $S = \{\alpha\}$ is a singleton, the trivial graph containing only one vertex, labelled α, is considered to be a binary leaf-labelled tree, and its vertex is called a leaf.

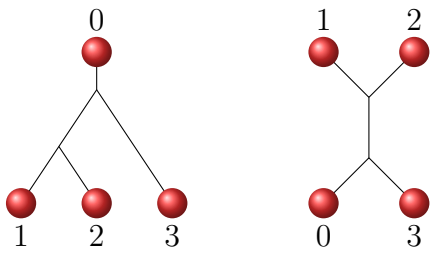


Figure 1.1: Two diagrams for the same binary tree.

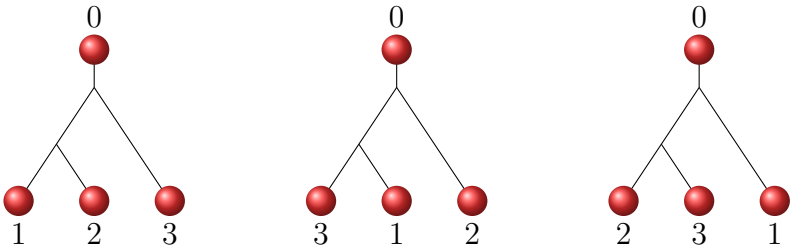


Figure 1.2: The three distinct trees in $\mathcal{B}(4)$.

Unless explicitly stated otherwise, when we say *binary tree*, we mean leaf-labelled binary tree. Formally, the edges of a binary tree are undirected. If $0 \in S$, it is possible to orient the edges *away from the root*; to orient the edges so that for any leaf $s \in S$, the path from 0 to s is a directed path. If the edges are oriented away from the root, then the orientation of an edge uv would specify the chronological order of the evolutionary events u and v. Although we don't formally orient the edges of a binary tree, we still make the following definition.

Definition 1.1.2. Let x and y be any two vertices in a binary tree with a root. The vertex y is called a *descendant* of x if and only if there is a path from the root to y which passes through x. The vertex x is called an *ancestor* of y if and only if y is a descendant of x. The neighbours of a vertex x which are descendants of x are called the *children* of x. If y is a child of x, then x is called the *parent* of y.

Definition 1.1.3. For any non-empty set S, let $\mathcal{B}(S)$ be the set of all leaf-labelled binary trees, with label set S. For a leaf-labelled tree A, let $L(A)$ denote the label set of A; so if $A \in \mathcal{B}(S)$ then $L(A) = S$. To simplify notation, for any positive integer n, let $[n] = \{1, 2, 3, \ldots, n\}$ and let $[n]_0 = \{0, 1, 2, \ldots, n-1\}$. For any integer $n \geq 3$, let $\mathcal{B}(n)$ denote the set of all binary trees with leaves labelled by $[n]_0$,

$$\mathcal{B}(n) = \mathcal{B}([n]_0) = \mathcal{B}(\{0, 1, 2, 3, \ldots, n-1\}).$$

A tree in $\mathcal{B}(n)$ has n leaves, $n-2$ tree-vertices (making $2n-2$ vertices in total) and $2n-3$ edges. A pair of leaves are called a *cherry* if the path between them has length 2; *i.e.* if they have a neighbour in common.

For trees T and T' in $\mathcal{B}(S)$, we say T is isomorphic to T' (denoted $T \equiv T'$) if there is a graph isomorphism preserving leaf labels; if there is a bijection $\psi : V(T) \to V(T')$ such that uv is an edge in T if and only if $\psi(x)\psi(y)$ is an edge in T' and for each leaf v

of T, v and $\psi(v)$ are labelled with the same element of S. For non-empty $S' \subseteq L(A)$, let $A|S'$ be the minimal subtree of A that contains all the leaves with labels in S'. Let A/S' be the tree formed by suppressing any vertices of degree 2 from $A|S'$. So

$$A|S' \text{ is a subgraph of } A \qquad \text{and} \qquad A/S' \in \mathcal{B}(S').$$

Definition 1.1.4. A leaf-labelled binary tree is called a *caterpillar* if all the tree-vertices form a path. A caterpillar is called a *rooted caterpillar* if it has a root, and the root is adjacent to one of the endpoints of the path containing all the tree vertices. For any ordering ϕ of the set $\{1, 2, 3, \ldots, n-1\}$, let $\text{cat}(\phi)$ denote the rooted caterpillar in $\mathcal{B}(n)$, in which the ordering of the leaves from the root is given by ϕ (see Figure 1.3 for example).

Definition 1.1.5. Let A be a tree in $\mathcal{B}(n)$ for some $n \geq 3$. For any two vertices u and v, let $l_A(u, v)$ denote the length of the path in A from u to v. Define the *center* of A to be the set of tree vertices $u \in A$ such that minimises the value of

$$\max_{v \in A} \{l_A(u, v)\}.$$

The center of A contains either one vertex of a pair of adjacent vertices. For each leaf $u \in A$, let $l_A(u)$ denote the length of the shortest path from u to a vertex in the center of A.

- The tree A is called *balanced* if $l_A(u) = l_A(v)$ for all leaves u and v.
- The tree A is called *almost balanced* if for any two leaves $u, v \in A$, we have $|l_A(u) - l_A(v)| \leq 1$.

For $n \geq 3$, there exists a balanced tree in $\mathcal{B}(n)$ if and only iff n can be written in the form 3×2^k or 4×2^k for some non-negative integer n. There exists an almost balanced tree in $\mathcal{B}(n)$ for all $n \geq 3$.

Binary trees can be used to convey phylogenetic data in multiple different ways.

- The evolutionary history of a set, S, of distinct species can be represented by an element of $\mathcal{B}(n)$ with $n = |S| + 1$, if we number the species from 1 to $n - 1$. The leaf labelled 0 represents a common ancestral species and the tree-vertices represent intermediate species (transitional forms).

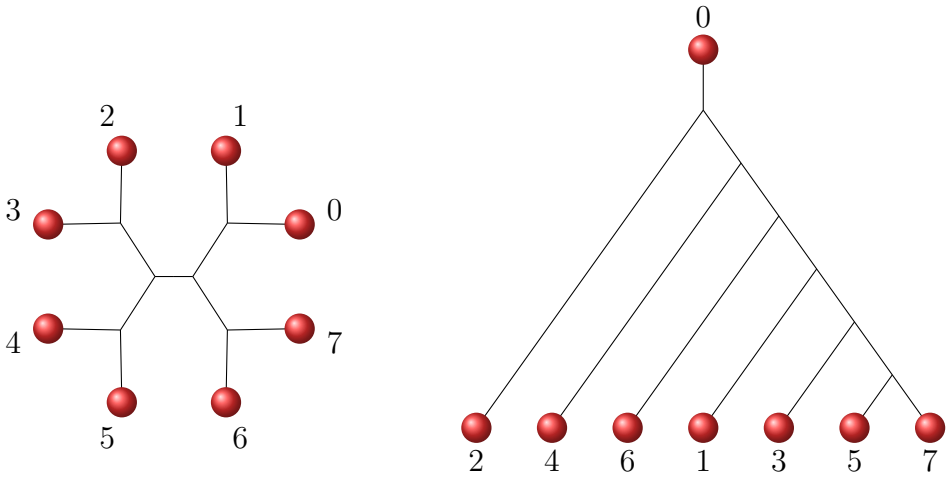


Figure 1.3: Two trees in $\mathcal{B}(8)$: a balanced tree (left) and the rooted caterpillar $\text{cat}(2, 4, 6, 1, 3, 5, 7)$ (right). Under the Yule-Harding distribution, the rooted caterpillar tree is less probable than the balanced tree.

- The genealogical history of a set, S, of gene-sequences observed within certain individuals in a species, by an element of $\mathcal{B}(n)$ with $n = |S|$. The common ancestral gene sequence is numbered 0, and the other gene-sequences are numbered from 1 to $n - 1$. Mutation can be assumed to have occurred along some of the edges.

For an in-depth discussion on how data can be represented by a binary tree, we refer the reader to [29].

Distributions over $\mathcal{B}(n)$

The concept of an exchangeable distribution over $\mathcal{B}(n)$ was introduced in [2]. Two commonly used exchangeable distributions are the uniform distribution and the Yule-Harding distribution (Definition 1.1.6).

Definition 1.1.6. Let T be a random tree in $\mathcal{B}(n)$. The distribution of T is called *exchangeable* iff for every $A, B \in \mathcal{B}(n)$, which have the same underlying unlabelled rooted tree structure (i.e. if B can be obtained from A by a permutation of the non-root leaf-labels) then $\mathbb{P}(T = A) = \mathbb{P}(T = B)$. Exchangeability is also known as leaf-invariance [40]. There are two common exchangeable distributions:

- In the *uniform* distribution, each tree $A \in \mathcal{B}(n)$ is equally probable. *i.e.*

$$\mathbb{P}(A) = \frac{1}{|\mathcal{B}(n)|} \qquad \forall\, A \in \mathcal{B}(n).$$

- The *Yule-Harding* distribution [47] over $\mathcal{B}(n)$ can be defined in the following way. Over $\mathcal{B}(3)$, the Yule-Harding distribution is identical to the uniform distribution. For $n \geq 4$, choose a tree $T' \in \mathcal{B}(n-1)$ using the Yule-Harding distribution, then construct T by choosing an edge incident with a non-root leaf, subdividing it and adjoining a new leaf labelled $n-1$, and then finally applying a random permutation of the non-root leaf labels. Or, equivalently, choosing $\alpha \in \{1, 2, 3, \ldots, n-2\}$ uniformly at random, replacing the leaf α with the cherry of α and $n-1$, and then applying a random permutation of the non-root leaf-labels.

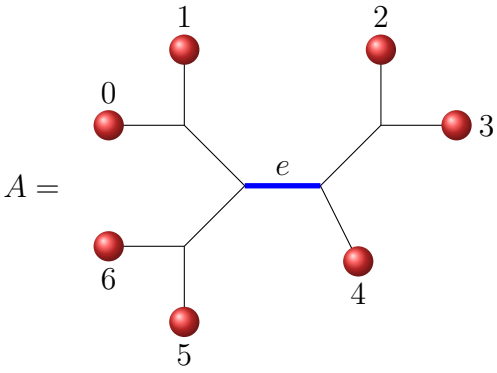
The final permutation of the labels is necessary in this definition of the Yule-Harding distribution, to ensure that the distribution is exchangeable. Otherwise, the procedure would never obtain the trees in $\mathcal{B}(n)$, for which the leaf labelled $n-1$ is not part of a cherry.

When constructing a binary tree under the Yule-Harding distribution, one could simply construct the unlabelled (rooted) tree-shape first and then assigning labels $\{1, 2, \ldots n-1\}$ to the non-root leaves at the end. The Yule-Harding distribution arises naturally in phylogenetics, if we assume that at any given time, any species is equally likely to speciate (bifurcate into two distinct species). Caterpillar-like trees tend to be less probable under the Yule Harding distribution, than under the uniform distribution.

For $i \geq 2$, any tree in $\mathcal{B}(i+1)$ can be constructed from a tree in $\mathcal{B}(i)$ by subdividing an edge and then adding a pendant leaf labelled i attached to the new vertex. This operation is reversible, by simply deleting the leaf labelled i, and suppressing the degree 2 vertex created this way. This is a many-to-one map from $\mathcal{B}(i+1)$ to $\mathcal{B}(i)$. Since there are $(2i-3)$ edges in any tree in $\mathcal{B}(i)$, this operation we see that $|\mathcal{B}(i+1)| = (2i-3)|\mathcal{B}(i)|$. Therefore, for $n \geq 3$,

$$|\mathcal{B}(n)| = 1 \times 3 \times 5 \times \cdots \times (2n-5)\,.$$

By this many-to-one map, we know that if T is a random tree chosen uniformly from $\mathcal{B}(i)$, and T' is obtained by uniformly at random, choosing an edge of T, subdividing it and then adjoining a pendant leaf labelled i, then T' will be uniformly distributed over $\mathcal{B}(i+1)$. So a quick method to sample from $\mathcal{B}(n)$ uniformly is to start with the [unique] tree in $\mathcal{B}(3)$ and then iteratively for $i = 3, 4, 5, \ldots, n-1$, subdivide a randomly and uniformly chosen edge and adjoin a pendant leaf labelled i. This method differs from the method of sampling from the Yule-Harding distribution because the pendant



$$\Sigma(A) = \left\{ \begin{array}{c} (0,1|2,3,4,5,6), \\ (0,1,5,6|2,3,4), \\ (0,1,2,3,4|5,6), \\ (0,1,4,5,6|2,3) \end{array} \right\}$$

Figure 1.4: The non-trivial splits of a tree $A \in \mathcal{B}(7)$. Edge e corresponds to the split $(0,1,5,6|2,3,4)$.

leaf is adjoined to any edge (rather than any edge incident with a non-root leaf, as in Definition 1.1.6) and there is no need for the final permutation of the leaf labels.

Splits

Definition 1.1.7. Let e be an edge in a tree $T \in \mathcal{B}(S)$. If $\{S_1, S_2\}$ is a bipartition of S such that deleting e from T yields the two components $T|S_1$ and $T|S_2$, then $\{S_1, S_2\}$ is called the *split* corresponding to e. If either S_1 or S_2 is a singleton, then $\{S_1, S_2\}$ is called a *trivial split*. The split $\{S_1, S_2\}$ is often denoted

$$(a_1, a_2, \ldots | b_1, b_2, \ldots),$$

where $S_1 = \{a_1, a_2, \ldots\}$ and $S_2 = \{b_1, b_2, \ldots\}$. For any $A \in \mathcal{B}(n)$, let $\Sigma(A)$ to be the set of all non-trivial-splits generated by edges in A.

Trivial splits are called "trivial", because the bipartition $\big\{\{\alpha\}, S\backslash\{\alpha\}\big\}$ is a split of any $T \in \mathcal{B}(S)$ for all $\alpha \in S$ (it is generated by the edge incident with the leaf labelled α). Any tree $A \in \mathcal{B}(n)$ contains $2n-3$ edges. Exactly n of these edges generate trivial splits, so by the one-to-one correspondence between the edges and the splits of a a binary tree, we have

$$|\Sigma(A)| = n - 3 \qquad \forall\, A \in \mathcal{B}(n).$$

Lemma 1.1.8 is a well known result [25]. It demonstrates that it is possible to reconstruct a binary tree $A \in \mathcal{B}(n)$, just knowing $\Sigma(A)$.

Lemma 1.1.8. *For any $A, B \in \mathcal{B}(S)$, if $\Sigma(A) = \Sigma(B)$, then $A \equiv B$.*

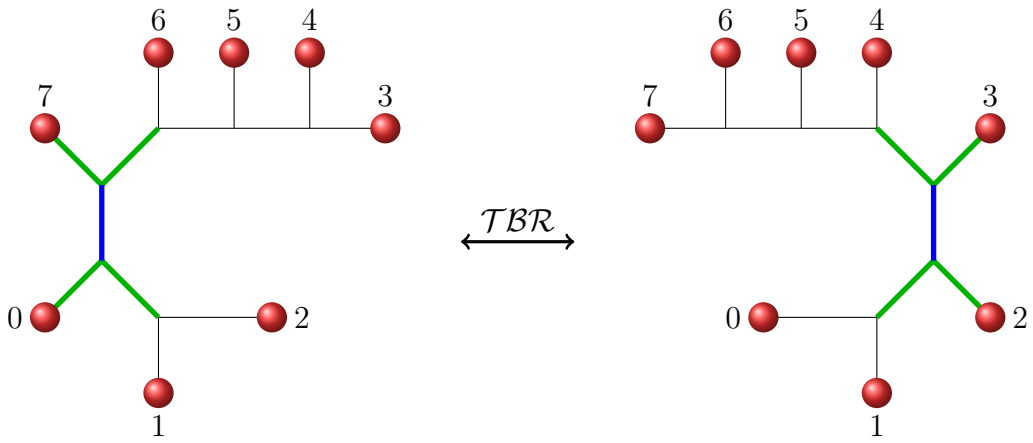


Figure 1.5: An example of the TBR operation.

Proof. Proof by induction on $|S|$. For the base case, $|S| = 3$, we have $|\mathcal{B}(S)| = 1$ and so $A = B$ trivially. Now assume $\Sigma(A) = \Sigma(B)$ for some $A, B \in \mathcal{B}(S)$ with $|S| \geq 4$. Since A is a binary tree, it contains a cherry and thus there is a split $s \in \Sigma(A)$ of the form $(\alpha, \beta|\ldots)$. Since $\Sigma(A) = \Sigma(B)$, we therefore can conclude that α and β form a cherry in B too. All other splits in $\Sigma(A) = \Sigma(B)$ have α and β on the same side of the split. Therefore, if A' and B' are the trees obtained by replacing the (α, β)-cherry with a single leaf labelled α in trees A and B respectively, then $\Sigma(A') = \Sigma(B')$. By the inductive assumption, this implies $A' \equiv B'$ and thus $A \equiv B$. □

Since a binary tree, $A \in \mathcal{B}(n)$ is uniquely determined by its set of non-trivial splits, $\Sigma(A)$, it is possible to reconstruct A from $\Sigma(A)$. Such a construction is given in [29].

Definition 1.1.9. Let $n \geq 4$ be an integer. A *phylogenetic triplet*, denoted $(a|b, c)$, is any three distinct integers a, b and c chosen from the set of non-root leaf labels, $\{1, 2, \ldots, n-1\}$. A binary tree $A \in \mathcal{B}(n)$ is said to be *consistent* with the phylogenetic-triplet $(a|b, c)$ if and only if the path from the root to the leaf labelled a is disjoint from the path between b and c. Equivalently, A is consistent with $(a|b, c)$ if and only if there exists a split $s \in \Sigma(A)$, such that 0 and a are on one side, while b and c are on the other side. *i.e.* a split of the form

$$s = (0, a, \ldots|b, c, \ldots) \in \Sigma(A).$$

For more information about phylogenetic triplets, we refer the reader to [45].

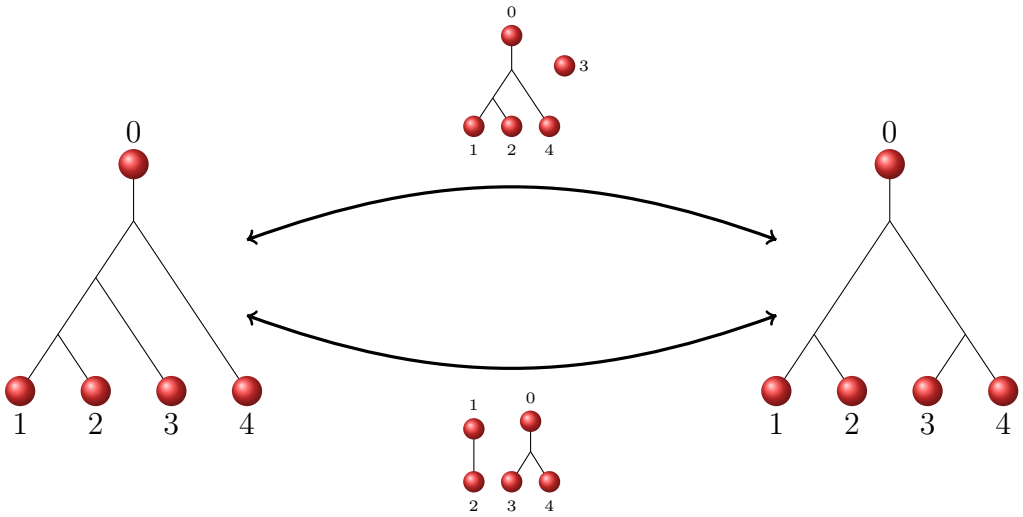


Figure 1.6: A transformation which can be achieved in two different ways.

1.2 Subtree transfer operations

A subtree transfer operation is a process for transforming one binary tree into another. Here we define four commonly used subtree transfer operations: TBR (*tree bisection and reconnection*), SPR (*subtree prune and regraft*), rSPR (*rooted subtree prune and regraft*) and NNI (*nearest neighbour interchange*), which act on the class $\mathcal{B}(n)$.

Tree bisection and reconnection

Definition 1.2.1. A tree-bisection-and-reconnection (TBR) on a tree $T \in \mathcal{B}(n)$, for $n \geq 3$, is a two step process:

1. *(bisection step)* Delete an edge e from T (this edge is called the *bisection* edge) and then suppress all degree 2 vertices (edges created by these suppressions are called *new* edges) to obtain a pair T_1, T_2 of binary trees with $L(T_1) \sqcup L(T_2) = [n]_0$. (The number of new edges is 2, unless the bisection edge was incident with a leaf. In the later case, there is only one new edge and either T_1 or T_2 is an isolated vertex.)

2. *(reconnection step)* Connect T_1 and T_2 by creating an edge between the midpoint of an edge in T_1 and the midpoint of an edge in T_2. This edge is called the *reconnecting* edge. (In the case that T_1 or T_2 is an isolated vertex, instead the reconnecting edge is incident to this vertex.)

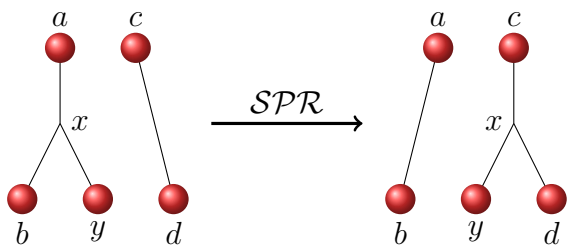


Figure 1.7: The SPR move

See Figure 1.5: on the left the bisection edge is blue and the green edges will form the new edges, and on the right the blue edge is the reconnection edge and the green edges are the new edges. If $A \in \mathcal{B}(n)$, and B is the result of performing a TBR on A, then $B \in \mathcal{B}(n)$ because B must be connected, acyclic, and all non-leaf vertices have degree 3.

Definition 1.2.2. Let $\mathcal{TBR} = \mathcal{TBR}(n)$ denote the set of pairs $(A, B) \in \mathcal{B}(n)^2$ such that B can be obtained from A by performing a single TBR move.

It is well known that TBR moves are reversible [45] that is, if B can be obtained from A via a TBR move, then A can be obtained from B via a TBR move. We can construct the reverse TBR move by switching the roles of the bisection edge and the reconnecting edge. Therefore

$$(A, B) \in \mathcal{TBR} \quad \text{if and only if} \quad (B, A) \in \mathcal{TBR}.$$

Typically, if $(A, B) \in \mathcal{TBR}$, there is a unique TBR move that transforms A into B. However, in special cases there might be several different TBR moves which transform A into B (an example of this is given in Figure 1.6).

Subtree prune and regraft

The SPR move (Definition 1.2.3) is an operation on binary trees whereby a subtree is removed from one part of the tree and regrafted to another part of the tree. SPR moves feature in many papers [3, 11, 32] and are used by tree-searching software packages, like PAUP [50] and Garli [53], for phylogenetic research.

Definition 1.2.3. A *subtree-prune-and-regraft* (SPR) move taking x from ab to cd is an operation which can be performed on a tree $T \in \mathcal{B}(n)$, as long as xy, ax, xb, cd are distinct edges (vertices a, b, c, d need not be distinct; possibly $a = c$, $b = c$, $a = d$ or $b = d$) and the path from c to y contains x. An SPR move is a two step process:

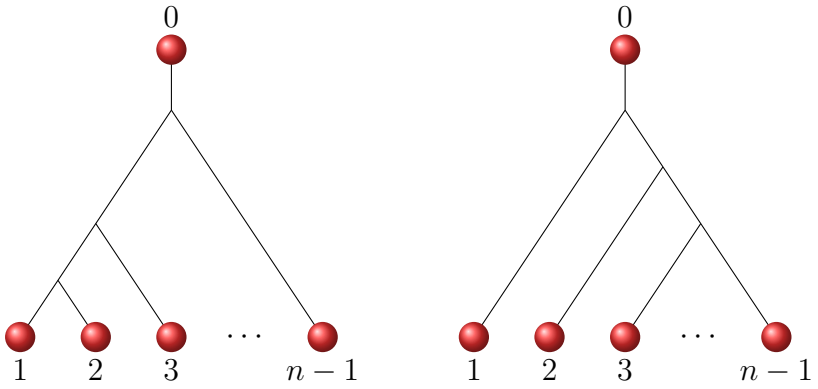


Figure 1.8: Trees that differ by a single SPR, but at least $\frac{n-3}{2}$ rSPR moves.

- *(prune step)* delete edges ax, xb and replace them with ab, then
- *(regraft step)* delete edge cd and replace it with cx and xd.

If B is the result of an SPR move on a tree $A \in \mathcal{B}(n)$, then B is acyclic (because of the path from c to y contains x), and so B is a binary tree. *i.e.* $B \in \mathcal{B}(n)$. Let $\mathcal{SPR} = \mathcal{SPR}(n)$ denote the set of pairs (A, B) such that $A \in \mathcal{B}(n)$ and B is the result of an SPR move on A. If y is a descendant of x, then this is called a *rooted-subtree-prune-and-regraft* (rSPR) move. Let $r\mathcal{SPR} = r\mathcal{SPR}(n)$ denote the set of pairs $(A, B) \in \mathcal{SPR}(n)$ such that B is the result of an rSPR move on A.

See Figure 1.7. By definition, an rSPR move is a special case of an SPR move. Furthermore an SPR move is a special case of a TBR move. Informally, an SPR move is any TBR move in which one of T_1 or T_2 is an isolated vertex or one of the endpoints of the reconnecting edge is the midpoint of one of the new edges. Therefore any valid SPR move performed on a binary tree in $\mathcal{B}(n)$ will result in a tree in $\mathcal{B}(n)$. Moreover,

$$r\mathcal{SPR} \subseteq \mathcal{SPR} \subseteq \mathcal{TBR}.$$

These inclusions have been noted in many publications [3, 11, 45]. If the edges were orientated away from the root, then an rSPR move is any SPR move that preserves these orientations. It is also known that SPR moves and rSPR moves are reversible [45].

$$(A, B) \in \mathcal{SPR} \iff (B, A) \in \mathcal{SPR} \quad \text{and} \quad (A, B) \in r\mathcal{SPR} \iff (B, A) \in r\mathcal{SPR}.$$

Not all SPR moves can be achieved with a single rSPR move. Indeed, we may need an arbitrary number of rSPR moves to simulate one SPR move [11]. For example,

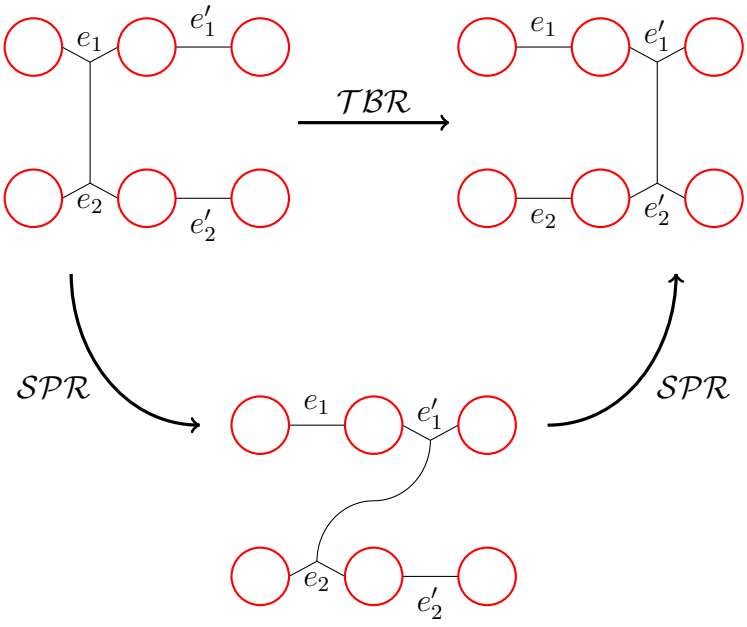


Figure 1.9: Any TBR can be simulated with 2 SPRs.

the trees in Figure 1.8 can be seen to differ by a single SPR by pruning off the single leaf labelled 0, and then regrafting it appropriately. However, as we will show In Example 1.4.3, at least $\frac{n-3}{2}$ rSPR moves are required to transform one into the other.

The following proposition is Lemma 2.7 (2b) in [3].

Proposition 1.2.4. *Any TBR move can be achieved using at most 2 SPRs; if $(A, B) \in \mathcal{TBR}$, then there exists some C such that $(A, C), (C, B) \in \mathcal{SPR}$.*

Proof. See Figure 1.9. Let the bisection step of the TBR, create two components T_1 and T_2. In the case that T_1 or T_2 was an isolated vertex, then we can trivially let $C = A$, since $(A, B) \in \mathcal{SPR}$ immediately. Otherwise the TBR had two new edges. Wlog let e_1 in T_1 and e_2 in T_2 be these new edges, and let the reconnecting edge join the midpoints of edge e'_1 in T_1 and $e'_2 \in T_2$. The two SPRs are as follows:

1. prune T_2 from e_1 to e'_1, to obtain C, and then
2. prune T_1 from e_2 to e'_2.

□

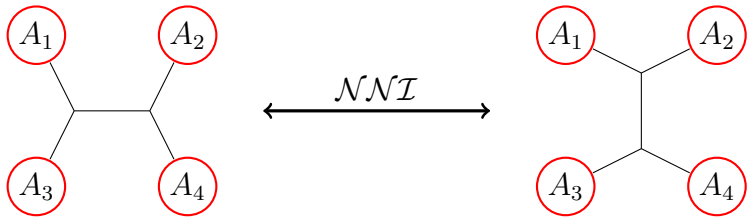


Figure 1.10: The NNI move.

Nearest neighbour interchange

The most elementary subtree transfer operation is the NNI (nearest neighbour interchange). The NNI operation has been used in several publications [3], [34], [46].

Definition 1.2.5. For an integer $n \geq 4$, let u and v be adjacent tree-vertices of a binary tree $T \in \mathcal{B}(n)$. Let a and b be the neighbours of u (other than v) and let c and d be the neighbours of v (other than u). A nearest-neighbour-interchange (NNI) on edge uv is a two step process:

- *(contraction step)* First contract edge uv to form a degree four vertex x, with neighbours a, b, c and d.
- *(resolution step)* Resolve vertex x into two tree-vertices u' and v' so that a and c (and v') are neighbours of u', while b and d (and u') are neighbours of v'.

For any edge joining two tree-vertices uv, there are four subtrees attached to it (the subtrees labeled A_1, A_2, A_3 and A_4 in Figure 1.10). An NNI can be interpreted as swapping two of these subtrees; replacing edges ub and vc with edges uc and vb.[1] The NNI move is a special case of the SPR move, simply prune off a subtree A_i and then regraft it appropriately. If we choose i so that the root is not in A_i, then this can be guaranteed to be an rSPR move.

Definition 1.2.6. Let $\mathcal{NNI} = \mathcal{NNI}(n)$ denote the set of pairs (A, B) such that $A \in \mathcal{B}(n)$ and B is the result of an NNI performed on A.

Since the result of an NNI applied to $A \in \mathcal{B}(n)$ is a tree $B \in \mathcal{B}(n)$, we must have $\mathcal{NNI} \subseteq \mathcal{B}(n)^2$. Furthermore, since the NNI is a special case of the rSPR, we also have

$$\mathcal{NNI} \subseteq r\mathcal{SPR}.$$

[1] This interpretation shows that the NNI move is a special case of the $\mathcal{EEX}_2$ move (Definition 4.2.1).

1.3 Tree metrics

A *tree metric* is a metric over the set $\mathcal{B}(n)$. Tree metrics are used to measure similarity between trees. Here we define: the Robinson-Foulds metric, the NNI metric, the rSPR metric, the SPR metric and the TBR metric.

The Robinson-Foulds metric

The Robinson Foulds metric [25] is a popular metric for phylogenetic software because it is relatively fast to compute. It can be computed in linear time [20], and approximated with bounded error in sublinear time [39].

Definition 1.3.1. The Robinson Foulds distance between two trees $A, B \in \mathcal{B}(n)$ is given by

$$d_{RF}(A, B) = \tfrac{1}{2}|\Sigma(A) \triangle \Sigma(B)|.$$

The Robinson Foulds distance is half the number of splits generated by A but not B, plus half the number of splits generated B but not A. Since $|\Sigma(A)| = |\Sigma(B)|$ for binary trees $A, B \in \mathcal{B}(n)$, we can deduce that $d_{RF}(A, B)$ is exactly the number of splits generated by A and that are not generated by B. *i.e.*

$$d_{RF}(A, B) = \tfrac{1}{2}|\Sigma(A) \triangle \Sigma(B)| = |\Sigma(A) \backslash \Sigma(B)|.$$

As a measure of how similar two trees are, the Robinson Foulds metric is often very counter intuitive. Figure 1.11 depicts two very similar binary trees (differing in a single rSPR move), however they have maximal Robinson Foulds distance. For two trees $A, B \in \mathcal{B}(n)$, the maximal possible value of the Robinson Foulds distance, is $n - 3$ because

$$\tfrac{1}{2}|\Sigma(A) \triangle \Sigma(B)| \leq \frac{|\Sigma(A)| + |\Sigma(B)|}{2} = n - 3.$$

Subtree transfer operation metrics

For any subtree transfer operation, $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}, \mathcal{NNI}\}$, we will define the χ-distance between two trees to be the minimum number of χ-moves required to transform one tree into the other, ensuring that all intermediate graphs are valid binary trees (*i.e.* acyclic). The definitions given in Definition 1.3.2 and Definition 1.3.3 are given in terms of a general class $\mathcal{C}$ of graphs, not just for the usual case that $\mathcal{C} = \mathcal{B}(n)$. We define it in general here, so that we can use the same notation later in Section 4 when we consider operations on certain classes of phylogenetic networks with recombination.

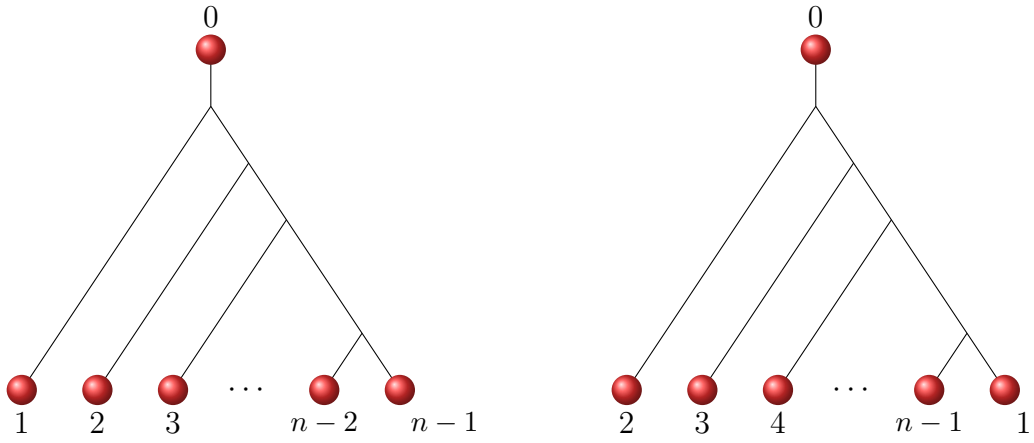


Figure 1.11: The trees $\mathrm{cat}(1, 2, \ldots, n-1)$ (left) and $\mathrm{cat}(2, 3, 4, \ldots, n-1, 1)$ (right) are phylogenetically very similar. The differ by a single rSPR move, but they have maximal Robinson-Foulds distance.

Definition 1.3.2. For a given class of graphs, $\mathcal{C}$, an operation χ over $\mathcal{C}$ means any subset of $\mathcal{C}^2$ (for example, if $\mathcal{C} = \mathcal{B}(n)$ and $\chi = \mathcal{SPR}$, then χ is the set of all pairs $(A, B) \in \mathcal{B}(n)^2$ such that B can be obtained from A via an SPR move). We say that $\mathcal{C}$ is *traversable* by χ operations, if and only if, for every pair $A, B \in \mathcal{C}$, there exists an integer $k \geq 0$ and a sequence

$$(A = A_0, A_1, A_2, \ldots, A_{k-1}, A_k = B)$$

such that A_i is a member of the class $\mathcal{C}$ (for all $0 \leq i < k$) and $(A_i, A_{i+1}) \in \chi$. The χ-distance (over $\mathcal{C}$) between a pair $A, B \in \mathcal{C}$, denoted $d_\chi^{\mathcal{C}}(A, B)$, is the minimum value of k, for which such a sequence exists. When the class $\mathcal{C}$ is clear from context, we will often simply write

$$d_\chi(A, B) \quad \text{to mean} \quad d_\chi^{\mathcal{C}}(A, B).$$

The $\mathcal{NNI}$-distance has been the subject of several publications, for example [3] and [30]. The $\mathcal{SPR}$-distance and $r\mathcal{SPR}$-distance is also well studied for example [11], [27], [30], and [31].

Definition 1.3.3. If $\mathcal{C}$ is a class of graphs, that is traversable by χ, then let $D_\chi(\mathcal{C})$ and $R_\chi(\mathcal{C})$ denote the χ-diameter and χ-radius (respectively) of $\mathcal{C}$. This is to say:

$$D_\chi(\mathcal{C}) := \max_{A,B \in \mathcal{C}} d_\chi^{\mathcal{C}}(A, B)$$

and moreover, let $R_\chi(n)$ denote the χ-*radius* of the class $\mathcal{B}(n)$;

$$R_\chi(\mathcal{C}) := \min_{A \in \mathcal{C}} \max_{B \in \mathcal{C}} d_\chi^{\mathcal{C}}(A, B).$$

As we shall see in Lemma 3.1.7, $\mathcal{B}(n)$ is traversable by $\mathcal{NNI}$ operations. Since $\mathcal{NNI} \subseteq r\mathcal{SPR} \subseteq \mathcal{SPR} \subseteq \mathcal{TBR}$, this implies that the χ-diameter and χ-radius (over $\mathcal{B}(n)$) is defined for all $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}, \mathcal{NNI}\}$. Moreover these inclusions imply the following proposition.

Proposition 1.3.4. *For any $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}, \mathcal{NNI}\}$, and any integer $n \geq 3$, we have $D_\chi(n) \geq R_\chi(n)$. Moreover,*

$$D_{\mathcal{NNI}}(\mathcal{B}(n)) \geq D_{r\mathcal{SPR}}(\mathcal{B}(n)) \geq D_{\mathcal{SPR}}(\mathcal{B}(n)) \geq D_{\mathcal{TBR}}(\mathcal{B}(n)),$$

and

$$R_{\mathcal{NNI}}(\mathcal{B}(n)) \geq R_{r\mathcal{SPR}}(\mathcal{B}(n)) \geq R_{\mathcal{SPR}}(\mathcal{B}(n)) \geq R_{\mathcal{TBR}}(\mathcal{B}(n)).$$

Example 1.3.5. For the trees A and B depicted in Figure 1.12 we prove that their $\mathcal{TBR}$, $\mathcal{SPR}$, $r\mathcal{SPR}$ and $\mathcal{NNI}$ distances are 1, 2, 3 and 4 respectively.

- Trivially $d_{\mathcal{TBR}}(A, B) \geq 1$ because $A \neq B$. To see that $(A, B) \in \mathcal{TBR}$, first note that A can be bisected into $T_1 = A/\{0, 1, 2\}$ and $T_2 = A/\{3, 4, 5, 6, 7\}$ by the deletion of a single edge. Then notice that T_1 and T_2 can be reconnected into B using a single reconnecting edge. This TBR move is depicted in Figure 1.5.

- By Proposition 1.2.4 we have $d_{\mathcal{SPR}}(A, B) \leq 2d_{\mathcal{TBR}}(A, B) = 2$ for any pair of trees. To see that $(A, B) \notin \mathcal{SPR}$, simply note that any SPR acting on A cannot make both $\{0, 1\}$ and $\{6, 7\}$ into cherries. Thus $d_{\mathcal{SPR}}(A, B) = 2$.

- As depicted in Figure 1.12, A can be transformed into B using the following three rSPR moves:

  1. Prune $\{3, 4\}$ and regraft it to the edge incident with the leaf labelled 2.
  2. Prune $\{5, 6\}$ and regraft it to the edge incident with the leaf labelled 4.
  3. Prune the leaf labelled 7 and regraft it to make a cherry with the leaf labelled 6.

  It is easy to check that $d_{r\mathcal{SPR}}(A, B) > 2$ using the Rooted Agreement Forest Lemma (Lemma 1.4.2). Suppose there exists a rooted agreement forest F three parts. No triple of labels from $\{3, 4, 5, 6, 7\}$ can exist in a single part of F. Therefore all three parts of F must contain an element from $\{3, 4, 5, 6, 7\}$. However, this would imply that the part containing 1 would also contain at least one element $\alpha \in \{3, 4, 5, 6, 7\}$, and this part would have to contain both 0 and 2. However, for any $3 \leq \alpha \leq 7$ we always have,

  $$A/\{0, 1, 2, \alpha\} \not\equiv B/\{0, 1, 2, \alpha\}.$$

  Therefore no such rooted agreement forest F exists.

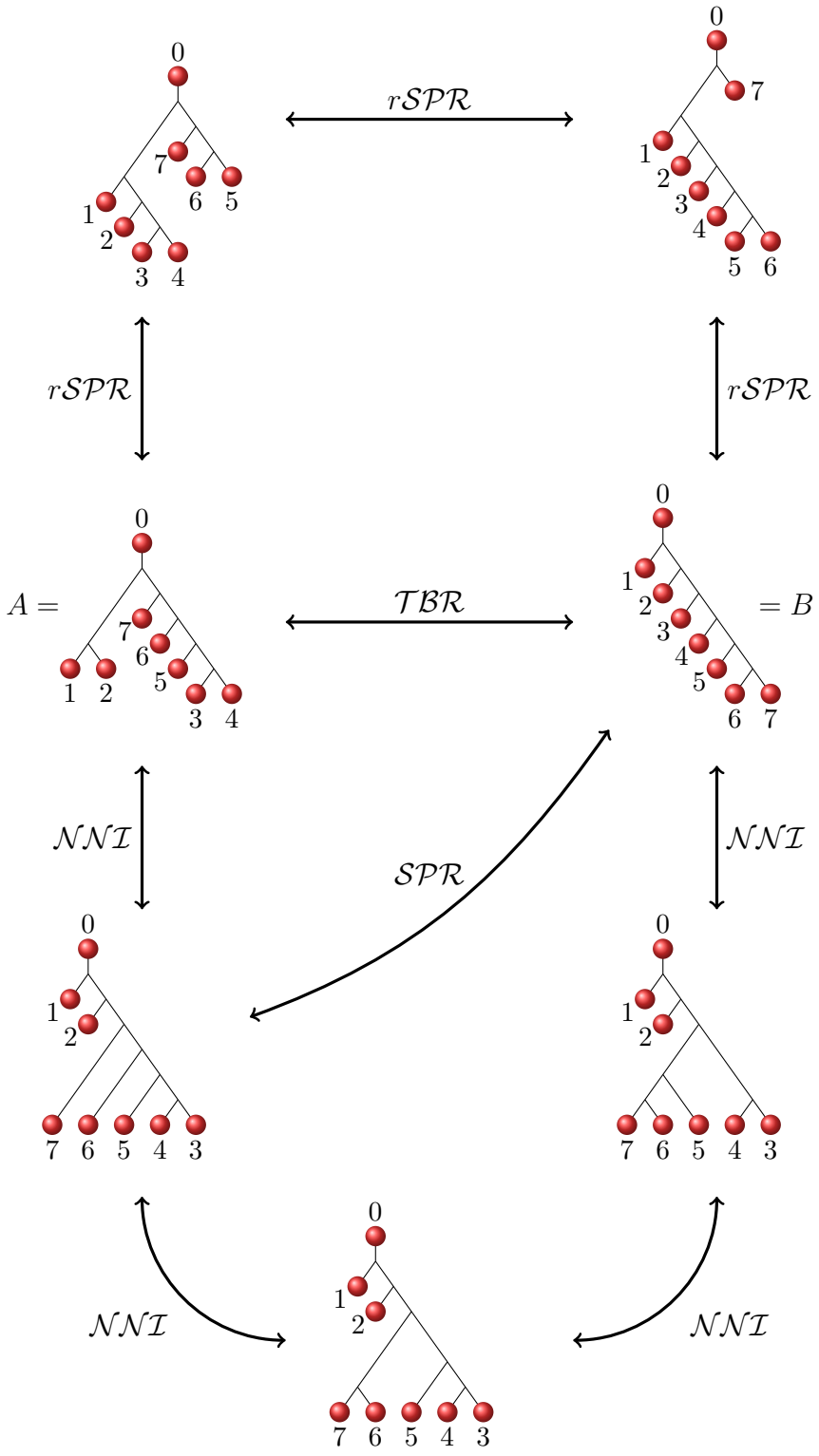


Figure 1.12: The $\mathcal{TBR}$, $\mathcal{SPR}$, $r\mathcal{SPR}$ and $\mathcal{NNI}$ distances between A and B are 1, 2, 3 and 4 respectively. Details given in Example 1.3.5.

- As we will see in Lemma 3.2.6, the $\mathcal{NNI}$-distance between any two trees is at least the Robinson-Foulds distance, and we can compute $d_{RF}(A, B)$ quickly:

$$\Sigma(A) = \{(12|034567), (012|34567), (0127|3456), (01267|345), (012567|34)\}$$
$$\Sigma(B) = \{(01|234567), (012|34567), (0123|4567), (01234|567), (012345|67)\}$$

  Therefore $4 = d_{RF}(A, B) \leq d_{\mathcal{NNI}}(A, B)$. To find a matching upper bound, Figure 1.12 depicts an example of how to obtain B from A using exactly 4 NNI moves.

1.4 Agreement forests

In this section, we define the notion of an *unrooted agreement forest*, and a *rooted agreement forest* for a set $\mathcal{A}$ of binary trees. The existence of an agreement forest with many edges (equivalently, with few components) can be interpreted as evidence that the trees in $\mathcal{A}$ are similar. We formally define the notion of an agreement forest here in Definition 1.4.1. The definition of a rooted agreement forest given here is equivalent to the definition given in [11]. The definition of a unrooted agreement forest given here is equivalent to the definition given in [21].

Definition 1.4.1. Let $\mathcal{A} \subset \mathcal{B}(n)$ be a set of $k \geq 2$ trees. An [unrooted] *agreement forest* of $\mathcal{A}$ is a partition $L_1, \ldots, L_m$ of $[n]_0$ (the label set) such that for all $A, B \in \mathcal{A}$:

- The subtrees $A|L_1$, $A|L_2$, ... , $A|L_m$ are disjoint, and
- $A/L_j \equiv B/L_j$ for $j = 1, \ldots m$.

Let $M_u(\mathcal{A})$ be the minimal possible value for m; the minimal number of parts in an agreement forest of $\mathcal{A}$. If in addition, we have

$$A/(L_j \cup \{0\}) \equiv B/(L_j \cup \{0\}), \qquad \text{for } j = 1, \ldots, m,$$

then the agreement forest is called a *rooted agreement forest*. Let $M_r(\mathcal{A})$ be the minimal number of parts in a rooted agreement forest of $\mathcal{A}$. When $k = 2$, we write $M_u(A, B)$ and $M_r(A, B)$ for $M_u(\{A, B\})$ and $M_r(\{A, B\})$ respectively.

We can always construct a trivial rooted agreement forest with $n-2$ parts (when $n \geq 3$) by setting $L_1 = \{0, 1, 2\}$ and $L_{i-1} = \{i\}$ for $i = 3, 4, 5, \ldots, n-1$. Therefore $M_r(A, B) \leq n - 2$ for any $A, B \in \mathcal{B}(n)$. Since a rooted agreement forests is a special case of an unrooted agreement forest, we must have $M_u(\mathcal{A}) \leq M_r(\mathcal{A}) \leq n-2$ for any

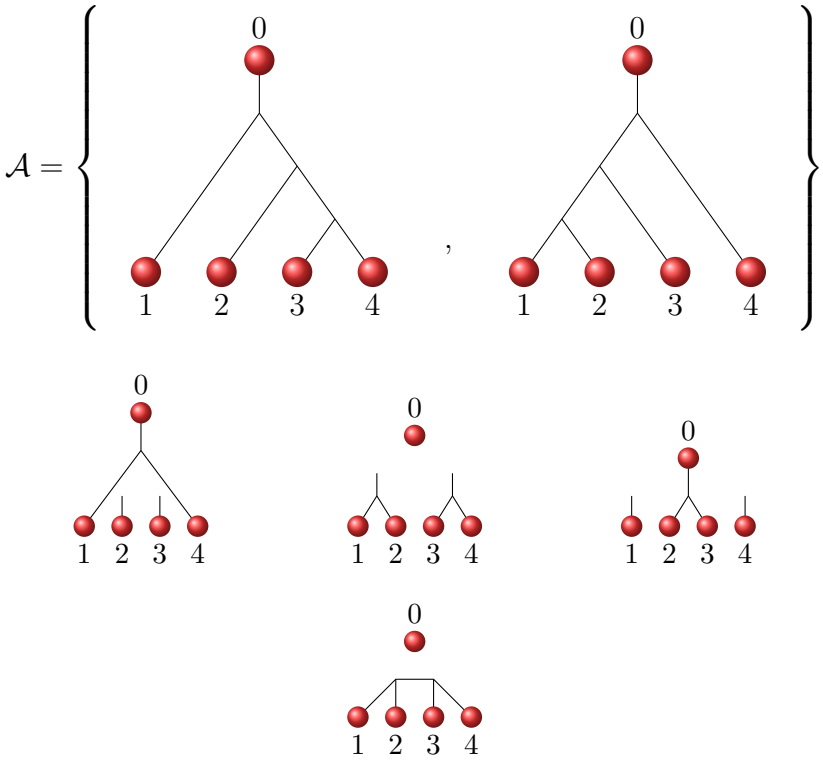


Figure 1.13: Three minimal rooted agreement forests and an unrooted agreement forest of $\mathcal{A} = \{\text{cat}(1,2,3,4), \text{cat}(4,3,2,1)\}$.

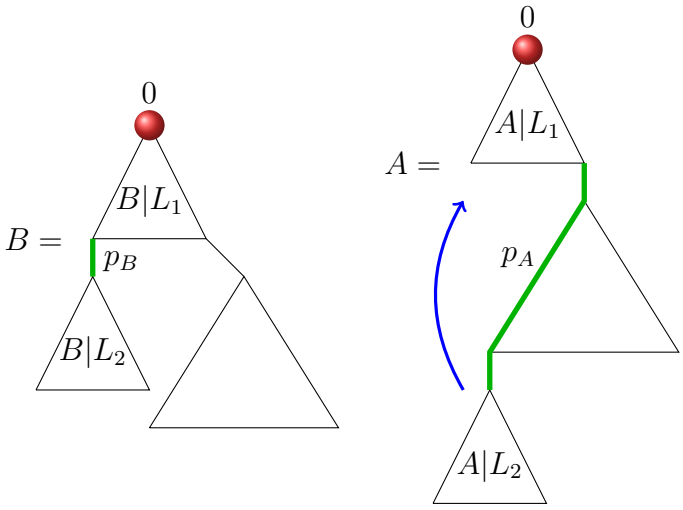


Figure 1.14: The construction in Lemma 1.4.2.

$\mathcal{A} \subseteq \mathcal{B}(n)$. For the set $\mathcal{A} = \{\text{cat}(1,2,3,4), \text{cat}(4,3,2,1)\} \subseteq \mathcal{B}(5)$, the unique minimal unrooted agreement forest and three minimal rooted agreement forests are depicted in Figure 1.13.

The following result is proved by Bordewich and Semple [11], building on the work of Hein et al [31]. A proof is provided here for completeness. This proof roughly follws the method presented in [11].

Lemma 1.4.2 (Rooted Agreement Forest Lemma)**.** *For any $A, B \in \mathcal{B}(n)$,*

$$M_r(A,B) = d_{r\mathcal{SPR}}(A,B) + 1.$$

Proof. Let $d = d_{r\mathcal{SPR}}(A,B)$ and let $A = A_0, A_1, \ldots, A_d = B$ be a sequence of trees $A_i \in \mathcal{B}(n)$ with $(A_{i-1}, A_i) \in r\mathcal{SPR}$ for all $i = 1, 2, \ldots, d$. If we perform all the *prunes* of these rSPR moves (and none of the *regrafts*), then the result will be a partition of the leaves into a valid rooted agreement forest for A and B. Therefore

$$M_r(A_0, A_d) \leq d + 1 = d_{r\mathcal{SPR}}(A_0, A_d) + 1.$$

Now we will show that $d_{r\mathcal{SPR}}(A,B)+1 \leq M_r(A,B)$, by induction on $m := M_r(A,B)$. For the base case $m = 1$, the only agreement forest with one component would be $A = B$ itself, thence $d_{r\mathcal{SPR}}(A,B) = 0$. For the inductive step $m \geq 2$, it suffices to show that there exists some $A' \in \mathcal{B}(X)$ such that $(A, A') \in r\mathcal{SPR}$ and $M_r(A', B) \leq m - 1$. Let $F = (L_1, \ldots, L_m)$ be an agreement forest for (A,B) of $m \geq 2$ parts and wlog $0 \in L_1$. We can construct A' from F explicitly:

- Wlog $B|L_2$ is one of the subtrees of B that is connected to $B|L_1$ by a path p_B, which does not intersect $B|L_i$ for any $i > 2$.

- Let $e = xy \in E(A)$ be the first edge on the path p_A from $A|L_1$ to $A|L_2$, such that y is in $A|L_2$ and x is not. If x is a leaf (i.e. if $L_1 = \{0\}$ and x is the root) then we do not perform an rSPR move (and let $A' = A$). Otherwise x is a tree vertex. Let a and b be the two neighbours of x other than y.

- Now we can prune A at e (that is, take xy from its current position on ab) and then regraft it to form A' such that

  $$A'/(L_1 \cup L_2) \equiv B/(L_1 \cup L_2).$$

  In the case that $L_1 = \{0\}$ is a singleton, this involves regrafting at the only edge incident with the root of A. Otherwise (if $|L_1| \geq 2$) this involves regrafting at an internal edge of $A|L_1$ corresponding to where the path p_B joins $B|L_1$.

We know that $\{L_1 \cup L_2, L_3, L_4, \ldots, L_m\}$ is an agreement forest for (A', B), because for all $i > 2$:

- $A'|(L_1 \cup L_2)$ does not intersect $A'|L_i$,

- $B|(L_1 \cup L_2) = B|L_1 \cup p_B \cup B|L_2$ does not intersect $B|L_i$,

- and $A'/(L_1 \cup L_2) \equiv B/(L_1 \cup L_2)$.

Hence $M_r(A', B) \leq m - 1$. By the inductive hypothesis, $d_{rSPR}(A', B) \leq m - 2$, and therefore $d_{rSPR}(A, B) \leq m - 1$ as required. □

Example 1.4.3. Let $A = \text{cat}(1, 2, \ldots, n-1)$ and $B = \text{cat}(n-1, n-2, \ldots, 2, 1)$ be the two caterpillar trees depicted in Figure 1.8. They differ by a single SPR changing the location of the root. Notice that $A/\{0, a, b, c\}$ is never isomorphic to $B/\{0, a, b, c\}$ because the leaf nearest to 0 in $A/\{0, a, b, c\}$ is the smallest of $\{a, b, c\}$ while the leaf nearest to 0 in $B/\{0, a, b, c\}$ is the largest. Therefore if $\{L_1, L_2, \ldots, L_m\}$ is a rooted agreement forest for $\{A, B\}$, with $0 \in L_1$, then $|L_1| \leq 3$ and $|L_i| \leq 2$ for all $i > 1$. So $n \leq 3 + 2(m-1)$ and therefore $m \geq \frac{n-1}{2}$. By the rooted agreement forest lemma (Lemma 1.4.2) this means $d_{rSPR}(A, B) \geq \frac{n-3}{2}$ and so

$$D_{r\mathcal{SPR}}(\mathcal{B}(n)) \geq \frac{n-3}{2}.$$

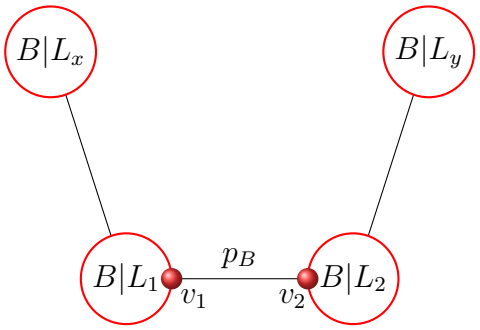


Figure 1.15: The construction in Lemma 1.4.4.

The next lemma is analogous to the rooted agreement forest lemma (Lemma 1.4.2). This lemma was proven by Ding et al. [21], but we present a proof here for completeness. The pattern of the proof follows that of the rooted agreement forest lemma, although the details are subtly different.

Lemma 1.4.4 (Unrooted Agreement Forest Lemma)**.** *If A and B are any two trees in $\mathcal{B}(n)$, then*

$$M_u(A, B) = d_{\mathcal{TBR}}(A, B) + 1.$$

Proof. Let $d = d_{\mathcal{TBR}}(A, B)$ and let $A = A_0, A_1, \ldots, A_d = B$ be a sequence of trees $A_i \in \mathcal{B}(n)$ satisfying $(A_{i-1}, A_i) \in \mathcal{TBR}$ for $i = 1, 2, \ldots, d$. If we perform all the *bisections* of these TBR moves (and none of the *reconnections*), then the result will be an agreement forest of at most $d + 1$ parts. Therefore

$$M_u(A_0, A_m) \leq d + 1 = d_{\mathcal{TBR}}(A_0, A_m) + 1.$$

Now we show $m = M_u(A, B) \geq d_{\mathcal{TBR}}(A, B) + 1$, by induction on m. For the base case $m = 1$, any agreement forest with one component must be $A = B$ itself, thence $d_{\mathcal{TBR}}(A, B) = 0$. For the inductive step $m \geq 2$, it suffices to show that there exists some A' with $(A, A') \in \mathcal{TBR}$, such that $M_u(A', B) \leq m - 1$. Let $F = \{L_1, \ldots, L_m\}$ be an agreement forest for (A, B). We can construct A' from F explicitly:

Step 1: Choose e, the bisection edge, arbitrarily from the edges not in $A|L_i$ for any i. We know that such an edge exists because $m \geq 2$.

The deletion of e forms two non-empty components A_1 and A_2. Note that each of the parts $A|L_i$ lie completely within one of A_1 or A_2.

Step 2: Now arbitrarily choose a pair of parts L_x and L_y, such that $A|L_x$ lies in A_1 and $A|L_y$ lies in A_2. Let p_B be the path in B from a vertex in $B|L_x$ to a vertex in $B|L_y$.

Step 3: Let v_1 be the last vertex on p_B which lies in a part $B|L_1$ (not necessarily distinct from $B|L_x$) such that $A|L_1$ lies in A_1. Let v_2 be first vertex after v_1 in p_B which lies in a part $B|L_2$ (not necessarily distinct from $B|L_y$). By definition, $A|L_2$ must lie in A_2 (see Figure 1.15).

So the path between v_1 and v_2 has no vertices in any $B|L_i$ except for its endpoints. Now v_1 corresponds to the midpoint of an edge in $B/L_1 \equiv A/L_1$, which in turn corresponds to a path p_1 in A_1 (or else $x = 1$ and L_1 is a singleton). Similarly v_2 corresponds to a path p_2 in A_2 (or $L_2 = \{v_2\}$).

Step 4: Construct A' to be the reconnection of A_1 and A_2 by an edge between the midpoint of an edge in p_1 and the midpoint of an edge in p_2 (if L_1 or L_2 is a singleton, then simply use its vertex as the endpoint of the reconnecting edge).

By construction, $(A, A') \in \mathcal{TBR}$ because we constructed A' by performing a TBR move on A. Moreover $A'/(L_1 \cup L_2) \equiv B/(L_1 \cup L_2)$, so

$$F' = \big\{(L_1 \cup L_2), L_3, \ldots, L_m\big\}$$

is an agreement forest for (A', B), of size $m - 1$. □

The $r\mathcal{SPR}$-distance and the $\mathcal{TBR}$-distance between a pair of trees is often used as a measure of the discrepancy between a pair of trees. So M_r and M_u can be seen as generalisation of this notion from pairs to arbitrary large sets of trees.

1.5 Agreement subtrees

Definition 1.5.1. An *agreement subtree* of a set of trees $\mathcal{A} = \{A_1, A_2, \ldots, A_k\} \in \mathcal{B}(n)$ is any binary tree $T \in \mathcal{B}(S)$ for some subset $S \subseteq [n]_0$ such that $A_i/S \equiv T$ for all $i = 1, 2, \ldots, k$. The number of leaves in a maximal agreement subtree of $\mathcal{A}$, is denoted $\mathrm{mast}(\mathcal{A})$, and the set of all maximal agreement subtrees of $\mathcal{A}$ is denoted $\mathrm{Mast}(\mathcal{A})$. To simplify notation, we sometimes write $\mathrm{mast}(A, B)$ and $\mathrm{Mast}(A, B)$ instead of $\mathrm{mast}(\{A, B\})$ and $\mathrm{Mast}(\{A, B\})$ respectively.

The maximum agreement subtree problem (the problem of determining $\mathrm{mast}(\mathcal{A})$ for a given $\mathcal{A} = \{A, B\} \subseteq \mathcal{B}(n)$) is attributed to Rosen [44] and Finden [28]. This problem can be solved in polynomial time [49].

Definition 1.5.2. For any integer $n \geq 2$, let $\mathrm{MAST}(n)$ denote the minimal size of a maximal agreement forest for a pair of trees $A, B \in \mathcal{B}(n)$. *i.e.*

$$\mathrm{MAST}(n) = \min_{A,B \in \mathcal{B}(n)} \Big\{ \mathrm{mast}(A, B) \Big\}.$$

Steel and Székely [48] proved that $\mathrm{MAST}(n) = \Omega(\log \log n)$. Martin and Thatte improved this bound to $\Omega(\sqrt{\log n})$ in [36]. An upper bound of $\mathcal{O}(\log n)$ can be found (even for unlabelled binary trees) by comparing a balanced tree to a caterpillar. Every subtree of a caterpillar must be a caterpillar, but the distance between any two leaves in a balanced tree is $\mathcal{O}(\log n)$. Hence

$$\Omega(\sqrt{\log n}) \leq \mathrm{MAST}(n) \leq \mathcal{O}(\log n).$$

The precise growth-rate of $\mathrm{MAST}(n)$ is still an open problem.

If A and B are chosen randomly from $\mathcal{B}(n)$ using the Yule-Harding or Uniform distribution, then $\mathbb{E}[\mathrm{mast}(A, B)]$ is known to be bounded above and below by a polynomial in n [10]. Explicitly, there exist positive constants λ_{YH} and λ'_{YH} such that for $A, B \in \mathcal{B}(n)$ chosen independently from the Yule-Harding distribution,

$$\lambda'_{YH} n^{0.344} < \mathbb{E}[\mathrm{mast}(A, B)] < \lambda_{YH} \sqrt{n}.$$

They also show that there exist constants λ_u and λ'_u such that for $A, B \in \mathcal{B}(n)$ chosen independently from the the uniform distribution,

$$\lambda'_u n^{1/8} < \mathbb{E}[\mathrm{mast}(A, B)] < \lambda_u \sqrt{n}.$$

The results in Lemma 1.5.3 are well-known, but they serve as a good introduction to the MAST problem.

Lemma 1.5.3. *For any two trees $A, B \in \mathcal{B}(n)$ and any $X \subseteq [n]_0$:*

1. $\mathrm{mast}(A, B) \geq \mathrm{mast}(A/X, B/X)$.
2. *If $n \geq 3$, then* $\mathrm{MAST}(n) \geq 3$.
3. *If $n \geq 6$, then* $\mathrm{MAST}(n) \geq 4$.

Proof. We prove each part separately.

1. For any $S \subseteq X$, we have $(A/X)/S \equiv A/S$ and $(B/X)/S \equiv B/S$.

$$\therefore \quad A/S \equiv B/S \qquad \text{iff} \qquad (A/X)/S \equiv (B/X)/S.$$

2. By the first part it suffices to consider only $n = 3$. This holds trivially because $|\mathcal{B}(3)| = 1$ (there is only one binary tree with 3 leaves).

3. By the first part we only need to consider when $n = 6$. In this case there are two possible underlying unlabelled trees that A and B might have: a caterpillar and a balanced tree. In either case we can construct $S = \{x_1, x_2, y_1, y_2\}$ such that $A/S = B/S$.

   - If A and B are both caterpillars, then we can divide each into disjoint subtrees of size 3:

     $$A = A_1 \sqcup A_2 \qquad \text{and} \qquad B = B_1 \sqcup B_2.$$

     Wlog $|L(A_1) \cap L(B_1)| \geq 2$ and so $|L(A_2) \cap L(B_2)| \geq 2$ (otherwise we would have $|L(A_1) \cap L(B_2)| \geq 2$ and $|L(A_2) \cap L(B_1)| \geq 2$). Then we can choose $x_1, x_2 \in L(A_1) \cap L(B_1)$ and $y_1, y_2 \in L(A_2) \cap L(B_2)$.
   - Otherwise one of A or B is not a caterpillar. Without loss of generality, assume B is balanced - so B has three cherries. Let $\{x_1, x_2\}$ be two leaves in a cherry of A, and then let $\{y_1, y_2\}$ be a cherry in B which doesn't include x_1 nor x_2.

□

We finish this section with a lemma relating maximal agreement subtrees and minimal agreement forests.

Lemma 1.5.4. *For any two binary trees $A, B \in \mathcal{B}(n)$, we have*

$$\text{mast}(A, B) + d_{\mathcal{TBR}}(A, B) \leq n.$$

Proof. For any agreement tree $A|S \equiv B|S$, we can always construct an agreement forest consisting of one large component S, and isolated singletons for all other leaves. If $S \in \text{Mast}(A, B)$ then $|S| = \text{mast}(A, B)$. This agreement forest has $n - |S| + 1$ components. Therefore (by Lemma 1.4.4) we have

$$d_{\mathcal{TBR}}(A, B) = M_u(A, B) - 1 \leq n - \text{mast}(A, B).$$

□

So if we can find a lower bound for $d_{\mathcal{TBR}}(A, B)$, then the above lemma can provide an upper bound for $\text{mast}(A, B)$. In particular, for any n, we will show that there exists $A, B \in \mathcal{B}(n)$, such that $d_{\mathcal{TBR}}(A, B) \geq n - \mathcal{O}(\sqrt{n})$ (Theorem 2.0.1), which by Lemma 1.5.4 implies $\text{MAST}(n) = \mathcal{O}(\sqrt{n})$. This is not very useful, since we already knew

$$\text{MAST}(n) = \mathcal{O}(\log n).$$

However, if A and B are chosen from $\mathcal{B}(n)$ with an exchangeable distribution, then (as we will see in Theorem 2.0.2), $\mathbb{E}[d_{\mathcal{TBR}}(A, B)] = n - \Theta(n^{2/3})$ and this implies

$$\mathbb{E}[\text{mast}(A, B)] = \mathcal{O}(n^{2/3}).$$

This is not an improvement on the polynomial bound given in [10]. However, this $\mathcal{O}(n^{2/3})$ upper bound applies to any exchangeable distribution.

Chapter 2

Subtree transfer operations on binary trees

In this chapter we focus on the maximal and typical χ-distances between a pair of trees chosen from $\mathcal{B}(n)$, for each $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}\}$. The values of the radius and diameter for $n \leq 9$ are given in Figure 5.1. The asymptotic values of the radius and diameter are given in the following theorem.

Theorem 2.0.1. *For each $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}\}$, both $D_\chi(n)$ and $R_\chi(n)$ are $n - \Theta(\sqrt{n})$.*

By Proposition 1.3.4, it suffices to give an upper bound for the $r\mathcal{SPR}$-diameter and a lower bound for the $\mathcal{TBR}$-radius, both of the form $n-\Theta(\sqrt{n})$. Suitable bounds are given explicitly, by setting $k = 2$ in Lemmas 2.2.1 and 2.4.3 respectively. In fact we prove the following stronger result for agreement forests. If k is a constant, then as $n \to \infty$,

$$\max_{\mathcal{A}} \Big\{ M_r(\mathcal{A}) \Big\} = n - \Theta(n^{1/k}) \qquad \text{and} \qquad \max_{\mathcal{A}} \Big\{ M_u(\mathcal{A}) \Big\} = n - \Theta(n^{1/k}),$$

where the maximums are taken over all $\mathcal{A} \subseteq \mathcal{B}(n)$ with $|\mathcal{A}| = k$.

Theorem 2.0.2. *Let $A, T \in \mathcal{B}(n)$ and $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}\}$ be fixed. Let B be obtained from T by a random permutation, σ, of the labels of the non-root leaves of T, where $\sigma \in S_{n-1}$ is chosen uniformly at random. If $X = d_\chi(A, B)$ then*

$$E[X] = n - \Theta(n^{2/3}).$$

Again Proposition 1.3.4, it suffices to give an upper bound for the expected $r\mathcal{SPR}$-distance and a lower bound for the expected $\mathcal{TBR}$-distance, both of the form $n - \Theta(n^{2/3})$. Suitable bounds are found by setting $k = 2$ in Lemmas 2.3.1 and 2.5.1

n	$R_{TBR}(n)$	$D_{TBR}(n)$	$R_{SPR}(n)$	$D_{SPR}(n)$	$R_{rSPR}(n)$	$D_{rSPR}(n)$
3	0	0	0	0	0	0
4	1	1	1	1	1	1
5	2	2	2	2	2	2
6	2	2	2	2	2	3
7	3	3	3	3	3	4
8	4	4	4	4	4	5
9	5	5	5	5	5	5

Figure 2.1: Small values of the χ-radius and χ-diameter of $\mathcal{B}(n)$ for all $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}\}$ and all $3 \leq n \leq 9$.

respectively. In fact we show the following stronger result for agreement forests. If k is a constant, then as $n \to \infty$,

$$\mathbb{E}[M_r(\mathcal{A})] = n - \Theta(n^{2/(k+1)}) \qquad \text{and} \qquad \mathbb{E}[M_u(\mathcal{A})] = n - \Theta(n^{2/(k+1)})$$

Where the maximums are taken over all $\mathcal{A} \in \mathcal{B}(n)$ such that $|\mathcal{A}| = k$. The third main result of this chapter is Theorem 2.0.3, which is proven Section 2.6

Theorem 2.0.3. *There exists constants $c_1 > c_2 > 0$ and $c_3 > 0$ such that, if X is the random variable as defined in Theorem 2.0.2 and μ denotes the expectation of X, then*

$$\mathbb{P}\left(n - c_1 n^{2/3} < X < n - c_2 n^{2/3}\right) \geq 1 - 2\exp\left(-c_3 n^{1/3}\right).$$

Theorem 2.0.2 shows that if A and B are chosen independently from any exchangeable distribution on $\mathcal{B}(n)$, then

$$E[d_\chi(A, B)] = n - \Theta(n^{2/3}).$$

for any $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}\}$. Moreover, Theorem 2.0.3 shows that this random distance $d_\chi(A, B)$ has a highly concentrated distribution.

In the next five sections (Sections 2.1 to 2.5) we prove Theorems 2.0.1 and 2.0.2. In fact, more general results involving agreement forests are established, and then the theorems regarding subtree-transfer-operation distances follow from the agreement forest lemmas (Lemma 1.4.4 and 1.4.2). This chapter concludes with a proof of Theorem 2.0.3 in Section 2.6.

2.1 Dividing a tree into almost-equal sized parts

The first two lemmas in this section provide an explicit method by which a tree can be divided into disjoint subtrees of a specific approximate size.

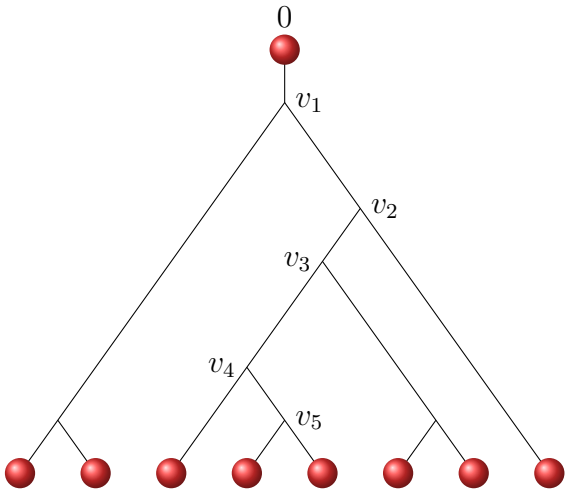


Figure 2.2: The construction in Lemma 2.1.1 for $n = 9$.

Lemma 2.1.1. *For an integers $N \geq n \geq 2$, let $S \subseteq [N]_0$ be a set of labels with $|S| = n$, let $T \in \mathcal{B}(S)$ be a binary tree with root $0 \in S$, and let a be a real number greater than 1 but at most $2(n-1)$. It is possible to remove a single edge from T, so that the number of leaves in the component not containing the root is in the interval $\left[\frac{a}{2}, a\right)$.*

Proof. For each vertex $x \in T$ let $g(x)$ be the set of leaves y such that x is on the path between the root and y. For every non-leaf vertex p,

$$g(p) = g(c_1) \sqcup g(c_2),$$

where $c_1, c_2 \in \Gamma(p)$ are the neighbours of p which are not on the path from p to the root. Moreover, if v is a leaf then $|g(v)| = 1$ (unless v is the root, in which case $g(v) = S$). Therefore, for each tree-node v_i, there exists a neighbour v_{i+1} such that

$$|g(v_i)| > |g(v_{i+1})| \geq \tfrac{1}{2}|g(v_i)|. \tag{2.1.1}$$

Now construct a path $(v_0, v_1, \ldots, v_m)$, such that v_0 is the root, v_1 is the neighbour of the root and equation (2.1.1) holds for each $i = 1, 2, \ldots, m-1$. This terminates when v_m is a leaf. So the sequence $|g(v_i)|$ decreases from

$$|g(v_1)| = n - 1 \geq \frac{a}{2} \quad \text{to} \quad |g(v_m)| = 1 < a,$$

and never decreases by a factor less than $\frac{1}{2}$ in a single step. Hence there must be some $1 \leq j \leq m$ such that

$$\frac{a}{2} \leq |g(v_j)| < a.$$

If edge $v_{j-1}v_j$ is deleted, then $g(v_j)$ is precisely the set of leaves in the component containing v_j. □

Lemma 2.1.2 is a direct consequence of Lemma 2.1.1.

Lemma 2.1.2. *For an integer $n \geq 2$, let $T \in \mathcal{B}(n)$ and let a be a real number greater than 1. It is possible to divide T into disjoint subtrees such that each leaf lies in a subtree, each subtree contains less than a leaves and every subtree that does not contain the root, contains at least $\frac{a}{2}$ leaves.*

Proof. (by induction) The case $a > n$ is trivial. If $n \geq a$, then we can use Lemma 2.1.1 iteratively to prune off subtrees with less than a (but at least $\frac{a}{2}$) leaves until the remaining tree has less than a leaves left (this remainder might have less than $\frac{a}{2}$ leaves). This final remaining tree contains the root. □

2.2 Upper bound for the rSPR diameter

In this section, we prove that the $rSPR$-distance between any two trees is never more than $n - \Omega(\sqrt{n})$. More generally we show that for a fixed integer k, a set of k binary trees in $\mathcal{B}(n)$, the size of a maximal rooted agreement forest is $n - \Omega(n^{1/k})$. In the proof, we find a $\Omega(n^{1/k})$ pairs of labels such that the paths between each pair are disjoint in each of the k trees. The rooted agreement forest we construct in the proof contains only parts of size 1 or 2.

Lemma 2.2.1. *For $k \geq 2$, and any set $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ of trees in $\mathcal{B}(n)$,*

$$M_r(\mathcal{A}) < n - \frac{1}{2(k+1)} \sqrt[k]{\frac{n}{k+1}}.$$

Proof. (by construction) Divide each A_i into t (possibly empty) disjoint connected parts,

$$A_i = A_{i,1} \sqcup A_{i,2} \sqcup \cdots \sqcup A_{i,t},$$

such that each part contains less than $\frac{2n}{t}$ leaves. This is possible by letting $a = \frac{2n}{t}$ in Lemma 2.1.2. Now, let us define a *token* to be any map $\tau : [k] \to [t]$. For any token τ, define

$$I(\tau) := \bigcap_{i=1}^{k} L(A_{i,\tau(i)}) \qquad \text{and} \qquad U(\tau) := \bigcup_{i=1}^{k} L(A_{i,\tau(i)}).$$

We say that τ is *nice* if $I(\tau) \geq 2$, and we say a pair τ, τ' are *friendly* if $\tau(i) \neq \tau'(i)$ for all i. Now if F is a set of pairwise friendly, nice tokens, then we can construct an agreement forest of $n - |F|$ parts in the following manner:

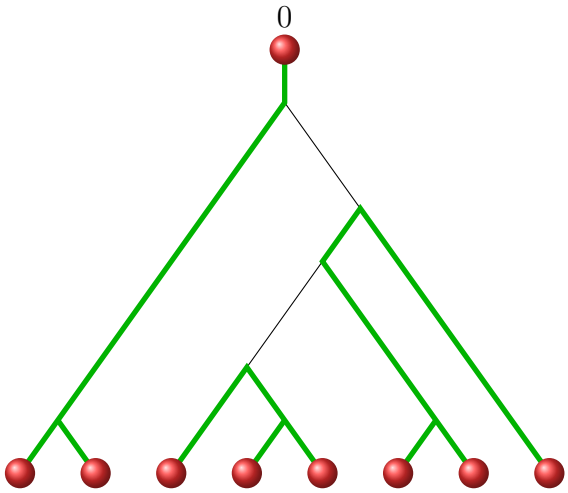


Figure 2.3: A subdivision of a tree into 3 disjoint subtrees, each with 3 leaves.

- for each $\tau \in F$, construct a part of size 2 using labels in $I(\tau)$,
- all other labels are put in their own part of size 1.

The paths between labels in each part of this partition will be disjoint because the tokens are friendly. Hence this is a valid agreement forest. So for any such set F, we have $M_r(\mathcal{A}) \leq n - |F|$. Now let F be any maximal set of friendly nice tokens (i.e. every nice token that is not in F is unfriendly with a token in F). Then for each label $x \in [n]_0$ either $x \in I(\sigma)$ for some not-nice token σ, or $x \in U(\tau)$ for some $\tau \in F$. So let us count:

- there are n labels in total,
- at most t^k not-nice tokens, each with at most 1 label in $I(\sigma)$, and
- less than ka labels in $U(\tau)$ for each token $\tau \in F$.

Therefore $t^k + ka|F| > n$. Substituting $t = \left\lfloor \sqrt[k]{\frac{n}{k+1}} \right\rfloor$, we can compute

$$|F| > \frac{n - t^k}{ka} \geq \frac{n - \left(\frac{n}{k+1}\right)}{ka} = \frac{t}{2(k+1)}.$$

Since the above inequality is strict and t, k are both integers, we can tighten the bound to:

$$|F| \geq \frac{t+1}{2(k+1)} > \frac{1}{2(k+1)} \sqrt[k]{\frac{n}{k+1}}$$

and we have $M(\mathcal{A}) \leq n - |F|$ as required. □

Set $k = 2$ in Lemma 2.2.1 and then apply the rooted agreement forest lemma (Lemma 1.4.2). We obtain,

$$D_{rSPR}(n) < n - \tfrac{1}{6\sqrt{3}}\sqrt{n} - 1,$$

which is a suitable upper bound in Theorem 2.0.1.

2.3 Upper bound for the expectation

In this section, we compute an upper bound of $n - \Omega(n^{2/3})$ for the expectation of the rSPR distance between two binary trees chosen randomly from an exchangeable distribution over $\mathcal{B}(n)$. More generally, For a set, $\mathcal{A}$, of k binary trees independently chosen from $\mathcal{B}(n)$ via an exchangeable distribution, we show that $\mathbb{E}[\mathcal{M}_r(\mathcal{A})]$ is $n - \Omega(n^{2/(k+1)})$.

Lemma 2.3.1. *Let $k \geq 2$ be a fixed integer and let $\mathcal{T}$ be an arbitrary set of k trees in $\mathcal{B}(n)$. If $\mathcal{A}$ is obtained by performing a random permutation of the non-root leaf-labels of each $T \in \mathcal{T}$ (permutations chosen independently and uniformly from S_{n-1}), then*

$$\mathbb{E}[M_r(\mathcal{A})] \leq n - \Omega\left(n^{2/(k+1)}\right)$$

as $n \to \infty$.

Proof. Set $b = \left\lfloor \sqrt[k+1]{\frac{n^{k-1}}{2k}} \right\rfloor$ and $s = \left\lfloor \frac{n}{2b} \right\rfloor$. Now for each $T \in \mathcal{T}$, let $T_1, T_2, \ldots, T_s$ be disjoint subtrees of T such that the number of leaves in each T_i is exactly b and none of them contain the root. This is possible by Lemma 2.1.2 (set $a = 2b$) and taking subtrees as necessary. Let $A \in \mathcal{A}$ be the tree obtained by the random labelling of T, and let $A_1, A_2, \ldots, A_s$ be the subtrees of A corresponding to $T_1, T_2, \ldots, T_s$ respectively. Call $\{x_1, \ldots, x_j\} \subseteq \{1, 2, \ldots, n-1\}$ a *good* set if for each $A \in \mathcal{A}$, the leaves labelled $x_1, \ldots, x_j$ all lie in the same A_i. We say that a good pair of distinct labels $\{x_1, x_2\}$ is *spoiled* if for some $A \in \mathcal{A}$, there is another good pair in the same subtree A_i of A as x_1 and x_2. Note that a good pair $\{x_1, x_2\}$ might be spoiled by a good pair $\{x_1, x_3\}$ (with $x_2 \neq x_3$) or by a disjoint good pair $\{x_3, x_4\}$. For good pairs $\{x_1, x_2\}$ and $\{x_3, x_4\}$, there only needs to be one A (such that x_1, x_2, x_3, x_4 are all on the same subtree A_i) to mean that the pairs spoil each other. We call a pair *super* if it is good but not spoiled. If $\{x_1, x_2\}$ is chosen uniformly at random from the set of pairs of non-root leaf-labels, then

$$\mathbb{P}\big(\{x_1, x_2\} \text{ is good}\big) = \left(\frac{sb}{n-1} \times \frac{b-1}{n-2}\right)^k$$

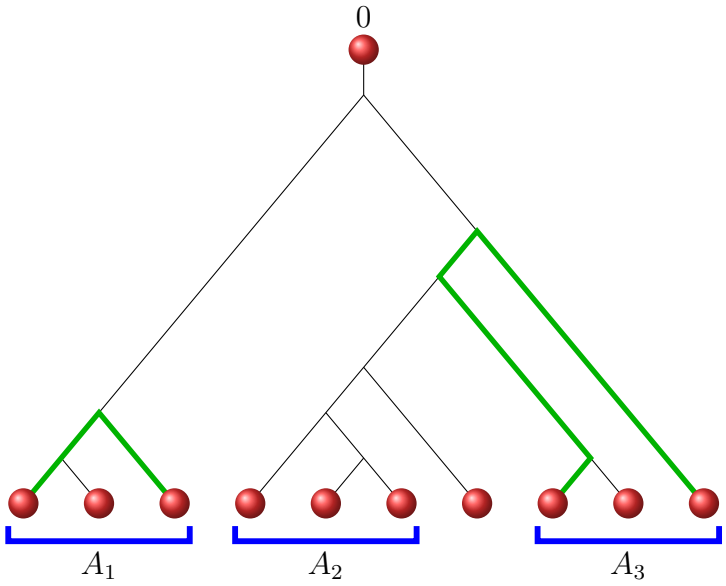


Figure 2.4: The paths between the two leaves in a super pair are disjoint because each super pair is contained within a single A_i and no two super pairs share the same A_i.

because for any $A \in \mathcal{A}$ the probability that $x_1 \in \bigcup_{i=1}^{s} L(A_i)$ is $\frac{sb}{n-1}$, and given that $x_1 \in L(A_i)$ the probability that $x_2 \in L(A_i)$ is $\frac{b-1}{n-2}$. Similarly, if $\{x_1, x_2, x_3\}$ is chosen uniformly at random from the set of triples of non-root leaf-labels, then

$$\mathbb{P}\big(\{x_1, x_2, x_3\} \text{ is good}\big) = \left(\frac{sb}{n-1} \times \frac{b-1}{n-2} \times \frac{b-2}{n-3}\right)^k,$$

because the probability that $\{x_1, x_2\} \subseteq \bigcup_{i=1}^{s} L(A_i)$ for some i is $\frac{sb}{n-1} \times \frac{b-1}{n-2}$, and under the assumption that $\{x_1, x_2\} \subseteq L(A_i)$ the probability that $x_3 \in L(A_i)$ is $\frac{b-2}{n-3}$. Since $\{x_1, x_2, x_3\}$ is good if and only if $\{x_1, x_2\}$ and $\{x_1, x_3\}$ are both good, we can see that

$$\mathbb{P}\big(\{x_1, x_3\} \text{ good} \big| \{x_1, x_2\} \text{ good}\big) = \left(\frac{b-2}{n-3}\right)^k \leq \left(\frac{b}{n}\right)^k. \tag{2.3.1}$$

Now fix a good pair $\{x_1, x_2\}$ with $\{x_1, x_2\} \subseteq L(A_i)$ for some $A \in \mathcal{A}$. If x_3 and x_4 are chosen from $L(A_i) \setminus \{x_1, x_2\}$, we compute an upper bound on the probability that $\{x_3, x_4\}$ is a good pair. For each $A' \in \mathcal{A} \setminus A$, the probability that $x_3 \in \bigcup_{j=1}^{s} L(A'_j)$ is $\frac{sb-2}{n-3}$ and given that $x_3 \in L(A'_j)$, the probability that $x_4 \in L(A'_j)$ is at most $\frac{b-1}{n-4}$. Therefore (for sufficiently large n),

$$\mathbb{P}\big(\{x_3, x_4\} \text{ good} \big| \{x_1, x_2\} \text{ good}\big) \leq \left(\frac{sb-2}{n-3} \times \frac{b-1}{n-4}\right)^{k-1} \leq \left(\frac{b}{n}\right)^{k-1}. \tag{2.3.2}$$

Now let Y denote the set of all pairs $\{y_1, y_2\}$ distinct from $\{x_1, x_2\}$, which lie in the same A_i as $\{x_1, x_2\}$ for some $A \in \mathcal{A}$. Using the union bound, the probability that $\{x_1, x_2\}$ is super is bounded below by

$$\mathbb{P}\Big(\{x_1, x_2\} \text{ good}\Big) \left[1 - \sum_{\{y_1, y_2\} \in Y} \mathbb{P}\Big(\{y_1, y_2\} \text{ good} \Big| \{x_1, x_2\} \text{ good}\Big)\right] \tag{2.3.3}$$

where the sum ranges over all $\{y_1, y_2\} \in Y$. There are at most $k\binom{b-2}{2}$ pairs such that $\{x_1, x_2\} \cap \{y_1, y_2\} = \emptyset$, and at most $2k(b-2)$ such pairs with $|\{x_1, x_2\} \cap \{y_1, y_2\}| = 1$. Substituting Equations (2.3.1) and (2.3.2) into (2.3.3) yields the lower bound

$$\left(\frac{sb}{n-1} \times \frac{b-1}{n-2}\right)^k \left[1 - k\binom{b-2}{2} \times \left(\frac{b}{n}\right)^{k-1} - 2k(b-2) \times \left(\frac{b}{n}\right)^{k-1}\right].$$

For fixed k, the values $b = \Theta\left(n^{\frac{k-1}{k+1}}\right)$ and $s = \Theta\left(n^{\frac{2}{k+1}}\right)$ were chosen so that $k\binom{b-2}{2} \times \left(\frac{b}{n}\right)^{k-1}$ and $2k(b-2) \times \left(\frac{b}{n}\right)^{k-1}$ are each less than $\frac{1}{4}$ for sufficiently large n. Therefore

$$\mathbb{P}(\{x_1, x_2\} \text{ super}) \geq \frac{1}{2}\left(\frac{sb}{n-1} \times \frac{b-1}{n-2}\right)^k = \Omega\big(n^{-2k/(k+1)}\big).$$

Therefore, if S is the number of super pairs, then

$$\mathbb{E}[S] = \binom{n-1}{2}\mathbb{P}\big(\{x_1, x_2\} \text{ super}\big) = \Omega\left(n^{2/(k+1)}\right).$$

Now let us partition the labels such that each super pair forms a part of size two (super pairs are disjoint by definition, see Figure 2.4), and all other labels are in individual parts. To prove that this partition is a valid agreement forest, it suffices to show that for any super pairs $\boldsymbol{x}$ and $\boldsymbol{y}$, the paths $A|\boldsymbol{x}$ and $A|\boldsymbol{y}$ are disjoint for all $A \in \mathcal{A}$. However $A|\boldsymbol{x}$ is contained within A_i and $A|\boldsymbol{y}$ is contained within A_j for some $i \neq j$. Since A_i and A_j are disjoint, this partition must be a valid rooted agreement forest. It has $n - S$ components, so $M_r(\mathcal{A}) \leq n - S$ and hence

$$\mathbb{E}[M_r(\mathcal{A})] \leq n - \mathbb{E}[S] = n - \Omega\big(n^{2/(k+1)}\big).$$

□

Now consider Lemma 2.3.1 with $k = 2$, so $\mathcal{T} = \{T_1, T_2\}$ and $\mathcal{A} = \{A_1, A_2\}$. If we apply the same permutation, σ, to the leaves of both A_1 and A_2, then the distance $d_{rSPR}(A_1, A_2)$ remains unchanged. In particular let σ be the inverse of the permutation by which A_1 was obtained from T_1. So, if $B \in \mathcal{B}(n)$ is the binary tree

formed by applying σ to the labels of the leaves of A_2, then B is uniformly distributed over the set of trees obtainable by a permutation of the non-root leaves of T_2 and

$$d_{rSPR}(A_1, A_2) = d_{rSPR}(T_1, B).$$

We can now set $A = T_1$ and $X = d_{rSPR}(A, B)$ apply the rooted agreement forest lemma (Lemma 1.4.2). We obtain

$$\mathbb{E}[X] \leq n - \Omega\big(n^{2/3}\big).$$

2.4 Lower bound for the TBR radius

In this section we present a result concerning permutations (Lemma 2.4.1) and a property of $\mathcal{A}$ that guarantees large $M_u(\mathcal{A})$ (Lemma 2.4.2). These two Lemmas are the two crucial steps required for the proof of the main result of this section (Lemma 2.4.3).

Lemma 2.4.1. *Let n, b and k be positive integers such that $2 \leq b \leq \sqrt[k]{n}$. There exist k permutations $\{\phi_j : [n]_0 \to [n]_0\}_{j=1}^{k}$ such that for any distinct $x, y \in [n]_0$ there exists some i such that*

$$|\phi_i(x) - \phi_i(y)| > b^{k-1} - 2b^{k-3}.$$

Proof. (by construction) Represent each $x \in [n]_0$ by the k-tuple $(x_1, x_2, \ldots, x_k)$ of non-negative integers such that

- $x = x_1 b^{k-1} + x_2 b^{k-2} + \cdots + x_k$, and
- $0 \leq x_i < b$ for all $1 < i \leq k$.

If $x < b^k$, then this is exactly representing x in base b. For larger x, we allow the coefficient of b^{k-1} to exceed $b - 1$. Now let ϕ_1 be the identity permutation, and for each $1 < j \leq k$ we define ϕ_j so that

$$\phi_j(x) > \phi_j(y) \quad \text{iff} \quad \begin{cases} x_j < y_j & \text{or} \\ x_j = y_j \text{ and } x > y. \end{cases}$$

See Figure 2.5 for an illustration. If $d_j = x_j - y_j$ then for $i \geq 2$ and $\phi_i(x) > \phi_i(y)$ (i.e. $d_i \leq 0$) we have

$$\phi_i(x) - \phi_i(y) \geq -d_i b^{k-1} + \sum_{j=1}^{i-1} d_j b^{k-j-1} + \sum_{j=i+1}^{k} d_j b^{k-j}. \tag{2.4.1}$$

This is an equality when $n = b^k$ (i.e. when n is a perfect k^{th} power), and increasing n can only increase the number of labels z such that $\phi_i(z)$ is between $\phi_i(x)$ and $\phi_i(y)$.

Now we fix distinct $x, y \in [n]_0$ and without loss of generality assume $x_1 \geq y_1$. There are four cases:

1. $|x_i - y_i| > 1$ for some i
2. $x_1 = y_1$
3. $x_1 > y_1$ and $x_i \geq y_i$ for all i.
4. $x_1 > y_1$ and $x_j < y_j$ for some $j \geq 2$.

We treat each of these cases separately.

Case 1: All z such that $z_i = \min(x_i, y_i) + 1$ must have $\phi_i(z)$ between $\phi_i(x)$ and $\phi_i(y)$. There are at least b^{k-1} such z.

Case 2: W.l.o.g. $d_i = -1$ and $d_j = 0$ for $j < i$ for some $i \geq 2$. Substituting this into equation (2.4.1) yields

$$\begin{aligned}\phi_i(x) - \phi_i(y) &\geq b^{k-1} - b^{k-i-1} - b^{k-i-2} - \cdots - 1 \\ &> b^{k-1} - 2b^{k-3}.\end{aligned}$$

Case 3: If $d_1 = 1$ and $d_j \geq 0$ for all j, then $|\phi_1(x) - \phi_1(y)| = x - y \geq b^{k-1}$.

Case 4: We have $d_1 = 1$ and there is some $i \geq 2$, $d_i = -1$ and $d_j \geq 0$ for all $j < i$. Substituting this into equation (2.4.1) yields

$$\begin{aligned}\phi_i(x) - \phi_i(y) &\geq b^{k-1} + b^{k-2} - b^{k-3} - b^{k-4} - \cdots - 1 \\ &> b^{k-1}.\end{aligned}$$

□

Lemma 2.4.2. *Let n, k, t be positive integers and let $\mathcal{A} \subseteq \mathcal{B}(n)$ be a set of k trees. For each $A \in \mathcal{A}$ let $A = A_1 \sqcup A_2 \sqcup A_3 \sqcup \cdots$ be a partition of the vertices of A into at most t connected parts. For any $z \geq 0$, if for all but z pairs of distinct labels $x, y \in [n]_0$ there is at least one $A \in \mathcal{A}$ such that the leaves labelled with x and y are in different parts of A, then*

$$M(\mathcal{A}) \geq n - kt + k - z.$$

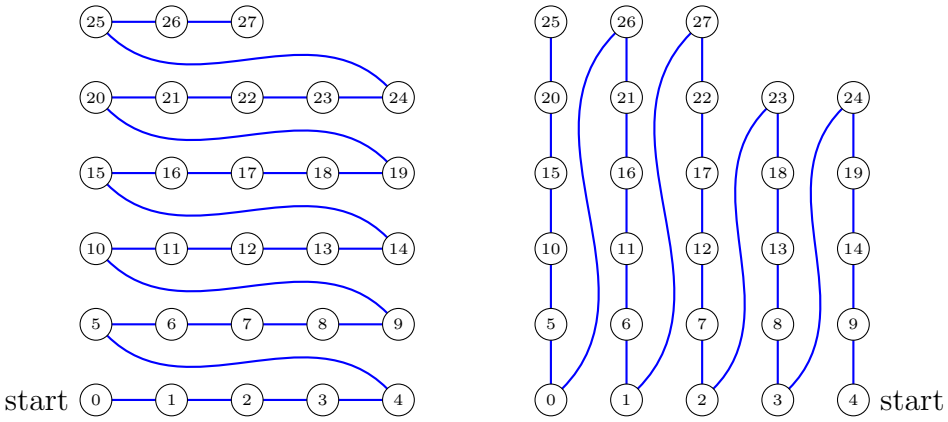


Figure 2.5: For $n = 28$, $k = 2$ and $b = 5$, the constructions of permutations ϕ_1 (left) and ϕ_2 (right) as constructed in Lemma 2.4.1 are shown above. The depicted paths trace from $\phi_i(0)$ to $\phi_i(n-1)$ in order.

Proof. Let $\{L_1, L_2, \ldots, L_m\}$ be an agreement forest of $\mathcal{A}$ and let $F = A/L_1 \sqcup A/L_2 \sqcup \cdots \sqcup A/L_m$ be a forest with leaves labelled by $[n]_0$ for some $A \in \mathcal{A}$ (by Definition 1.4.1 it doesn't matter which $A \in \mathcal{A}$ we choose). Each tree $A \in \mathcal{A}$ can be divided into the forest $A = A_1 \sqcup A_2 \sqcup A_3 \sqcup \cdots$ by deleting at most $t-1$ edges. Let these edges be called *hot* edges. There is a total of at most $k(t-1)$ hot edges (at most $t-1$ in each $A \in \mathcal{A}$). An edge in A might not correspond to any edge in F, moreover there may be several edges that correspond to the same edge in F. In any case, each edge corresponds to at most one edge in F. In particular this holds for hot edges. Therefore, if we delete all the edges in F corresponding to hot edges then we delete at most $k(t-1)$ edges.

The result of the deletion of these edges would be a forest of at least $n-z$ components because z is an upper bound on the number of pairs of leaves in the same component. Therefore F must have had $m \geq n - z - k(t-1)$ components. □

Lemma 2.4.3. *Let $k \geq 2$ and $n \geq (3k)^k$ be integers. If $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ is a set of binary trees each with n leaves, then there exists $\mathcal{A} = \{A_1, A_2, \ldots, A_k\} \subseteq \mathcal{B}(n)$ such that A_i is a labelling of the leaves of T_i, and*

$$M(\mathcal{A}) \geq n - 3k\sqrt[k]{n}.$$

Proof. First set

$$b = \lfloor \sqrt[k]{n} \rfloor \ , \quad t = \left\lceil \frac{2n}{b^{k-1} - 2b^{k-3}} \right\rceil \quad \text{and} \quad a = \frac{2n}{t} \leq b^{k-1} - 2b^{k-3}.$$

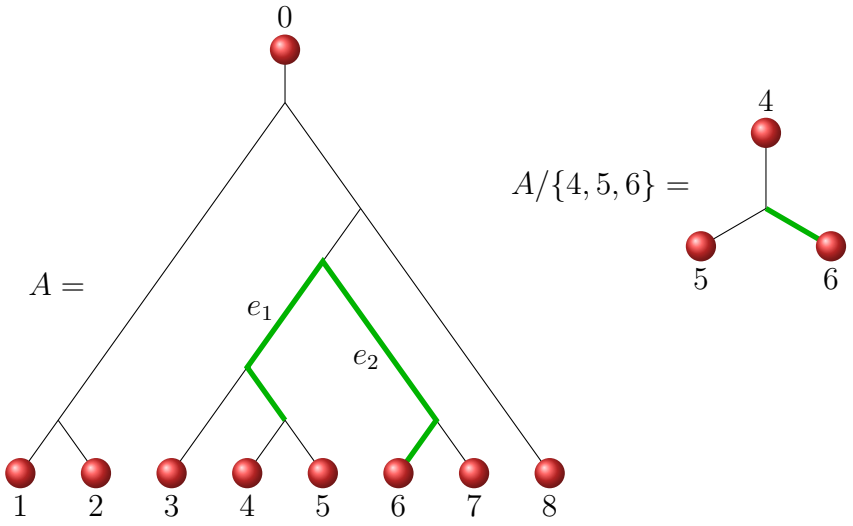


Figure 2.6: Edges e_1, e_2 in A correspond to the same edge in $A/\{4,5,6\}$.

Then divide each T_i into at most t connected parts, $T_i = T_{i1} \sqcup T_{i2} \sqcup T_{i3} \sqcup \cdots$ (using Lemma 2.1.2) so each part contains less than a leaves. Now it suffices to label the leaves of each T_i such that for any pair of distinct labels $x, y \in [n]_0$, there is some i such that x and y are in different parts of T_i, and then apply Lemma 2.4.2 (with $z = 0$).

To achieve this: first (by Lemma 2.4.1) find k permutations of $[n]_0$ such that for any two distinct labels $x, y \in [n]_0$ there is some $1 \leq i \leq k$ such that $|\phi_i(x) - \phi_i(y)| \geq a$. Then assign labels to the leaves of T_i in the order given by ϕ_i, firstly to all the leaves of T_{i1}, then T_{i2}, then T_{i3}, and so on. For any distinct x and y, consider a permutation ϕ_i in which $|\phi_i(x) - \phi_i(y)| \geq a$. If some T_{ij} contained both x and y, then it would have to also contain the labels

$$\{z : \phi_i(x) > \phi_i(z) > \phi_i(y) \text{ or } \phi_i(x) < \phi_i(z) < \phi_i(y)\},$$

but this is impossible since each T_{ij} has size less than a. Thus by Lemma 2.4.2,

$$M(\mathcal{A}) \geq n - k(t-1).$$

For constant k, one can easily observe that $t \sim 2\sqrt[k]{n}$ as $n \to \infty$. For an explicit

bound (using $n \geq (3k)^k$), we can show $t \leq 3\sqrt[k]{n} + 1$, in the following manner:

$$\begin{aligned} b = \lfloor \sqrt[k]{n} \rfloor &> \sqrt[k]{n}\left(1 - \frac{1}{\sqrt[k]{n}}\right) \\ &\geq \sqrt[k]{n}\left(1 - \frac{1}{3k}\right) \\ \implies b^2 - 2 &> \sqrt[k]{n^2}\left(\left(1 - \frac{1}{3k}\right)^2 - \frac{1}{9k^2}\right) \\ &> \sqrt[k]{n^2}\left(1 - \frac{1}{k}\right). \end{aligned}$$

Therefore $b^{k-3}(b^2-2) > n^{k-1}\left(1-\frac{1}{3k}\right)^{k-3}\left(1-\frac{1}{k}\right) \geq n^{k-1}\left(1-(k-3)\frac{1}{3k}-\frac{1}{k}\right)$. Then, by definition, $t = \left\lceil \frac{2n}{b^{k-3}(b^2-2)} \right\rceil$ and so

$$t \leq \left\lceil \frac{2n}{n^{(k-1)/k}\left(1-\frac{k}{3k}\right)} \right\rceil \leq 3n^{1/k} + 1.$$

□

If $\mathcal{A}' = \{A'_1, \ldots, A'_k\}$ is obtained from $\mathcal{A}$ by a permutation, π, of the labels (i.e. the leaf labeled x in A_i is labelled $\pi(x)$ in A'_i for all $x \in [n]_0$ and all $i \in [k]$) then $M(\mathcal{A}') = M(\mathcal{A})$. Therefore in the above Lemma, we could have constructed $\mathcal{A}$ so that one of the trees has a particular labelling, say A_1. Now if we set $k = 2$ and apply the unrooted agreement forest lemma (1.4.4) then

$$R_{TBR}(n) \geq n - 6\sqrt{n} - 1,$$

for all integers $n \geq 36$. Actually this holds for all integers $n \geq 2$ because, for $n < 36$, this bound is trivial since $R_{TBR}(n) \geq 0 > n - 6\sqrt{n} - 1$.

2.5 Lower bound for the expectation

Lemma 2.5.1. *Let $k \geq 2$ and $n \geq 2$ be integers and let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\} \subseteq \mathcal{B}(n)$ be a set of k binary trees. For each $i = 1, 2, \ldots, k$, let A_i be a random relabelling of the non-root labels of T_i (chosen independently and uniformly from S_n). If $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$, then*

$$\mathbb{E}[M(\mathcal{A})] > n - (2k+1)n^{2/(k+1)}.$$

Proof. Set $a = n^{\frac{k-1}{k+1}} + 1$ and $t = \lceil 2n/a \rceil$ so that $t - 1 < \frac{2n}{a} < 2n^{2/(k+1)}$. Now, using Lemma 2.1.2, divide each A_i into t (possibly empty) connected parts, so that each part has less than a leaves: $A_i = A_{i1} \sqcup \cdots \sqcup A_{it}$. For distinct $x, y \in [n]_0$, notice that

$$\mathbb{P}\big(y \in L(A_{ij}) | x \in L(A_{ij})\big) = \frac{|L(A_{ij})| - 1}{n-1} < \frac{a-1}{n-1}.$$

Now let Z be the set of pairs of distinct labels $\{x, y\}$ such that for each i, x and y are in the same part A_{ij}. Explicitly $Z := \big\{\{x, y\} \ : \ x \neq y \text{ and } \forall i \ \exists j \ \text{ s.t. } \ x, y \in L(A_{ij})\big\}$. Since the labellings are chosen independently, $\mathbb{P}(\{x, y\} \in Z) < \left(\frac{a-1}{n-1}\right)^k$ for all $x, y \in [n]_0$. Now let $z = |Z|$. By the linearity of expectation

$$\mathbb{E}[z] = \binom{n}{2}\mathbb{P}(\{x, y\} \in Z) < \binom{n}{2}\left(\frac{a-1}{n-1}\right)^k < n^{2/(k+1)},$$

for large enough n. Since $t - 1 < 2n^{2/(k+1)}$ and $\mathbb{E}[z] < n^{2/(k+1)}$, we can applying Lemma 2.4.2 to get

$$\begin{aligned}\mathbb{E}[M(\mathcal{A})] &\geq n - k(t-1) - \mathbb{E}[z] \\ &> n - (2k+1)n^{2/(k+1)}.\end{aligned}$$

□

Setting $k = 2$ in Lemma 2.5.1 gives $\mathbb{E}[M(A_1, A_2)] \geq n - 5n^{2/3}$. Applying the unrooted agreement forest lemma (1.4.4) to get

$$\mathbb{E}[d_{TBR}(A_1, A_2)] \geq n - 5n^{2/3} - 1.$$

If we apply the same permutation, σ, to the leaves of both A_1 and A_2, then the distance $d_{TBR}(A_1, A_2)$ remains unchanged. In particular let σ is chosen to be the inverse of the permutation by which A_1 was obtained from T_1. So, if $B \in \mathcal{B}(n)$ is the binary tree formed by applying σ to the labels of the leaves of A_2, then B is uniformly distributed over the set of trees obtainable by a permutation of the leaves of T_2 and $d_{TBR}(T_1, B) = d_{TBR}(A_1, A_2)$. Therefore for $X = d_{TBR}(T_1, B)$ we have

$$\mathbb{E}[X] \geq n - 5n^{2/3} - 1.$$

This is a suitable lower bound for Theorem 2.0.2 (letting $A = T_1$).

2.6 Concentration results

Let $n \geq 2$ be a fixed integer and let $A, T \in \mathcal{B}(n)$ be fixed binary trees. Let σ be a random permutation of $\{1, 2, \ldots, n-1\}$ chosen uniformly from S_{n-1}, and let

$B \in \mathcal{B}(n)$ be the random binary tree obtained by performing σ on the non-root leaves of T. In this section, for each $\chi \in \{TBR, SPR, rSPR\}$, we investigate the distribution of the random variable $X = d_\chi(A, B)$. Specifically we will show that X is highly concentrated about its mean, using the following result from [37] (Theorem 7.4 applied to Example 7.2).

Lemma 2.6.1 ([37])**.** *Let S_{n-1} denote the group of all permutations of $\{1, 2, \ldots, n-1\}$ and let $k > 0$ be a fixed constant. Let $f : S_{n-1} \to \mathbb{R}$ be a function such that for any $\sigma \in S_{n-1}$ and any transposition τ, we have*

$$|f(\sigma \circ \tau) - f(\sigma)| \leq k.$$

Let σ be chosen uniformly at random from S_{n-1} and let X be the random variable $X = f(\sigma)$. If μ denotes $\mathbb{E}[X]$ then for any $\alpha > 0$, we have

$$\mathbb{P}(|X - \mu| \geq \alpha) \leq 2 \exp\left(\frac{-2\alpha^2}{(n-2)k^2}\right).$$

For any binary tree $T \in \mathcal{B}(n)$, if T' is the result of a transposition of the labels of two non-root leaves of T, then $d_\chi(T, T') \leq 2$ because we can obtain T' from T by swapping the two leaves with 2 rSPR moves. Therefore we can apply Lemma 2.6.1 to the random variable $X = d_\chi(A, B)$ with $k = 2$. By Theorem 2.0.2 we know there must be constants $\lambda_1 > \lambda_2 > 0$ such that

$$n - \lambda_1 n^{2/3} < \mu < n - \lambda_2 n^{2/3}$$

for all integers $n \geq 2$. If we let $\alpha = \frac{1}{2}\lambda_2 n^{2/3}$ in Lemma 2.6.1, then

$$\begin{aligned}\mathbb{P}\left(|X - \mu| \geq \alpha\right) &\leq 2 \exp\left(\frac{-2\left(\frac{1}{2}\lambda_2 n^{2/3}\right)^2}{4(n-2)}\right) \\ &\leq 2 \exp\left(-\tfrac{1}{8}\lambda_2^2 n^{1/3}\right).\end{aligned}$$

Therefore we have proven Theorem 2.0.3 for positive constants $c_1 = \lambda_1 + \frac{1}{2}\lambda_2$ and $c_2 = \frac{1}{2}\lambda_2$ and $c_3 = \frac{1}{8}\lambda_2^2$; that is,

$$\mathbb{P}\left(n - c_1 n^{2/3} < X < n - c_2 n^{2/3}\right) \geq 1 - 2 \exp\left(-c_3 n^{1/3}\right).$$

Chapter 3

The NNI metric and the RF metric

3.1 The NNI metric

In this section we compute the typical and maximal $\mathcal{NNI}$-distance between two trees chosen from $\mathcal{B}(n)$. In 1996, Li, Tromp and Zhang [34] (using the technique described in [46]), showed that the number of trees obtainable by a sequence of at most k NNI moves from a fixed tree in $\mathcal{B}(n)$ is at most exponential in k (Theorem 3.1.2). This is then compared to the value of $|\mathcal{B}(n)|$, to achieve a lower bound on the $\mathcal{NNI}$-radius (Lemma 3.1.3).

Definition 3.1.1. For any $A \in \mathcal{B}(n)$, let $\text{ball}_k(A)$ be the number of trees which are at most k NNI moves from A.

Theorem 3.1.2 ([34])**.** *For any $A \in \mathcal{B}(n)$, we have the following bound,*

$$\text{ball}_k(A) \leq 3^{n-2}2^{4k}.$$

For a proof of Theorem 3.1.2, we refer the reader to [34] and [46]. We now use this theorem to show that the $\mathcal{NNI}$-distance between a pair of trees in $\mathcal{B}(n)$ is typically $\Omega(n \log n)$.

Lemma 3.1.3. *Let $A \in \mathcal{B}(n)$ be any binary tree, and let $B \in \mathcal{B}(n)$ be a random binary tree chosen uniformly. If X denotes the $\mathcal{NNI}$-distance between A and B, then*

$$\mathbb{E}[X] > \tfrac{1}{4}n \log_2 n - n.$$

Moreover, there exists a constant $c > 0$ such that

$$\mathbb{P}(X \leq \tfrac{1}{4}n \log_2 n - n) < \exp(-cn).$$

Proof. For $n < 12$, the result is trivial because $\frac{1}{4}n\log_2 n - n < 0$ and $R_{\mathcal{NNI}}(\mathcal{B}(n)) \geq 0$. We now show, by induction, that $|\mathcal{B}(n)| > n!$ for all $n \geq 12$. For the base case, $|\mathcal{B}(12)| = 19!! > 12!$. For the inductive step, since $(2n-5) > n$, we have

$$|\mathcal{B}(n)| = (2n-5)|\mathcal{B}(n-1)| > n(n-1)! = n!.$$

Therefore $|\mathcal{B}(n)| > n!$. Now, by considering the Taylor expansion of e^x, we can see that $e^x > \frac{x^n}{n!}$ for all positive x. In particular, if we set $x = n$, then $e^n > \frac{n^n}{n!}$. Therefore

$$|\mathcal{B}(n)| > n! > \left(\frac{n}{e}\right)^n.$$

Now, by definition, the probability that $d_{\mathcal{NNI}}(A, B) \leq k$ is exactly $\text{ball}_k(A)$ divided by $|\mathcal{B}(n)|$. For $n \geq 12$ and $k = \frac{1}{4}n\log_2 n - n + 1$, applying Theorem 3.1.2 yields,

$$\mathbb{P}(d_{\mathcal{NNI}}(A, B) \leq k) = \frac{\text{ball}_k(A)}{|\mathcal{B}(n)|} < \frac{3^{n-2}2^{4k}}{\left(\frac{n}{e}\right)^n} = \frac{1}{9}\left(\frac{3e}{16}\right)^n.$$

This establishes the second part of the lemma, by setting $c = \log\left(\frac{16}{3e}\right)$ (in fact this is slightly stronger, since $k > \frac{1}{4}n\log_2 n - n$). Now we can apply the Markov's inequality to get:

$$\begin{aligned}
\mathbb{E}[d_{\mathcal{NNI}}(A, B)] &\geq k\mathbb{P}(d_{\mathcal{NNI}}(A, B) \geq k) \\
&\geq \left(\frac{1}{4}n\log_2 n - n + 1\right)\left(1 - \frac{1}{9}e^{-cn}\right) \\
&= \frac{1}{4}n\log_2 n - n + \left(1 - \frac{n\log_2 n - 4n + 4}{36e^{cn}}\right) \\
&> \frac{1}{4}n\log_2 n - n.
\end{aligned}$$

The last line follows because $36e^{cn} = 36\left(\frac{16}{3e}\right)^n > n\log_2 n - 4n + 4$ for all $n \geq 12$. □

Lemma 3.1.3 establishes a $\Theta(n\log n)$ lower bound for $R_{\mathcal{NNI}}(\mathcal{B}(n))$. We now search for an upper bound for $D_{\mathcal{NNI}}(\mathcal{B}(n))$. It is a known result that $D_{\mathcal{NNI}}(A, B) \leq \mathcal{O}(n\log n)$, because Culik and Wood [19] found a bound of $2n\log n + \mathcal{O}(n)$ in 1982. This bound was improved in 1996 to $n\log n + \mathcal{O}(n)$ by Li, Tromp and Zhang [34] by simulating a merge-sort algorithm on caterpillar trees. Here we provide a different construction that also yields a similar worst case time. Specifically, for any pair of trees $A, B \in \mathcal{B}(n)$, we will show explicitly (Lemma 3.1.7) how to transform A into B using less than $n\log_2 n + 3n$ NNI moves. Lemmas 3.1.4 and 3.1.5 are used in this construction.

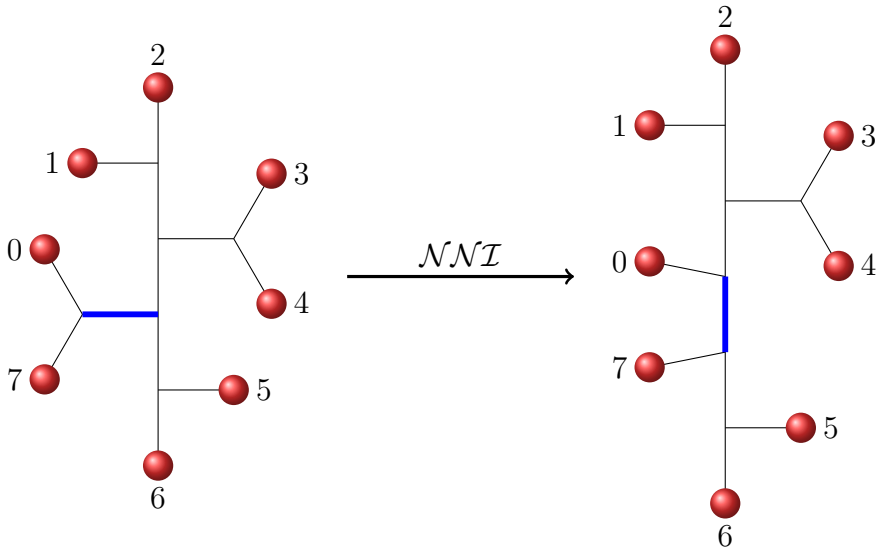


Figure 3.1: The length of the longest path in any non-caterpillar tree, can be increased by 1 using a single NNI move.

Lemma 3.1.4. *It is possible to change any tree into a caterpillar using less than n NNI moves.*

Proof. A tree in $\mathcal{B}(n)$ is a caterpillar if and only if the longest path between any two leaves has length exactly $n-1$. So it suffices to show that from any non-caterpillar $T \in \mathcal{B}(n)$, you can increase the length of the longest path in T by making a single NNI move. Let $v_0 v_1 v_2 \dots v_m \in T$ be a path of greatest length, so v_0 and v_m are leaves which are the furthest apart in T. If every v_i had neighbours v_{i-1}, v_{i+1} and c, where c is a leaf, then T would be a caterpillar. Therefore there is some v_i such that its neighbours are v_{i-1}, v_{i+1} and u where u is a tree vertex. The length of the path from v_0 to v_m can be increased by a single NNI move performed on edge $v_i u$. □

The length of the longest path in a tree $T \in \mathcal{B}(n)$, is minimal when T is balanced. The exact value of this minimal longest path can be shown to equal

$$\lceil \log_2 n \rceil + \lceil \log_2 3n \rceil ,$$

for any integer $n \geq 3$. This could be used to tighten the bound in Lemma 3.1.4. In order to simplify our expressions, we don't bother with the tight bound.

Lemma 3.1.5. *The $\mathcal{NNI}$-distance between any two unlabelled binary trees is less than $2n$.*

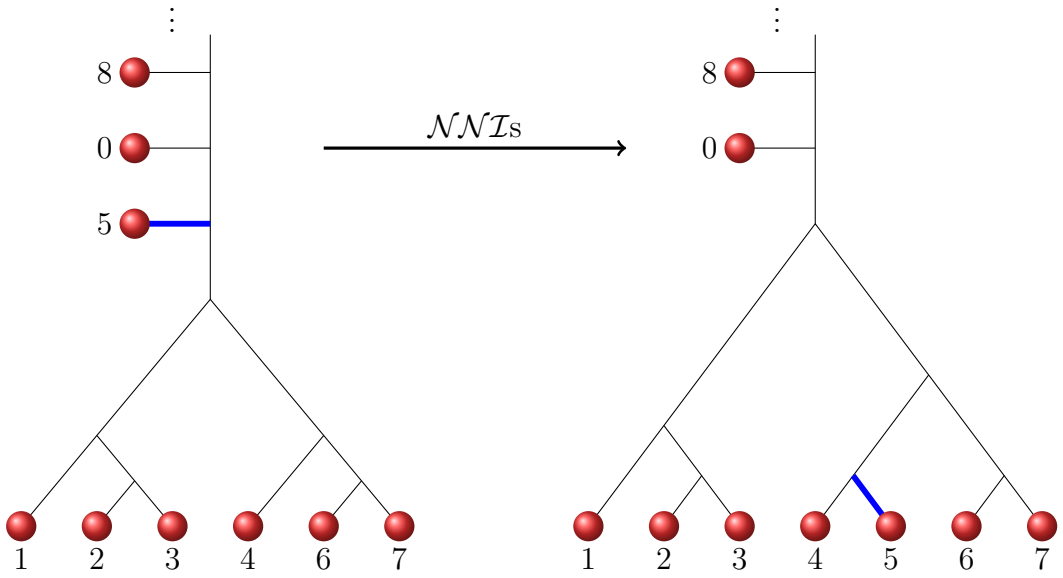


Figure 3.2: The construction in Lemma 3.1.6. The balanced tree B grows, leaf by leaf, from one end of the caterpillar A.

Proof. Let C be the unlabelled caterpillar tree. Lemma 3.1.4 tells us that $d_{\mathcal{NNI}}(A, C)$ and $d_{\mathcal{NNI}}(C, B)$ are both less than n, for any unlabelled binary trees A, B. Therefore

$$d_{\mathcal{NNI}}(A, B) \leq d_{\mathcal{NNI}}(A, C) + d_{\mathcal{NNI}}(C, B) < 2n.$$

□

Lemma 3.1.5 tells us that we can change the shape of a tree arbitrarily using $\mathcal{O}(n)$ NNI moves. Any two labels in a balanced tree can be swapped using $2\log_2 n$ NNI moves and at most n such swaps are required to perform an arbitrary permutation of the labels. This then implies that any two balanced trees can be transformed into each other using $2n\log_2 n$ NNI moves. Culik and Wood [19] used this result (along with Lemma 3.1.5) to deduce that the $\mathcal{NNI}$-diameter of $\mathcal{B}(n)$ is at most $2n\log_2 n + \mathcal{O}(n)$. However, the following construction decreases this upper bound for $D_{\mathcal{NNI}}(\mathcal{B}(n))$ to $n\log_2 n + \mathcal{O}(n)$.

Lemma 3.1.6. *If $A, B \in \mathcal{B}(n)$ such that A is a balanced and B is a caterpillar then*

$$d_{\mathcal{NNI}}(A, B) < n\log_2 n.$$

Proof. by construction. Let e be an edge incident with the center of A; an edge which has distance at most $\log_2 n$ from all the leaves. Now iteratively move each leaf to e,

in the order specified by B. So the caterpillar B grows from e leaf-by-leaf. Each leaf requires at most $\log_2 n$ NNI moves to reach e, so this will take at most $(n-2)\log_2 n$ NNI moves to move the first $(n-2)$ leaves (and the other 2 leaves will be in their correct locations automatically). □

Lemma 3.1.7. *The class $\mathcal{B}(n)$ is traversable by $\mathcal{NNI}$ moves. Moreover,*

$$D_{\mathcal{NNI}}(\mathcal{B}(n)) < n \log n + 3n.$$

Proof. Let $A, B \in \mathcal{B}(n)$ be an arbitrary pair of trees. By Lemma 3.1.5, there exists some balanced tree $A' \in \mathcal{B}(n)$ such that $d_{\mathcal{NNI}}(A, A') < 2n$. By Lemma 3.1.4, there exists a caterpillar $B' \in \mathcal{B}(n)$ such that $d_{\mathcal{NNI}}(B, B') < n$. By Lemma 3.1.6, we have $d_{\mathcal{NNI}}(A', B') < n \log_2 n$. Putting all this together, we get

$$d_{\mathcal{NNI}}(A, B) \leq d_{\mathcal{NNI}}(A, A') + d_{\mathcal{NNI}}(A', B') + d_{\mathcal{NNI}}(B', B) < n \log_2 n + 3n.$$

□

As a result of Lemma 3.1.7, and Lemma 3.1.3, we deduce that

$$\frac{1}{4} n \log_2 n - n \leq R_{\mathcal{NNI}}(\mathcal{B}(n)) \leq D_{\mathcal{NNI}}(\mathcal{B}(n)) \leq n \log_2 n + 3n.$$

Moreover, for any $A \in \mathcal{B}(n)$, if $B \in_u \mathcal{B}(n)$, then

$$\frac{1}{4} n \log_2 n - n \leq \mathbb{E}[d_{\mathcal{NNI}}(A, B)] \leq n \log_2 n + 3n.$$

Recall (Theorem 2.0.3) that the TBR, SPR and rSPR distances between A and B are all highly concentrated about their means. No such result is known for the $\mathcal{NNI}$-metric over $\mathcal{B}(n)$.

3.2 Robinson Foulds metric

The Robinson Foulds metric is easier to compute than the other metrics discussed in this thesis. For this reason, it is used in many phylogenetic software packages.

Definition 3.2.1. In a tree T, an edge *contraction* of $xy \in E(T)$ means deleting edge xy and identifying x and y. So x and y are replaced by a new vertex whose neighbourhood is the (disjoint) union of the neighbourhoods of x and y excluding x and y themselves.

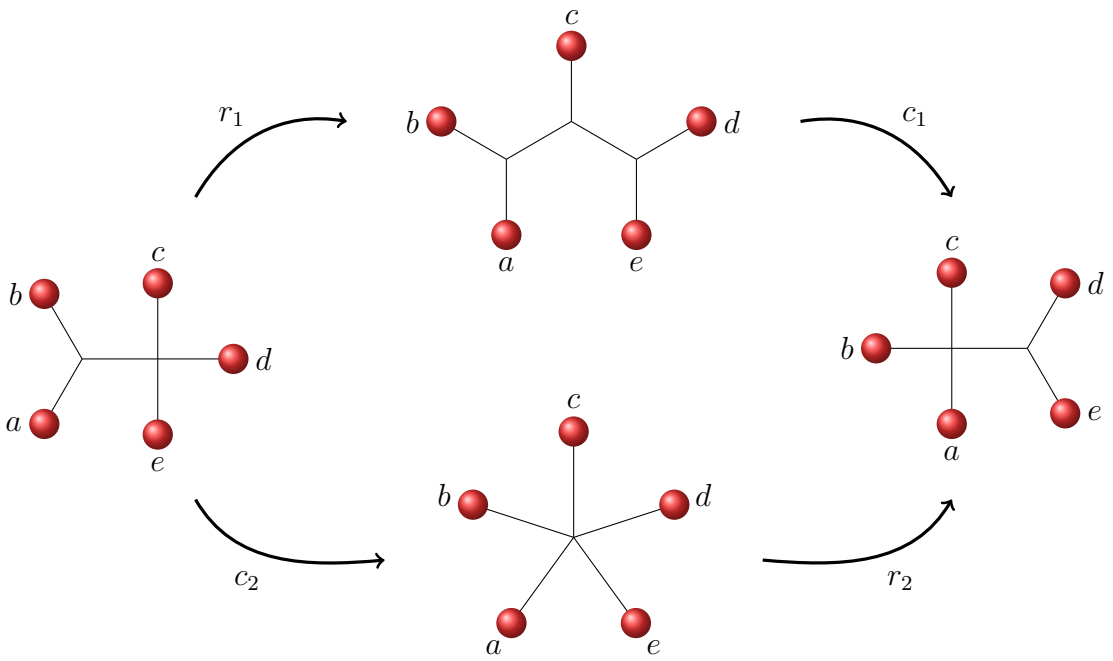


Figure 3.3: The result of any resolution, r_1, followed by a contraction, c_1, can always be achieved by some contraction, c_2, followed by some resolution, r_2.

Definition 3.2.2. The reverse of an edge contraction is called a vertex *resolution*. If v is a vertex in a tree T, then v can be *resolved* if $\deg(v) \geq 4$. To resolve v means to replace v with a pair of vertices x and y, each with at least 2 neighbours, such that the neighbourhoods of x and y form a partition of what the neighbourhood of v was, and then adding edge xy. (We insist that x and y each take at least two of the neighoburs of v so that the final degree of both x and y is at least 3.)

The definitions of a contraction and a resolution are reminisent of the definition of the NNI operation (Definition 1.2.5). Indeed an NNI move on a tree in $\mathcal{B}(n)$ is precisely a contraction of an edge between two tree vertices followed immediately by a resolution of the newly created degree 4 vertex. However, in this section, we consider contractions and resolutions performed on arbitrary leaf-labelled trees (not just binary trees in $\mathcal{B}(n)$).

Lemma 3.2.3. *The process of performing a resolution followed by contraction on a tree, can be achieved by performing the contraction first, and then a resolution.*

Proof. See Figure 3.3. If the vertex being resolved is not an endpoint of the edge being contracted, then the two operations commute trivially. So let v be the vertex to be resolved into xy, and let yz be the edge which would be subsequently contracted

(so vz is an edge in the original tree). Then we can instead contract vz to form a vertex v' (with a neighbourhood of $\Gamma[x] \cup \Gamma[y] \cup \Gamma[z]$). The desired result can then be achieved by resolving v' into a vertex with neighbourhood $\Gamma[x]$ and a vertex with neighbourhood $\Gamma[y] \cup \Gamma[z]$. □

The converse of Lemma 3.2.3 is not true; contractions and resolutions do not commute in general (See Figure 3.4). We can now reformulate the Robinson Foulds metric in terms of edge contractions and resolutions, and thus relate it to the $\mathcal{NNI}$ metric.

Proposition 3.2.4. *For any $A, B \in \mathcal{B}(n)$, the $\mathcal{NNI}$-distance between A and B is half the length of the shortest sequence of alternating contractions and resolutions, which can be performed on A to produce B.*

Proof. An NNI move is precisely an edge contraction followed by the resolution of the newly created vertex. So if $d_{\mathcal{NNI}}(A, B) = k$, then the sequence of k NNIs moves that transform A into B is exactly a sequence of $2k$ moves, alternatively contractions and resolutions.
Conversely, any edge contraction followed by the resolution of the formed newly formed, is exactly an NNI move. So let

$$m_1, m_2, m_3, \ldots, m_t$$

be an arbitrary sequence of alternating contractions and resolutions, which can be performed on A to produce B. Iteratively, for $i = 1, 2, \ldots, t/2$: the contraction, m_{2i-1}, must act on a pair of degree 3 vertices (as all non-leaf vertices have degree 3) and the resolution, m_{2i}, must act on the only vertex with degree > 3, the degree 4 vertex just created by m_{2i-1}. So the composition of m_{2i-1} with m_{2i} is an NNI move, thus $d_{\mathcal{NNI}}(A, B) \leq \frac{t}{2}$. The number t had to be even since B doesn't contain a vertex of degree 4. □

Proposition 3.2.5. *The Robinson Foulds distance between a pair of trees $A, B \in \mathcal{B}(n)$, is half the length of the shortest sequence of (not necessarily alternating) sequence of contractions and resolutions, which can be performed on A to produce B.*

Proof. For any leaf labelled trees T and T', if T' is the result of contracting edge xy in T, then

$$\Sigma(T') \sqcup \{s_{xy}\} = \Sigma(T),$$

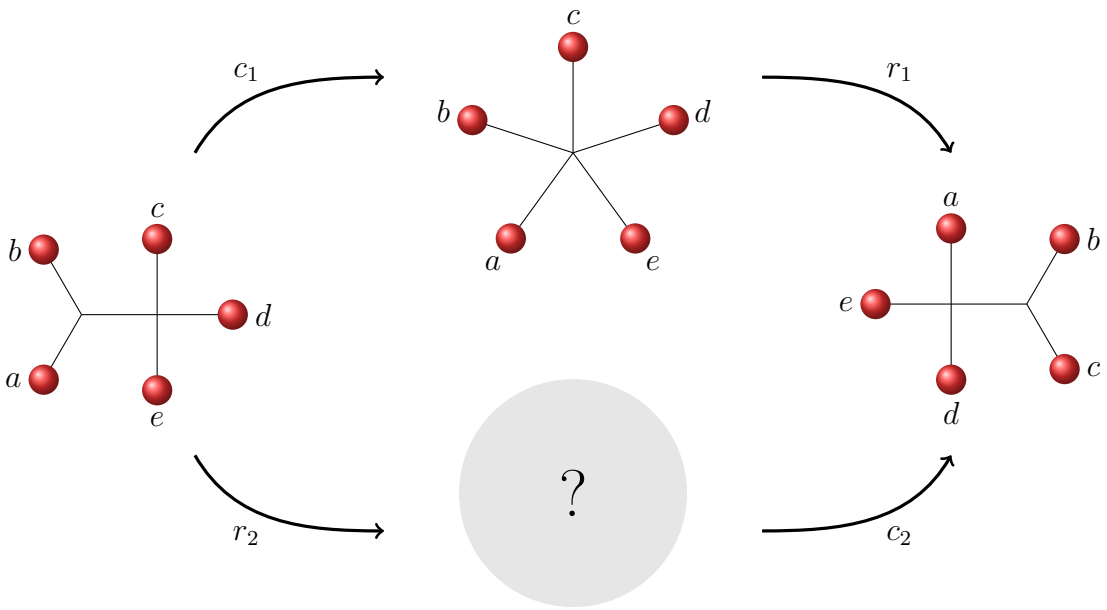


Figure 3.4: The result of the contraction, c_1, followed by the resolution, r_1, cannot be achieved by a resolution, r_2, followed by a contraction, c_2.

where s_{xy} is the split corresponding to edge xy. Now let $d_{RF}(A, B) = k$ and let T be the leaf-labelled (non-binary) tree obtained by contracting the k edges, in A, corresponding to splits in $\Sigma(A) \backslash \Sigma(B)$. So $\Sigma(T) = \Sigma(A) \cap \Sigma(B)$. Since $\Sigma(T) \subseteq \Sigma(B)$, it follows that B is obtainable from T by k resolutions.

To demonstrate the converse, suppose there exists a sequence of $2m$ moves (m resolutions and m contractions, because A and B have the same number of edges) which can transform A into B. By Lemma 3.2.3 we can assume that this is a sequence of m contractions followed by m resolutions. If T is the intermediate tree, which is the result of the first m contractions on A, then $\Sigma(T)$ is obtained from $\Sigma(A)$ by removing m splits. Similarly, since a resolution is the inverse of a contraction, $\Sigma(T)$ is a subset of $\Sigma(B)$, and therefore $\Sigma(T) \subseteq \Sigma(A) \cap \Sigma(B)$. Therefore

$$\begin{aligned} d_{RF}(A, B) &= \tfrac{1}{2}|\Sigma(A)| + \tfrac{1}{2}|\Sigma(B)| - |\Sigma(A) \cap \Sigma(B)| \\ &\leq \tfrac{1}{2}|\Sigma(A)| + \tfrac{1}{2}|\Sigma(B)| - |\Sigma(T)| = m. \end{aligned}$$

□

Combining Propositions 3.2.4 and 3.2.5, we can discover the following simple lemma.

Lemma 3.2.6. *For any pair of trees $A, B \in \mathcal{B}(n)$, we have*

- $d_{RF}(A, B) = 1$ *if and only if* $d_{\mathcal{NNI}}(A, B) = 1$, *and*
- *in general:* $d_{RF}(A, B) \leq d_{\mathcal{NNI}}(A, B)$.

Proof. The only difference between the $\mathcal{NNI}$ metric, and the Robinson Foulds metric (when computing them as a minimal sequence of contractions and resolutions) is that this sequence must be alternating for the $\mathcal{NNI}$-distance. So for $\mathcal{NNI}$, we are more constrained and thus

$$d_{RF}(A, B) \leq d_{\mathcal{NNI}}(A, B).$$

However, both A and B contain no vertices with degree ≥ 4. So if $d_{RF}(A, B) = 1$ then it must be a contraction followed by a resolution, and so $d_{\mathcal{NNI}}(A, B) = 1$. □

For trees in $\mathcal{B}(n)$, the maximum value of the Robinson Foulds distance is $|\Sigma(A)| = n - 3$. This can be achieved from an arbitrary tree $A \in \mathcal{B}(n)$, by choosing a cherry $\{x, y\}$ in A, and then choosing B to be a caterpillar in which the path between x and y is a longest path. Any split in $|\Sigma(A)|$ will not separate x and y while every split $|\Sigma(B)|$ will separate x and y. This can be counter-intuitive as there are many examples of trees with high Robinson Foulds distance but are similar by other measures. For example, Figure 1.11 depicts a pair of trees with maximal Robinson Foulds distance, but have an agreement subtree with $n - 1$ leaves and differ by only one rSPR move.

We now turn our attention to the distribution of $d_{RF}(A, B)$ for $A, B \in \mathcal{B}(n)$ chosen randomly and independently. Let X_u denote $d_{RF}(A, B)$ for $A, B \in \mathcal{B}(n)$ chosen independently from the uniform distribution over $\mathcal{B}(n)$. Let X_{YH} denote $d_{RF}(A, B)$ for $A, B \in \mathcal{B}(n)$ chosen independently from the Yule-Harding distribution over $\mathcal{B}(n)$. The following result appears in [40].

Theorem 3.2.7 ([40])**.** *If $A, B \in \mathcal{B}(n)$ are random trees chosen independently with the uniform distribution, then let X_u denote the random variable*

$$X_u = n - 3 - d_{RF}(A, B).$$

If $A, B \in \mathcal{B}(n)$ are random variables chosen independently with the Yule-Harding distribution, then let X_{YH} denote the random variable

$$X_{YH} = n - 3 - d_{RF}(A, B).$$

As $n \to \infty$, the distributions of the variables X_u and X_{YH} tend to the Poisson distribution $\mathrm{Po}(\lambda)$ *with $\lambda = 1/8$ and $\lambda = 2/9$ respectively.*

So for these specific cases, we know the distribution of $d_{RF}(A, B)$ exactly. Comparing this with the results of $\mathcal{TBR}$, $\mathcal{SPR}$ and $r\mathcal{SPR}$ metrics, we see that all four are highly concentrated about their means. However, for the Robinson Foulds metric, the maximum is $\Theta(1)$ away from the mean, whereas, for the $\mathcal{TBR}$, $\mathcal{SPR}$ and $r\mathcal{SPR}$ metrics, the mean is $\Theta(n^{2/3})$ below their maximum value.

Chapter 4

Subtree transfer operations on networks with recombination

"the history of life cannot properly be represented as a tree." – Ford Doolittle

Sometimes trees can be inadequate for representing genetic data. Meiotic recombination occurs in populations of individuals inside a single species. More general recombination events occur between different species, for example by lateral gene transfer or hybrid speciation. For a more thorough discussion of how recombination vertices can represent real life phylogenetic events, we refer the reader to [29].

In this chapter we define three phylogeny rearrangement operations, which act on phylogenetic networks with recombination: the $\mathcal{EEX}$, $p\mathcal{SPR}$ and $pNNI$ in Definitions 4.2.1, 4.3.1 and 4.4.1 respectively. Similarly with previous chapters, we will only consider networks in which the non-leaf vertices have total degree three (in-degree 2 and out-degree 1 or in-degree 1 and out degree 2). We then prove some results regarding the traversability of the class of phylogenies (Definitions 4.1.1) by these rearrangement operations. The results in this chapter are all original work; these results have not yet been published in literature, as far as I am aware.

4.1 Basic definitions

Definition 4.1.1. In this chapter, a *network* refers to a directed multigraph with 4 types of vertex:

- One *root* vertex of in-degree 0 and out-degree 1, labelled 0.
- *Tree* vertices of in-degree 1 and out-degree 2 (unlabelled).
- *Recombination* vertices of in-degree 2 and out-degree 1 (unlabelled).

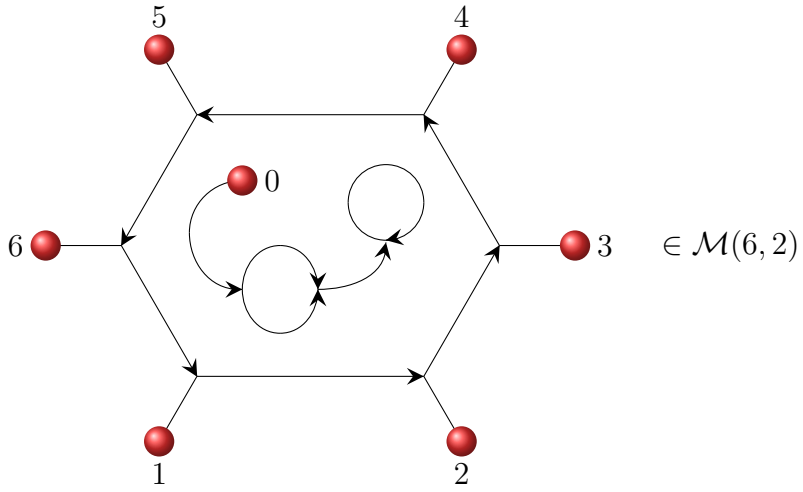


Figure 4.1: An example of a network in $\mathcal{M}(6,2)$ with $l = 6$ leaves, $r = 2$ recombination vertices, and $t = 7$ tree vertices. This network is particularly pathological, as it is disconnected, contains a loop, contains a double edge and it contains a cycle.

- *Leaf* vertices of in-degree 1 and out-degree 0, labelled from 1 to l, where l is the number of leaf vertices.

Let $\mathcal{M}(l,r)$ denote the class of all networks with r recombination vertices and l leaves labelled with $1, 2, \ldots, l$. In the context of networks with recombination, the root is not considered to be a leaf. A *phylogeny* is any acyclic network which doesn't contain any double edges nor loops[1]. Let $\mathcal{P}(l,r)$ denote the class of all phylogenies with r recombination vertices and l leaves. When $r = 0$, $\mathcal{P}(l,0)$ is naturally isomorphic to $\mathcal{B}(l+1)$ with all edges oriented away from the root. In this chapter, a path will always mean a directed path. For any vertices u and v in a phylogeny, such that there is a path from u to v, we say u is an ancestor of v and v is a descendant of u. If the path was a single edge uv, then we say u is a *parent* of v and v is a *child* of u. Let $\Gamma_-(v)$ denote the set of parents of v and let $\Gamma_+(v)$ denote the set of children of V.

A network in $\mathcal{M}(l,r)$ has l leaves, r recombination vertices, 1 root, and t tree vertices. Since, the number of edges equals the sum of the in-degrees equals the sum of the out-degrees, we have $l + 2r + t = 1 + 2t + r$ and therefore $t = r + l - 1$. So every network in $\mathcal{M}(l,r)$ has exactly $r+l-1$ tree vertices and thus $3r+2l-1$ edges. The following proposition (Proposition 4.1.2) is a standard result.

[1]A loop is an edge from a vertex to itself. A loop contributes 1 to the in-degree and 1 to the out-degree of the vertex.

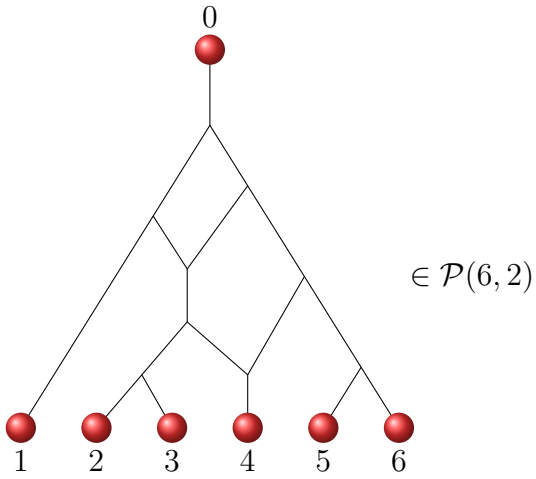


Figure 4.2: An example of a phylogeny in $\mathcal{P}(6,2)$ with $l = 6$ leaves and $r = 2$ recombination vertices. The edges are all oriented downwards.

Proposition 4.1.2. *Let G be any network in $\mathcal{M}(l,r)$. If G contains no cycles, then there is a path from the root to any other vertex.*

Proof. Let $u = u_0$ be an arbitrary vertex. Iteratively, if u_i is not the root, let u_{i+1} be a parent of u_i. Since G is acyclic, the sequence $(u_i)_{i\geq 0}$ can never repeat. Because G has a finite number of vertices the sequence must terminate at some u_k with in-degree 0. Therefore the sequence terminates at the root because the root is the only vertex with zero in-degree. The reverse of this sequence is a path from the root to u. □

By Proposition 4.1.2, the root vertex is a common ancestor of all vertices in a phylogeny. The following Lemma is quite simple, but is used in several proofs in this chapter.

Lemma 4.1.3. *If $r \geq 1$, then in any phylogeny $A \in \mathcal{P}(l,r)$, there exists a tree vertex u and a recombination vertex v such that:*

- *The vertex v is not a descendant of any other recombination vertex,*
- *There is a unique path, p_1, from the root to u, and p_1 contains only tree vertices (except for the root).*
- *There are two distinct paths, p_2 and p_3, from u to v, each containing only tree vertices (except for v) such that the only vertices in common to both paths are u and v.*

Proof. Choose a recombination vertex $v_0 \in A$ arbitrarily. Then iteratively, for $i = 1, 2, 3, \ldots$, if v_i is the descendant of a recombination vertex, then arbitrarily choose v_{i+1} to be a recombination vertex which is an ancestor of v_i. No vertex v_i can appear twice in the sequence $v_0, v_1, v_2, \ldots, v_k$ because A is acyclic. Moreover, since there are a finite number, r, of recombination vertices in A, this process must terminate, and the final vertex, v_j, of this sequence is a recombination vertex which is not the descendant of any other recombination vertex. Let $v = v_j$, and let a and b be the parents of v. By Proposition 4.1.2, there exist a path p_a from the root to a, and a path p_b from the root to b. These paths cannot contain any leaves (because leaves have no out-degree) nor recombination vertices (because v is not the descendant of any recombination vertices) and therefore p_a and p_b contain only tree vertices. If for any i, the i^{th} vertex on the paths p_a and p_b were distinct, then all vertices after the i^{th} must also be distinct, because the paths contain only tree vertices. Therefore p_a and p_b coincide for some initial path p_1, and if the initial common segment p_1 is removed from both p_a and p_b, then remainders must be disjoint. The vertex u is the last vertex of p_1; the last common vertex of p_a and p_b. The paths p_2 and p_3 can be constructed by removing the initial segment p_1, and adding v to the ends of paths p_a and p_b. □

4.2 Edge exchanges

Networks may contain multiple edges, cycles or even loops (edges from a vertex to itself). The edge exchange operation (Definition 4.2.1) is simple to define on the class of networks, because there is no need to worry about whether a double edge or a cycle is created. However, they become quite difficult to work with, if we restrict ourselves to a subclass, like phylogenies.

Definition 4.2.1. An $\mathcal{EEX}_k$ (an edge exchange of order k) is a operation on a k-tuple, $(a_1b_1, a_2b_2, \ldots, a_kb_k)$, of distinct edges in a network. The operation is performed by deleting the edges $\{a_ib_i\}_{i=1}^{k}$ and replacing them with the edges $\{a_ib_{i+1}\}_{i=1}^{k-1}$ and a_kb_1. When acting on a class of networks $\mathcal{C}$ (for example we often let $\mathcal{C} = \mathcal{P}(l, r)$) we only allow $\mathcal{EEX}_k$ moves for which the resulting directed graph is in $\mathcal{C}$.

Proposition 4.2.2. *Over the class of networks, for $k \geq 2$ any $\mathcal{EEX}_k$ operation can be expressed as the composition of $(k-1)$ $\mathcal{EEX}_2$ operations.*

Proof. (by induction on k) For the base case, $k = 2$, the $\mathcal{EEX}_k$ operation is trivially an $\mathcal{EEX}_2$ operation. For the inductive step, $k > 2$, it suffices to show that any $\mathcal{EEX}_k$

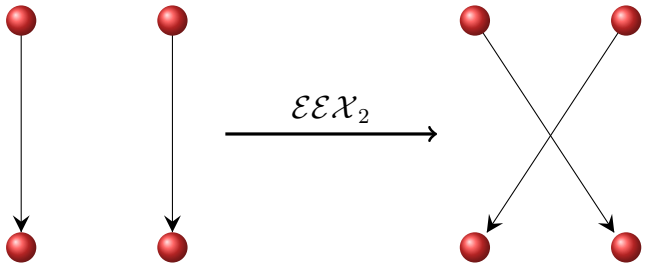


Figure 4.3: The $\mathcal{EEX}_2$ operation.

operation can be expressed as the composition of an $\mathcal{EEX}_{k-1}$ and an $\mathcal{EEX}_2$. Explicitly, one can verify that the composition of the $\mathcal{EEX}_{k-1}$ on $(a_1b_1, a_2b_2, \ldots, a_{k-1}b_{k-1})$ with the $\mathcal{EEX}_2$ on $(a_{k-1}b_1, a_kb_k)$, is the $\mathcal{EEX}_k$ on $(a_1b_1, a_2b_2, \ldots, a_kb_k)$. □

Proposition 4.2.3. *Let k be an integer with $2 \leq k < 2r + l$. Over the class $\mathcal{M}(l, r)$:*

1. *Any $\mathcal{EEX}_3$ operation can be expressed as the composition of two $\mathcal{EEX}_k$ operations.*

2. *Any $\mathcal{EEX}_2$ operation can be expressed as the composition of at most two $\mathcal{EEX}_3$ operations.*

Proof. We prove each part separately.

1. Consider the $\mathcal{EEX}_3$ move on edges (a_1b_1, a_2b_2, a_3b_3). If $u_1v_1, u_2v_2, \ldots, u_{k-2}v_{k-2}$ are any $k-2$ other edges then this $\mathcal{EEX}_3$ can be achieved by performing the $\mathcal{EEX}_k$ on $(a_1b_1, a_2b_2, u_1v_1, \ldots, u_{k-2}v_{k-2})$ followed by the $\mathcal{EEX}_k$ on

$$(a_2v_1, a_3b_3, u_{k-2}b_1, u_{k-3}v_{k-2}, \ldots, u_2v_3, u_1v_2).$$

2. Consider the $\mathcal{EEX}_2$ operation on the edges (a_1b_1, a_2b_2). There are three cases:

   - If $a_1 = a_2$, then this $\mathcal{EEX}_2$ operation makes no change to the network, and thus can be expressed as a composition of zero $\mathcal{EEX}_3$ operations.

   From now on assume $a_1 \neq a_2$.

   - If at least one of a_1 or a_2 is a tree vertex, then wlog a_2 is a tree vertex. Let b_3 be the child of a_2 distinct from b_2. The $\mathcal{EEX}_2$ operation on (a_1b_1, a_2b_2) is exactly the same as the $\mathcal{EEX}_3$ operation on (a_1b_1, a_2b_2, a_2b_3).

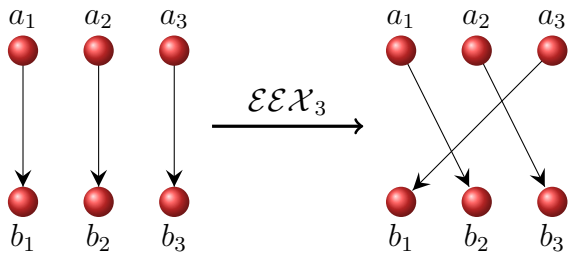


Figure 4.4: The $\mathcal{EEX}_3$ operation.

- If neither a_1 nor a_2 are tree vertices, then let u be an arbitrary tree vertex in the network. Note that there must be a tree vertex, u, because the network has more than 1 edge (and it doesn't matter if we choose $u = b_1$ or $u = b_2$, because are working over $\mathcal{M}(l, r)$). Let v_1 and v_2 be the children of u. The $\mathcal{EEX}_2$ operation on (a_1b_1, a_2b_2) is exactly the same as the $\mathcal{EEX}_3$ operation on (a_1b_1, a_2b_2, uv_1) followed by the $\mathcal{EEX}_3$ on (a_2v_1, ub_1, uv_2).

□

In permutation theory, transpositions are often considered to be the most elementary permutation. In a similar sense, edge exchanges of order 2 can be considered the most elementary network operation, because they involve deleting-and-replacing only 2 edges. Proposition 4.2.2 shows that, over the class $\mathcal{M}(l, r)$, any sequence of edge exchanges (of any orders) can be decomposed into a composition of only $\mathcal{EEX}_2$s. Therefore, for any distinct integers k and k', an edge exchange of order k can be achieved using a composition of edge exchanges of order k'. We will show now, in Lemma 4.2.4, that over the class $\mathcal{M}(l, r)$ it is possible to transform any network into any other using only $\mathcal{EEX}_2$ moves. However, as we will see in Lemma 4.2.7, this is not the case over the class of phylogenies; *i.e.* for certain $A, B \in \mathcal{B}(n)$, it is not possible to transform A into B using $\mathcal{EEX}_2$ moves unless some of the intermediate networks are not phylogenies (like the example depicted in Figure 4.1).

Lemma 4.2.4. *Let l and r be any two non-negative integers such that $l + r \geq 1$. The class $\mathcal{M}(l, r)$ is traversable by $\mathcal{EEX}_2$ moves, and*

$$D_{\mathcal{EEX}_2}(\mathcal{M}(l, r)) \leq 3r + 2l - 2.$$

Proof. Let $A, B \in \mathcal{M}(l, r)$ be chosen arbitrarily. It suffices to show that A can be turned into B using at most $(3r + 2l - 2)$ $\mathcal{EEX}_2$ moves. Let π be any bijection from the vertices of A to the vertices of B, such that π maps: the root to the root,

tree vertices to tree vertices, recombination vertices to recombination vertices and for $i = 1, 2, \ldots, l$, the leaf labelled i, in A, maps to the leaf labelled i in B. Now arbitrarily choose a bijection τ from the edges of A to the edges of B, such that for all edges $e = uv$ of A, if

$$\tau(e) = u'v' \qquad \text{then} \qquad \pi(u) = u'.$$

We now iteratively perform $\mathcal{EEX}_2$ moves until τ becomes a graph isomorphism. Let H denote the set of edges $e = uv$ in A such that if $\tau(e) = u'v'$ then $\pi(v) = \pi(v')$. We show now that if H doesn't contain all the edges of A, then there exists an $\mathcal{EEX}_2$ move that increases $|H|$. Let $e_1 = uv$ be an edge not in H, let $\tau(e) = u'v'$, and let y be the vertex in A such that $\pi(y) = v'$. Let x be a vertex in A such that $e_2 = xy$ is an edge of A not in H. Such a vertex x must exist, because it is not possible that $py \in H$, for all parents p of y, because $\pi(y)$ (which has the same number of parents as y) would have all $\pi(p)$ and u' for parents. Now consider the $\mathcal{EEX}_2$ move on (uv, xy). Initially $e_1 = uv$ and $e_2 = xy$ are both not in H, whereas, in the resulting graph, e_1 becomes uy and so now,

$$\tau(e_1) = u'v' = \pi(u)\pi(y).$$

So this $\mathcal{EEX}_2$ added e_1 into H without removing anything. By performing this process iteratively, we can increase the size of H until it contains all the edges in A using as most $e(A) - 1 = 3r + 2l - 2$ edge exchanges of order 2 (We can say $e(A) - 1$ because when the penultimate edge is included in H, the final edge must also be in H automatically). After performing these edge exchanges, we must have $A \equiv B$ because π is now a graph isomorpism. *i.e.* for all edges uv of A,

$$\pi(u)\pi(v) = \tau(uv) \quad \text{ is an edge of } B.$$

□

The traversability of $\mathcal{P}(l, r)$ by $\mathcal{EEX}_2$ operations.

Lemma 4.2.4 demonstrated that the edge exchange operation is sufficient to transform between any two networks over the class $\mathcal{M}(l, r)$. The same result over the more restrictive class $\mathcal{P}(l, r)$ doesn't immediately follow, as we can't guarantee that all intermediate networks are in $\mathcal{P}(l, r)$. In fact, we will show that it is not always possible to transform a phylogeny into any other, using only $\mathcal{EEX}_2$ operations, if we require that all intermediate networks are phylogenies.

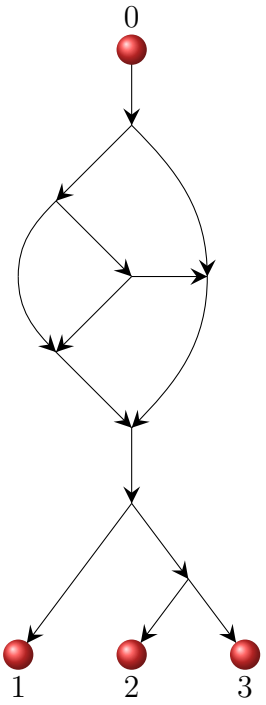


Figure 4.5: A phylogeny in $\mathcal{P}(3,3)$ with a noah vertex.

Proposition 4.2.5. *A non-trivial $\mathcal{EEX}_2$ move, (a_1b_1, a_2b_2) on a phylogeny $A \in \mathcal{P}(l,r)$ is valid over the class $\mathcal{P}(l,r)$ if and only if both a_1b_2 and a_2b_1 were not initially edges in A, and there is no path in A containing both edges a_1b_1 and a_2b_2.*

Proof. Assume that the condition holds. The new edges do not create double edges, because both a_2b_1 and a_1b_2 were not initially joined in A. By assumption there is no path from b_1 to a_2 nor a path from b_2 to a_1, and since the original graph is acyclic, there is no path from b_1 to a_1 nor from b_2 to a_2. *i.e.* there is no path from b_j to a_i for any i, j. Therefore the addition of any edges a_ib_j cannot create any cycles. Hence the result is a phylogeny.
Conversely, assume that there is a path in A containing both a_1b_1 and a_2b_2; wlog assume there is a path from b_1 to a_2. The path from b_1 to a_2 cannot contain edges a_1b_1 nor a_2b_2 since A is acyclic. Therefore the proposed $\mathcal{EEX}_2$ move on (a_1b_1, a_2b_2) is invalid because the addition of edge a_2b_1 would create a cycle with the path from b_1 to a_2. □

Definition 4.2.6. If a recombination vertex which lies on every path from the root to a leaf, then we shall call it a *Noah* vertex.

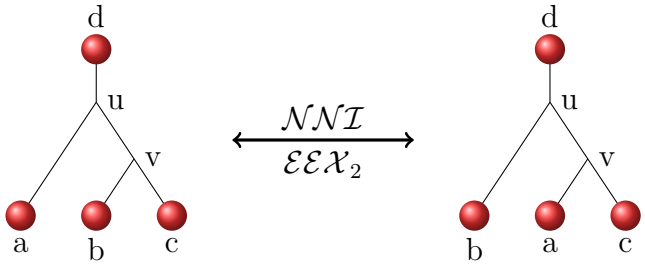


Figure 4.6: An NNI on an edge between two tree vertices, u and v, is exactly the $\mathcal{EEX}_2$ is on (ua, vb) where $a \neq v$ is the child of u, and b is one of the children of v.

Lemma 4.2.7. *Let $A, B \in \mathcal{P}(l, r)$ be phylogenies such that B is obtainable from A by an $\mathcal{EEX}_2$ move. If A contains a Noah vertex, then so does B.*

Proof. Because every edge lies on a path from the root to a leaf, any Noah vertex v, naturally partitions the edges into edges which are ancestral to v or descendant from v. By Proposition 4.2.5, it is impossible to perform an $\mathcal{EEX}_2$ move on a pair of edges, one being ancestral to the Noah vertex, and the other being a descendant of it. So a Noah vertex will remain a Noah vertex after the $\mathcal{EEX}_2$ move is performed. Since the inverse of an $\mathcal{EEX}_2$ is an $\mathcal{EEX}_2$, we can conclude that Noah vertices can be neither created nor destroyed by $\mathcal{EEX}_2$ operations. □

In general (*i.e.* if $r \geq 2$ and $l \geq 2$), the class $\mathcal{P}(l, r)$ contains a phylogeny with a Noah vertex, and a phylogeny without a Noah vertex. So Lemma 4.2.7 demonstrates that $\mathcal{P}(l, r)$ is not traversable by $\mathcal{EEX}_2$ moves in general. Conversely, Lemma 4.2.11 shows that the class of phylogenies in $\mathcal{P}(l, r)$ with no Noah vertices, is traversable by $\mathcal{EEX}_2$ moves. As a corollary, this shows that $\mathcal{P}(l, 0)$ and $\mathcal{P}(l, 1)$ are both traversable by $\mathcal{EEX}_2$ moves. Although we already knew $\mathcal{P}(l, 0)$ was traversable, because $\mathcal{P}(l, 0) \equiv \mathcal{B}(l + 1)$ is traversable by NNI moves (Lemma 3.1.7) and all NNI moves on binary trees are edge exchanges of order 2 (see Figure 4.6).

After some preliminary work, we will conclude this section with Lemma 4.2.13; a characterisation of all pairs $A, B \in \mathcal{P}(l, r)$ such that B can be obtained from A by a sequence of $\mathcal{EEX}_2$ moves.

Definition 4.2.8. Let ρ be the root of a phylogeny $A \in \mathcal{P}(l, r)$ and let x be the child of ρ. The phylogeny A is said to have a *k-ladder* if there exists tree vertices $u_1, u_2, \ldots, u_k$ and recombination vertices $v_1, v_2, \ldots, v_k$ such that $u_i v_i$ is an edge for all i and both

$$\rho x u_1 u_2 \ldots u_k \qquad \text{and} \qquad \rho x v_1 v_2 \ldots v_k$$

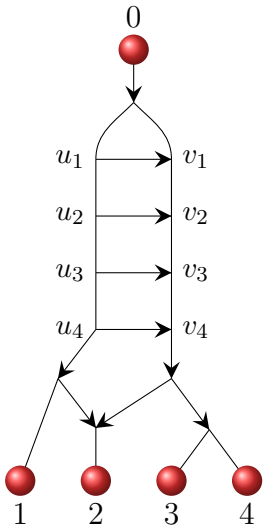


Figure 4.7: A phylogeny with an open 4-ladder.

are paths. The k-ladder is called a *closed* k-ladder if the child of v_k is also a child of u_k. Otherwise the k-ladder is called an *open* k-ladder (see Figure 4.7).

Lemma 4.2.9. *Let r, l be positive integers, and let $A \in \mathcal{P}(l,r)$ be a phylogeny with no Noah vertices. It is possible to turn A into a phylogeny with an open 1-ladder using at most four $\mathcal{EEX}_2$ moves over the class $\mathcal{P}(l,r)$. Moreover, it is possible to do this without affecting the edge incident with the root.*

Proof. Let x be the child of the root and let a and b be the children of x. By Lemma 4.1.3, there exists a recombination vertex v that is not a descendant of any other recombination vertex, and a tree vertex y, such that there are two disjoint paths (of tree vertices) from y to v, and a unique path (of tree vertices) from the root to y.

Step 1. If y is the child of the root, then do nothing. Otherwise y is a descendant of (or equal to) exactly one of a or b. Wlog y is a descendent of (or equal to a). Since there are two disjoint paths from y to v, and no double edges, at least one of these paths is at least two edges long. Hence there exists a tree vertex z, distinct from v, which is a child of y and an ancestor of v. Perform the $\mathcal{EEX}_2$ on (xb, yz). By Proposition 4.2.5, this is a valid $\mathcal{EEX}_2$ over $\mathcal{P}(l,r)$ because we can guarantee yb and xz are initially not edges, and the neither b nor z could possibly be an ancestor of x or y.

At this point we have performed at most one $\mathcal{EEX}_2$ move so that A is still in $\mathcal{P}(l,r)$, but there now exists disjoint paths of tree vertices from x (the child of the root) to v. Let these paths be

$$(x, \alpha_1, \alpha_2, \ldots, \alpha_i, v) \qquad \text{and} \qquad (x, \beta_1, \beta_2, \ldots \beta_j, v)$$

respectively. Wlog $i \geq j$. Also, since there are no double edges, we know $i + j > 0$.

Step 2. If $j = 0$ (*i.e.* if v is already a child of x), then do nothing. Otherwise make v a child of x by performing the $\mathcal{EEX}_2$ on $(x\beta_1, \alpha_i v)$. We know this is a valid $\mathcal{EEX}_2$ move over $\mathcal{P}(l,r)$ by Proposition 4.2.5.

Now we have performed at most two $\mathcal{EEX}_2$ so that x is the child of the root and a parent of the recombination vertex v, and there exists a directed path (of at least two edges),

$$(x, \gamma_1, \gamma_2, \ldots, \gamma_s, v),$$

of tree vertices from x to v (this path is in fact the concatenation of $x, \alpha_1, \alpha_2, \ldots, \alpha_i$ with $\beta_1, \beta_2, \ldots \beta_j, v$). If $s = 1$ then the phylogeny now has an open 1-ladder, with $u_1 = \gamma_1$ and $v_1 = v$. So let us assume $s \geq 2$ and let w be the child of γ_1.

Step 3. If w is not a child of γ_s, then do nothing and go to step 4. Otherwise make γ_1 and γ_s have no children in common by performing the $\mathcal{EEX}_2$ on $(\gamma_1 w, v w')$, where w' is the child of v. We know this is a valid $\mathcal{EEX}_2$ over $\mathcal{P}(l,r)$ by Proposition 4.2.5.

Step 4. Now that w is not a child of γ_s, we can perform the $\mathcal{EEX}_2$ on $(\gamma_1 w, \gamma_s v)$ (we needed w to not already be a child of γ_s in order to avoid creating a double edge).

Our phylogeny now has an open 1-ladder, with $u_1 = \gamma_1$ and $v_1 = v$. We know the $\mathcal{EEX}_2$ move in Step 4 was a valid move over $\mathcal{P}(l,r)$ by Proposition 4.2.5 (before the move: $\gamma_s w$ and $\gamma_1 v$ were not edges, and w cannot be an ancestor of γ_s). The 1-ladder created must be open, because the phylogeny cannot have any Noah vertices. □

Lemma 4.2.10. *Let r, l and k be positive integers such that $k < r$, and let $A \in \mathcal{P}(l,r)$ be a phylogeny with an open k-ladder and no Noah vertices. It is possible to turn A into a phylogeny with an open $(k+1)$-ladder using at most five $\mathcal{EEX}_2$ moves over the class $\mathcal{P}(l,r)$.*

Proof. Let a be the child of u_k (other than v_k) and let b be the child of v_k. Vertices a and b must be distinct otherwise they would be a Noah vertex. Let $A' \in \mathcal{P}(l, r-k)$ be the phylogeny obtained by contracting all the set $\{x, u_i, v_i | i = 1, 2, \ldots, k\}$ into a single tree vertex, x', which has children a and b. Applying Lemma 4.2.9 to A', we there exists a sequence of $\mathcal{EEX}_2$s so that the children of x' become a tree vertex u and a recombination vertex v such that u is the parent of v (other than x').

If we apply this same sequence of $\mathcal{EEX}_2$ moves to the phylogeny A, then u and v become the children (distinct from v_k) of u_k and v_k. If u is the child of u_k (other than v_k), then v must be the child of v_k and we are done.

Otherwise u is the child of v_k, and v is the child of u_k (other than v_k). In this case, we can create a $(k+1)$-ladder by performing the $\mathcal{EEX}_2$ on the pair $(u_k v, v_k u)$. The $(k+1)$-ladder created must be open because the phylogeny cannot have any Noah vertices. □

Lemma 4.2.11. *Let $l > 0$ and $r \geq 0$ be integers and let $A, B \in \mathcal{P}(l, r)$. If neither A nor B contain any Noah vertices, then phylogeny B can be obtained from A by a sequence of $\mathcal{EEX}_2$ operations over the class $\mathcal{P}(l, r)$.*

Proof. First we prove that there exists a phylogeny A' with an open r-ladder, that is obtainable from A, via a sequence of less than $5r$ $\mathcal{EEX}_2$ moves over $\mathcal{P}(l, r)$. We prove this by applying Lemma 4.2.9 to put an open 1-ladder in A, then iteratively for $k = 1, 2, \ldots, k-1$ apply Lemma 4.2.10. Similarly we have a phylogeny B with an open r-ladder obtainable from B, via a sequence of less than $5r$ $\mathcal{EEX}_2$ moves over $\mathcal{P}(l, r)$. It now suffices to show that B' can be obtained from A' via a sequence of $\mathcal{EEX}_2$ moves.

Let A'' be the phylogeny in $\mathcal{P}(l, 0)$ obtained by identifying all the vertices in the r-ladder (u_i and v_i for $i = 1, 2, \ldots, r$) and the child of the root, into a single tree vertex x whose children are the vertices that were the child of v_r, and the child of u_r other than v_r. Similarly construct $B'' \in \mathcal{P}(l, r)$. Now, since $\mathcal{P}(l, 0)$ is isomorphic to $\mathcal{B}(l+1)$, which is traversable by $\mathcal{NNI}$ moves. Since the $\mathcal{NNI}$ move over $\mathcal{B}(n)$ are a special case of the $\mathcal{EEX}_2$ move, it follows that $\mathcal{P}(l, 0)$ is traversable by $\mathcal{EEX}_2$ moves. Therefore B'' is obtainable from A'' by $\mathcal{EEX}_2$ moves over $\mathcal{P}(l, 0)$. This sequence of $\mathcal{EEX}_2$ moves corresponds exactly to a sequence of $\mathcal{EEX}_2$ moves changing A' into B' over $\mathcal{P}(l, r)$, without affecting the r-ladder. □

Definition 4.2.12. For any tree vertex, u, of a phylogeny $A \in \mathcal{P}(l, r)$, let $h(u)$ denote the number of Noah vertices on a path from the root to u. By Definition 4.2.6, $h(u)$

does not depend on the choice of path. We define here, the *signature* of A to be the sequence

$$(s_0, s_1, s_2, \ldots, s_k)$$

where k is the number of Noah vertices in A, and for all i, s_i is the number of tree vertices u in A such that $h(u) = i$.

Lemma 4.2.13. *Let $A, B \in \mathcal{P}(l, r)$ for some positive integers l and r. The phylogeny B can be obtained from A by a sequence of $\mathcal{EEX}_2$ operations if and only if A and B have the same signature.*

Proof. Let ρ_0 be the root of A, and let $\rho_1, \rho_2, \ldots, \rho_k$ be the Noah vertices in A such that ρ_i is a descendant of ρ_{i+1} for all $i < k$. If the signature of A is $(s_0, s_1, s_2, \ldots, s_k)$ then for $i = 0, 1, 2, \ldots, k-1$, let A_i denote the subnetwork consisting of all descendants of ρ_i which are ancestors of ρ_{i+1}. Additionally, let A_k denote the subnetwork consisting of all descendants of ρ_k. By Proposition 4.2.5, any valid $\mathcal{EEX}_2$ can only act on a pair of edges in the same A_i. Such operations won't change the signature, so the signature is invariant under any sequence of $\mathcal{EEX}_2$ moves over $\mathcal{P}(l, r)$.
For $i < k$, consider the subnetwork A_i and pretend that ρ_i is the root, and replace ρ_{i+1} with two of leaves labelled 1 and 2, one adjacent to each the vertices that were the parents of ρ_{i+1}. In this way, A_i is naturally isomorphic to a phylogeny A'_i with 2 leaves and s_i tree vertices, so $A'_i \in \mathcal{P}(2, s_i - 1)$. Similarly, if we consider the subnetwork A_k and pretend that ρ_k is the root, then A_k is naturally isomorphic to a phylogeny A'_k which has l leaves and s_k tree vertices. Again, by equating the sum of the in-degrees with the sum of the out degrees, we see that A'_k contains $s_k - l + 1$ recombination vertices, and so $A'_k \in \mathcal{P}(l, s_k - l + 1)$.

Construct subnetworks $B_0, B_1, B_2, \ldots, B_k$ in a similar manner. Since A and B have the same signature, $(s_0, s_1, \ldots, s_k)$, we have $A'_k, B'_k \in \mathcal{P}(l, s_k - l + 1)$ and for $i = 0, 1, 2, \ldots, k-1$, we have $A'_i, B'_i \in \mathcal{P}(2, s_i - 1)$, and none of these have any Noah vertices. So for all $j = 0, 1, 2, \ldots, k$, by Lemma 4.2.11, we can transform each A'_j into B'_j without creating any loops, double edges or cycles, and therefore the same transformation applied to each subnetwork A_j of A, will obtain B. □

4.3 SPR moves on phylogenies with recombination

In this section, we generalise the rSPR move (which acts over the class $\mathcal{B}(n)$ of binary trees), to the pSPR move (Definition 4.3.1) which acts on networks with

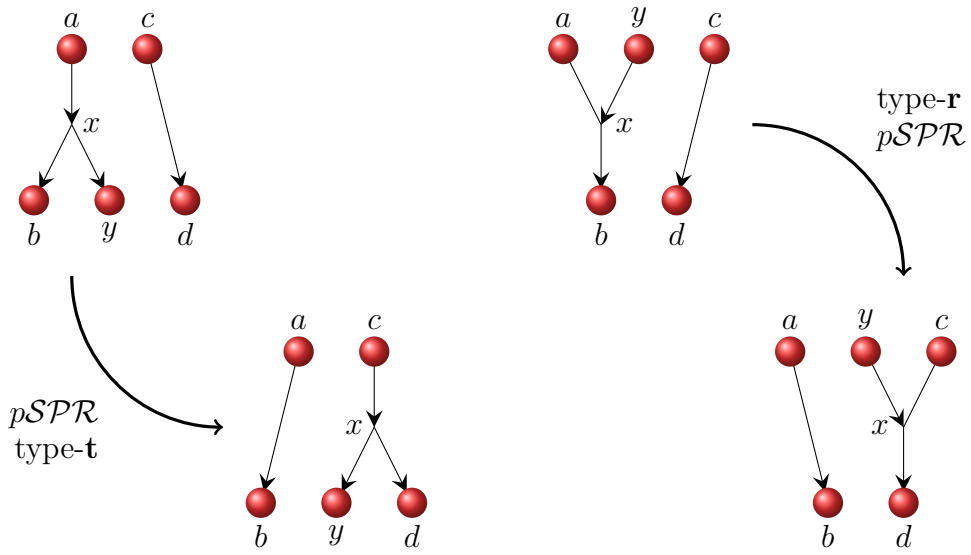


Figure 4.8: The type-**t** (left) and type-**r** (right) pSPR moves.

recombination. When $r = 0$, two phylogenies in $\mathcal{P}(l, 0)$ will differ by a pSPR if and only if the corresponding binary trees (in $\mathcal{B}(l+1)$) differ by a rSPR.

Definition 4.3.1. Let x be a vertex in a phylogeny $A \in \mathcal{P}(l, r)$ that is neither the root nor a leaf. Let a be a parent of x, let b be a child of x and let cd be an edge of A other than ax and xb. The *phylogeny subtree prune and regraft* (pSPR) move taking x from ab to cd is the $\mathcal{EEX}_3$ move on the edges (ax, xb, cd), and is allowed if and only if the resulting network is a phylogeny. This pSPR move is said to be *type-**t***, if x is a tree vertex, otherwise x is a recombination vertex and this pSPR move is said to be *type-**r***. For a class $\mathcal{C}$ of phylogenetic networks, a pSPR move is said to be *over the class* $\mathcal{C}$ if both the initial network, A, and the result of the pSPR move are members of class $\mathcal{C}$.

Formally, a pSPR move over $\mathcal{P}(l, r)$ is any $\mathcal{EEX}_3$ move, say (a_1b_1, a_2b_2, a_3b_3) such that $b_1 = a_2$ and the resulting network is a phylogeny (acyclic with no double edges nor loops). For $r = 0$, a pSPR move over $\mathcal{P}(l, 0)$ is exactly the same as a rSPR over $\mathcal{B}(l+1)$ (Definition 1.2.3). Informally, one can visualise the pSPR move as 'plucking' x from the path ab and then gluing x into the middle of edge cd (see Figure 4.8). The following two lemmas (Lemma 4.3.2 and 4.3.3) serve as useful lemmas, for determining when a pSPR is valid over $\mathcal{P}(l, r)$.

Lemma 4.3.2. *Let x be a tree vertex in a phylogeny $A \in \mathcal{P}(l, r)$, let a be the parent of x, let b and y be the children of x and let cd be an edge, distinct from ax, xb and*

xy. The type-$\mathbf{t}$ pSPR taking x from ab to cd is a valid pSPR over $\mathcal{P}(l,r)$ if and only if:

- *initially, ab is not an edge in A,*
- *vertices x, y, c and d are distinct, and*
- *the vertex y is not an ancestor of c.*

Proof. Let $B \in \mathcal{M}(l,r)$ be the result of this pSPR move. We will prove the 'if' part and the 'only if' part separately.

- Assume $B \in \mathcal{P}(l,r)$. Since B doesn't have any double edges, then ab must not have been an edge in A. The vertex x is distinct from y, c and d, because cx, xd and xy are all edges in B, and B has no loops. c must be distinct from d and y otherwise B would contain the the cycle $\{cx, xc\}$. Vertices y and d must also be distinct, otherwise B would have a double edge coming out of x. Therefore x, y, c and d are distinct. Finally, there cannot be a directed path from y to c, otherwise it would complete a cycle in B with edges cx and xy.
- Now assume the conditions of the proposition, and we shall show that B must be a phylogeny. Firstly if B had a loop, then (because A had no loops), it must be either ab, cx or xd. However, a and b are distinct because otherwise there would have been a cycle ax, xb in A, and neither cx nor xd can be loops as x, c and d are distinct vertices. Secondly we need to show that none of the new edges form double edges. We know ab cannot be a double edge, since it was not an edge initially in A. Moreover, neither cx nor xd could be edges in A, since we already knew all edges incident with x; they are ax, xb and xy. Finally, since A is acyclic, if B were to contain a cycle, then it would have to involve at least one of the edges ab, cx or xd. Edge ab cannot complete a cycle in B because the path (ax, xb) didn't complete a cycle in A. If xd completed a cycle in B, then this cycle would have to involve edge cx too (c is the unique parent of x in B). This wouldn't complete a cycle since d was not an ancestor of c in A. Therefore, if this pSPR were, to complete a cycle it would have to involve edge cx and thus also xy. However, y is not an ancestor of c, so B must be acyclic.

□

Lemma 4.3.3. *Let x be a recombination vertex in a phylogeny $A \in \mathcal{P}(l, r)$, let a and y be the parents of x, let b be the child of x and let cd be an edge, distinct from ax, xb and yx. The type-**r** pSPR taking x from ab to cd is a valid pSPR over $\mathcal{P}(l, r)$ if and only if:*

- *initially, ab is not an edge in A,*
- *vertices x, y, c and d are distinct, and*
- *the vertex y is not a descendant of d.*

Proof. Let $B \in \mathcal{M}(l, r)$ be the result of this pSPR move. We will prove the 'if' part and the 'only if' part separately.

- Assume $B \in \mathcal{P}(l, r)$. Since B doesn't have any double edges, then ab must not have been an edge in A. The vertex x is distinct from y, c and d, because cx, xd and yx are all edges in B, and B has no loops. d must be distinct from c and y otherwise B would contain the the cycle $\{xd, dx\}$. Vertices y and c must also be distinct, otherwise B would have a double edge coming into x. Therefore x, y, c and d are distinct. Finally, there cannot be a directed path from d to y, otherwise it would complete a cycle in B with edges yx and xd.

- Now assume the conditions of the proposition, and we shall show that B must be a phylogeny. Firstly if B had a loop, then (because A had no loops), it must be either ab, cx or xd. However, a and b are distinct because otherwise there would have been a cycle ax, xb in A, and neither cx nor xd can be loops as x, c and d are distinct vertices. Secondly we need to show that none of the new edges form double edges. We know ab cannot be a double edge, since it was not an edge initially in A. Moreover, neither cx nor xd could be edges in A, since we already knew all edges incident with x; they are ax, xb and yx. Finally, since A is acyclic, if B were to contain a cycle, then it would have to involve at least one of the edges ab, cx or xd. Edge ab cannot complete a cycle in B because the path (ax, xb) didn't complete a cycle in A. If cx completed a cycle in B, then this cycle would have to involve edge xd too, since d is the unique child of x (in B). Moreover this wouldn't complete a cycle since c was not a descendant of c in A. Therefore, if this pSPR were, to complete a cycle it would have to involve edge xd and thus also yx. However, y is not a descendant of d, so B must be acyclic.

□

Lemma 4.3.4. *Let l and r be positive integers. If $A \in \mathcal{P}(l,r)$ is any phylogeny, then it is possible to obtain a phylogeny with a 1-ladder, from A, by performing at most three pSPR moves, such that none of the pSPR moves affect the edge incident with the root.*

Proof. Let ρ be the root of A, and let x be the child of ρ. By lemma 4.1.3, there must exist a recombination vertex v and a two distinct directed paths

$$\rho x u_1 u_2 \dots u_i v \qquad \text{and} \qquad \rho x u'_1 u'_2 \dots u'_j v,$$

both consisting only of tree vertices (except for ρ and v).

Step 1. If $i = 0$ or $j = 0$, then do nothing. Otherwise let b be the child of v and wlog assume u'_j is not a parent of b (otherwise u_i would be a non-parent of b and we can swap the roles of u_i and u'_j). Let d be the child of x such that d is not an ancestor of u_i (d may or may not be u'_1). Perform the type-**r** pSPR move taking v from $u'_j b$ to xd. We know this is a valid pSPR move by Lemma 4.3.3.

Now we have performed at most one pSPR move, so that one of the children of x is a recombination vertex v and there is a directed path

$$x u_1 u_2 \dots u_i v$$

consisting of only tree vertices (apart from v). If $i = 1$, then we have a 1-ladder so we are done. So we asssume $i \geq 2$. Let c be the child of u_i, other than v and let c' be the child of u_{i-1} other than u_i.

Step 2. If c and c' are distinct, then do nothing. Otherwise perform the type-**t** pSPR taking u_i from $u_{i-1}v$ to vb, where b is the child of v. We know this is a valid type-**t** pSPR by Lemma 4.3.2.

Now we have performed at most one pSPR move, so that one of the children of x is a recombination vertex v and there is a directed path

$$x u_1 u_2 \dots u_k v$$

consisting of only tree vertices (apart from v). Again, if $k = 1$, then we have a 1-ladder so we are done. So we asssume $k \geq 2$. Let c be the child of u_k, other than v and note that c cannot be a child of u_{k-1}.

Step 3. Perform the type-**t** pSPR taking u_k from $u_{k-1}c$ to xu_1. By Lemma 4.3.2, this is a valid type-**t** pSPR move.

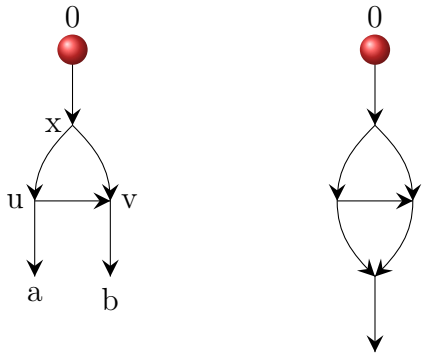


Figure 4.9: A phylogeny with an open 1-ladder (left) and a phylogeny with a closed 1-ladder (right).

A now has a 1-ladder, and we achieved this by performing at most three pSPR moves. None of these pSPRs touched the edge ρx. □

Lemma 4.3.5. *Let l and r be positive integers with $r \geq 2$. If $A \in \mathcal{P}(l, r)$ is any phylogeny with an open 1-ladder, then it is possible to create a phylogeny with a closed k-ladder in A, for some $k \geq 1$, by performing at most four pSPR moves.*

Proof. Let ρ be the root of A and let x be the child of ρ. Since A has a 1-ladder, it's children must be a tree vertex u and a recombination vertex v such that uv is an edge. Let a be the child of u, other than v, and let b be the child of v. Since the 1-ladder is open, a and b must be distinct (as depicted in Figure 4.9). Since a and b are distinct, the phylogeny A' obtained by identifying x, u and v into a new tree vertex which is a child of ρ and a parent of both a and b, then $A' \in \mathcal{P}(l, r-1)$. We can apply Lemma 4.3.4 to find a sequence of at most three pSPR moves to put a 1-ladder into A' without effecting edge ρx. If we apply the same sequence of pSPR moves to A, then we arrive at a phylogeny such that the children of u and v are connected by an edge. Now let a' and b' be the children of u and v respectively with $a' \neq v$.

Case 1. If the edge between a' and b' is oriented $b'a'$, and a' and b' have a common child, then we can create a closed 2-ladder by performing the type-**t** pSPR taking b' from va' to ua'. This pSPR is a valid pSPR over $\mathcal{P}(l, r)$ by Lemma 4.3.2.

Case 2. If the edge between a' and b' is oriented $b'a'$, and a' and b' have no common child, then we can create a closed 1-ladder by performing the type-**t** pSPR taking b' from va' to $a'c$, where c is the child of a'. This pSPR is a valid pSPR over $\mathcal{P}(l, r)$ by Lemma 4.3.2.

Case 3. If the edge between a' and b' is oriented $a'b'$, then we either already have a closed 2-ladder, or we can create a closed 1-ladder by performing the type-**t** pSPR taking a' from ub' to $b'c$, where c is the child of b'. This pSPR is a valid pSPR over $\mathcal{P}(l,r)$ by Lemma 4.3.2.

□

Lemma 4.3.6. *Let k, l and r be positive integers with $k \leq r-2$. If $A \in \mathcal{P}(l,r)$ is any phylogeny with a closed k-ladder, then for some $k' > k$, it is possible to obtain a phylogeny with a closed k'-ladder, from A, by performing at most five pSPR moves.*

Proof. Let x be the most ancestral Noah vertex of A; the common child of vertices u_k and v_k, that closes the k-ladder, and let y be the child of x. Let A' be the phylogeny obtained by deleting all ancestors of x and letting x be the new root. Applying Lemma 4.3.4 to $A' \in \mathcal{P}(l, r-k)$ we can perform at most 3 pSPR moves so that y now has children u and v such that u is a tree vertex and v is a recombination vertex and uv is an edge. We can now obtain an $(k+2)$-ladder by performing the pSPR taking y from xv to $u_k x$. If this $(k+2)$-ladder is closed then set $k' = k+2$ and we are done. Otherwise this $(k+2)$ ladder is open, and so if b is the child of v then b is not a child of u. We can construct a closed $(k+1)$-ladder by performing the pSPR move taking u from yv to vb. □

Lemma 4.3.7. *For any integers $l \geq 1$ and $r \geq 2$, the class $\mathcal{P}(l,r)$ is traversable by pSPR moves. Moreover,*

$$D_p\mathcal{SPR}(\mathcal{P}(l,r)) \leq l + 10r - \Theta(\sqrt{l}).$$

Proof. Let A and B be chosen arbitrarily from the set of phylogenies in $\mathcal{P}(l,r)$. Applying Lemma 4.3.4 we can put a 1-ladder into A using at most 3 pSPR moves. Then, since $r \geq 2$, we can use Lemma 4.3.5 to put a closed ladder into A, using a further 4 pSPR moves. Then iteratively apply Lemma 4.3.6 until we obtain a phylogeny A', with a closed $(r-1)$-ladder. A' was obtained from A by at most $7 + 5(r-2)$ pSPR moves. Similarly let B' be a phylogeny with a $(r-1)$-ladder, obtainable from B by at most $7 + 5(r-2)$ pSPR moves.

Without the ladders, A' and B' have no recombination vertices, and so the tree of descendants of their Noah vertices, are isomorphic to binary trees in $\mathcal{B}(l+1)$.

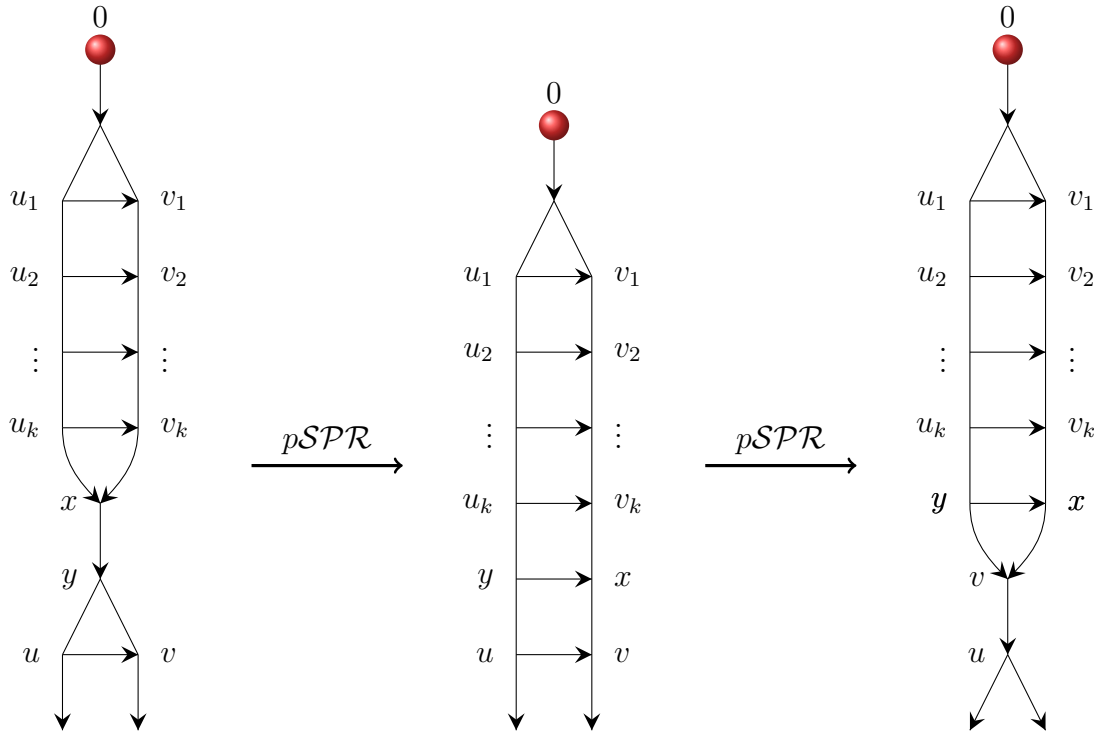


Figure 4.10: The construction in the proof of Lemma 4.3.6.

Therefore, B' can be obtained from A' using $l - \Omega\left(\sqrt{l}\right)$ rSPR moves (Theorem 2.0.1, which are pSPR moves. Therefore, over the class $\mathcal{P}(l, r)$,

$$\begin{aligned} d_{p\mathcal{SPR}}(A, B) &\leq d_{p\mathcal{SPR}}(A, A') + d_{p\mathcal{SPR}}(A', B') + d_{p\mathcal{SPR}}(B', B) \\ &\leq (7 + 5(r-2)) + l - \Omega\left(\sqrt{l}\right) + (7 + 5(r-2)) \\ &= l + 10r - \Omega\left(\sqrt{l}\right). \end{aligned}$$

□

Theorem 4.3.7 tells us that $\mathcal{P}(l, r)$ is traversable using both type-**t** and type-**r** pSPR moves. However, as Lemma 4.3.8 will demonstrate, if $l > 3r + 7$, then $\mathcal{P}(l, r)$ is traversable by type-**t** pSPR moves, because a type-**r** pSPR move can be simulated by type-**t** pSPR moves.

Lemma 4.3.8. *Let l and r be positive integers and let $A, B \in \mathcal{P}(l, r)$ such that B can be obtained from A via a the type-**r** pSPR move. If $l > 3r + 7$, then it is possible to change A into B using at most five type-**t** pSPR moves over the class $\mathcal{P}(l, r)$.*

Proof. Suppose B is obtained from A by the type-**r** pSPR move that takes a recombination vertex x from ab to cd. Let $y \neq a$ be the other parent of x in A. So A can be

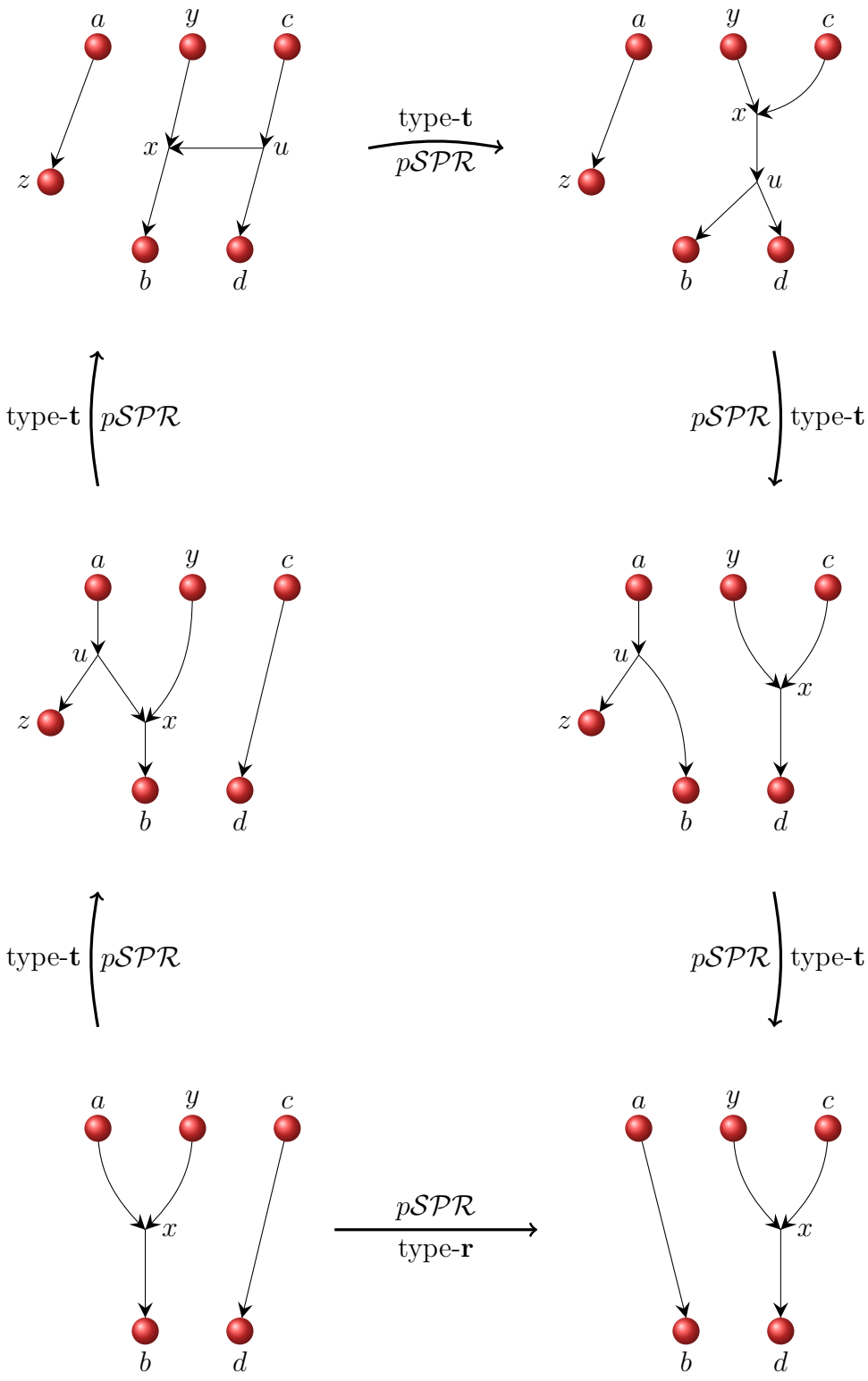


Figure 4.11: The construction in the proof of Lemma 4.3.8.

obtained from B by the type-**r** pSPR taking x from cd to ab. Consider the following statements:

- Vertex b is an ancestor of c.
- Vertex d is an ancestor of a.

Each statement is either true in both A and B, or neither A nor B. Moreover both statements cannot hold, since the phylogenies are acyclic. Hence, without loss of generality, we assume b is not an ancestor of c (otherwise we simply swap the roles of ab and cd, and find a sequence of 5 type-**t** pSPRs that transform B into A).

Now we find a leaf z such that the parent of z is a tree vertex, u, which is distinct from a, b, c, d, x and y, such that the other child of u (other than z) is not a recombination vertex. We know such a leaf z exists because $l > 3r + 7$ (each recombination vertex can ruin at most three leaves, the vertices a, c and y can ruin at most one leaf each, and b and d could potentially ruin 2 leaves each). Let the parent of u be p, and let $z' \neq z$ be the other child of u.

If b and d are distinct vertices, the we simply perform the following sequence of type-**t** pSPR moves (depicted in Figure 4.11).

Step 1. Perform the pSPR taking u from pz' to ax.

Step 2. Perform the pSPR taking u from az to cd.

Step 3. Perform the pSPR taking u from cx to xb.

Step 4. Perform the pSPR taking u from xd to az.

Step 5. Perform the pSPR taking u from ab to pz'.

However, if $b = d$, then its even easier. We can perform Step 3 and Step 4 simultaneously with a single type-**t** pSPR. Explicitly, the pSPR taking u from cx to az. □

It would be nice if we could relax the $l > 3r + 7$ condition in Lemma 4.3.8. Even when $l = 1$, it is typically quite easy to find a suitable tree vertex u in A, such that the method presented in the proof works. However, sometimes no such vertex u exists. In the small examples I have investigated, it has always been possible to create such a vertex u, using a type-**t** pSPR move (which we would then have to 'undo' afterwards), however it is difficult to see how to do this in general.

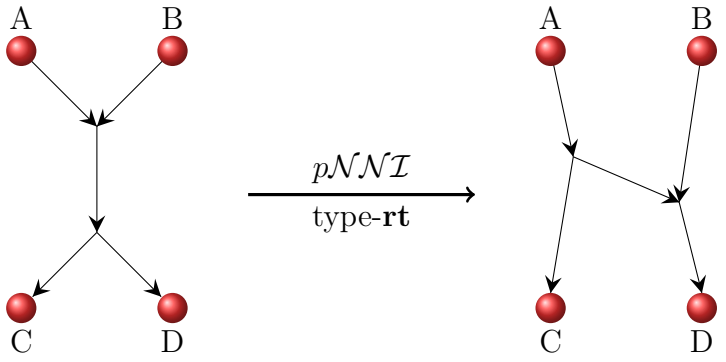


Figure 4.12: The type-**rt** pNNI move. This is never an $\mathcal{EEX}_2$ move.

4.4 Nearest neighbour interchange operations on phylogenies

In an analogous manner to the way we generalised rSPR moves on binary trees, to pSPR moves on phylogenies, the NNI move on binary trees can be generalised to the *phylogeny nearest neighbour interchange* (pNNI move) which acts on the class of phylogenies. What we mean by this, is that a pair of phylogenies in $\mathcal{P}(l, 0)$ differ by a pNNI move if any only if the corresponding binary trees in $\mathcal{B}(l+1)$ differ by a NNI move. In this section we will formally define the pNNI move, and then show that, for $l > 3r + 7$, the class $\mathcal{P}(l, r)$ is traversable by pNNI moves.

Definition 4.4.1. A *phylogeny nearest neighbour interchange* (pNNI) is an operation which can be applied to any edge uv such that each of u and v are either a tree vertex or a recombination vertex. The operation is a two step process:

1. Contract edge uv into a new vertex x with $\Gamma_+(x) = \Gamma_+(u) \sqcup \Gamma_+(v) \backslash \{u\}$ and $\Gamma_-(x) = \Gamma_-(u) \sqcup \Gamma_-(v) \backslash \{v\}$. ($u$ and v are both deleted in this step)

2. Resolve x into a pair of vertices u' and v' such that $\Gamma_+(x) = \Gamma_+(u') \sqcup \Gamma_+(v')$ and $\Gamma_-(x) = \Gamma_-(u') \sqcup \Gamma_-(v')$, and then add edge $u'v'$.

This process is only allowed, so long as the resulting digraph is a phylogeny. In particular, it doesn't matter if any double-edges or cycles are created during the contraction of uv, so long as they are subsequently destroyed in the resolution of x. More generally, a pNNI over a class $\mathcal{C}$ of phylogenies, is allowed only if the resulting phylogeny is in $\mathcal{C}$. Let $p\mathcal{NNI} = p\mathcal{NNI}(l, r)$ denote the set of pairs $(A, B) \in \mathcal{P}(l, r)^2$ such that B is the result of an pNNI performed on A. The pNNI operation is:

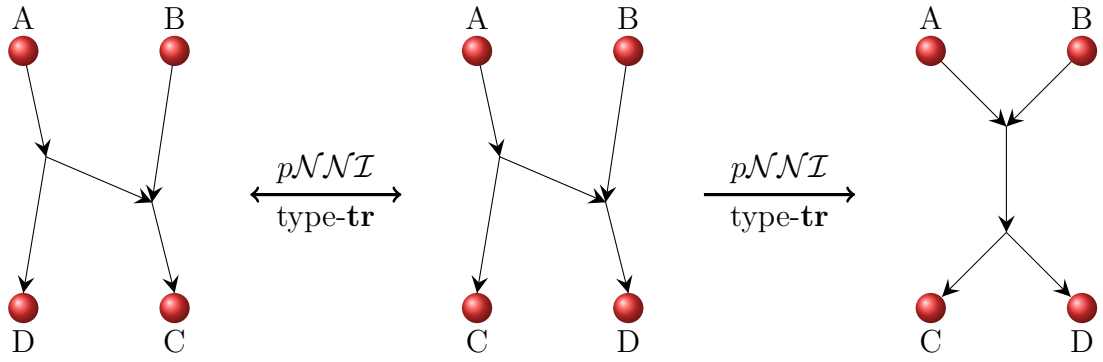


Figure 4.13: Some type-**tr** pNNI moves. One of them (right) is not an $\mathcal{EEX}_2$ move.

- type-**tt** if both u and v are tree vertices (and hence u' and v' must both be tree vertices).
- type-**tr** if u is a tree vertex and v is a recombination vertex.
- type **rt** if u is a recombination vertex and v is a tree vertex.
- type-**rr** if both u and v are recombination vertices (and hence u' and v' must both be recombination vertices).

A type-**tt** pNNI move is always an $\mathcal{EEX}_2$ move (Figure 4.6). The type-**rr** pNNI move is another special case of the $\mathcal{EEX}_2$ move. Type-**tr** pNNI moves are not $\mathcal{EEX}_2$ moves in general (Figure 4.13), and any non-identical type-**rt** pNNI move is never an $\mathcal{EEX}_2$ move (Figure 4.12).

Arguably, one might suggest that the generalisation of the NNI operation should only include type-**tt** pNNI moves, because the pNNI moves over $\mathcal{P}(l, 0)$ which correspond to NNI moves over $\mathcal{B}(l+1)$ are always type-**tt** (there are no recombination vertices). However, if we were to exclude type-**rt** pNNI moves, then the class $\mathcal{P}(l, r)$ would not be traversable, since we could never destroy any Noah vertices. For every operation we include, we should include the inverse too. So if we include type-**rt** pNNIs, then might as well include type-**tr** pNNIs, because any type-**tr** pNNI is the composition of a type-**rt** pNNI and the inverse of a type-**rt** pNNI (see Figure 4.14).

We now prove a little lemma (Lemma 4.4.2), which is used to show that the class $\mathcal{P}(l, r)$ is traversable by $p\mathcal{NNI}$ moves, in Theorem 4.4.3.

Lemma 4.4.2. *Let x be any tree vertex in a phylogeny $A \in \mathcal{P}(l, r)$. Let a be the parent of x and let b and y be the children of x. If ab is not an edge in A, then it is*

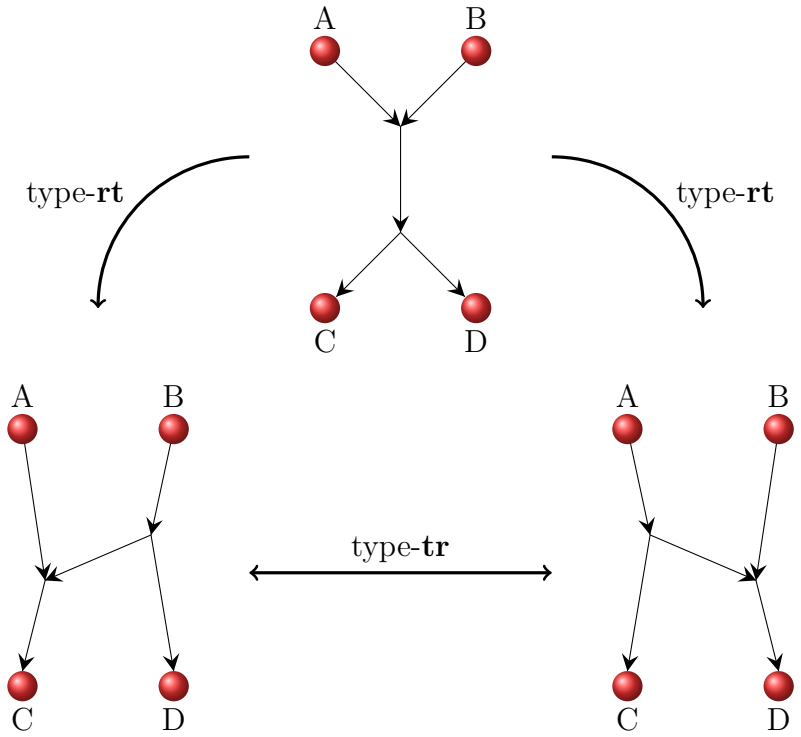


Figure 4.14: Any type-**tr** move can be achieved by the inverse of a type-**rt** pNNI followed by a type-**rt** pNNI.

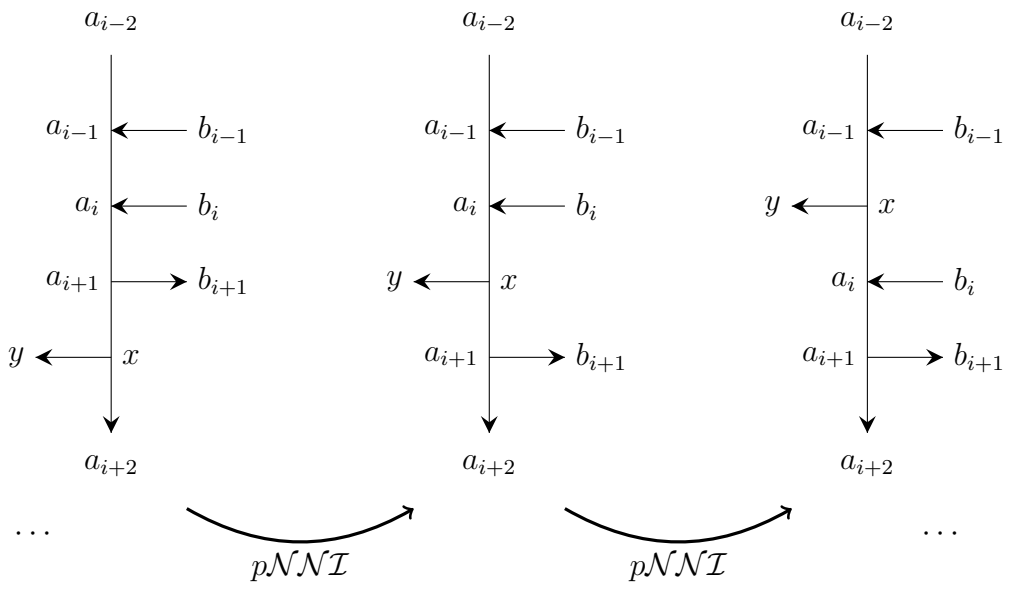


Figure 4.15: The construction in Lemma 4.4.2.

*possible to perform the type-**t** pSPR move taking x from ab to the edge incident with the root, using only pNNI moves, over the class $\mathcal{P}(l, r)$.*

Proof. Consider the shortest path in A,

$$a_0 a_1 a_2 \dots a_k,$$

such that a_0 is the root and $a_k = a$, and let $a_{k+1} = b$. For each tree vertex a_i in the path let $b_i \neq a_{i+1}$ be the other child of a_i, and for each recombination vertex a_j in the path, let $b_j \neq a_{j-1}$ be the other parent of a_j. Since A is acyclic, and this path has minimal length, none of the vertices $\{b_i\}_{i=1}^{k}$ lie in the path. Moreover y is not on the path, because A is acyclic (although y may or may not be one of the b_is - this doesn't matter).

Now consider the following iteration: for $i = k, k-1, k-2, \dots, 2, 1$, perform the type-**t** pSPR taking x from $a_i a_{i+1}$ to $a_{i-1} a_i$. The move is a pNNI on the edge $a_i x$ (either a type-**tt** or a type-**rt**, depending on whether a_i is a tree vertex or a recombination vertex). Furthermore, each intermediate network is a valid phylogeny, because it cannot have any loops, double edges or cycles. In particular, we needed ab to be a non-edge so that whenever x is between a_i and a_{i+1}, the edge $a_i a_{i+1}$ is not present. □

Theorem 4.4.3. *For any integers $r \geq 0$ and $l > 3r + l$, the class $\mathcal{P}(l, r)$ is traversable by $p\mathcal{NNI}$ operations.*

Proof. By Lemma 4.3.7, is sufficies to show that any $p\mathcal{SPR}$ can be achieved by $p\mathcal{NNI}$s over $\mathcal{P}(l, r)$. Furthermore, by Lemma 4.3.8, it suffices to show this for a type-**t** pSPR. So we consider arbitrary $A, B \in \mathcal{P}(l, r)$, such that B is obtained from A, by performing the type-**t** pSPR move which takes tree vertex x from ab to cd. Let ρ be the root of A and let z be the child of ρ. Let $C \in \mathcal{P}(l, r)$ be the phylogeny obtained from A, by performing type-**t** pSPR which takes x from ab to ρz. By Lemma 4.3.2, C is a valid phylogeny.

If we apply Lemma 4.4.2 to A, then deduce that C can be obtained from A by a sequence of $p\mathcal{NNI}$ operations such that all intermediate networks are valid phylogenies. Similarly (applying Lemma 4.4.2 to B), we see that C can be obtained from B by $p\mathcal{NNI}$s over the class $\mathcal{P}(l, r)$. Since the inverse of a pNNI is a pNNI, we see that B can be obtained from A by performing the sequence of pNNIs that turn A into C followed by the inverses of the pNNIs (in reverse order) that turn B into C. □

Chapter 5

Betweenness and nonbetweenness

The betweenness function bet(n) is the minimum number of total orderings of n objects such that for any three distinct objects a, b and c, there is an ordering in which b is between a and c. The nonbetweenness function nbet(n) is the minimum number of total orderings such that for any three distinct objects a, b and c, there is an ordering in which b is not between a and c. Our motivation for studying betweenness and nonbetweenness comes from the application to phylogenetic triplets. However, since betweenness and nonbetweenness are rich topics in their own right, the phylogenetic applications are left until Section 5.5.

In Section 5.1, we formally define betweenness and nonbetweenness, and state the main results of this Chapter. In Section 5.2, we compute the exact value of nbet(n) for all n. In Section 5.3, we show that bet(n) = $\Theta(\log n)$. Section 5.4 focuses on a generalisation called ECP[II] (the extreme constraint problem). These results are then used in Section 5.5 to show that the size of a minimal set $\mathcal{A} \subseteq \mathcal{B}(n)$, such that every phylogenetic triplet is consistent with at least one tree in $\mathcal{A}$, is $\Theta(\log\log(n))$.

5.1 Notation and definitions

In this section, we give formal definitions of orderings, relative orders, order-systems and constraints. We also define the betweenness function, bet(n), and the nonbetweenness function nbet(n).

Definition 5.1.1. Let $S_3 = \{123, 132, 213, 231, 312, 321\}$ be the group of permutations on 3 objects and let $n \geq 3$ be an integer. An *ordering* (total ordering on $[n]$) is a permutation $\phi : [n] \to [n]$. For an ordering ϕ and any distinct $x, y \in [n]$, we sometimes say x *comes before* y *in* ϕ to mean $\phi(x) < \phi(y)$. Otherwise $\phi(x) > \phi(y)$

and we would say x *comes after* y. We sometimes express ϕ as the n-tuple:

$$\phi = \big(\phi^{-1}(1), \phi^{-1}(2), \phi^{-1}(3), \ldots, \phi^{-1}(n)\big).$$

For any ordering ϕ, let ϕ^R denote the reverse ordering; for all $i = 1, 2, \ldots, n$, we have $\phi^R(i) = n - \phi(i) + 1$. For example, if $n = 3$ and $\phi = (2, 3, 1)$ then $\phi(1) = 3$, $\phi(2) = 1$ and $\phi(3) = 2$ and

$$\phi = (2, 3, 1) \quad \Longleftrightarrow \quad \phi^R = (1, 3, 2).$$

A *constraint* is a triple $\mathbf{x} = (x_1, x_2, x_3)$ of distinct integers: $x_1, x_2, x_3 \in [n]$. For an ordering ϕ and a constraint $\mathbf{x} = (x_1, x_2, x_3)$, let the *relative order* of $\mathbf{x}$ given by ϕ be

$$\mathrm{ord}(\phi, \mathbf{x}) = abc$$

where $abc \in S_3$ such that $\phi(x_a) < \phi(x_b) < \phi(x_c)$.

Definition 5.1.2. An ordering ϕ is said to *between-satisfy* a constraint (x_1, x_2, x_3) if and only if x_2 comes before one of x_1 or x_3, and comes after the other. *i.e.* ϕ between-satisfies $\mathbf{x}$ if and only if

$$\mathrm{ord}(\phi, \mathbf{x}) = 123 \quad \text{or} \quad \mathrm{ord}(\phi, \mathbf{x}) = 321.$$

An ordering is said to *nonbetween-satisfy* a constraint if and only if it does not between-satisfy the constraint. *i.e.*

$$\phi \text{ nonbetween-satisfies } \mathbf{x} \quad \Longleftrightarrow \quad \mathrm{ord}(\phi, \mathbf{x}) \in \{132, 213, 231, 312\}.$$

Any set of orderings on $[n]$ is called an *order-system*. An order-system, Φ, is said to *between-satisfy* a constraint, $\mathbf{x}$, if and only if $\mathbf{x}$ is between-satisfied by at least one $\phi \in \Phi$. Similarly, Φ *nonbetween-satisfies* $\mathbf{x}$, if and only if $\mathbf{x}$ is nonbetween-satisfied by at least one $\phi \in \Phi$.

Example 5.1.3. For $n = 5$ consider the order-system $\Phi = \{\phi, \psi\}$ with $\phi = (1, 2, 3, 4, 5)$ and $\psi = (4, 5, 1, 2, 3)$. *i.e.* $\phi(i) = i$ for all i and:

$$\psi(4) = 1, \psi(5) = 2, \psi(1) = 3, \psi(2) = 4 \text{ and } \psi(3) = 5.$$

Also consider the constraints $\mathbf{x} = (1, 2, 3)$, $\mathbf{y} = (2, 4, 5)$ and $\mathbf{z} = (1, 5, 4)$. We can verify that Φ between-satisfies $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ because:

- Both ϕ and ψ between-satisfy $\mathbf{x} = (1, 2, 3)$ because $\mathrm{ord}(\phi, \mathbf{x}) = \mathrm{ord}(\psi, \mathbf{x}) = 123$.

n	3	4	5	6	7
bet(n)	3	4	5	5	5
nbet(n)	2	2	3	3	3

Figure 5.1: The table of bet(n) and nbet(n) for $3 \leq n \leq 7$. The values of nbet come from Theorem 5.1.5 and the values of bet are computed in Section 5.3.

- ϕ between-satisfies $\mathbf{y} = (2, 4, 5)$ because $\phi(2) < \phi(4) < \phi(5)$.
  (*i.e.* $\text{ord}(\phi, \mathbf{y}) = 123$)

- ψ between-satisfies $\mathbf{z} = (1, 5, 4)$ because $\psi(1) > \psi(5) > \psi(4)$.
  (*i.e.* $\text{ord}(\psi, \mathbf{z}) = 321$)

We can also say that Φ nonbetween-satisfies $\mathbf{y}$ and $\mathbf{z}$ because:

- ψ nonbetween-satisfies $\mathbf{y}$ since 4 comes before both 2 and 5 in ψ.
  (*i.e.* $\text{ord}(\psi, \mathbf{y}) = 231$)

- ϕ nonbetween-satisfies $\mathbf{z}$ since 5 comes after both 1 and 4 in ϕ.
  (*i.e.* $\text{ord}(\phi, \mathbf{z}) = 132$)

However Φ does not nonbetween-satisfy $\mathbf{x}$.

Definition 5.1.4. For any integer $n \geq 3$:

- Let NBET = NBET(n) be the collection of all order-systems on $[n]$ which nonbetween-satisfy all constraints.

- Let BET = BET(n) be the collection of all order-systems on $[n]$ which between-satisfy all constraints.

- Let nbet(n) and bet(n) denote the minimum size of an order-system in NBET(n) and BET(n) respectively. *i.e.*

$$\text{nbet}(n) = \min\big\{|\Phi| : \Phi \in \text{NBET}(n)\big\}$$
$$\text{and} \qquad \text{bet}(n) = \min\big\{|\Phi| : \Phi \in \text{BET}(n)\big\}.$$

Figure 5.1 displays the values of nbet(n) and bet(n) for all n up to 7. Theorem 5.1.5 presents the value of nbet(n) and Theorem 5.1.6 presents the asymptotics of bet(n) as $n \to \infty$.

Theorem 5.1.5. *For all $n \geq 3$,*

$$\mathrm{nbet}(n) = \lceil \log_2 \log_2 n \rceil + 1.$$

Theorem 5.1.6. *As $n \to \infty$,*

$$\mathrm{bet}(n) = \Theta(\log n).$$

Theorem 5.1.5 is proven in Section 5.2 and Theorem 5.1.6 is proven in Section 5.3. We now conclude this section with a Proposition.

Proposition 5.1.7. *If $\Phi \in \mathrm{BET}(n)$, then for any permutation $\sigma \in S_n$ we have $\Phi\sigma \in \mathrm{BET}(n)$, where*

$$\Phi\sigma = \{\phi \circ \sigma : \phi \in \Phi\}.$$

Similarly, $\Phi \in \mathrm{NBET}(n)$ if and only if $\Phi\sigma \in \mathrm{NBET}(n)$.

Proof. For an arbitrary constraint $\mathbf{x} = (x_1, x_2, x_3)$, consider the constraint $\mathbf{x}'$, defined to be $\mathbf{x}' = (\sigma(x_1), \sigma(x_2), \sigma(x_3))$. For any ordering ϕ, if $abc \in S_3$ is such that $\phi(\sigma(x_a)) < \phi(\sigma(x_b)) < \phi(\sigma(x_c))$ then

$$\mathrm{ord}(\phi \circ \sigma, \mathbf{x}) = abc = \mathrm{ord}(\phi, \mathbf{x}').$$

- If $\Phi \in \mathrm{BET}(n)$ then, for any constraint $\mathbf{x}$, there is some $\phi \in \Phi$ such that ϕ between-satisfies $\mathbf{x}'$ and so $\mathrm{ord}(\phi, \mathbf{x}')$ is either 123 or 321. So $\Phi\sigma \in \mathrm{BET}(n)$ because
$$\mathrm{ord}(\phi \circ \sigma, \mathbf{x}) = \mathrm{ord}(\phi, \mathbf{x}') \in \{123, 321\}.$$
- If $\Phi \in \mathrm{NBET}(n)$ then, for any constraint $\mathbf{x}$, there is some $\phi \in \Phi$ such that ϕ nonbetween-satisfies $\mathbf{x}'$ and so $\mathrm{ord}(\phi, \mathbf{x}')$ is 132, 213, 231 or 312. Therefore $\Phi\sigma \in \mathrm{NBET}(n)$ because
$$\mathrm{ord}(\phi \circ \sigma, \mathbf{x}) = \mathrm{ord}(\phi, \mathbf{x}') \in \{132, 213, 231, 312\}.$$

□

5.2 Nonbetweenness

The results in this section are dedicated to the proof of Theorem 5.1.5. The proofs in this section fundamentally rely on Theorem 5.2.1 which is due to Erdős and Szekeres [24]. A proof is given here for completeness. Lemma 5.2.4 demonstrates that the bound given in the Erdős-Szekeres theorem is tight.

Theorem 5.2.1. *[24] Any sequence of m^2+1 real numbers contains a monotonic subsequence of length $m+1$.*

Proof. Let $x_1, x_2, \ldots, x_{m^2+1}$ be an arbitrary sequence of real numbers and let $f(i)$ denote the maximum length of all non-decreasing subsequences beginning with the i^{th} number x_i. For $i < j$, if $f(i) = f(j)$ then $x_i > x_j$. Therefore for any k, the subsequence $(x_i : f(i) = k)$ must be decreasing. Either $f(i) > m$ for some i, or $f(i) \leq m$ for all i. In the former case, there is a non-decreasing subsequence of length $m+1$ beginning with x_i. In the later case $f : [m^2+1] \to [m]$, so by the pigeon-hole principle there must be some $k \in [m]$ such that $|\{x_i : f(x_i) = k\}| > m$. In this case, the subsequence $(x_i : f(i) = k)$ is a decreasing subsequence of length at least $m+1$. □

Definition 5.2.2. A sequence of points in $\mathbb{R}^d$ is called monotonic if and only if it is monotonic in each coordinate.

Corollary 5.2.3. *Let d and m be positive integers. Any sequence of $N = m^{2^d}+1$ points in $\mathbb{R}^d$ contains a monotonic subsequence of length $m+1$.*

Proof. (by induction on d) The base case, $d = 1$, is exactly the Erdős-Szekeres theorem (Theorem 5.2.1). For the inductive step, $d \geq 2$, we can use Theorem 5.2.1 to find a subsequence of length M which is monotonic in the first coordinate, such that

$$M = \sqrt{N-1} + 1 = m^{2^{d-1}} + 1.$$

Now we can apply the inductive assumption to this sequence (the first coordinates can now be guaranteed to be monotonic, so they are ignored) to find the required subsequence of length $m+1$. □

Lemma 5.2.4. *For positive integers d and m, there exists a sequence $x_0, x_1 \ldots, x_{N-1}$, of $N = m^{2^d}$ points in $\mathbb{R}^d$ with no monotonic subsequence of length $m+1$. Moreover, one can construct such a sequence for which the k^{th} coordinates are a rearrangement of $[N]$ for each $k = 1, 2, \ldots, d$.*

In the following proof, we construct a suitable sequence in which the k^{th} coordinates are a rearrangement of $\{0, 1, 2, \ldots, N-1\}$, not the conventional $[N] = \{1, 2, 3, \ldots, N\}$. This can be overcome by simply adding one to each coordinate.

Proof. When $m = 1$ the result is trivial, so we only consider when $m \geq 2$. We construct a suitable sequence by induction on d.

Base case For $d = 1$ we have $N = m^2$. For all a, b in the range $\{0, 1, 2, \ldots, m-1\}$, let

$$x_{am+b} = am + (m - b - 1).$$

Now let $(x_{k_i})_{i=1}^{m+1}$ be an arbitrary subsequence of $(x_k)_{k=0}^{N-1}$ with length $m+1$.

- By the pigeon-hole principle, there must be two terms $x_{k_i}, x_{k_{i'}}$ (wlog $k_i < k_{i'}$) such that k_i and $k_{i'}$ have equal remainders upon division by m. We have $x_{k_i} - x_{k_{i'}} = k_i - k_{i'} < 0$ and so the subsequence cannot be decreasing.
- Similarly by the pigeon-hole principle, there are two terms $x_{k_j}, x_{k_{j'}}$ (wlog $k_j < k_{j'}$) such that $\lfloor k_j/m \rfloor = \lfloor k_{j'}/m \rfloor$. We have $x_{k_j} - x_{k_{j'}} = k_{j'} - k_j > 0$ and so the subsequence cannot be increasing.

Therefore all monotonic subsequences have length at most m.

Inductive step: Let $M = \sqrt{N} = m^{2^{d-1}}$. By the inductive assumption, let $(y_i)_{i=0}^{M-1}$ be a sequence of points in $\mathbb{R}^{d-1}$ with no monotonic subsequence of length $m+1$ such that the k^{th} coordinates are $\{0, 1, \ldots, M-1\}$ in some order for each $k = 1, \ldots, d-1$. Let y_i^k be the k^{th} coordinate of y_i. We construct a sequence $(x_i)_{i=0}^{N-1}$ where the k^{th} coordinate of x_i is denoted x_i^k. For all $a, b \in \{0, 1, 2, \ldots, M-1\}$ define

$$x_{aM+b}^k = y_a^k M + y_b^k,$$

for $k = 1, 2, \ldots, d-1$. Also define $x_{aM+b}^d = aM + (M - b - 1)$. Now partition the sequence into M equal parts (each part has size M) in the following way.

$$P_j = (x_{jM+i})_{i=0}^{M-1}$$

for each $j = 0, 1, 2, \ldots, M-1$. Now we show by contradiction that the sequence $(x_i)_{i=0}^{N-1}$ contains no monotonic subsequence of length $m+1$:

- Any subsequence which is increasing in its d^{th} coordinate must have at most one term in each P_j. However, this could not have length $m+1$ and be monotonic in the other coordinates by the inductive assumption.
- Any subsequence which is decreasing in its d^{th} coordinate must be contained within a single P_j. Such a subsequence could not have length $m+1$ and be monotonic in the other coordinates by the inductive assumption.

□

We now have all the tools required to prove Lemmas 5.2.5 and 5.2.6. Together, these lemmas imply Theorem 5.1.5.

Lemma 5.2.5. $\text{nbet}(n) \geq \log_2 \log_2 n + 1$.

Proof. Let $\Phi = \{\phi_0, \phi_1, \phi_2, \ldots, \phi_d\} \in \text{NBET}(n)$ be an order-system of size $\text{nbet}(n) = d + 1$. Without loss of generality, let ϕ_0 be the identity ordering (if not then we can let $\sigma = \phi_0^{-1}$ in Proposition 5.1.7 and consider $\Phi\sigma \in \text{NBET}(n)$). Now consider the sequence $(\mathbf{p}_i)_{i=1}^n$ of points in $\mathbb{R}^d$ defined by

$$\mathbf{p}_i = (\phi_1(i), \phi_2(i), \ldots, \phi_d(i)).$$

If this sequence contained a monotonic subsequence of length 3, say $(\mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c)$, then ϕ_j would between-satisfy the constraint (a, b, c) for all j, but this would contradict $\Phi \in \text{NBET}(n)$. Therefore the sequence $(\mathbf{p}_i)_{i=1}^n$ contains no monotonic subsequence of length 3. By Corollary 5.2.3 (with $m = 2$) this means $n \leq 2^{2^d}$. Hence

$$\text{nbet}(n) = d + 1 \geq \log_2 \log_2 n + 1.$$

□

Lemma 5.2.6. $\text{nbet}\left(2^{2^d}\right) = d + 1$.

Proof. Let $n = 2^{2^n}$. Because of Lemma 5.2.5, it suffices to construct an order-system $\Phi \in \text{NBET}(n)$ with $|\Phi| = d + 1$. Using Lemma 5.2.4 (with $m = 2$), let $(\mathbf{p}_i)_{i=1}^n$ be a sequence of n points in $\mathbb{R}^d$ containing no monotonic subsequence of length 3, such that for each coordinate k, the k^{th} coordinates are an ordering of $[n]$. Let ϕ_0 be the identity ordering and for each $k = 1, 2, \ldots, d$, let ϕ_k be the ordering given by the k^{th} coordinate of this sequence. It suffices to show

$$\Phi = \{\phi_i : i = 0, 1, 2, \ldots, d\} \in \text{NBET}(n).$$

Let $\mathbf{x} = (x_1, x_2, x_3)$ be an arbitrary constraint. If x_2 was not between x_1 and x_3 then ϕ_0 would nonbetween-satisfy $\mathbf{x}$. So we assume x_2 is between x_1 and x_3, and without loss of generality let $x_1 < x_2 < x_3$. Since $\big(\mathbf{p}(x_1), \mathbf{p}(x_2), \mathbf{p}(x_3)\big)$ is not monotonic in $\mathbb{R}^d$; there must be some coordinate k such that $\phi_k(x_2)$ is not between $\phi_k(x_1)$ and $\phi_k(x_3)$, and so ϕ_k nonbetween-satisfies $\mathbf{x}$. □

Proof of Theorem 5.1.5. If n is a double power of 2, then we are done Lemma 5.2.6. In general, let d be the positive integer such that $2^{2^{d-1}} < n \leq 2^{2^d}$ (i.e. $d = \lceil \log_2 \log_2 n \rceil$). So

$$\begin{aligned} d &< \log_2 \log_2 n + 1 \\ &\leq \text{nbet}(n) && \text{(Lemma 5.2.5)} \\ &\leq \text{nbet}\left(2^{2^d}\right) && \text{(nbet is non-decreasing)} \\ &= d + 1. && \text{(Lemma 5.2.6)} \end{aligned}$$

The value $\text{nbet}(n)$ is an integer strictly greater than d and at most $d+1$. Therefore

$$\text{nbet}(n) = d + 1 = \lceil \log_2 \log_2 n \rceil + 1.$$

□

5.3 Betweeness

In this section, we prove Theorem 5.1.6. Lemma 5.3.1, establishes the lower bound, and the upper bound is established in Lemma 5.3.2.

Lemma 5.3.1. $\text{bet}(n) \geq \log_2(n-1) + 1$.

Proof. Let $\text{bet}(n) = k+1$, let $\Phi = \{\phi_0, \phi_1, \phi_2, \ldots, \phi_k\} \in \text{BET}(n)$ and without loss of generality, let ϕ_0 be the identity ordering (if not then by Proposition 5.1.7 we can let $\sigma = \phi_0^{-1}$ and consider $\Phi\sigma \in \text{BET}(n)$). Note that ϕ_0 does not between-satisfy (x, n, y) for any x, y. Consider the function $\mathbf{f} : [n-1] \to \{0,1\}^k$ defined by $\mathbf{f}(x) = (I_1, I_2, \ldots, I_k)$ where I_i is the indicator function of $\phi_i(x) < \phi_i(n)$, for all $1 \leq i \leq k$.

$$I_i = \begin{cases} 1 & : \text{ if } \phi_i(x) < \phi_i(n) \\ 0 & : \text{ otherwise.} \end{cases}$$

For any distinct $x, y \in [n-1]$ we must have $\mathbf{f}(x) \neq \mathbf{f}(y)$, otherwise the constraint (x, n, y) would not be between-satisfied by Φ. Therefore $\mathbf{f}$ is injective and its domain is smaller than or equal to its codomain. So

$$n - 1 \leq \left|\{0,1\}^k\right| = 2^{\text{bet}(n)-1}.$$

□

Lemma 5.3.2. $\text{bet}(n) \leq 2\lceil \log_2(n) \rceil$.

Proof. Since $\mathrm{bet}(n)$ is a non-decreasing integer function, it suffices to consider only the case that $n = 2^k$ for some integer $k \geq 2$. We do this by explicitly constructing an order-system $\Phi \in \mathrm{BET}(2^k)$ with $|\Phi| = 2k$. First let $\{\psi_i\}_{i=1}^{k}$ be a set of k distinct orderings such that for each $x, y \in [n]$, there is at least one ψ_i such that $\psi_i(x) \leq \frac{n}{2} < \psi_i(y)$ or $\psi_i(y) \leq \frac{n}{2} < \psi_i(x)$.[1] For each ψ_i, let ϕ_i be an ordering defined by

$$\phi_i(x) = \begin{cases} \psi_i(x) - n/2 & : \text{ if } \psi_i(x) > \frac{n}{2} \\ \psi_i(x) + n/2 & : \text{ otherwise.} \end{cases}$$

Now consider an arbitrary constraint (x, α, y), and let i be such that $\psi_i(x) \leq \frac{n}{2} < \psi_i(y)$ or $\psi_i(y) \leq \frac{n}{2} < \psi_i(x)$. If ψ_i does not between-satisfy (x, α, y) then there are four cases:

- If $\psi(\alpha) < \psi_i(x) \leq \frac{n}{2} < \psi_i(y)$ then $\phi_i(y) \leq n/2 < \phi_i(\alpha) < \phi_i(x)$.
- If $\psi_i(x) \leq \frac{n}{2} < \psi_i(y) < \psi(\alpha)$ then $\phi_i(y) < \psi(\alpha) \leq n/2 < \phi_i(x)$.
- If $\psi(\alpha) < \psi_i(y) \leq \frac{n}{2} < \psi_i(x)$ then $\phi_i(x) \leq n/2 < \phi_i(\alpha) < \phi_i(y)$.
- If $\psi_i(y) \leq \frac{n}{2} < \psi_i(x) < \psi(\alpha)$ then $\phi_i(x) < \psi(\alpha) \leq n/2 < \phi_i(y)$.

In any case, if ψ_i does not between-satisfy (x, α, y) then ϕ_i does. Therefore

$$\Phi = \bigcup_{i=1}^{k} \{\phi_i, \psi_i\} \in \mathrm{BET}(n).$$

□

We now conclude this section, by computing the values of $\mathrm{bet}(n)$ for all $3 \leq n \leq 7$ (Figure 5.1). We don't consider the betweenness problem for smaller n, because there are no constraints over $[n]$ for $n < 3$. The exact value of $\mathrm{bet}(n)$ for $n \geq 8$ is left as an open problem. Lemma 5.3.3 deals with $n = 3$ and $n = 4$, and Lemma 5.3.4 deals with $n = 5$, $n = 6$ and $n = 7$.

Lemma 5.3.3. *For $n = 3, 4$ we have $\mathrm{bet}(n) = n$.*

Proof. We consider the two cases $n = 3$ and $n = 4$ separately.

- For $n = 3$, any ordering between-satisfies 2 constraints and the total number of constraints is 6. Hence $\mathrm{bet}(3) \geq 3$. To see that $\mathrm{bet}(3) \leq 3$, simply observe that

$$\{(1, 2, 3), (2, 3, 1), (3, 1, 2)\} \in \mathrm{BET}(3).$$

[1] There are many ways to find such set $\{\psi_i\}_{i=1}^{k}$. One way is to write the elements of $[n]$ in base two, and then let ψ_i be the i^{th} cyclic shift of the digits.

		$A(i)$				
		0	1	2	3	4
$B(i)$	0	✓	✓	✗	✗	✗
	1	✓	✓	✗	✗	✗
	2	✓	✓	✓	✗	✗
	3	✓	✓	✗	✗	✗
	4	✓	✗	✗	✗	✗

Figure 5.2: Possible values of $A(i)$ and $B(i)$ as defined in the proof of Lemma 5.3.4.

- For $n = 4$, any ordering between-satisfies 8 constraints and the total number of constraints is 24. Therefore if $\text{bet}(4) = 3$, then for any $\Phi \in \text{BET}(4)$ with $|\Phi| = 3$, we must have each constraint between-satisfied by exactly one ordering in Φ. Now consider the constraints:

$$(1, 4, 2)\,, \quad (2, 4, 3) \quad \text{and} \quad (3, 4, 1).$$

  Any ordering either between-satisfies either none of them or exactly two of these three constraints. So it is not possible for Φ to between-satisfy all three of these constraints unless one of them is between-satisfied by more than one ordering in Φ. This is a contradiction, so $\text{bet}(4) \geq 4$. To show $\text{bet}(4) = 4$, simply observe that

$$\{(1, 2, 3, 4), (2, 3, 4, 1), (3, 4, 1, 2), (4, 1, 2, 3)\} \in \text{BET}(4).$$

□

Lemma 5.3.4. *For $n = 5, 6, 7$ we have $\text{bet}(n) = 5$.*

Proof. Consider the order-system $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4, \psi_5\} \in \text{BET}(7)$ defined by

$$\begin{aligned}
\psi_1 &= (1, 6, 4, 3, 2, 7, 5) \\
\psi_2 &= (2, 6, 5, 4, 3, 7, 1) \\
\psi_3 &= (3, 6, 1, 5, 4, 7, 2) \\
\psi_4 &= (4, 6, 2, 1, 5, 7, 3) \\
\psi_5 &= (5, 6, 3, 2, 1, 7, 4)\,.
\end{aligned}$$

Therefore $\text{bet}(7) \leq 5$. Since bet is a non-decreasing function of n, it is now sufficient to show $\text{bet}(5) > 4$. To do this, we will assume there exists an order-system $\Phi =$

$\{\phi_1, \phi_2, \phi_3, \phi_4\} \in \text{BET}(5)$ and find a contradiction. Now for $i = 1, 2, 3, 4, 5$ let $A(i)$ and $B(i)$ denote the following cardinalities:

$$A(i) = |\{j \mid \phi_j(i) = 1 \text{ or } \phi_j(i) = 5\}| \quad \text{and} \quad B(i) = |\{j \mid \phi_j(i) = 3\}|.$$

i.e. $A(i)$ is the number of orderings in which i comes either first or last, and $B(i)$ is the number or orderings in which i is in the middle. Each of the orderings (ϕ_1, ϕ_2, ϕ_3 and ϕ_4) contribute a $+1$ to two different $A(i)$s and a $+1$ to a single $B(i)$. So

$$\sum_{i=1}^{5} A(i) = 8 \qquad \text{and} \qquad \sum_{i=1}^{5} B(i) = 4.$$

We now establish the table of possibilities displayed in Figure 5.2. If for some i, we had $A(i) \geq 2$ then at least two orderings (say ϕ_3 and ϕ_4) between satisfy none of the constraints of the form (x, i, y). The other orderings (ϕ_1 and ϕ_2) can only between-satisfy all constraints of the form (x, i, y) if i is in the middle of both ϕ_1 and ϕ_2. Therefore, if $A(i) \geq 2$ then $B(i) = 2$ and $A(i) = 2$. Now that we have proven which of the combinations of $A(i)$ and $B(i)$, shown in Figure 2.1, are possible, we can see that $B(i) \geq 2(A(i) - 1)$ for all i. However, this is a contradiction since

$$4 = \sum_{i=1}^{5} B(i) \geq \sum_{i=1}^{5} 2(A(i) - 1) = 6.$$

□

5.4 The extreme constraint problem

The extreme constraint problem is a generalisation of both betweeness and nonbetweenness. In this section we define ECP[Π] (Definition 5.4.1) and determine the growth-rate of $\text{ecp}_\Pi(n)$ for all proper subsets $\Pi \subset S_3$ (Theorem 5.4.4).

Definition 5.4.1. Let $\Pi \subset S_3$ be a non-empty set of permutations.

- An order-system Φ on $[n]$ is said to Π-solve the *extreme constraint problem* if for every constraint $\mathbf{x}$, there exists an order $\phi \in \Phi$ such that $\text{ord}(\phi, \mathbf{x}) \in \Pi$.
- Let ECP[Π] denote the set of all order-systems that Π-solve the extreme constraint problem.
- Let $\text{ecp}_\Pi(n)$ denote the minimum size of an order-system in ECP[Π].

We only consider non-empty subsets Π because $\text{ecp}_\emptyset(n)$ is not defined.

Proposition 5.4.2. *For $\Pi \subseteq S_3$ and any permutation $\sigma \in S_3$, we have* $\mathrm{ECP}[\Pi] = \mathrm{ECP}[\sigma\Pi]$ *where* $\sigma\Pi = \{\sigma \circ \pi \mid \pi \in \Pi\}$*, and so* $\mathrm{ecp}_\Pi(n) = \mathrm{ecp}_{\sigma\Pi}(n)$*.*

Proof. Since $\Pi = \sigma^{-1}(\sigma\Pi)$, it suffices to show that $\mathrm{ECP}[\Pi] \subseteq \mathrm{ECP}[\sigma\Pi]$. To show this, we will prove that for any $\Phi \in \mathrm{ECP}[\Pi]$, we must have $\Phi \in \mathrm{ECP}[\sigma\Pi]$. For any constraint $\mathbf{x} = (x_1, x_2, x_3)$ there exists a constraint $\mathbf{x}' = (x_{\sigma^{-1}(1)}, x_{\sigma^{-1}(2)}, x_{\sigma^{-1}(3)})$. By construction, for any ordering ϕ we have

$$\mathrm{ord}(\phi, \mathbf{x}) = \sigma \circ \mathrm{ord}(\phi, \mathbf{x}').$$

Therefore if $\Phi \in \mathrm{ECP}[\Pi]$ then for any constraint $\mathbf{x}$, there exists some $\phi \in \Phi$ such that $\mathrm{ord}(\phi, \mathbf{x}') \in \Pi$. For this ordering ϕ, we must have

$$\mathrm{ord}(\phi, \mathbf{x}) = \sigma \circ \mathrm{ord}(\phi, \mathbf{x}') \in \sigma\Pi.$$

□

Definition 5.4.3. For $a = 1, 2, 3$ let M_a denote the set of elements of S_3 with a in the middle. i.e.

$$M_1 = \{213, 312\}$$
$$M_2 = \{123, 321\}$$
$$M_3 = \{132, 231\}$$

By definition: BET = $\mathrm{ECP}[M_2]$ and NBET = $\mathrm{ECP}[M_1 \cup M_3]$. So the extreme constraint problem is a generalisation of betweenness and nonbetweenness. The asymptotics of $\mathrm{ecp}_{(M_1 \cup M_3)}(n)$ and $\mathrm{ecp}_{M_2}(n)$ are $\Theta(\log \log n)$ and $\Theta(\log n)$ respectively by Theorems 5.1.5 and 5.1.6. For $\Pi = S_3$, we trivially have $\mathrm{ecp}_\Pi(n) = 1$ for all n, and for $\Pi = \emptyset$, the function ecp_Π is not defined. For other sets Π, the asymptotics of $\mathrm{ecp}_\Pi(n)$ are presented in Theorem 5.4.4.

Theorem 5.4.4. *Let $\emptyset \neq \Pi \subset S_3$ and let c be the number of distinct M_a such that $\Pi \cap M_a = \emptyset$.*

- *If $c = 0$ then* $\mathrm{ecp}_\Pi(n) = 2$*.*
- *If $c = 1$ then* $\mathrm{ecp}_\Pi(n) = \Theta(\log \log n)$*.*
- *If $c = 2$ then* $\mathrm{ecp}_\Pi(n) = \Theta(\log n)$*.*

The three parts of Theorem 5.4.4 are proven separately in Lemmas 5.4.5. 5.4.6 and 5.4.7.

Lemma 5.4.5. *If $\Pi \neq S_3$ and Π intersects each of M_1, M_2 and M_3, then* $\mathrm{ecp}_\Pi(n) = 2$*.*

Proof. Since $S_3 \backslash \Pi \neq \emptyset$, for each ordering ϕ there will be some constraint $\mathbf{x}$ such that $\text{ord}(\phi, \mathbf{x}) \in S_3 \backslash \Pi$. Therefore $\{\phi\} \notin \text{ECP}[\Pi]$ for any ordering ϕ. So $\text{ecp}_\Pi(n) \neq 1$ and thus $\text{ecp}_\Pi(n) \geq 2$. However for any order, ϕ, suppose $\text{ord}(\phi, \mathbf{x}) = \sigma \notin \Pi$ for some constraint $\mathbf{x}$. There must be some $a \in \{1, 2, 3\}$ such that $\sigma \in M_a$. So let $M_a = \{\sigma, \tau\}$ and let ϕ^R be the reverse of ϕ. Since $\Pi \cap M_a \neq \emptyset$ we must have $\tau \in \Pi$ and (since τ is the reverse of σ)

$$\text{ord}(\phi^R, \mathbf{x}) = \tau \in \Pi.$$

Thus for any ordering ϕ and any constraint $\mathbf{x}$, either $\text{ord}(\phi, \mathbf{x}) \in \Pi$ or $\text{ord}(\phi^R, \mathbf{x}) \in \Pi$. For any ϕ, we have $\{\phi, \phi^R\} \in \text{ECP}[\Pi]$. Hence $\text{ecp}_\Pi(n) = 2$. □

Lemma 5.4.6. *If* Π *intersects exactly* 2 *of* $\{M_1, M_2, M_3\}$*, then for all* $n \geq 3$

$$\lceil \log_2 \log_2 n \rceil + 1 \leq \text{ecp}_\Pi(n) \leq 2 \lceil \log_2 \log_2 n \rceil + 2.$$

Proof. Let a be the index such that $\Pi \cap M_a = \emptyset$. If $a \neq 2$ then we can apply Proposition 5.4.2 (with either $\sigma = 231$ or $\sigma = 312$) so that $\sigma\Pi$ intersects M_1 and M_3 but not M_2. So without loss of generality assume $a = 2$; *i.e.* assume $\Pi \subseteq M_1 \cup M_3$. This implies $\text{ECP}[\Pi] \subseteq \text{ECP}[M_1 \cup M_3] = \text{NBET}$ and therefore

$$\text{ecp}_\Pi(n) \geq \text{nbet}(n) = \lceil \log_2 \log_2 n \rceil + 1.$$

To show the upper bound, consider some $\Phi \in \text{NBET}(n)$ with $|\Phi| = \text{nbet}(n) = \lceil \log_2 \log_2 n \rceil + 1$. Let $\mathbf{x} = (x_1, x_2, x_3)$ be an arbitrary constraint. There must be some $\phi \in \Phi$ such that x_2 is not between x_1 and x_3 in ϕ. Let $\text{ord}(\phi, \mathbf{x}) = \sigma \in M_1 \cup M_3$.

- If $\sigma \in \Pi$ then ϕ Π-satisfies $\mathbf{x}$.
- If for $b = 1$ or 3, we have $\sigma \in M_b \backslash \Pi$ then $M_b = \{\sigma, \tau\}$ where $\tau \in \Pi$ and τ is the reverse of σ. In this case $\text{ord}(\phi^R, \mathbf{x}) = \tau \in \Pi$, where ϕ^R is the reverse of ϕ.

So either $\text{ord}(\phi, \mathbf{x}) \in \Pi$ or $\text{ord}(\phi^R, \mathbf{x}) \in \Pi$. Hence $\Psi = \{\phi, \phi^R | \phi \in \Phi\} \in \text{ECP}[\Pi]$. Therefore

$$\text{ecp}_\Pi(n) \leq |\Psi| = 2 \lceil \log_2 \log_2 n \rceil + 2.$$

□

Lemma 5.4.7. *If* Π *is non-empty and intersects exactly* 1 *of* $\{M_1, M_2, M_3\}$ *then for all* $n \geq 3$

$$\log_2(n-1) + 1 \leq \text{ecp}_\Pi(n) \leq 4 \lceil \log_2 n \rceil .$$

Proof. Let a be the index such that $\Pi \subseteq M_a$. If $a \neq 2$ then we can apply Proposition 5.4.2 (with either $\sigma = 231$ or $\sigma = 312$) so that $\sigma\Pi \subseteq M_2$. So without loss of generality assume $a = a$; *i.e.* assume $\Pi \cap M_1$ and $\Pi \cap M_3$ are empty, and so $\Pi \subseteq M_2$. Therefore $\mathrm{ECP}[\Pi] \subseteq \mathrm{ECP}[M_2] = \mathrm{BET}$ and therefore (by Lemma 5.3.1)

$$\mathrm{ecp}_\Pi(n) \geq \mathrm{bet}(n) \geq \log_2(n-1) + 1.$$

To show the upper bound, let $\Phi \in \mathrm{BET}(n)$ be chosen arbitrarily with $|\Phi| = \mathrm{bet}(n) \leq 2\lceil \log_2 n \rceil$ (Lemma 5.3.2). Let $\mathbf{x} = (x_1, x_2, x_3)$ be an arbitrary constraint. Since Φ solves BET, there must be some $\phi \in \Phi$ such that x_2 is between x_1 and x_3 in ϕ. Without loss of generality $M_2 = \{\sigma, \tau\}$ and $\mathrm{ord}(\phi, \mathbf{x}) = \sigma$. If $\sigma \notin \Pi$ then $\Pi = \{\tau\}$ and $\mathrm{ord}(\phi^R, \mathbf{x}) = \tau \in \Pi$ where ϕ^R is the reverse of ϕ. Either way $\mathrm{ord}(\phi, \mathbf{x})$ or $\mathrm{ord}(\phi^R, \mathbf{x})$ is in Π. Hence

$$\Psi = \{\phi, \phi^R | \phi \in \Phi\} \in \mathrm{ECP}[\Pi]$$

and therefore $\mathrm{ecp}_\Pi(n) \leq |\Psi| \leq 4\lceil \log_2 n \rceil$. □

5.5 The extreme phylogenetic triplet problem

The extreme phylogenetic triplet problem (Definition 5.5.1) asks, what is the minimal size of a set of binary trees which is consistent with all phylogenetic triplets? In this section we show how this problem is related to the extreme constraint problem, and thus show that the answer is $\Theta(\log \log n)$.

Recall that a phylogenetic triplet, say $(a|bc)$, is consistent with a leaf-labelled binary tree if and only if the path from the root to the leaf labelled a, is disjoint from the path between the leaves labelled b and c.

Definition 5.5.1. Let $\mathcal{T}$ be a set of rooted phylogenetic trees with n non-root leaves labelled by $[n]$. *i.e.* $\mathcal{T} \subseteq \mathcal{B}(n+1)$. We say $\mathcal{T}$ solves the *extreme phylogenetic triplet* (EPT) problem if and only if, for every triplet $(a|bc)$ over $[n]$, at least one tree in $\mathcal{T}$ is consistent with $(a|bc)$. Let $\mathrm{EPT} = \mathrm{EPT}(n)$ denote the set of all $\mathcal{T} \subseteq \mathcal{B}(n+1)$ such that $\mathcal{T}$ solves the extreme phylogenetic triplet problem. Denote the minimal size of such a set by $\mathrm{ept}(n)$. *i.e.*

$$\mathrm{ept}(n) = \min\{|\mathcal{T}| : \mathcal{T} \in \mathrm{EPT}(n)\}.$$

Recall that for any ordering ϕ on $[n]$ (where n is a positive integer), $\mathrm{cat}(\phi)$ is the rooted caterpillar in $\mathcal{B}(n+1)$ with leaves in the order given by ϕ. For example, if $n = 4$ and $\phi = (3, 1, 2, 4)$ then the tree depicted in Figure 5.3, is $\mathrm{cat}(\phi) \in \mathcal{B}(5)$.

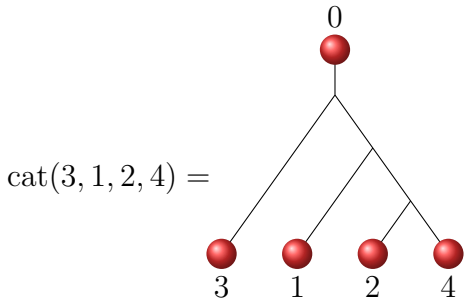


Figure 5.3: The binary tree $\text{cat}(3,1,2,4) \in \mathcal{B}(5)$.

Proposition 5.5.2. *Let* $\Pi = \{123, 132\} \subseteq S_3$*, let* $\mathbf{x} = (x_1, x_2, x_3)$ *be any constraint and let* ϕ *be any ordering. We have* $\text{ord}(\phi, \mathbf{x}) \in \Pi$ *if and only if* $\text{cat}(\phi)$ *is consistent with the triplet* $(x_1|x_2, x_3)$*.*

Proof. We have $\text{ord}(\phi, \mathbf{x}) \in \Pi$ if and only if x_1 comes before x_2 and x_3 in ϕ. Equivalently x_1 is a leaf in $\text{cat}(\phi)$ nearer to the root than x_2 and x_3. This is exactly what it means for a rooted caterpillar to be consistent with $(x_1|x_2, x_3)$. □

Proposition 5.5.3. *Let* $\Pi = \{123, 132\} \subseteq S_3$*. For any* $\Phi \in \text{ECP}[\Pi]$*, the set of trees* $\text{cat}(\Phi) = \{\text{cat}(\phi) : \phi \in \Phi\}$ *is in* EPT *and so, for all* $n \geq 3$*,*

$$\text{ept}(n) \leq \text{ecp}_{\Pi}(n).$$

Proposition 5.5.3 is a direct consequence of Proposition 5.5.2. These two Propositions highlights a way in which the extreme phylogenetic triplet problem is related to the extreme constraint problems. The value of $\text{ept}(n)$ would be exactly $\text{ecp}_{\{123,132\}}(n)$ if we only allowed phylogenetic trees that were caterpillars. This relationship between constraint satisfaction problems and phylogenetic triplet consistency problems is a known result [33].

Definition 5.5.4. For any rooted planar embedding E of a phylogenetic tree with leaf label set $[n]$, let $t(E)$ be the total ordering on $[n]$ given by listing the non-root leaf labels from left to right (with the root at the top). For example, the two embeddings, E_1 and E_2, depicted in Figure 5.4, are both embeddings of the same tree, but $t(E_1) \neq t(E_2)$.

There are many[2] different planar embeddings of a phylogenetic tree. As E varies over all planar embeddings of a fixed tree, $t(E)$ will result in a variety of different orders. For our purposes, it will not matter which planar embedding is used.

[2]For any rooted binary tree with n leaves, there are exactly 2^{n-1} different planar embeddings.

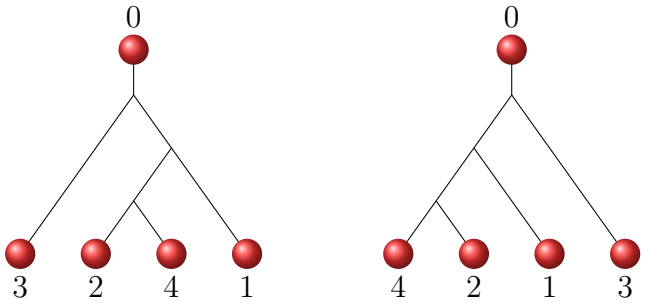


Figure 5.4: Two different embeddings, E_1 (left) and E_2 (right) of the same tree in $\mathcal{B}(5)$, however $t(E_1) = (3, 2, 4, 1)$ and $t(E_2) = (4, 2, 1, 3)$.

Lemma 5.5.5. $\text{ept}(n) \geq nbet(n)$.

Proof. Let E be any planar embedding of a phylogenetic tree T. If T is consistent with $(x_2|x_1, x_3)$ then x_1 and x_3 must be on the same side of x_2 in any planar embedding of T. So in $t(E)$, x_2 is not between x_1 and x_3 and so $t(E)$ would nonbetween-satisfy (x_1, x_2, x_3).
Now let $p = \text{ept}(n)$, let $\{T_1, T_2, \ldots T_p\} \in \text{EPT}(n)$ and for each i let E_i be a planar embedding of T_i. Consider the order-system $\Phi = \{t(E_1), t(E_2), \ldots, t(E_p)\}$. For any $\mathbf{x} = (x_1, x_2, x_3)$, there is some T_i which is consistent with $(x_2|x_1, x_3)$ and so there is some $\phi = t(E_i) \in \Phi$ which nonbetween-satisfies $\mathbf{x}$. Hence $\Phi \in \text{NBET}(n)$ and so $\text{nbet}(n) \leq p$. □

Theorem 5.5.6. $\text{ept}(n) = \Theta(\log \log n)$.

Proof. Let $\Pi = \{123, 132\}$. Proposition 5.5.3 and Lemma 5.5.5 provide the bounds:

$$\text{nbet}(n) \leq \text{ept}(n) \leq \text{ecp}_\Pi(n).$$

By Theorem 5.1.5 we know $\text{nbet}(n) = \lceil \log_2 \log_2 n \rceil + 1 = \Theta(\log \log n)$, and by Theorem 5.4.4 we know $\text{ecp}_\Pi(n) = \Theta(\log \log n)$ too. □

The explicit lower and upper bounds obtained this way are

$$\lceil \log_2 \log_2 n \rceil + 1 \leq \text{ept}(n) \leq 2 \lceil \log_2 \log_2 n \rceil + 2.$$

It is certainly true that $\text{ept}(n) \to \infty$ as $n \to \infty$, however even if we substitute an unrealistically large value of n, the result is surprisingly small. If n is the number of atoms on Earth, then $\text{ept}(n) \leq 18$.

Chapter 6

The guessing game on undirected cycles

For a given number of colours, s, the guessing number of a graph is the base s logarithm of the size of the largest family of colourings of the vertex set of the graph such that the colour of each vertex can be determined from the colours of the vertices in its neighbourhood. An upper bound for the guessing number of the n-vertex cycle graph C_n is $n/2$. It is known that the guessing number equals $n/2$ whenever n is even or s is a perfect square. We show that, for any given integer $s \geq 2$, if a is the largest factor of s less than or equal to $\sqrt{s}$, for sufficiently large odd n, the guessing number of C_n with s colours is $(n-1)/2 + \log_s(a)$. This answers a question posed by Christofides and Markström in 2011 [17]. We also present an explicit protocol which achieves this bound for every n.

In Section 6.7, these results are linked to index coding with side information. Specifically, we deduce that the information defect of C_n with s colours is $(n+1)/2 - \log_s(a)$ for sufficiently large odd n. The results in this section are a generalisation of the $s = 2$ case which was proven in [9].

6.1 Introduction

Computing the guessing number (Definition 6.1.3) of a graph G, can be equivalent to determining whether the multiple unicast coding problem [22] is solvable on a network related to G. The guessing number of a graph, G, is also studied for its relation to the information defect of G and index coding with side information [4, 26]. Exact guessing numbers are known only for a small number specific classes of graphs, such as perfect graphs, or small cases of non-perfect graphs [8, 13, 16, 52]. In particular, the guessing number of odd cycles, which is the focus of this paper, was not known,

except for small cases [9, 17]. Here we compute the guessing number of the cycle graph, C_n, by analysing optimal protocols for the "guessing game".

The guessing game was introduced by Riis in 2007 [41, 43]. It is a cooperative n-player information game played on a graph with n vertices with s colours. The guessing game on the complete graph K_n with $s = 2$ colours is played as follows. Each of the n players are given a hat that is red or blue uniformly and independently at random. Each player can see everyone else's hat, but not their own. The players collaboratively aim to maximise the probability that all players guess the colour of their hats correctly. Much of the popularity of this puzzle is owed to the striking difference between the success probability achieved by uncoordinated random guessing and an optimal protocol, which are $1/2^n$ and $1/2$ respectively.

The general guessing game considered here differs from many other variants of multiplayer information games (for example: the "hat guessing game" [12], "Ebert's game" [23] and the "hats-on-a-line game" [38]) in the following critical ways:

- The colours are assigned to each player independently and uniformly.
- Every player must guess (no passing or remaining silent).
- Each player does not necessarily see every other player's colours; two players can see each other if and only if they are joined by an edge in a given graph G.
- The players guess simultaneously so no communication is possible once the colours are assigned.
- The guessing game is won only if *all* the players guess correctly. An incorrect guess by any single player would mean that the whole team of n players collectively lose the guessing game (unlike [12], for example which seeks to optimise the number of players who guess correctly).

It is known that the greatest probability of winning the guessing game can be achieved by a deterministic protocol [13]. In this chapter, we restrict our attention to undirected graphs, but the problem is generalizes to directed graphs in an obvious way.

Definition 6.1.1. For any positive integer s, we let $\mathbb{Z}_s$, the group of all residues modulo s, denote the *colour set*. A *colouring* of G with s colours is an n-tuple $c = (c_1, c_2, \ldots, c_n)$ such that $c_i \in \mathbb{Z}_s$. The set of all colourings of G with s colours is denoted $\mathbb{Z}_s^n$. A *protocol* on G with s colours is any n-tuple $\mathcal{P} = (f_1, f_2, f_3, \ldots, f_n)$ where for each i, the [deterministic] function $f_i : \mathbb{Z}_s^n \to \mathbb{Z}_s$ is such that $f_i(c)$ is

dependent only on c_j for all j such that $v_iv_j \in E$, *i.e.* for any i and any two colourings $c = (c_1, c_2, \ldots, c_n)$ and $c' = (c'_1, c'_2, \ldots, c'_n)$, if $c'_j = c_j$ for all j such that $v_jv_i \in E$ then $f_i(c) = f_i(c')$. The *fixed set* of $\mathcal{P}$, $\mathrm{Fix}(\mathcal{P})$, is the set of all invariant colourings:

$$\mathrm{Fix}(\mathcal{P}) = \big\{c \in \mathbb{Z}_s^n \mid c_i = f_i(c)\ \forall i\big\}.$$

Definition 6.1.2. The *fixed number* of $\mathcal{P}$ is the size of its fixed set; $\mathrm{fix}(\mathcal{P}) = |\mathrm{Fix}(\mathcal{P})|$. A protocol $\mathcal{P}$ is called *non-trivial* if $\mathrm{Fix}(\mathcal{P}) \neq \emptyset$. A protocol is called *optimal* if it has maximal fixed number.

Definition 6.1.3. The *guessing number* of G with s colours is defined as

$$\mathrm{gn}(G, s) = \log_s \max_{\mathcal{P}} \left[\mathrm{fix}(\mathcal{P})\right].$$

We assign the n-tuple of colours $c \in \mathbb{Z}_s^n$ uniformly at random to the set of players, who are each identified with a vertex of G. The guesses of the players are given by $\mathcal{P}(c)$, so the players win if and only if $c = \mathcal{P}(c)$. Hence, the probability that an optimal protocol $\mathcal{P}$ wins is

$$\mathbb{P}\big(c = \mathcal{P}(c)\big) = \frac{\mathrm{fix}(\mathcal{P})}{|\mathbb{Z}_s^n|} = s^{\mathrm{gn}(G,s)-n}.$$

Christofides and Markström [17] showed that, for a perfect graph G and any s, $\mathrm{gn}(G, s) = n - \alpha$ where α is the size of the largest independent set in G. For example, the complete graph K_n is a perfect graph with $\alpha = 1$, so an optimal protocol on K_n, wins with probability $1/s$. The 3-cycle and the even-cycle C_{2k} (for any positive integer k) are both perfect graphs with $\alpha(C_3) = 1$ and $\alpha(C_{2k}) = k$ so

$$\mathrm{gn}(C_3, s) = 2 \quad \text{and} \quad \mathrm{gn}(C_{2k}, s) = k \quad \forall\, k. \tag{6.1.1}$$

For the rest of this chapter, we consider only the cycle graphs C_n for odd $n \geq 5$. In [17], it is shown that

$$\mathrm{gn}(C_5, 2) = \log_2 5,$$

and the analysis in [9] shows that

$$\mathrm{gn}(C_n, 2) = \frac{n-1}{2}, \ \text{ for odd } n \geq 7.$$

For general s, Christofides and Markström define protocols called "the clique strategy" and "the fractional-clique strategy" [17]. The fractional clique strategy is only defined

when the number of colours s is a perfect power, and it is shown to be optimal on the odd cycle whenever s is a perfect square, *i.e.*

$$\mathrm{gn}(C_n, m^2) = \frac{n}{2} \qquad \forall\, n, m. \tag{6.1.2}$$

In Definition 6.3.2, a protocol $\mathcal{P}_{fcp}$ is defined on odd cycles for any number of colours s. The protocol $\mathcal{P}_{fcp}$ is equivalent to the clique-strategy when s is prime, and to the fractional-clique-strategy when s is a perfect square. The protocol $\mathcal{P}_{fcp}$ is called the *fractional-clique-partition protocol* to emphasise that it is very closely related to Christofides and and Markström's fractional-clique strategy. Our main result in Theorem 6.5.6 states that, for any given s, this fractional-clique-partition protocol is optimal on any large enough odd cycle.

The rest of this chapter is organised as follows. In Section 6.2, we summarise a few of the known results on guessing numbers, and introduce the concepts of entropy and mutual information, which we will use heavily in our proofs. In Section 6.3, we define the fractional-clique-partition protocol, which is a refinement of the protocol introduced in [17] and we prove that for odd n, as the number of colours grows, this protocol achieves a $\mathrm{fix}(\mathcal{P})$ lies between $s^{n/2}$ and $s^{n/2}(1-\mathcal{O}(n/\sqrt{s}))$ (Theorem 6.3.5). In Section 6.4, we lay the technical groundwork which is needed for Section 6.5. Then, in Section 6.5, we focus on the case of large n compared to s, and we prove that the fractional-clique-partition protocol is in fact optimal on large enough odd cycles (Theorem 6.5.6). In Section 6.7, we link this to index coding with side information and compute the size of an optimal index code for C_n with s colours when n is odd and sufficiently large.

6.2 Backround material and notation

Many of our proofs will use the concept of the entropy of a random variable. Entropy is defined in Definition 6.2.2 and we list three crucial properties in Proposition 6.2.3. In this paper we take most logarithms base s, including inside the definitions of entropy. In the rest of this section, we present a few known results on the guessing number, define some useful random variables on the cycle graph and a notion of entropy, all of which will be used extensively in our proofs. When possible, we are consistent with the definitions and notations given in [8, 13, 16, 17, 42, 43]. We start with a small, useful result that shows, intuitively, that we are allowed to "forget" some colours.

Proposition 6.2.1. *Let G be a graph, let s and s' be positive integers with $s' \leq s$, and let $\mathcal{P}$ be any protocol on G with s colours. There exists a protocol $\mathcal{P}'$ on G with s' colours such that*

$$\left\{c \in \mathrm{Fix}(\mathcal{P}) \;\middle|\; 0 \leq c_i < s' \; \forall i\right\} \subseteq \mathrm{Fix}(\mathcal{P}').$$

Proof. If $\mathcal{P} = (f_1, f_2, \ldots f_n)$ then define $\mathcal{P}' = (f_1', f_2', \ldots, f_n')$ in the following way.

- If $0 \leq c_j < s'$ for all j such that $v_j v_i \in E$, and $0 \leq f_i(c) < s'$ then $f_i'(c) = f_i(c)$.
- If $s' \leq c_j < s$ for any j such that $v_j v_i \in E$, or $s' \leq f_i(c) < s$ then $f_i'(c) = 0$.

For any colouring $c \in \mathrm{Fix}(\mathcal{P})$, if $0 \leq c_i < s'$ for all i, then $\mathcal{P}'(c) = \mathcal{P}(c) = c$ so $c \in \mathrm{Fix}(\mathcal{P}')$. □

Definition 6.2.2. Let $A_1, \ldots, A_k$ be random variables which take values in a finite set $\mathcal{A}$. The *entropy* of $A_1, \ldots, A_k$ is denoted $H(A_1, \ldots, A_k)$ and is given by:

$$H(A_1, \ldots, A_k) = -\sum_{a_1, \ldots, a_k \in \mathcal{A}^k} \mathbb{P}(A_1 = a_1, \ldots, A_k = a_k) \log_s \mathbb{P}(A_1 = a_1, \ldots, A_k = a_k).$$

The *mutual information* of A_1 and A_2 is denoted $I(A_1; A_2)$ and is given by:

$$I(A_1; A_2) = H(A_1) + H(A_2) - H(A_1, A_2).$$

Let B be another random variable taking values in $\mathcal{A}$. The conditional mutual information of $I(A_1; A_2|B)$ is given by

$$I(A_1; A_2|B) = H(A_1, B) + H(A_2, B) - H(A_1, A_2, B) - H(B). \tag{6.2.1}$$

Proposition 6.2.3. *Let A_1, A_2 be random variables which take values in a finite sets $\mathcal{A}$.*

1. *$H(A_1) \leq \log |\mathcal{A}|$ with equality if and only if A_1 is uniformly distributed.*
2. *$I(A_1; A_2) \geq 0$ with equality if and only if A_1 and A_2 are independent.*
3. *$I(A_1; A_2|B) \geq 0$ with equality if and only if A_1 and A_2 are independent conditional on B.*

For a proof of the results in Proposition 6.2.3 we refer the reader to [18].

Definition 6.2.4. For a non-empty set S, we use the notation $A \in_u S$ to mean A is a random variable distributed uniformly over all elements in S.

Definition 6.2.5. The cycle graph, C_n, has n vertices $V = \{v_1, v_2, \ldots, v_n\}$. The edge set of C_n is

$$E = \{v_i v_{i+1} \mid i = 1, 2, 3, \ldots, n\}$$

(indices are always taken modulo n). In a slight abuse of notation, for any protocol $\mathcal{P} = (f_1, f_2, f_3, \ldots, f_n)$ on C_n with s colours, we say $f_i : \mathbb{Z}_s^2 \to \mathbb{Z}_s$ where

$$f_i(c) = f_i(c_{i-1}, c_{i+1}).$$

Recall that a protocol $\mathcal{P}$ is non-trivial if $\mathrm{Fix}(\mathcal{P}) \neq \emptyset$. For a given non-trivial protocol $\mathcal{P}$ on C_n, define $X = (X_1, X_2, \ldots, X_n)$ to be a colouring chosen uniformly at random from $\mathrm{Fix}(\mathcal{P})$. *i.e.*

$$X \in_u \mathrm{Fix}(\mathcal{P}).$$

Note that the random colouring $X = (X_1, X_2, \ldots, X_n)$ is only defined for non-trivial protocols $\mathcal{P}$. To simplify notation we will sometimes denote the entropy of a tuple of X_is by

$$h(i_1, i_2, i_3, \ldots) = H(X_{i_1}, X_{i_2}, X_{i_3}, \ldots).$$

Since X_i is determined by (X_{i-1}, X_{i+1}) we must have $H(X_{i-1}, X_i, X_{i+1})$ equal to $H(X_{i-1}, X_{i+1})$ so $h(i-1, i, i+1) = h(i-1, i+1)$. In general we can freely remove the argument i from $h(\ldots, i-1, i, i+1, \ldots)$ as long as we don't remove the arguments $i-1$ and $i+1$.

$$h(\ldots, i-1, i, i+1, \ldots) = h(\ldots, i-1, i+1, \ldots) \tag{6.2.2}$$

To simplify notation even further, for integers $j < k$, let H_j^k denote the quantity

$$H_j^k = h(j, j+1, j+2, \ldots, k-1) + h(j+1, j+2, j+3, \ldots, k).$$

Proposition 6.2.6. *For any three integers i, j, k such that $1 \leq i < j$ and $j + 1 < k \leq n$.*

$$H_i^k \leq H_i^j + H_{j+1}^k.$$

Proof. We add up the following inequalities:

$$\begin{aligned} h(i, i+1, \ldots, k-1) &= h(i, \ldots, j-1, j+1, \ldots, k-1) \\ &\leq h(i, \ldots, j-1) + h(j+1 \ldots, k-1), \\ \text{and} \quad h(i+1, i+2, \ldots, k) &= h(i+1, \ldots, j, j+2, \ldots, k) \\ &\leq h(i+1, \ldots, j) + h(j+2, \ldots, k). \end{aligned}$$

□

Lemma 6.2.7. *If $\mathcal{P}$ is a non-trivial protocol on C_n with $s \geq 2$ colours and $X \in_u \text{fix}(\mathcal{P})$, then, for all i,*

$$\log_s \text{fix}(\mathcal{P}) = H(X), \quad h(i) \leq 1.$$

Proof. The entropy of any random variable over a finite domain is maximised when the variable is uniformly distributed. Therefore, $h(i) = H(X_i) \leq H(U)$ where U is a random variable uniformly distributed over $\mathbb{Z}_s$. Hence,

$$h(i) \leq H(U) = -\sum \frac{1}{s} \log_s \frac{1}{s} = 1.$$

The variable X is uniformly distributed over $\text{Fix}(\mathcal{P})$. Therefore,

$$H(X) = -\sum \frac{1}{\text{fix}(\mathcal{P})} \log_s \frac{1}{\text{fix}(\mathcal{P})} = \log_s \text{fix}(\mathcal{P}).$$

□

Lemma 6.2.8. *If $\mathcal{P}$ is a non-trivial protocol on C_n with $s \geq 2$ colours and X is chosen uniformly from* $\text{fix}(\mathcal{P})$ *(*i.e. *$X \in_u \text{fix}(\mathcal{P})$), then*

$$H_j^k \leq \sum_{i=j}^{k} H(X_i),$$

for any $j \leq 1$ and $j + 3 \leq k \leq n$.

Proof. We prove this by induction on $(k - j)$. Recall that $h(i_1, i_2, i_3, \ldots)$ denotes $H(X_{i_1}, X_{i_2}, X_{i_3}, \ldots)$.

- **Base case:** $k = j + 3$. Since X_{j+1} is a function of X_j and X_{j+2}, and X_{j+2} is a function of X_{j+1} and X_{j+3} ($X_{j+1} = f_{j+1}(X_j, X_{j+2})$ and $X_{j+2} = f_{j+2}(X_{j+1}, X_{j+3})$), we have

$$h(j, j+1, j+2) = h(j, j+2) \leq h(j) + h(j+2)$$
$$\text{and} \qquad h(j+1, j+2, j+3) = h(j+1, j+3) \leq h(j+1) + h(j+3),$$

  respectively. Adding these together yields:

$$H_j^{j+3} = h(j, j+1, j+2) + h(j+1, j+2, j+3) \leq h(j) + h(j+1) + h(j+2) + h(j+3).$$

- **Inductive step:** $k \geq j + 4$. Since $X_{k-1} = f_{k-1}(X_{k-2}, X_k)$ we have

$$\begin{aligned} h(j+1, j+2, \ldots, k) &= h(j+1, j+2, \ldots, k-2, k) \\ &\leq h(j+1, j+2, \ldots, k-2) + h(k). \end{aligned}$$

By Proposition 6.2.3, $I(X_j; X_{k-1}|X_{j+1}, X_{j+2}, \ldots, X_{k-2}) \geq 0$. Adding these together yields

$$H_j^k \leq H_j^{k-1} + h(k).$$

This completes the proof.

□

Lemma 6.2.9. *Let $\mathcal{P}$ be a non-trivial protocol on C_n with $s \geq 2$ colours and let $X \in_u \text{fix}(\mathcal{P})$. Suppose $1 = d(1), d(2), d(3), \ldots, d(k) = n$ is a sequence of positive integers with $k \geq 2$. If $d(i+1) \geq d(i) + 2$ for all i, then*

$$2 \log_s \text{fix}(\mathcal{P}) \leq H_{d(1)}^{d(2)} + H_{d(2)+1}^{d(3)} + \cdots + H_{d(k-1)+1}^{d(k)}.$$

Proof. We proceed by induction on k.

- **Base case:** $k = 2$. Since $X_1 = f_1(X_n, X_2)$ and $X_n = f_n(X_{n-1}, X_1)$, we have

  $$H(X) = h(2, 3, 4, \ldots, n) \qquad \text{and} \qquad H(X) = h(1, 2, 3, \ldots, n-1),$$

  respectively. Adding these together gives $H_1^n = 2H(X) = 2 \log_s \text{fix}(\mathcal{P})$.

- **Inductive step.** By Proposition 6.2.6, for any $d(k-1) + 2 \leq d(k) \leq n - 2$, we have

  $$H_{d(k-1)+1}^n \leq H_{d(k-1)+1}^{d(k)} + H_{d(k)+1}^n.$$

□

6.3 The fractional-clique-partition protocol

In this section, we define the fractional-clique-partition protocol, $\mathcal{P}_{fcp}$, on odd cycles C_n with $s \geq 2$ colours. Theorem 6.3.4 appears in [17] and serves as a good upper bound for any $n \geq 4$ and all numbers of colours.

Definition 6.3.1. It is easy to see that for any factorization $ab = s$, there exists a bijection between $\mathbb{Z}_s$ and $\mathbb{Z}_a \times \mathbb{Z}_b$. Let $\phi(z) \times \psi(z)$ be such a bijection. For ease of notation, a and b are assumed to be given in context. Let π be the inverse of this bijection, so that $\pi(\phi(z), \psi(z)) = z$ for all $z \in \mathbb{Z}_s$.

Definition 6.3.2. Let $n \geq 3$ be an odd integer, let s be a positive integer, let a be the greatest factor of s less than or equal to $\sqrt{s}$ and let $b = s/a$. For any colouring $c = (c_1, c_2, \ldots, c_n) \in \mathbb{Z}_s^n$, let $\phi(c_i)$ and $\psi(c_i)$ be referred to as the first and second coordinates respectively of vertex v_i. The *fractional-clique-partition protocol* is the protocol $\mathcal{P}_{fcp} = (f_1, f_2, \ldots, f_n)$ on C_n defined by:

$$\begin{aligned} f_i(c_{i-1}, c_{i+1}) &= \pi\big(\phi(c_{i-1}), \psi(c_{i+1})\big) && \text{for } i = 2, 4, 6, \ldots, n-1 \\ f_i(c_{i-1}, c_{i+1}) &= \pi\big(\phi(c_{i+1}), \psi(c_{i-1})\big) && \text{for } i = 3, 5, 7, \ldots, n-2 \\ f_1(c_n, c_2) &= \pi\big(\phi(c_2), \phi(c_n)\big) && \text{and} \\ f_n(c_{n-1}, c_1) &= \pi\big(\psi(c_1) (\text{mod } a), \psi(c_{n-1})\big). \end{aligned}$$

Informally, vertices v_{2k-1} and v_{2k} are copying each others first coordinate and vertices v_{2k} and v_{2k+1} are copying each others second coordinate (for $k = 1, 2, 3, \ldots, (n-1)/2$). Additionally, the second coordinate of vertex v_1 and the first coordinate of vertex v_n copy each other as much as possible - whenever the second coordinate of vertex v_1 is less than a. An example of $\mathcal{P}_{fcp}$ on C_7 is illustrated in Figure 6.1.

Proposition 6.3.3. *For a given integer $s \geq 2$ and odd integer $n \geq 3$, if a is the greatest factor of s less than or equal to $\sqrt{s}$, then we have* $\text{fix}(\mathcal{P}_{fcp}) = as^{(n-1)/2}$.

Proof. Let $n = 2k + 1$. We count the number of colourings of C_n for which the protocol $\mathcal{P}_{fcp}$ guesses correctly. For any colouring $c \in \text{Fix}(\mathcal{P}_{fcp})$, there are k pairs of vertices copying each other's first coordinates and there are a different choices for ϕ for each pair. Similarly, for each of the k pairs of vertices copying each other's second coordinates, there are b different choices for ψ. This yields $a^k b^k$ possibilities. Additionally, the first coordinate of vertex v_n must equal the second coordinate of vertex v_1, for which there are a possible colours. Multiplying these together yields

$$\text{fix}(\mathcal{P}_{fcp}) = a^{k+1} b^k = as^{(n-1)/2}.$$

□

Theorem 6.3.4. *[17] For any integer $n \geq 4$, we have* $\text{gn}(C_n, s) \leq \frac{n}{2}$, *with equality only if for any optimal protocol, $\mathcal{P}$ the following is satisfied. If $X \in_u \text{Fix}(\mathcal{P})$ then $H(X_i) = 1$ for all i.*

Proof. Let $\mathcal{P}$ be an optimal protocol on C_n with s colours. By Lemmas 6.2.7, 6.2.8 and 6.2.9, we have

$$\text{gn}(C_n, s) = \log_s \text{fix}(\mathcal{P}) = H(X) = \tfrac{1}{2} H_1^n \leq \tfrac{1}{2} \sum_{i=1}^{n} h(i) \leq \frac{n}{2}.$$

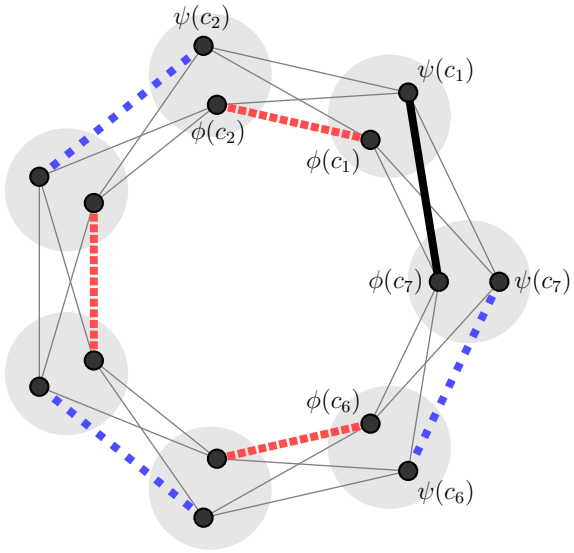


Figure 6.1: The protocol $\mathcal{P}_{fcp}$ on C_7 with $s = ab$ colours, where $a < b$. Each vertex v_i is subdivided into two nodes representing the first and second components ($\phi(c_i)$ and $\psi(c_i)$, respectively). The red edges (▬) represent pairs of first-components that are copying each other. The blue edges (▬) represent pairs of second-components that are copying each other. The black edge (▬) joins a first-component ($\phi(c_n)$) and a second-component ($\psi(c_1)$) which are copying each other as much as possible. For a colouring $c \in \mathrm{Fix}(\mathcal{P}_{fcp})$ on C_7, there are a different choices for each red edge, b different choices for each blue edge and a different choices for the black edge. Therefore, $\mathrm{fix}(\mathcal{P}_{fcp}) = a^4b^3 = as^3$ for $n = 7$.

If $\mathrm{gn}(C_n, s) = n/2$, then we must have equality throughout, which means that $h(i) = 1$ for all i. □

Theorem 6.3.4 appears in [17]. This same paper also shows that the limit of $\mathrm{gn}(C_n, s) \to n/2$ as $s \to \infty$. We give a bound on the rate convergence to this limit in Theorem 6.3.5.

Theorem 6.3.5. *If $n \geq 4$ and $s = m^2 - t$ for integers m and $t \geq 0$ then there exists a protocol $\mathcal{P}$ on C_n with s colours such that*

$$\mathrm{fix}(\mathcal{P}) \geq s^{n/2}\left(1 - \frac{tn}{s}\right).$$

Proof. Consider the protocol $\mathcal{P}' = \mathcal{P}_{fcp}$ on C_n with $s' = m^2$ colours and let $X' \in_u \mathrm{Fix}(\mathcal{P}')$. By Theorem 6.3.4, we must have $H(X'_i) = 1$ and therefore X'_i is uniformly distributed over $\mathbb{Z}_{s'}$ for all i. By the union bound,

$$\mathbb{P}\left(X'_i < s \;\forall\, i\right) \geq 1 - \sum_{i=1}^{n} \mathbb{P}(X'_i \geq s) = 1 - \sum_{i=1}^{n} \frac{t}{m^2} = 1 - \frac{tn}{m^2}.$$

Now, let $\mathcal{P}$ be a protocol on C_n with s colours such that $c \in \mathrm{Fix}(\mathcal{P})$ for all colourings $c \in \mathrm{Fix}(\mathcal{P}')$ such that $c_i < s$ for all i (such a protocol must exist by Proposition 6.2.1). For this protocol,

$$\begin{aligned}
\mathrm{fix}(\mathcal{P}) &\geq \mathrm{fix}(\mathcal{P}')\,\mathbb{P}\left(X'_i < s \;\forall\, i\right) \\
&\geq \mathrm{fix}(\mathcal{P}')\left(1 - \frac{tn}{m^2}\right) \\
&= (s+t)^{n/2}\left(1 - tn(s+t)^{-1}\right) \\
&\geq s^{n/2}\left(1 - \frac{tn}{s}\right).
\end{aligned}$$

□

Corollary 6.3.6. *If $n \geq 4$ then $\mathrm{gn}(C_n, s) = \frac{n}{2} - \mathcal{O}\left(\frac{n}{\sqrt{s}\log_e s}\right)$ as $s \to \infty$.*

Proof. For even n we have $\mathrm{gn}(C_n, s) = \frac{n}{2}$. For odd n, let m be the smallest positive integer such that $m^2 \geq s$. This gives $t = m^2 - s = \mathcal{O}(\sqrt{s})$. If $\mathcal{P}$ is the protocol constructed in Theorem 6.3.5, then

$$\mathrm{gn}(C_n, s) \geq \log_s \mathrm{fix}(\mathcal{P}) \geq \frac{n}{2} + \log_s\left(1 - \frac{tn}{s}\right) = \frac{n}{2} - \mathcal{O}\left(\frac{n}{\sqrt{s}\log_e s}\right).$$

□

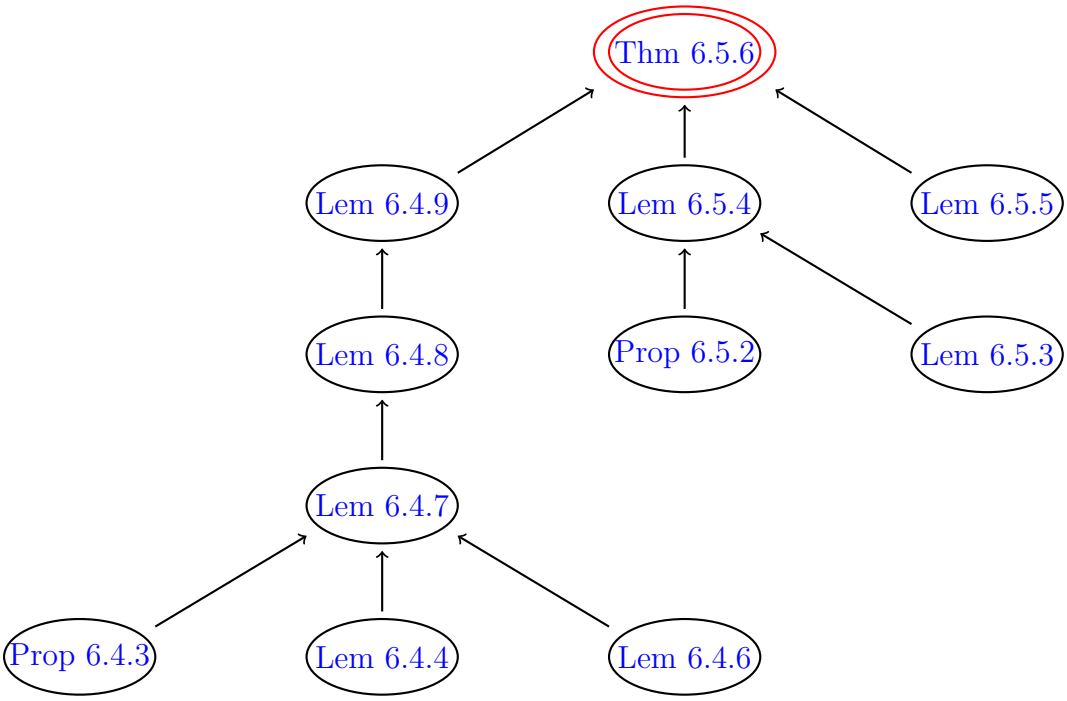


Figure 6.2: The structure of Sections 6.4 and 6.5. An arrow $A \to B$ indicates that A is used in the proof of B.

6.4 Entropy results

The bounds in Theorem 6.3.5 are only useful when n is small relative to s. In contrast, the purpose of the results in this section is to establish Lemma 6.4.9, which in turn will be used to prove Theorem 6.5.6 which only applies when n is large relative to s. To help orientate the reader through this section (and the next), Figure 6.2 shows which results are used to prove other results.

Definition 6.4.1. For any $z \in \mathbb{Z}_s$ and for any function $f : \mathbb{Z}_s^2 \to \mathbb{Z}_s$ let $f^{-1}(z) = \{(x, y) \mid f(x, y) = z\}$. The function f is called *flat* if and only if $|f^{-1}(z)| = s$ for all z. Let $U = (U_1, U_2) \in_u \mathbb{Z}_s^2$. A *semi-perfect* function, f, is any flat function such that the U_1 and U_2 are conditionally independent given $f(U)$ (Definition 6.2.2), *i.e.*

$$I(U_1; U_2 \mid f(U)) = 0.$$

Definition 6.4.2. For any positive integer k and any $\epsilon > 0$, a random variable Y is called (k, ϵ)-uniform if Y takes values in a finite set $\mathcal{Y}$ with $|\mathcal{Y}| = k$ and, for any $y \in \mathcal{Y}$,

$$\left| \mathbb{P}(Y = y) - \frac{1}{k} \right| \leq \epsilon.$$

Proposition 6.4.3. *For any integer $k \geq 2$, any integer $s \geq 2$ and any $\epsilon > 0$, there exists $\delta > 0$ such that, for any random variable Y which takes k distinct values, if $H(Y)$ is the entropy of Y (base s), then*

$$H(Y) \geq \log_s k - \delta \qquad \Longrightarrow \qquad Y \text{ is } (k, \epsilon)\text{-uniform.}$$

Proof. For each k, it suffices to show this for all small enough ϵ. Assume $7k\epsilon < 1$. We prove the contrapositive:

- Suppose that $\mathbb{P}(Y = y) \geq \frac{1}{k} + \epsilon$ for at least one value y. Entropy is greatest when Y is as uniformly distributed as possible. Therefore,

$$\begin{aligned} H(Y) &= -\sum_i \mathbb{P}(Y = i) \log_s \mathbb{P}(Y = i) \\ &\leq -\left(\tfrac{1}{k} + \epsilon\right) \log_s \left(\tfrac{1}{k} + \epsilon\right) - (k-1)\left(\tfrac{1}{k} - \tfrac{\epsilon}{k-1}\right) \log_s \left(\tfrac{1}{k} - \tfrac{\epsilon}{k-1}\right) \\ &= \log_s k - \left(\tfrac{1}{k} + \epsilon\right) \log_s (1 + k\epsilon) - \left(\tfrac{k-1}{k} - \epsilon\right) \log_s \left(1 - \tfrac{k\epsilon}{k-1}\right). \end{aligned}$$

  Since $0 < k\epsilon < \frac{1}{7}$, we can use the identity, $-\log_s(1-\gamma) \leq (\gamma + \frac{5}{9}\gamma^2) \log_e s$ (valid for $|\gamma| \leq 1/7$), to simplify this expression.

$$\begin{aligned} H(Y) &\leq \log_s k - \frac{k\epsilon^2}{9}\left(4 - 5k\epsilon + \tfrac{4}{k-1} + \tfrac{5k\epsilon}{(k-1)^2}\right) \log_e s \\ &\leq \log_s k - \frac{k\epsilon^2}{3} \log_e s. \end{aligned}$$

- Now suppose $\mathbb{P}(Y = y) \leq \frac{1}{k} - \epsilon$ for at least one value y. The entropy would be greatest when Y is as uniformly distributed as possible. Therefore,

$$\begin{aligned} H(Y) &= -\sum_i \mathbb{P}(Y = i) \log_s \mathbb{P}(Y = i) \\ &\leq -\left(\tfrac{1}{k} - \epsilon\right) \log_s \left(\tfrac{1}{k} - \epsilon\right) - (k-1)\left(\tfrac{1}{k} + \tfrac{\epsilon}{k-1}\right) \log_s \left(\tfrac{1}{k} - \tfrac{\epsilon}{k-1}\right) \\ &= \log_s k - \left(\tfrac{1}{k} - \epsilon\right) \log_s (1 - k\epsilon) - \left(\tfrac{k-1}{k} + \epsilon\right) \log_s \left(1 + \tfrac{k\epsilon}{k-1}\right). \end{aligned}$$

  We can use the identity, $-\log_s(1-\gamma) \leq (\gamma + \frac{5}{9}\gamma^2) \log_e s$, again to simplify this expression.

$$\begin{aligned} H(Y) &\leq \log_s k - \tfrac{k\epsilon^2}{9}\left(4 - 5k\epsilon + \tfrac{4}{k-1} - \tfrac{5k\epsilon}{(k-1)^2}\right) \log_e s \\ &\leq \log_s k - \frac{k\epsilon^2}{3} \log_e s. \end{aligned}$$

In either case, $H(Y) < \log_s k - \delta$ for any $\delta < \frac{k\epsilon^2}{3} \log_e s$. □

Lemma 6.4.4. *For any integer $s \geq 2$, there exists positive constant $\epsilon = \epsilon(s)$ that satisfies the following property. For any non semi-perfect function $f : \mathbb{Z}_s^2 \to \mathbb{Z}_s$ and for any three (s, ϵ)-uniform random variables Y_1, Y_2, Y_3 over $\mathbb{Z}_s$ satisfying $Y_2 = f(Y_1, Y_3)$, if (Y_1, Y_3) is (s^2, ϵ)-uniform, then*

$$I(Y_1; Y_3|Y_2) \geq \tfrac{1}{2} \min \big\{ I(U_1; U_2|f(U)) \;\big|\; f \text{ is a flat but not semi-perfect} \big\} = \delta_1,$$

where $U = (U_1, U_2) \in_u \mathbb{Z}_s^2$.

Proof. The value,

$$\delta_1 = \delta_1(s) = \tfrac{1}{2} \min \big\{ I(U_1; U_2|f(U)) \;\big|\; f \text{ is a flat but not semi-perfect} \big\},$$

is well-defined for any $s \geq 2$, because there are only a finite number of functions $f : \mathbb{Z}_s^2 \to \mathbb{Z}_s$, and at least one of them is flat and not semi-perfect hence we can take the minimum of these. For example, the function $f(x, y) = x + y \pmod{s}$ is flat but not semi-perfect. First, let $\epsilon < \frac{1}{s^2(s+2)}$, so that

$$\frac{1}{s^2} - (s-1)\epsilon > \frac{1}{s^2} - (s+1)\epsilon > \epsilon.$$

We show that f is flat by contradiction. Since (Y_1, Y_3) is (s^2, ϵ)-uniform:

- If $|f^{-1}(z)| \geq s + 1$ then

  $$\mathbb{P}(Y_2 = z) = \mathbb{P}((Y_1, Y_3) \in f^{-1}(z)) \geq (s+1)\left(\tfrac{1}{s^2} - \epsilon\right) = \tfrac{1}{s} + \left(\tfrac{1}{s^2} - (s+1)\epsilon\right) > \tfrac{1}{s} + \epsilon.$$

- If $|f^{-1}(z)| \leq s - 1$ then

  $$\mathbb{P}(Y_2 = z) = \mathbb{P}((Y_1, Y_3) \in f^{-1}(z)) \leq (s-1)\left(\tfrac{1}{s^2} + \epsilon\right) = \tfrac{1}{s} - \left(\tfrac{1}{s^2} - (s-1)\epsilon\right) < \tfrac{1}{s} - \epsilon.$$

Both cases contradict the assumption that Y_2 is (s, ϵ)-uniform. Therefore f is a flat function and so

$$I(U_1; U_2|f(U)) \geq 2\delta_1.$$

Moreover, since (Y_1, Y_3) is (s^2, ϵ)-uniform, then U and (Y_1, Y_3) differ in distribution by less than ϵ. Since mutual information is continuous, we can choose ϵ small enough so that

$$\big| I(U_1; U_2|f(U)) - I(Y_1; Y_3|Y_2) \big| \leq \delta_1.$$

Then, by the triangle inequality, $I(Y_1; Y_3|Y_2) \geq \delta_1$. □

Definition 6.4.5. From now on, for any integer $s \geq 2$, let $\epsilon = \epsilon(s) > 0$ be chosen small enough so that $\epsilon \leq \frac{1}{s^2(2s+1)}$ and ϵ satisfies Lemma 6.4.4. Then let $\delta_2 = \delta_2(s) > 0$ be chosen small enough to satisfy Proposition 6.4.3 for both $k = s$ and $k = s^2$ for this value ϵ. Then, with δ_1 as defined in Lemma 6.4.4, let $\delta = \min(\delta_1, \delta_2)$.

Lemma 6.4.6. *Let $n \geq 5$ be an integer and let $\mathcal{P}$ be any non-trivial protocol on C_n with $s \geq 2$ colours. The random variables X_1, X_2, X_3, X_4, X_5 (Definition 6.2.5) satisfy:*

$$H_1^5 \leq 3 + h(2,4) - I(X_2; X_4 | X_3).$$

Proof. By Lemma 6.2.7, it suffices to show $H_1^5 \leq h(1) + h(3) + h(5) + h(2,4) - I(X_2; X_4|X_3)$. By Shannon's Inequality (Proposition 6.2.3) we have:

$$h(2,3,4) + h(3) = h(2,3) + h(3,4) - I(X_2; X_4|X_3), \tag{6.4.1}$$

$$h(1,2,3,4) + h(2,3) \leq h(1,2,3) + h(2,3,4), \tag{6.4.2}$$

$$\text{and} \quad h(2,3,4,5) + h(3,4) \leq h(2,3,4) + h(3,4,5). \tag{6.4.3}$$

Also, since $X_i = f_i(X_{i-1}, X_{i+1})$ for $i = 2, 3, 4$ respectively we have:

$$h(1,2,3) = h(1,3) \leq h(1) + h(3), \tag{6.4.4}$$

$$h(2,3,4) = h(2,4), \tag{6.4.5}$$

$$\text{and} \quad h(3,4,5) = h(3,5) \leq h(3) + h(5). \tag{6.4.6}$$

The required result is the sum of equations (6.4.1), (6.4.2), (6.4.3), (6.4.4), (6.4.5) and (6.4.6). □

Lemma 6.4.7. *Let $n \geq 5$ be an integer and let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ be a non-trivial protocal on C_n with $s \geq 2$ colours and let $X \in_u \text{fix}(\mathcal{P})$. For any j, if f_{j+2} is not semi-perfect or (X_{j+1}, X_{j+3}) is not (s^2, ϵ)-uniform then $H_j^{j+4} \leq 5 - \delta$, for δ as in Definition 6.4.5.*

Proof. Without loss of generality let $j = 1$. There are 3 cases.

- If, for any $i \in \{1, 2, 3, 4, 5\}$, the variable X_i is not (s, ϵ)-uniform, then $h(i) \leq 1 - \delta_2$ (Proposition 6.4.3). In this case, by Lemma 6.2.8,

$$H_1^5 \leq \sum_{i=1}^{5} h(i) \leq 5 - \delta_2.$$

- If (X_2, X_4) is not (s^2, ϵ)-uniform, then $h(2,4) \leq 2 - \delta_2$ (Proposition 6.4.3). Therefore, by Lemma 6.4.6, we have

$$H_1^5 \leq 3 + h(2,4) - I(X_2; X_4|X_3) \leq 5 - \delta_2.$$

- Otherwise, X_2, X_3, X_4 are each (s, ϵ)-uniform and (X_2, X_4) is (s^2, ϵ)-uniform and f_3 is not semi-perfect. In this case, by Lemma 6.4.4, we have $I(X_2; X_4|X_3) \geq \delta_1$. By Lemma 6.4.6, we have

$$H_1^5 \leq 3 + h(2,4) - I(X_2; X_4|X_3) \leq 5 - \delta_1.$$

In all cases, we have $H_1^5 \leq 5 - \delta$ because $\delta = \min\{\delta_1, \delta_2\}$. □

Lemma 6.4.8. *Let $n \geq 7$ be an integer and let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ a non-trivial protocol on C_n with $s \geq 2$ colours and let $X \in_u \text{fix}(\mathcal{P})$. For any j, if any of f_{j+2}, f_{j+3} or f_{j+4} are not semi-perfect, or any of (X_{j+1}, X_{j+3}), (X_{j+2}, X_{j+4}) or (X_{j+3}, X_{j+5}) are not (s^2, ϵ)-uniform, then $H_j^{j+6} \leq 7 - \delta$.*

Proof. Without loss of generality let $j = 1$. We treat each case individually, and use Lemma 6.4.7.

- If f_3 is not semi-perfect or (X_2, X_4) is not (s^2, ϵ)-uniform then

$$\begin{aligned} H_1^7 &= h(1,2,3,4,5,6) + h(2,3,4,5,6,7) \\ &= h(1,2,3,4,6) + h(2,3,4,5,7) \\ &\leq H_1^5 + h(6) + h(7) \\ &\leq (5 - \delta) + 1 + 1. \end{aligned}$$

- If f_4 is not semi-perfect or (X_3, X_5) is not (s^2, ϵ)-uniform then

$$\begin{aligned} H_1^7 &= h(1,2,3,4,5,6) + h(2,3,4,5,6,7) \\ &= h(1,3,4,5,6) + h(2,3,4,5,7) \\ &\leq h(1) + H_2^6 + h(7) \\ &\leq 1 + (5 - \delta) + 1. \end{aligned}$$

- If f_5 is not semi-perfect or (X_4, X_6) is not (s^2, ϵ)-uniform then

$$\begin{aligned} H_1^7 &= h(1,2,3,4,5,6) + h(2,3,4,5,6,7) \\ &= h(1,3,4,5,6) + h(2,4,5,6,7) \\ &\leq h(1) + h(2) + H_3^7 \\ &\leq 1 + 1 + (5 - \delta). \end{aligned}$$

□

Lemma 6.4.9. *Let $n \geq 7(\delta^{-1}+2)$. Suppose $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ is a non-trivial protocol on C_n with $s \geq 2$ colours and let $X \in_u \mathrm{fix}(\mathcal{P})$ such that, for each j, either*

- *at least one of f_{j-1}, f_j, f_{j+1} is not semi-perfect, or*
- *at least one of (X_{j-2}, X_j), (X_{j-1}, X_{j+1}), (X_j, X_{j+2}) is not (s^2, ϵ)-uniform,*

then $\mathrm{fix}(\mathcal{P}) < s^{(n-1)/2}$.

Proof. Let m be an odd integer such that $m > \delta^{-1}$ and $7m \leq n$. By Lemma 6.2.9 and Lemma 6.4.8, we have

$$\begin{aligned} 2H(X) &\leq \sum_{j=0}^{m-1} H_{7j+1}^{7j+7} + \sum_{i=7m}^{n-1} h(i) \\ &\leq m(7-\delta) + (n-7m) \\ &= n - m\delta. \end{aligned}$$

Since $m > \delta^{-1}$, this means that $H(X) < \frac{n-1}{2}$. Therefore $\mathrm{fix}(\mathcal{P}) < s^{(n-1)/2}$ by Lemma 6.2.7. □

6.5 Guessing numbers of large odd cycles

In this section, we prove our main result in Theorem 6.5.6, which states that, for any given s, this fractional-clique-partition protocol is optimal on any large enough odd cycle.

Definition 6.5.1. For any function $f : \mathbb{Z}_s^2 \to \mathbb{Z}$, let $L(f,z)$ and $R(f,z)$ denote the subsets

$$\begin{aligned} L(f,z) &= \{x \mid f(x,y) = z \text{ for some } y\} \\ \text{and} \quad R(f,z) &= \{y \mid f(x,y) = z \text{ for some } x\}. \end{aligned}$$

The function f is called a *perfect* function if it is semi-perfect and the cardinalities $|L(f,z)|$ and $|R(f,z)|$ do not depend on z, *i.e.* if $|L(f,z)| = |L(f,z')|$ and $|R(f,z)| = |R(z')|$ for all $z, z' \in \mathbb{Z}_s$.

Proposition 6.5.2. *If f is a semi-perfect function then for all $z \in \mathbb{Z}_s$ then*

$$f^{-1}(z) = L(f,z) \times R(f,z).$$

Moreover $|L(f,z)||R(f,z)| = s$.

Proof. Let $U = (U_1, U_2) \in_u \mathbb{Z}_s^2$ and for a given z, let $L = L(f, z)$ and $R = R(f, z)$. Since f is semi-perfect, we have $I(U_1, U_2 \mid f(U)) = 0$. Therefore, U_1 and U_2 are conditionally independent given $f(U)$. For any $x \in L$ and any $y \in R$, we must have

$$\mathbb{P}(U_1 = x \wedge U_2 = y | f(U) = z) = \mathbb{P}(U_1 = x | f(U) = z)\mathbb{P}(U_2 = y | f(U) = z) > 0,$$

and so $f^{-1}(z) = L \times R$. Furthermore, since U_1 and U_2 are independently uniformly distributed over $\mathbb{Z}_s$ and U is uniformly distributed over $\mathbb{Z}_s^2$, we have

$$\frac{1}{s} = \mathbb{P}(f(U) = z) = \mathbb{P}(U_1 \in L \wedge U_2 \in R) = \mathbb{P}(U_1 \in L) \times \mathbb{P}(U_2 \in R) = \frac{|L|}{s} \times \frac{|R|}{s}.$$

Therefore, $|L||R| = s$. □

Lemma 6.5.3. *Let $s \geq 2$ be an integer, let $0 < \epsilon \leq \frac{1}{s^2(2s+1)}$ be a constant. Let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ be any non-trivial protocol on C_n with s colours and let $X \in_u \mathrm{Fix}(\mathcal{P})$. If f_1 and f_2 are semi-perfect functions and (X_0, X_2) and (X_1, X_3) are (s^2, ϵ)-uniform, then, for any $c_1, c_2 \in \mathbb{Z}_s$, we have*

$$|\{c_0 | f_1(c_0, c_2) = c_1\}| = |\{c_3 | f_2(c_1, c_3) = c_2\}|.$$

Proof. We proceed by contradiction. Let:

$$S_0 = \{c_0 | f_1(c_0, c_2) = c_1\} \qquad \text{and} \qquad S_3 = \{c_3 | f_2(c_1, c_3) = c_2\}.$$

Without loss of generality assume $|S_0| < |S_3|$ so since $|S_0| < s$ we must have $|S_3| > \left(1 + \frac{1}{s}\right)|S_0|$. Now since (X_0, X_2) is (s^2, ϵ)-uniform,

$$\mathbb{P}(X_1 = c_1 \wedge X_2 = c_2) = \sum_{x \in S_0} \mathbb{P}\big((X_0, X_2) = (x, c_2)\big) \leq |S_0| \left(\frac{1}{s^2} + \epsilon\right).$$

Similarly, since (X_1, X_3) is (s^2, ϵ)-uniform,

$$\mathbb{P}(X_1 = c_1 \wedge X_2 = c_2) = \sum_{x \in S_3} \mathbb{P}\big((X_1, X_3) = (c_1, x)\big) \geq |S_3| \left(\frac{1}{s^2} - \epsilon\right).$$

However, since $\epsilon \leq \frac{1}{s^2(2s+1)}$, this implies

$$1 + \frac{1}{s} < \frac{|S_3|}{|S_0|} \leq \frac{s^{-2} + \epsilon}{s^{-2} - \epsilon} \leq \frac{\frac{1}{s^2} + \frac{1}{s^2(2s+1)}}{\frac{1}{s^2} - \frac{1}{s^2(2s+1)}} = 1 + \frac{1}{s},$$

which is a contradiction. □

Lemma 6.5.4. *Let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ be a non-trivial protocol on C_n with $s \geq 2$ colours, let $X \in_u \mathrm{Fix}(\mathcal{P})$ and let j be any index (indices taken modulo n). If f_{j-1}, f_j and f_{j+1} are semi-perfect functions and (X_{j-2}, X_j), (X_{j-1}, X_{j+1}) and (X_j, X_{j+2}) are (s^2, ϵ)-uniform, then f_j is a perfect function.*

Proof. We proceed by contradiction. Without loss of generality, assume $j = 0$ and fix $c_0, c_0' \in \mathbb{Z}_s$ arbitrarily. Now choose $c_{-1}, c_1 \in \mathbb{Z}_s$ such that $f_0(c_{-1}, c_1) = c_0$ and choose $c_{-1}', c_1' \in \mathbb{Z}_s$ such that $f_0(c_{-1}', c_1') = c_0'$. Also let $c_0'' = f_0(c_{-1}', c_1)$. Now by Lemma 6.5.3,

$$|L(f_0, c_0)| = |\{x | f_0(x, c_1) = c_0\}| = |\{x | f_1(c_0, x) = c_1\}| = |R(f_1, c_1)|$$
$$\text{and} \qquad |L(f_0, c_0'')| = |\{x | f_0(x, c_1) = c_0''\}| = |\{x | f_1(c_0'', x) = c_1\}| = |R(f_1, c_1)|.$$

Similarly

$$|R(f_0, c_0'')| = |\{x | f_0(c_{-1}', x) = c_0''\}| = |\{x | f_{-1}(x, c_0'') = c_{-1}'\}| = |L(f_{-1}, c_{-1}')|$$
$$\text{and} \qquad |R(f_0, c_0')| = |\{x | f_0(c_{-1}', x) = c_0'\}| = |\{x | f_{-1}(x, c_0') = c_{-1}'\}| = |L(f_{-1}, c_{-1}')|.$$

Recall that $|L(f_0, z)| \cdot |R(f_0, z)| = s$ for all $z \in \mathbb{Z}_s$ (Proposition 6.5.2). Therefore, $|R(f_0, c_0')| = |R(f_0, c_0'')|$ if and only if $|L(f_0, c_0')| = |L(f_0, c_0'')|$. Hence,

$$|L(f_0, c_0)| = |L(f_0, c_0'')| = |L(f_0, c_0')|.$$

Similarly, $|R(f_0, c_0)| = |R(f_0, c_0')|$ (for arbitrary $c_0, c_0' \in \mathbb{Z}_s$) and therefore f_0 is a perfect function. □

Lemma 6.5.5. *Let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ be a non-trivial protocol on C_n with $s \geq 2$ colours, such that f_j is a perfect function for some j. Then $\mathrm{fix}(\mathcal{P}) \leq a s^{(n-1)/2}$, where a is the greatest factor of s less than or equal to $\sqrt{s}$.*

Proof. Without loss of generality, assume $j = 2$. Since f_2 is perfect, let $l = |L(f_2, z)|$ and $r = |R(f_2, z)|$. Without loss of generality, assume $l \leq r$ and therefore $l \leq a$. Then X_2 takes at most s different values and X_1, conditioned on $X_2 = z$ for any $z \in \mathbb{Z}_s$, takes at most l different values. Therefore, the pair (X_1, X_2) takes at most ls different values in $\mathbb{Z}_s^2$ and $h(1, 2) \leq \log_s(ls)$. We have

$$\begin{aligned} H(X) &= h(1, 2, 3, \ldots, n) \\ &= h(1, 2, 4, 6, \ldots, n-3, n-1) \\ &\leq h(1, 2) + \sum_{i=1}^{(n-3)/2} h(2i + 2) \\ &\leq \log_s(ls) + \frac{n-3}{2}. \end{aligned}$$

Therefore $\mathrm{fix}(\mathcal{P}) = s^{H(X)} \leq l s^{(n-1)/2} \leq a s^{(n-1)/2}$. □

Theorem 6.5.6. *For any integer $s \geq 2$, let a be the greatest factor of s less than or equal to $\sqrt{s}$. There exists some $N \in \mathbb{N}$ such that*

$$\mathrm{gn}(C_n, s) = \begin{cases} \frac{n}{2}, & \text{for even } n, \\ \frac{n-1}{2} + \log_s a, & \text{for odd } n > N, \end{cases}$$

and $\mathcal{P}_{fcp}$ is an optimal protocol on C_n with s colours for any odd $n \geq N$.

Proof. Let ϵ and δ be the values given in Definition 6.4.5, let $N = 7(\delta^{-1} + 2)$ and let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ be any non-trivial protocol on C_n with s colours. We have two cases:

Case one For all j, either:

- at least one of the functions f_{j-1}, f_j and f_{j+1} is not semi-perfect or
- at least one of (X_{j-2}, X_j), (X_{j-1}, X_{j+1}), (X_j, X_{j+2}) is not (s^2, ϵ)-uniform.

Case two There exists some j such that:

- the functions f_{j-1}, f_j and f_{j+1} are all semi-perfect and
- (X_{j-2}, X_j), (X_{j-1}, X_{j+1}) and (X_j, X_{j+2}) are all (s^2, ϵ)-uniform.

For case one, we can conclude that $\mathrm{fix}(\mathcal{P}) \leq s^{(n-1)/2} \leq \mathrm{fix}(\mathcal{P}_{fcp})$ by Lemma 6.4.9. In case two, f_j must be a perfect function (Lemma 6.5.4) and then $\mathrm{fix}(\mathcal{P}) \leq as^{(n-1)/2} = \mathrm{fix}(\mathcal{P}_{fcp})$ (Lemma 6.5.5). In either case, $\mathrm{fix}(\mathcal{P}_{fcp}) \geq \mathrm{fix}(\mathcal{P})$. Hence $\mathcal{P}_{fcp}$ is optimal. □

6.6 Two colours

In this section we compute the guessing number of C_n with $s = 2$ colours. For $n \leq 12$, the values of $\mathrm{gn}(C_n, 2)$ was computed by a computer search in 2011 [17], but the value of $\mathrm{gn}(C_n, 2)$ for odd $n \geq 13$ was an open problem. We show here that $\mathcal{P}_{fcp}$ (which, on $s = 2$ colours, is equivalent to Christofides and Markström's clique strategy) is optimal on C_n with 2 colours for all odd $n \geq 7$.

Definition 6.6.1. Let $g : \mathbb{Z}_2^2 \to \mathbb{Z}_2$ be the function defined by

$$g(x, y) = 1 - xy \qquad \forall\, x, y \in \mathbb{Z}_2.$$

For each positive integer n, let $\mathcal{P}_5$ denote the protocol on C_n defined by

$$\mathcal{P}_5 = (g, g, g, \ldots, g).$$

The reason for the subscript '5', in the notation for this protocol $\mathcal{P}_5$, is (as we will discover later in this section) it is an optimal protocol on the 5-cycle.

Proposition 6.6.2. *For odd $n \geq 7$, we have* $\mathrm{fix}(\mathcal{P}_5) < 2^{(n-1)/2}$.

Proof. Let $A(n)$ denote the set of all binary n-tuples containing no two consecutive 0s (cyclically) and no three consecutive 1s (cyclically). Using the protocol $\mathcal{P}_5$, the player at vertex i will guess incorrectly if and only if the triple (c_{i-1}, c_i, c_{i+1}) is assigned

$$(0,0,0), (0,0,1), (1,0,0) \text{ or } (1,1,1),$$

with indicies taken modulo n, *i.e.* if and only if (c_{i-1}, c_i, c_{i+1}) does not contain two consecutive 0s nor three consecutive 1s. Therefore, $A(n) = \mathrm{Fix}(\mathcal{P}_5)$ for each n. Now, for $n \geq 6$, a binary sequence in $A(n)$ must contain at least two 0s. Therefore, the following transformation $\tau : A(n) \to A(n-2) \sqcup A(n-3)$ is well-defined. For any $c = (c_1, c_2, \ldots, c_n)$, let i and j (with $i < j$) be the smallest two indices such that $c_i = c_j = 0$, and define τ the be the transformation which replaces the subsequence $(c_i, c_{i+1}, \ldots, c_j)$ with a single 0. Then,

$$\tau(c_1, c_2, c_3, \ldots, c_n) = (c_1, c_2, \ldots, c_{i-1}, 0, c_{j+1}, \ldots, c_n).$$

The transformation is invertible, because we can find an inverse $\tau^{-1} : A(n-2) \sqcup A(n-3) \to A(n)$ explicitly. For any $c = (c_1, c_2, \ldots, c_m) \in A(n-2) \sqcup A(n-3)$, if i is the smallest index such that $c_i = 0$, let

$$\tau^{-1}(c_1, c_2, c_3, \ldots, c_m) = \begin{cases} (c_1, \ldots, c_{i-1}, 0, 1, 1, 0, c_{i+1}, \ldots, c_m) & : \text{ if } m = n-3 \\ (c_1, \ldots, c_{i-1}, 0, 1, 0, c_{i+1}, \ldots, c_m) & : \text{ if } m = n-2. \end{cases}$$

So τ is a bijection and thus $|A(n)| = |A(n-2) \sqcup A(n-3)|$. Then, the sequence $\bar{A}(n) = |A(n)|$ satisfies the recurrence

$$\bar{A}(n) = \bar{A}(n-2) + \bar{A}(n-3) \qquad \forall\, n \geq 6.$$

It is sufficient to show that $\bar{A}(n) = \mathrm{fix}(\mathcal{P}_5) < 2^{(n-1)/2}$ for all $n \geq 6$. We will do this by induction. The initial terms of the sequence $\bar{A}(3) = 3$, $\bar{A}(4) = 2$ and $\bar{A}(5) = 5$ can be found be inspection:

$$\begin{aligned} A(3) &= \{011, 101, 110\}, \\ A(4) &= \{0101, 1010\}, \text{ and} \\ A(5) &= \{01011, 10110, 01101, 11010, 10101\}. \end{aligned}$$

Then, using the recurrence we can compute some more terms:

n	3	4	5	6	7	8
$\bar{A}(n)$	3	2	5	5	7	10

Thus, we have the base cases for our induction: $\bar{A}(n) < 2^{(n-1)/2}$ for $n = 6, 7, 8$. For $n \geq 9$, the inductive step is

$$\bar{A}(n) = \bar{A}(n-2) + \bar{A}(n-3) < 2^{(n-3)/2} + 2^{(n-4)/2} < 2^{(n-1)/2}.$$

This completes the induction. Therefore, for odd $n \geq 7$, $\mathrm{fix}(\mathcal{P}_5) = \bar{A}(n) < 2^{(n-1)/2}$. □

Definition 6.6.3. There are only two bijections on the set $\mathbb{Z}_2$, the identity function and the *toggle* function $\mathbf{t} : \mathbb{Z}_s \to \mathbb{Z}_s$ defined by

$$\mathbf{t}(x) = 1 - x.$$

Two quadruples $(x_1, x_2, x_3, x_4), (x_1', x_2', x_3', x_4') \in \mathbb{Z}_2^4$ are called *incompatible* if either

- $x_1 = x_1'$ and $x_3 = x_3'$ but $x_2 \neq x_2'$, or
- $x_2 = x_2'$ and $x_4 = x_4'$ but $x_3 \neq x_3'$.

Two quadruples are called *compatible* if they are not incompatible. If Q and Q' are two sets of quadruples, then we say that Q is *equivalent* to Q' if there exists bijections $b_i : \mathbb{Z}_2 \to \mathbb{Z}_2$ for $i = 1, 2, 3, 4$ such that

$$Q' = \big\{(b_1(x_1), b_2(x_2), b_3(x_3), b_4(x_4)) \big| (x_1, x_2, x_3, x_4) \in Q\big\}.$$

If $\{i_1, i_2, \ldots, i_k\}$ are the indices such that $b_i = \mathbf{t}$ then the action of replacing Q with Q' is called *toggling* the $i_1^{\text{th}}, i_2^{\text{th}}, \ldots, i_k^{\text{th}}$ coordinates. Finally, let $\mathcal{J}$ denote the following set of 5 pairwise compatible quadruples:

$$\mathcal{J} = \{(0,1,0,1), (0,1,1,0), (1,0,1,0), (1,0,1,1), (1,1,0,1)\}.$$

Lemma 6.6.4. *Suppose $J \subseteq \mathbb{Z}_2^4$ is a set of pairwise compatible quadruples such that $|J| \geq 5$. If $|K| \geq 3$ where $K = \{(b, c) \mid \exists\, a, d \in \mathbb{Z}_2$ such that $(a, b, c, d) \in J\}$, then J is equivalent to $\mathcal{J}$.*

Proof. It suffices to show that we can toggle some coordinates of J until $J = \mathcal{J}$. For $i, j \in \mathbb{Z}_2$ let T_{ij} denote the set of all quadruples $(x_1, x_2, x_3, x_4) \in \mathbb{Z}_2^4$ such that $x_1 = i$ and $x_4 = j$. For example

$$T_{11} = \{(1,0,0,1), (1,0,1,1), (1,1,0,1), (1,1,1,1)\}.$$

By the pigeon-hole principle, J must contain at least 2 elements from at least one of T_{00}, T_{01}, T_{10} or T_{11}. We can toggle the 1^{st} and/or 4^{th} coordinates as needed, such that

$$|J \cap T_{11}| \geq 2.$$

We can then toggle the 2^{nd} and/or 3^{rd} coordinates of J such that $(1,0,1,1) \in J$. Now, since $(1,0,1,1)$ is incompatible with everything in T_{11} except for $(1,1,0,1)$, we must have $(1,1,0,1) \in J$ too. Therefore,

$$(1,0,1,1), (1,1,0,1) \in J.$$

Now, we know $(0,1)$ and $(1,0)$ are in K and $|K| \geq 3$. If $(1,1) \notin K$, then we would have $(0,0) \in K$ and we could toggle both the 2^{nd} and 3^{rd} coordinates of J such that $K = \{(0,1), (1,0), (1,1)\}$. Without loss of generality, we can assume $(1,1) \in K$. There must be some quadruple in J of the form

$$q = (x_1, 1, 1, x_4) \in J.$$

We deduce that $x_1 = 0$. Otherwise, q and $(1,0,1,1)$ would be incompatible. Additionally, $x_4 = 0$. Otherwise, q and $(1,1,0,1)$ would be incompatible. Therefore,

$$(0,1,1,0), (1,0,1,1), (1,1,0,1) \in J.$$

Of the other 13 quadruples in $\mathbb{Z}_2^4$, only 3 are compatible with $(0,1,1,0)$, $(1,0,1,1)$ and $(1,1,0,1)$, namely,

$$(0,0,0,0), (0,1,0,1), \text{ and } (1,0,1,0).$$

Note that $(0,0,0,0)$ is incompatible with both $(0,1,0,1)$ and $(1,0,1,0)$. If $(0,0,0,0)$ were in J, then we would have $|J| = 4$ which is a contradiction. Therefore $(0,0,0,0) \notin J$ and hence

$$J \subseteq \{(0,1,1,0), (1,0,1,1), (1,1,0,1), (0,1,0,1), (1,0,1,0)\} = \mathcal{J}.$$

Finally, since $|J| \geq 5 = |\mathcal{J}|$, we must have $J = \mathcal{J}$. □

Definition 6.6.5. For any $F, F' \in \mathbb{Z}_2^n$, we say F is *equivalent* to F' if there exists bijections $b_i : \mathbb{Z}_2 \to \mathbb{Z}_2$ for $i = 1, 2, \ldots, n$ such that

$$F' = \big\{(b_1(x_1), b_2(x_2), \ldots, b_n(x_n))|(x_1, x_2, \ldots, x_n) \in F\big\}.$$

Two protocols $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ and $\mathcal{P}' = (f_1', f_2', \ldots, f_n')$ are called *equivalent* if there exists bijections $b_i : \mathbb{Z}_2 \to \mathbb{Z}_2$ such that for all i,

$$f_i'(x, y) = b_i(f_i(b_{i-1}(x), b_{i+1}(y))) \qquad \forall\, x, y \in \mathbb{Z}_2.$$

Proposition 6.6.6. *Let $F, F' \in \mathbb{Z}_2^n$ be any two equivalent sets of n-tuples. If F is the fixed set of some protocol $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ then F' is the fixed set of some protocol $\mathcal{P}'$ and $\mathcal{P}$ is equivalent to $\mathcal{P}'$.*

Proof. Let $b_i : \mathbb{Z}_2 \to \mathbb{Z}_2$ be the bijections between F and F'. Let $\mathcal{P}' = (f_1', f_2', \ldots, f_n')$ be the protocol defined by

$$f_i'(x, y) = b_i(f_i(b_{i-1}(x), b_{i+1}(y))) \qquad \forall\, x, y \in \mathbb{Z}_2.$$

By definition, $\mathcal{P}'$ is equivalent to $\mathcal{P}$. Moreover, for any $(x_1', x_2', \ldots, x_n') \in F'$, we must have a corresponding $(x_1, x_2, \ldots, x_n) \in F$ such that $x_i = b_i(x_i')$ for all i, and

$$f_i'(x_{i-1}', x_{i+1}') = b_i(f_i(b_{i-1}(x_{i-1}'), b_{i+1}(x_{i+1}'))) = b_i(f_i(x_{i-1}, x_{i+1})) = x_i',$$

for all i. □

Lemma 6.6.7. *Let $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ be a protocol on C_n with $s = 2$ colours, such that $\mathrm{fix}(\mathcal{P}) > 2^{(n-1)/2}$. As $c = (c_1, c_2, \ldots, c_n)$ ranges over all colourings in $\mathrm{Fix}(\mathcal{P})$, we have that, for any i,*

- *the set of pairs that (c_{i+2}, c_{i+3}) takes has size at least 3, and*
- *the set of values that $(c_{i+1}, c_{i+2}, c_{i+3}, c_{i+4})$ takes is a pairwise compatible set of at least 5 distinct quadruples.*

Proof. By the symmetry of C_n, it suffices to show this for $i = 0$. We prove both parts independently by contradiction.

- Suppose the pair (c_2, c_3) only takes 2 different values. This would imply that $(c_2, c_3, c_5, c_7, \ldots, c_n)$ takes at most $2^{(n-1)/2}$ different values, as $c_5, c_7, c_9, \ldots c_n$ take at most 2 different values each. Since c can be determined knowing onlt the value of $(c_2, c_3, c_5, c_7, \ldots, c_n)$, this would imply $\mathrm{fix}(\mathcal{P}) \leq 2^{(n-1)/2}$, which is a contradiction.

- Now suppose the quadruple (c_1, c_2, c_3, c_4) only takes 4 different values. This implies that $(c_1, c_2, c_3, c_4, c_6, c_8, \ldots, c_{n-1})$ takes at most $2^{(n-1)/2}$ different values, because $c_6, c_8, c_{10}, \ldots c_{n-1}$ can take at most 2 different values each. Since c can be determined from $(c_1, c_2, c_3, c_4, c_6, c_8, \ldots, c_{n-1})$, this implies $\text{fix}(\mathcal{P}) \leq 2^{(n-1)/2}$, which is a contradiction.

- Let J is the set of values that (c_1, c_2, c_3, c_4) takes. For any two quadruples (x_1, x_2, x_3, x_4) and (x'_1, x'_2, x'_3, x'_4) in J,

  - if $x_1 = x'_1$ and $x_3 = x'_3$, then we would have

    $$x_2 = f_2(x_1, x_3) = f_2(x'_1, x'_3) = x'_2.$$

  - if $x_2 = x'_2$ and $x_4 = x'_4$, then we would have

    $$x_3 = f_3(x_2, x_4) = f_3(x'_2, x'_4) = x'_3.$$

  Therefore J is pairwise compatible.

□

Theorem 6.6.8. *If $\mathcal{P} = (f_1, f_2, \ldots, f_n)$ is a protocol on C_n with $s = 2$ colours such that $\text{fix}(\mathcal{P}) > 2^{(n-1)/2}$, then $\mathcal{P}$ is equivalent to $\mathcal{P}_5$.*

Proof. As $c = (c_1, c_2, \ldots, c_n)$ ranges over all colourings in $\text{Fix}(\mathcal{P})$, let $J_j \subset \mathbb{Z}_2^4$ denote the set of quadruples that $(c_{j+1}, c_{j+2}, c_{j+3}, c_{j+4})$ ranges over. By Lemma 6.6.7 and Lemma 6.6.4, J_j must be equivalent to $\mathcal{J}$ for all j. For an arbitrary j, let b_{j+1},b_{j+2},b_{j+3} and b_{j+4} be the bijections through which J_j is equivalent to $\mathcal{J}$, and let b'_{j+2},b'_{j+3},b'_{j+4} and b'_{j+5} be the bijections through which J_{j+1} is equivalent to $\mathcal{J}$. Since for all $x, y \in \mathbb{Z}_2$ we have

$$\begin{aligned} f_{j+3}(x, y) &= b_{j+3}(g(b_{j+2}(x), b_{j+4}(y))) \\ \text{and } \ f_{j+3}(x, y) &= b'_{j+3}(g(b'_{j+2}(x), b'_{j+4}(y))). \end{aligned}$$

We must have $b_{j+2} = b'_{j+2}$ and $b_{j+3} = b'_{j+3}$ and $b_{j+4} = b'_{j+4}$. Therefore, the bijections through which all the J_is are equivalent to $\mathcal{J}$ all agree. Hence, there exists an n-tuple of bijections $(b_1, b_2, \ldots, b_n)$ such that

$$f_i(x, y) = b_i(g(b_{i-1}(x), b_{i+1}(y))) \quad \forall i,$$

i.e. the protocol $\mathcal{P}$ is equivalent to $\mathcal{P}_5$. □

Corollary 6.6.9. *For all odd $n > 6$, $\mathrm{gn}(C_n, 2) = \frac{n-1}{2}$.*

Proof. If $\mathcal{P}_{fcp}$ is not optimal, then there exists some protocol $\mathcal{P}$ with $\mathrm{fix}(\mathcal{P}) > \mathrm{fix}(\mathcal{P}_{fcp}) = 2^{(n-1)/2}$. By Theorem 6.6.8, this would mean $\mathcal{P}$ is equivalent to $\mathcal{P}_5$ which is a contradiction (Proposition 6.6.2). Hence, $\mathcal{P}_{fcp}$ is optimal and

$$\mathrm{gn}(C_n, 2) = \log_2 \mathrm{fix}(\mathcal{P}_{fcp}) = \frac{n-1}{2}.$$

□

The guessing game on C_5 with $s = 2$ is a special case, because

$$4 = \mathrm{fix}(\mathcal{P}_{fcp}) < \mathrm{fix}(\mathcal{P}_5) = 5,$$

and $\mathcal{P}_5$ is the unique (up to equivalence) optimal protocol on C_5. This special case is a known result [4].

6.7 Index coding with side information

In the problem of index coding with side information [4] on a graph G, a sender aims communicate n messages $c_1, c_2, \ldots, c_n$ (where $c_i \in \mathbb{Z}_s$) to n receivers $v_1, v_2, \ldots, v_n$ (the vertices of G). Each receiver, c_i, knows c_j in advance, for each j such that $v_j v_i$ is an edge in G. The sender is required to broadcast a message to all receivers (the same message to all receivers) so that each receiver, v_i, can recover c_i. The information defect [26, 42] (Definition 6.7.1) is a measure of the minimum amount of information that the sender would need to broadcast to achieve this.

Definition 6.7.1. Let G be a directed graph on n vertices, and let $s \geq 2$ be an integer. Suppose $B = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$ is a set of m protocols on G with s colours, such that

$$\bigcup_{i=1}^{m} \mathrm{Fix}(\mathcal{P}_i) = \mathbb{Z}_s^n.$$

The set B is said to be a solution to the *index coding with side information* problem on G with s colours. The *information defect* of G with s colours is defined to be

$$\beta(G, s) = \log_s(m),$$

where $m = |B|$ is the minimum size of a set B of protocols that solves the index-coding with side information problem on G with s colours.

Informally, we can interpret m as the smallest integer such that the sender can achieve this by broadcasting one of only m different messages. *i.e.* if the protocols $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$ are known in advance, then the sender can simply broadcast an index k, such that $c = (c_1, c_2, \ldots, c_n) \in \mathrm{Fix}(\mathcal{P}_k)$, to all the receivers, and then each receiver, v_i, can recover c_i by evaluating $f_k(c)$.

The relationship between the guessing number and information defect of a graph is well known. Explicitly, let $\mathcal{C}_s(G)$ be the *confusion graph* [4, 9] (also known as the "code graph" [17]), defined to have vertex set $\mathbb{Z}_s^n$, in which two vertices $c, c' \in \mathbb{Z}_s^n$ are adjacent if and only if for some $i \in [n]$, $c_i \neq c'_i$ but for each j such that $v_j v_i \in E(G)$ we have $c_j = c'_j$. Intuitively $c, c' \in Z_s^n$ are 'confusable' (joined by an edge in the confusion graph) if there is no protocol $\mathcal{P}$, for the guessing game on G, such that both $c, c' \in \mathrm{Fix}(\mathcal{P})$ (*i.e.* c and c' cannot both be encoded with the same message from the sender.). If $\chi(\mathcal{C}_s(G))$ is chromatic number of the confusion graph of G and $\alpha(\mathcal{C}_s(G))$ is the size of the largest independent set in the confusion graph of G, then

$$\beta(G, s) = \log_s \chi(\mathcal{C}_s(G)) \qquad \text{and} \qquad \mathrm{gn}(G, s) = \log_s \alpha(\mathcal{C}_s(G)).$$

For a graph H, in general, there is the identity $\chi(H)\alpha(H) \geq |H|$ and so we have

$$\beta(G, s) + \mathrm{gn}(G, s) \geq \log_s |\mathcal{C}_s(G)| = n.$$

This also appears in [42]. We use this identity and the fact that the fractional-clique protocol $\mathcal{P}_{fcp}$ is optimal (Theorem 6.5.6) to prove Theorem 6.7.2. This theorem in general is a new result, although the case $s = 2$ was proven combinatorially in [9]. Theorem 6.7.2 shows that the size of an optimal index code, $\beta(G, s)$, depends on the factorisation structure of the size of the alphabet, s, used for the input.

Theorem 6.7.2. *For a given s, let b be the smallest factor of s which is at least $\sqrt{s}$. There exists some N such that for all odd $n > N$,*

$$\beta(C_n, s) = \frac{n-1}{2} + \log_s b.$$

Proof. Write $a = s/b$. First by Theorem 6.5.6, $\mathrm{gn}(C_n, s) = (n-1)/2 + \log_s a$ for all large enough odd n. Therefore,

$$\beta(C_n, s) \geq n - \mathrm{gn}(C_n, s) = \frac{n-1}{2} + \log_s b.$$

To show that we in fact get equality, we define a set of $bs^{(n-1)/2}$ possible messages with which the sender can solve the index coding with side information problem on C_n. Let ϕ and ψ be defined as in Definition 6.3.1. This means that $\phi \times \psi$ is a bijection from $\mathbb{Z}_a \times \mathbb{Z}_b$ to $\mathbb{Z}_s$. Now for any colouring $c = (c_1, c_2, \ldots, c_n) \in \mathbb{Z}_s^n$ let the sender broadcast the following values:

- For $i = 1, 2, 3, \ldots, \frac{n-1}{2}$, the sender broadcasts the residue $\phi(c_{2i-1}) + \phi(c_{2i})$ modulo a and the residue $\psi(c_{2i}) + \psi(c_{2i+1})$ modulo b.
- Additionally, the sender broadcasts the residue $\psi(c_1) + \phi(c_n)$ modulo b.

The sender broadcasts $\frac{n-1}{2}$ residues modulo a and $\frac{n+1}{2}$ residues modulo b, and so the total number of possible messages that the sender might send is

$$m = a^{(n-1)/2} b^{(n+1)/2} = b s^{(n-1)/2}.$$

Furthermore, each receiver, v_i, knows c_{i-1} and c_{i+1}, and so can recover both c_i because she can recover both $\phi(c_i)$ and $\psi(c_i)$. □

Chapter 7

The information flow problem

The information flow problem on a network asks whether r senders, $v_1, v_2, \ldots, v_r$ can each send messages to r corresponding receivers $v_{n+1}, \ldots, v_{n+r}$ via intermediate nodes $v_{r+1}, \ldots, v_n$. For a given finite $R \subset \mathbb{Z}^+$, the clock network $N_n(R)$ (Definition 7.1.7 has edge $v_i v_k$ if and only if $k > r$ and $k - i \in R$. We show that the information flow problem on $N_n([r])$ can be solved for all $n \geq r$. We also show that for any finite R such that $\gcd(R) = 1$ and $r = \max(R)$, we show that the information flow problem can be solved on $N_n(R)$ for all $n \geq 3r^3$. This is an improvement on the bound given in [52] and answers an open question from [43].

7.1 The information flow problem

The information flow problem (Definition 7.1.4) is an important problem for multiuser information theory. This problem was introduced in [1] to formalise the multiple unicast problem. It was shown that the information flow problem is equivalent to the guessing number of a related digraph [43]. The same paper poses an open question regarding the guessing number of a class of digraphs known as clock digraphs (Definition 7.1.8). Corollary 7.4.7 answers this question.

Definition 7.1.1. A *information flow network* of length n and width r is an acyclic digraph N with vertex set $\{v_i\}_{i=1}^{n+r}$ such that the *input nodes* (vertices $v_1, v_2, \ldots, v_r$) have no incoming edges. Vertices $v_{n+1}, v_{n+2}, \ldots, v_{n+r}$ are called the *output nodes* and vertices $v_{r+1}, v_{r+2}, \ldots, v_n$ are called *intermediate nodes*. For any $r < k \leq n + r$, let $\Gamma_-(k)$ denote set of all indices, i, such that $v_i v_k$ is an edge.

In this chapter, we will simply say 'network' when we mean 'information flow network'.

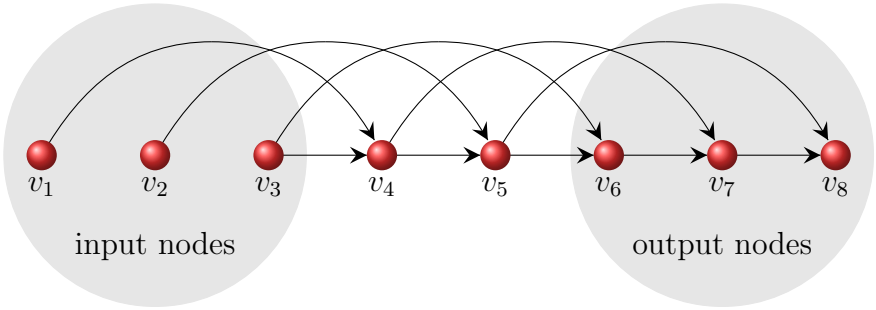


Figure 7.1: The network $N_5(\{1,3\})$ has 3 input nodes (v_1, v_2, v_3) and 3 output nodes (v_6, v_7, v_8) and 2 intermediate nodes (v_4, v_5).

Definition 7.1.2. For any network N and any integer $s \geq 2$, a *circuit* on N over $\mathbb{Z}_s$, is a n-tuple of functions $F = (f_{r+1}, f_{r+2}, \ldots, f_{n+r})$,

$$f_k : \mathbb{Z}_s^{\Gamma_-(k)} \to \mathbb{Z}_s \qquad \forall\, r < k \leq n + r,$$

where n and r are the length and width respectively of N. For each input $c = (c_1, c_2, \ldots, c_r) \in \mathbb{Z}_s^r$, let $X = (X_1, X_2, \ldots, X_{n+r})$ denote the unique $(n+r)$-tuple in $\mathbb{Z}_s^{n+r}$ such that $X_i = c_i$ for $i = 1, 2, \ldots, r$ and

$$X_k = f_k\left(X_i \,\middle|\, i \in \Gamma_-(k)\right) \qquad \forall\, r < k \leq n + r.$$

X is called the *valuation* of F.

Definition 7.1.3. A circuit, $F = (f_{r+1}, f_{r+2}, \ldots, f_{n+r})$, is called *linear* if and only if each function f_k is a linear map. For any linear circuit F, let M_F denote the *R-circuit matrix* of F; the linear map $M_F : \mathbb{Z}_s^r \to \mathbb{Z}_s^{n+r}$ such that $X = M_F(c)$ for all inputs $c \in \mathbb{Z}_s^r$. *i.e.* M_F is a $(n+r) \times r$ matrix such that

$$X^T = M_F c^T$$

where X^T and c^T are the column vectors of the valuation X and the input c respectively. The first r rows of the R-circuit matrix M_F are a copy of the $r \times r$ identity matrix, I_r.

Definition 7.1.4. A network N of width r is *s-solvable* if and only if there exists a circuit on N over $\mathbb{Z}_s$ such that for all inputs $c \in \mathbb{Z}_s^r$, the valuation satisfies

$$(X_1, X_2, \ldots, X_r) = c = (X_{n+1}, X_{n+2}, \ldots, X_{n+r})$$

A network N of width r is *linearly s-solvable* if and only if there exists a linear circuit F on N over $\mathbb{Z}_s$ such that the final r rows of M_F are a copy of I_r.

Definition 7.1.5. For a given network N and an integer $s \geq 2$, the *Information Flow Problem* asks whether or not N is s-solvable. Similarly, the *Linear Information Flow Problem* asks whether or not N is linearly s-solvable.

It is natural to consider the information flow problem as an information theory problem in the following way. Each input node, v_i, is a sender trying to send a message to its corresponding receiver at node v_{n+i} via the network of internal nodes. The elements of the group $\mathbb{Z}_s$ correspond to the s distinct messages that could potentially be sent along each edge. There is a traditional method for solving the information flow problem, called "routing", in which each intermediate node simply passes on one of the messages it receives. A network can only be solved by routing if and only if there exist vertex disjoint paths from each sender to its corresponding receiver. There are many examples in which a network is solvable, but cannot be solved by routing alone [14, 35]. Instead we allow each non-input node, v_k, to perform some function, f_k, on the messages it receives from nodes $\Gamma_-(k)$. Each node v_i must send the same message to all nodes v_k such that $i \in \Gamma_-(k)$. Linear circuits are of interest because they are fast to compute and linear circuits are sufficient to solve a large family of networks (Theorems 7.2.5 and 7.4.6).

The information flow problem also has an application to computing the guessing number (Definition 6.1.3) and the information defect (Definition 6.7.1) of directed graphs. Specifically, for any network N with input nodes $v_1, \ldots, v_r$ and output nodes $v_{n+1}, v_{n+2}, \ldots, v_{n+r}$, let G_N denote the digraph obtained by identifying vertex v_i with v_{n+i} for all $1 \leq i \leq r$. The relationship between the s-solvability of a network N and the guessing number (and information defect) of G_N is presented in Theorem 7.1.6 which originally appears in [43]. Note that for our purposes it does not matter if edge $v_{n+i}v_k$ is replaced with v_iv_k (nor would it make any difference if both edges were included) because, for a circuit which solves the network, the valuation would satisfy $X_i = X_{n+i}$.

Theorem 7.1.6. *[43] For any network N of length n and width r,*

$$\mathrm{gn}(G_N, s) \leq r \qquad \text{and} \qquad \beta(G_N, s) \geq n - r.$$

We get the equality $\mathrm{gn}(G_N, s) = r$ if and only if N is s-solvable. Moreover, if N is linearly s-solvable then $\beta(G_N, s) = n - r$.

Definition 7.1.7. For any finite $R \subset \mathbb{Z}^+$ let $r = \max(R)$. For any integer $n > r$ let $N_n(R)$ denote the *clock* network; the network with vertex set is $V = \{v_1, v_2, \ldots, v_{n+r}\}$

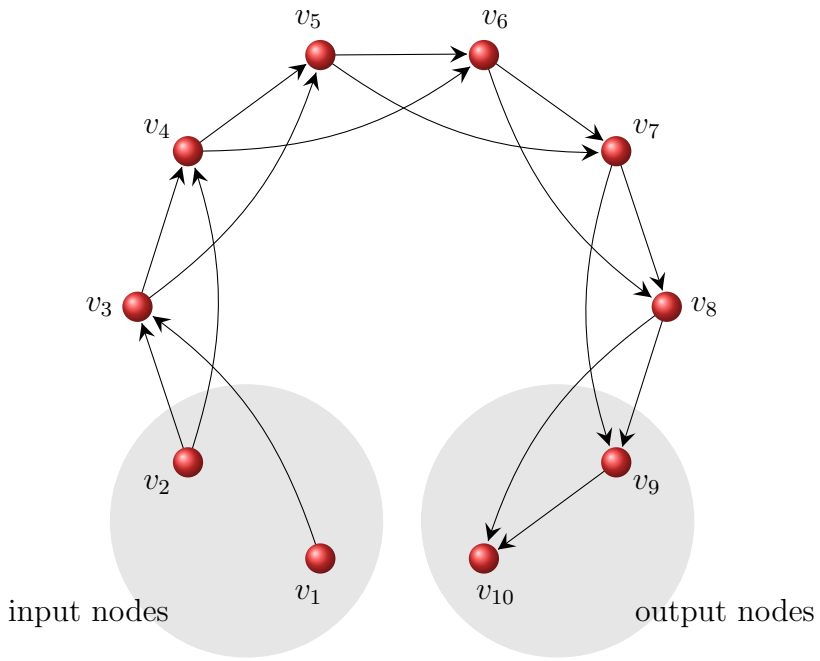


Figure 7.2: The full clock network $N_8(2)$.

and edge set

$$E = \{v_i v_k \mid k > r \text{ and } k - i \in R\}.$$

The network $N_n([r])$ is called the *full* clock network. To simplify notation, we sometimes write

$$N_n(r) = N_n([r]).$$

Definition 7.1.8. For any finite $R \subset \mathbb{Z}^+$ let $r = \max(R)$. For any integer $n > r$ let $G_{clock}(n, R)$ denote the *clock* digraph which has n vertices $\{v_i\}_{i=1}^n$, where $v_i v_j$ is an edge if and only if $j - i$ (modulo n) is an element of R. To simplify notation, for any positive integer r, we say

$$G_{clock}(n, r) = G_{clock}(n, [r]).$$

The clock network inherits it's name from the clock digraph, $G_{clock}(n, R)$, as defined in [43]. When $|R| = 2$, the clock digraph is also known as the Cayley graph $\text{Cay}(n, R)$, or the "shift graph" [52]. The clock digraph, $G_{clock}(n, R)$, can be obtained from the clock network, $N_n(R)$, by identifying nodes v_i and v_{n+i} for all $1 \leq i \leq r$. *i.e.*

$$G_{clock}(n, R) = G_{N_n(R)}.$$

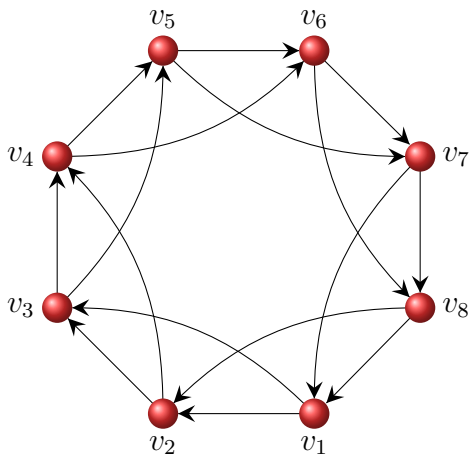


Figure 7.3: The clock digraph $G_{clock}(8, 2)$.

We show in Theorem 7.2.5 that $N_n(r)$ is always linearly s-solvable. By Theorem 7.1.6 (which originally appears in [43]) this implies that the guessing number and information defect of $G_{clock}(n, r)$ are r and $n - r$ respectively.

Proposition 7.1.9. *For a given finite $R \subset \mathbb{Z}^+$, let $r = \max(R)$, let M be a $(n+r) \times r$ matrix with entries in $\mathbb{Z}_s$ and for $i = 1, 2, \ldots, n + r$, let $\omega(i)$ be the i^{th} row of M. If*

- *the first r rows of M form a copy of the identity matrix I_r, and*
- *for all $r < k \leq n+r$, the row $\omega(k)$ is a linear combination of the rows $\{\omega(i) \,|\, k - i \in R\}$,*

then there exists a circuit, F, on $N_n(R)$ such that $M_F = M$.

Proof. For $k = r + 1, r + 2, \ldots, n + r$, and $j \in R$, let $\lambda_{kj} \in \mathbb{Z}_s$ be the constants by which $\omega(k)$ is a linear combination of $\{\omega(k - j) \mid j \in R\}$. *i.e.*

$$\omega(k) = \sum_{j \in R} \lambda_{kj} \omega(k - j).$$

For all pairs (k, j) such that $k - j \notin R$ we set $\lambda_{kj} = 0$. Now let $F = (f_{r+1}, \ldots, f_{n+r})$ be the circuit on $N_n(R)$ defined by

$$X_k = f_k \left(X_i \,\middle|\, i \in \Gamma_-(k) \right) = \sum_{j \in R} \lambda_{kj} X_{k-j},$$

and for $i = 1, 2, \ldots, n + r$, let ω'_i be the i^{th} row of M_F. Since the first r rows of any R-circuit matrix form a copy of I_r, we must have $\omega(i) = \omega'(i)$ for $i = 1, 2, \ldots, r$.

$$\mathcal{I}_{4,3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathcal{I}_{30,43} = \begin{array}{|c|c|} \hline \multicolumn{1}{|c}{\multirow{2}{*}{I_{13}}} & \multirow{3}{*}{I_{30}} \\ \multicolumn{1}{|c}{} & \\ \cline{1-1} I_{13} & \\ \hline \end{array}$$

Figure 7.4: The matrices $\mathcal{I}_{4,3}$ and $\mathcal{I}_{30,43}$. Each square represents a copy of an identity matrix.

Then, inductively, for all $k > r$ we must have

$$\omega(k) = \sum_{j \in R} \lambda_{kj} \omega(k-j) = \sum_{j \in R} \lambda_{kj} \omega'(k-j) = \omega'(k).$$

□

7.2 Full clock networks

As Theorem 7.2.5 shows, the full clock network is linearly s-solvable for all s. This is equivalent to Proposition A in [43], however their proof is incomplete (see Example 7.2.6). We show that the full clock network is linearly s-solvable by finding a valid $[r]$-circuit matrix explicitly.

Definition 7.2.1. For any integers $a, b > 0$ we can define $\mathcal{I}_{a,b}$ in the following recursive manner.

- If $a = b$, then $\mathcal{I}_{a,a} = I_a$ (the $a \times a$ identity matrix).
- If $a < b$ then $\mathcal{I}_{a,b} = [\mathcal{I}_{a,b-a}, I_a]$ (the horizontal concatenation of $\mathcal{I}_{a,b-a}$ and I_a).
- If $a > b$ then $\mathcal{I}_{a,b} = \begin{bmatrix} \mathcal{I}_{a-b,b} \\ I_b \end{bmatrix}$ (the vertical concatenation of $\mathcal{I}_{a-b,b}$ and I_b).

For example: $\mathcal{I}_{4,3}$ and $\mathcal{I}_{30,43}$ are depicted in Figure 7.2.

Proposition 7.2.2. *If A is the topleft-most $a \times a$ sub-matrix of $\mathcal{I}_{n,r}$, then $|\det(A)| = 1$.*

Proof. Let A be the topleft-most $a \times a$ submatrix of $\mathcal{I}_{n,r}$. We now construct the pair of integers p and q in the following way. Initially let $x = n$ and $y = r$. Then

iteratively perform the following process.

$$\begin{aligned}&\text{while } x > a \text{ or } y > a:\\ &\qquad \text{if } x > y \text{ replace } x \text{ with } x - y,\\ &\qquad \text{otherwise replace } y \text{ with } y - x.\end{aligned}$$

Throughout this process (by Definition 7.2.1) topleft-most $x \times y$ submatrix of A is always a copy of $\mathcal{I}_{x,y}$. As soon as both x and y are less than or equal to a, we set $p = x$ and $q = y$, and terminate this process. Just before the final iteration, we must have had one of x or y greater than a, so $a < \max(x, y) = p + q$. Now A must be in the following form.

$$A = \begin{bmatrix} \mathcal{I}_{p,q} & P \\ Q & S \end{bmatrix} \tag{7.2.1}$$

where P, Q and S are matrices with dimensions $p\times(a-q)$, $(a-p)\times q$ and $(a-p)\times(a-q)$ respectively. Now, there are two cases:

- If P is the left-most $(a-q)$ columns of a copy of I_p, then $[Q, S]$ is the topleft-most $(a-p)\times a$ submatrix of a large identity matrix.
- If Q is the top-most $(a-p)$ rows of a copy of I_q, then $\begin{bmatrix} P \\ S \end{bmatrix}$ is the topleft-most $a \times (a-q)$ submatrix of a large identity matrix.

In either case, all the entries of S must be zero because $a-p < q$ and $a-q < p$. Now consider $\mathcal{I}_{p,q}$ in which the bottomright-most square submatrix must be a copy of an identiy matrix. Explicitly, for any integer b such that $0 \le b \le \min(p, q)$, we have

$$\mathcal{I}_{p,q} = \begin{bmatrix} * & * \\ * & I_b \end{bmatrix}, \tag{7.2.2}$$

where $*$ denotes arbitrary entries. In particular $\mathcal{I}_{p,q}$ has this form for $b = p + q - a$. Substituting Equation (7.2.2) into Equation (7.2.1), we see that A has the form:

$$A = \begin{bmatrix} \mathcal{I}_{p,q} & P \\ Q & 0 \end{bmatrix} = \begin{bmatrix} * & * & I_{a-q} \\ * & I_{p+q-a} & 0 \\ I_{a-p} & 0 & 0 \end{bmatrix}$$

where a 0 denotes a submatrix full of zeros, and $*$ denotes a submatrix with arbitrary entries. The only non-zero terms in the Leibniz formula for the determinant of A must come exclusively from the submatrices labelled I_{a-q}, I_{p+q-a} and I_{a-p}. Therefore

$$|\det(A)| = |\det(I_{a-q})| \times |\det(I_{p+q-a})| \times |\det(I_{a-p})| = 1.$$

□

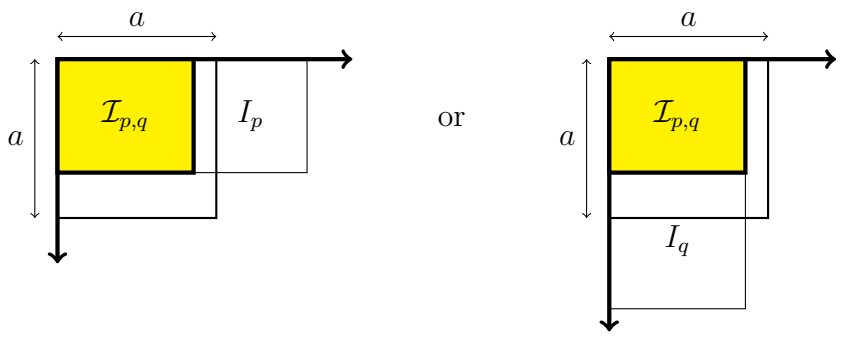


Figure 7.5: In the proof of Proposition 7.2.2, $\mathcal{I}_{p,q}$ is the topleft-most $p \times q$ submatrix of A and A is the topleft-most $a \times a$ submatrix of $\mathcal{I}_{n.r}$.

Definition 7.2.3. For any positive integers n and r with $n > r$ we define the $(n+r)\times r$ matrix $\mathcal{M}_{n,r}$ formed by concatenating a copy of I_r ontop of $\mathcal{I}_{n,r}$. *i.e.*

$$\mathcal{M}_{n,r} = \begin{bmatrix} I_r \\ \mathcal{I}_{n,r} \end{bmatrix}.$$

Proposition 7.2.4. *For any positive integers $n > r$, if M is an $r \times r$ submatrix of $\mathcal{M}_{n,r}$ formed by r consecutive rows then $|\det(M)| = 1$.*

Proof. Let M be the rows $\omega_{a+1}, \omega_{a+2}, \ldots, \omega_{a+r}$. There are two cases: either $0 \leq a < r$ or $r \leq a \leq n$.

- If $a < r$ then M consists of the final $r - a$ rows of I_r followed by the initial a rows of $\mathcal{I}_{n,r}$. In this case, M must have the form:

  $$M = \begin{bmatrix} 0 & I_{r-a} \\ A & * \end{bmatrix}$$

  Where A is the top-leftmost $a \times a$ submatrix of $\mathcal{I}_{n,r}$ and $*$ denotes a submatrix with arbitrary entries. In this case, by Proposition 7.2.2, we have

  $$|\det(M)| = |\det(I_{r-a})| \times |\det(A)| = 1$$

- If $a \geq r$ then M must have the form:

  $$M = \begin{bmatrix} * & I_b \\ I_{r-b} & 0 \end{bmatrix}$$

  where $*$ denotes a submatrix with arbitrary entries and b is the remainder when $n - a$ is divided by r. In this case we have

  $$|\det(M)| = |\det(I_{r-b})| \times |\det(I_b)| = 1.$$

□

Theorem 7.2.5. *For any $n \geq r > 0$ and any $s \geq 2$, the full clock network, $N_n(r)$, is linearly s-solvable.*

Proof. Consider the matrix $\mathcal{M}_{n,r}$ as defined in Definition 7.2.3. By Proposition 7.2.4, for any s, the integer span of any r consecutive rows of $\mathcal{M}_{n,r}$ is all $\mathbb{Z}_s^r$. So any row can be expressed as a linear combination of the preceding r rows. Moreover, the first r rows of $\mathcal{M}_{n,r}$ form a copy of I_r. Therefore $\mathcal{M}_{n,r}$ satisfies the conditions of Proposition 7.1.9, and so there is a circuit F on $N_n(r)$ such that $M_F = \mathcal{M}_{n,r}$. This circuit linearly solves $N_n(r)$ because the final r rows of $\mathcal{M}_{n,r}$ form a copy of I_r. □

Example 7.2.6 is the construction used in the incomplete proof of Proposition A in [43]. For certain integers n and r, this construction does not solve $N_n(r)$.

Example 7.2.6. Let F be a circuit on $N_n(r)$ over $\mathbb{Z}_s$ such that the valuation of F satisfies

$$X_i + X_{i+1} + \cdots + X_{i+r} \equiv 0 \pmod{s} \qquad \text{for } i = 1, 2, 3, \ldots, n - r$$

for any input $c \in \mathbb{Z}_s^r$. To see that this circuit does not solve $N_n(r)$ in general, simply observe that when $n = 7$ and $r = 2$, this does not solve $N_7(2)$. Explicitly, for any $c = (c_1, c_2) \in \mathbb{Z}_s^2$, the valuation must satisfy:

$$\begin{aligned} & X_1 = c_1 \\ \text{and} \quad & X_2 = c_2, \end{aligned}$$

because for any valuation of a circuit on $N_7(2)$, we have $(X_1, X_2) = c$. Moreover, for $j = 3, 4, 5, 6$ and 7, we can deduce:

$$\begin{aligned} & X_3 = -c_1 - c_2, \\ & X_4 = c_1, \\ & X_5 = c_2, \\ & X_6 = -c_1 - c_2 \\ \text{and} \quad & X_7 = c_1, \end{aligned}$$

because $X_{j-2} + X_{j-1} + X_j \equiv 0$. Finally, if F solved $N_7(2)$, then we would have:

$$\begin{aligned} & X_8 = c_1 \\ \text{and} \quad & X_9 = c_2. \end{aligned}$$

However, this is not possible; $X_9 = c_2$ cannot be determined from only $X_7 = c_1$ and $X_8 = c_1$.

n	4	5	6	7	8	9	10	11	≥ 12
Is $N_n(\{1,3\})$ linearly 2-solvable?	✗	✗	✓	✓	✗	✓	✓	✗	✓
Is $N_n(\{1,3\})$ linearly 3-solvable?	✗	✗	✓	✗	✓	✓	✓	✓	✓

Figure 7.6: The linear 2-solvability and linear 3-solvability of $N_n(\{1,3\})$ for all $n \geq 4$.

7.3 Analysis of a specific case

In this section, for various values n, we investigate the 2-solvability and 3-solvability of the network $N_n(\{1,3\})$. Firstly, we consider $n = 7$ and $n = 8$ in the following example.

Example 7.3.1. Let $n = 7$, $m = 3$ and $R = \{1,3\}$ and consider the following two matrices A and B defined as follows.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 1 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The matrix A is constructed so that the i^{th} row is the sum of the $(i-1)^{\text{th}}$ row and the $(i-3)^{\text{th}}$ row modulo 2. So, by Proposition 7.1.9, A is a valid $\{1,3\}$-circuit matrix (over $\mathbb{Z}_2$) and since the bottom 3 rows of A form an identity matrix, this demonstrates that $N_7(\{1,3\})$ is linearly 2-solvable. Similarly the matrix B demonstrates that $N_8(\{1,3\})$ is linearly 3-solvable. It can be verified by a brute force computer search that $N_7(\{1,3\})$ is not linearly 3-solvable and $N_8(\{1,3\})$ is not linearly 2-solvable.

In general, the s-solvability of a network depends on s. However, for any $n \geq 12$, we can construct a $\{1,3\}$-circuit matrix of length n which is valid over Z_s for any $s \geq 2$ in the following way. For $n \equiv 0, 1, 2 \pmod 3$ iteratively concatenate copies of I_3 to the bottom of I_3, M_{10} or M_{14} respectively, where M_{10} and M_{14} are given in Figure 7.7. We know that no such $\{1,3\}$-circuit matrix exists for $n = 7$ nor $n = 11$ because (by brute force computer search) we computed that $N_7(\{1,3\})$ is not linearly 3-solvable and $N_{11}(\{1,3\})$ is not 2-solvable.

$$M_{10} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ -1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{and} \qquad M_{14} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \\ 1 & 0 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 7.7: Matrices M_{10} and M_{14} show that $N_{10}(\{1,3\})$ and $N_{14}(\{1,3\})$ are linearly s-solvable for all $s \geq 2$.

7.4 General clock networks

We saw in Section 7.3, that the network $N_n(\{1,3\})$ is linearly s-solvable for any s, for all $n \geq 12$. In this section we generalise this result to arbitrary finite sets of positive integers. Specifically, we determine for which finite $R \subset \mathbb{Z}^+$, does there exist a constant n_0 such that $N_n(R)$ is s-solvable for all s and all $n \geq n_0$. We deduce (by Lemma 7.4.1 and Corollary 7.4.7), that such an integer n_0 exists if and only if $\gcd(R) = 1$.

Lemma 7.4.1. *If n is not a multiple of $\gcd(R)$, then $N_n(R)$ is not s-solvable for any $s \geq 2$.*

Proof. Let $d = \gcd(R) > 1$. By definition, each edge $v_i v_k$ only joins vertices such that $i \equiv k \pmod{d}$. Therefore $N_n(R)$ is disconnected with at least one component for each residue modulo d. Now consider some index a, and an input $c = (c_1, c_2, \ldots, c_r)$. If we keep c_i constant for all $i \neq a$ and let c_a vary, then the valuation will only change on vertices in the same component as v_a. Since n is not a multiple of d,

$$(n + a) - a = n \not\equiv 0 \pmod{d}.$$

So the input node, v_a, and its corresponding output node, v_{n+a}, are in a different components of $N_n(R)$. Therefore $N_n(R)$ is not s-solvable for any $s \geq 2$. □

Now consider any $R \subset \mathbb{Z}^+$ such that $\gcd(R) > 1$, there are an infinite number of integers n which are not a multiple of $\gcd(R)$. By Lemma 7.4.1, this is an infinite number of integers n such that $N_n(R)$ is not s-solvable (for any s). Therefore there cannot exist any n_0 such that $N_n(R)$ is s-solvable for all $n \geq n_0$. However, if n is a multiple of $d = \gcd(R) > 1$, then the network $N_n(R)$ is a disjoint union of d copies of

$$N' = N_{n/d}(R') \quad \text{where} \quad R' = \{j/d \mid j \in R\}.$$

So $N_n(R)$ is s-solvable if and only if N' is s-solvable. Since $\gcd(R') = 1$, we can now assume $\gcd(R) = 1$ without losing any generality. We now make the following definition and propositions, used in Theorem 7.4.6 and Corollary 7.4.7.

Definition 7.4.2. Let $s \geq 2$ be an integer, let R be a finite set of positive integers, and let $r = \max(R)$.

- An *R-atomic* matrix is any $r \times r$ matrix, with entries in $\mathbb{Z}_s$, of the form:

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \alpha_r & \alpha_{r-1} & \alpha_{r-2} & \cdots & \alpha_1 \end{bmatrix}$$

  such that $\alpha_j = 0$ for all $j \notin R$.

- A *R-step* matrix is any $r \times r$ matrix, with entries in $\mathbb{Z}_s$, formed by starting with I_r, and then for some $1 \leq t \leq r$, replacing the t^{th} row with $[\beta_1, \beta_2, \ldots, \beta_r]$,

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ \beta_1 & \beta_2 & \cdots & \beta_t & \cdots & \beta_r \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

  where β_i is non-zero only if there is some $j \in R$ such that $i + j \equiv t \pmod{r}$. Since $r \in R$, we always allow β_t to be non-zero.

- For any $1 \leq t \leq r$, the *t-toggle* matrix is the following $r \times r$ matrix, $T(t)$, with

entries in $\mathbb{Z}_s$.

$$T(t) = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ -1 & -1 & \cdots & -1 & \cdots & -1 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}.$$

i.e. The t-toggle matrix is formed from I_r by replacing row t with a row of -1s. A matrix is called a *toggle* matrix iff it is a t-toggle matrix for some t.

Proposition 7.4.3. *Any R-step matrix can be expressed as a product of r R-atomic matrices.*

Proof. Let P denote the only R-atomic matrix which is also a permutation matrix; the R-atomic matrix for which $\alpha_r = 1$ and $\alpha_i = 0$ for all $i < r$. For any $1 \leq t \leq r$ consider the product

$$A_r A_{r-1} \dots A_t \dots A_2 A_1 = S,$$

where A_t is an arbitrary R-atomic matrix and $A_i = P$ for all $i \neq t$. This product, S, is an arbitrary R-step matrix. □

Proposition 7.4.4. *If $\gcd(R) = 1$ and $r = \max(R) > 1$ then for any $1 \leq t \leq r$ the t-toggle matrix can be expressed as a product of $(2r-3)$ R-step matrices.*

Proof. We inductively define a sequence of subsets,

$$U_2 \subset U_3 \subset U_4 \subset \cdots \subset U_r = [r],$$

in the following manner. Let $U_2 = \{x, t\}$ where $x \in [r]$ is chosen so that $t - x \pmod r \in R$. For $k = 3, 4, 5, \dots, r$, iteratively define $U_k = U_{k-1} \cup \{b\}$ for some $b \in [r] \backslash U_{k-1}$ such that there exists some $a \in U_{k-1}$ such that $a - b \pmod r \in R$. We know a and b exist because $\gcd(R) = 1$. Now let S_k be the matrix formed from I_r by replacing the t^{th} row with

$$(x_1, x_2, x_3, \dots, x_n) \quad \text{where} \quad x_i = \begin{cases} -1 & : \text{ if } i \in U_k \\ 0 & : \text{ otherwise.} \end{cases}$$

Now we prove that S_k can be expressed as a product of $(2k-3)$ R-step matrices, by induction on $k = 2, 3, 4, \dots, r$. For the base case ($k = 2$), S_2 is a R-step matrix. For the inductive step,

$$S_k = E_{ab}(-1) S_{k-1} E_{ab}(1)$$

where $E_{ij}(\lambda)$ is the matrix formed from I_r by replacing the ij^{th} entry with λ. Note that $E_{ab}(1)$ and $E_{ab}(-1)$ are both R-step matrices because $a - b \pmod r \in R$, and S_{k-1} can be expressed as a product of $(2(k-1)-3)$ R-step matrices by the inductive assumption. Therefore S_k can be expressed as a product of

$$1 + (2(k-1) - 3) + 1 = 2k - 3 \quad R\text{-step matrices.}$$

This completes the induction. For $k = r$ we have $U_r = [r]$ and so the t-toggle matrix is $T(t) = S_r$, which can be expressed as a product of $(2r - 3)$ R-step matrices. □

Proposition 7.4.5. *Any $r \times r$ permutation matrix can be expressed as the product of at most $\frac{3r}{2}$ toggle matrices.*

Proof. First we show that an arbitrary k-cycle can be expressed as the product of $k+1$ toggle matrices. Explicitly, if Q is the $r \times r$ matrix corresponding to the k cycle, $(a_1, a_2, \ldots, a_k)$, can be expressed as the product

$$Q = T(a_1)T(a_2)T(a_3)\cdots T(a_{k-1})T(a_k)T(a_1).$$

Now consider the cyclic decomposition of the permutation; the permutation expressed as the composition of at most $n/2$ cycles, such that the sum of the lengths of these cycles is at most n. If each of these cycles are expressed as a product of toggle matrices, then this is a total of at most $\frac{3n}{2}$ toggle matrices. □

Theorem 7.4.6. *For any finite $R \subset \mathbb{Z}^+$, the network $N_n(R)$ is linearly s-solvable if and only if the identity matrix can be expressed as a product of n R-atomic matrices with entries in $\mathbb{Z}_s$.*

Proof. For any valuation, $X = (X_1, X_2, \ldots, X_n)$, of a linear circuit on $N_n(R)$, let Y_i denote the column vector $Y_i = (X_{i+1}, X_{i+2}, X_{i+3}, \ldots, X_{i+r})^T$ for $i = 0, 1, 2, \ldots, n$. Since f_k is linear,

$$X_k = f_k\left(X_{k-j} \mid j \in R\right) = \sum_{j \in R} \alpha_{kj} X_{k-j}.$$

Therefore we must have $Y_i = A_i Y_{i-1}$ where A_i is a R-atomic matrix. Inductively this implies that $Y_i = A_i A_{i-1} \ldots A_2 A_1 Y_0$ for all $i \geq 0$ and thus $Y_n = \mathcal{A} Y_0$, where $\mathcal{A} = A_n A_{n-1} \cdots A_2 A_1$ is a product of n R-atomic matrices. For all inputs $c \in \mathbb{Z}_s^r$ we have

$$F(c) = Y_n = \mathcal{A} Y_0 = \mathcal{A} c.$$

So if X is the valuation of a circuit which linearly s-solved $N_n(R)$, then $c = \mathcal{A}c$ for all $c \in \mathbb{Z}_s^r$. Hence $\mathcal{A} = I_r$, and $\mathcal{A}$ is a product of exactly n R-atomic matrices. Conversely the function f_k can be reconstructed from the R-atomic matrix A_{k-r}, for each $k = r+1, r+2, \ldots, n+r$, so this construction is reversible. □

Corollary 7.4.7. *Let R be any finite set of positive integers with* $\gcd(R) = 1$ *and let* $r = \max(R)$. *For any* $n \geq 3r^3$, *the network* $N_n(R)$ *is linearly* s*-solvable for any integer* $s \geq 2$.

Proof. It suffices to show that the identity matrix can be expressed as a product of exactly n R-atomic matrices. Let P denote the only atomic matrix which is also a permutation matrix; the atomic matrix for which $\alpha_r = 1$ and $\alpha_i = 0$ for all $i < r$. Note that P is a r-cycle and so $P^r = I_r$. Let $Q = P^{-n}$ and note that Q is a permutation matrix. By Proposition 7.4.5, we can write Q as a product of $k \leq \frac{3r}{2}$ toggle matrices. By Proposition 7.4.4, we can write each of these toggle matrices as a product of $(2r-3)$ R-step matrices, and by Proposition 7.4.3 we can write each of these step matrices as a product of r R-atomic matrices. Therefore Q can be expressed as a product of $kr(2r-3)$ R-atomic matrices. Since $k \leq \frac{3r}{2}$ and $n \geq 3r^3 > \frac{3r}{2}r(2r-3)$, we must have $n - kr(2r-3) \geq 0$ and so

$$Q \times P^{n-kr(2r-3)} = (Q \times P^n) \times (P^r)^{k(2r-3)} = I_r.$$

Since P is a R-atomic matrix and Q can be expressed as a product of $kr(2r-3)$ R-atomic matrices, we can express I_r as a product of n R-atomic matrices. □

For finite $R \subset \mathbb{Z}^+$, let $n_0 = n_0(R)$ be the minimum integer such that $N_n(R)$ is s-solvable for all $s \geq 2$ for all $n \geq n_0$. Corollary 7.4.7 shows that n_0 is well defined and that $n_0 \leq 3r^3$ (where $\max(R) = r$). Theorem 7.2.5 shows that $n_0([r]) = r$ and in Section 7.3, we deduced that $n_0(\{1,3\}) = 12$. The value $n_0(R)$ for R in general, remains an open question. We conclude this section with an example which demonstrates that the cubic bound $n_0 \leq 3r^3$ in Corollary 7.4.7 cannot be replaced with any bound less than $r^2 - r$.

Example 7.4.8. For any integer $r \geq 3$, let $n = r^2 - r - 1$ and consider the networks $N = N_n(\{1, r\})$ and $N' = N_n(\{1, r-1\})$, and consider the digraphs $G = G_{clock}(n, \{1, r\})$ and $G' = G_{clock}(n, \{1, r-1\})$. Using Theorem 7.1.6, the existence of the acyclic network N' of width $r-1$ such that $G' = G_{N'}$ implies that

$$\mathrm{gn}(G', s) \leq r - 1.$$

Note that $r(r-1) \equiv 1$ modulo n, so G and G' are isomorphic, and so $\mathrm{gn}(G, s) = \mathrm{gn}(G', s) < r$. Now using Theorem 7.1.6 again (since $G = G_N$) we can conclude that N is not s-solvable. Thus

$$n_0(\{1, r\}) \geq r^2 - r,$$

for all $r \geq 3$.

Chapter 8

Future research

The maximum agreement subtree problem

The current best bounds we have for MAST(n) are $\mathcal{O}(\log n)$ and $\Omega(\sqrt{\log n})$. At first, it might seem like the $\mathcal{O}(\log n)$ bound is very weak, because it even applies to unlabelled trees. However, in the proof of the lower bound (as presented in [36]), they construct a $\Omega(\log n)$ agreement subtree in two cases:

- if both are caterpillars, or
- if one is balanced.

The $\sqrt{\cdot}$ the crops up because a tree in $\mathcal{B}(n)$ is guaranteed to have either a caterpillar of length $\Omega(\sqrt{n})$ or a balanced tree of $\Omega(\sqrt{n})$ leaves. Intuitively, it seems as if $\Theta(\log n)$ might be the correct growth rate for MAST(n).
For a pair (or indeed a set of k) trees in $\mathcal{B}(n)$. We have a polynomial bound for the expectaion $\mathbb{E}[\text{mast}(A, B)]$. It is not known whether there exists some constant $c > 0$ such that

$$\mathbb{E}[\text{mast}(A, B)] = n^{c+o(1)}.$$

Subtree transfer operations on binary trees

In chapter 2, for each move $\chi \in \{TBR, SPR, rSPR\}$ we have estimated the diameter $D_\chi(n)$ and the radius $R_\chi(n)$ quite precisely. We saw in Theorem 2.0.1 that both $D_\chi(n)$ and $R_\chi(n)$ are $n - \Theta(\sqrt{n})$. It would be interesting to learn about finer behaviour. For example:

- Are there constants $c'_\chi \geq c'_\chi > 0$ such that

$$D_\chi(n) = n - \big(c_\chi + o(1)\big)\sqrt{n} \qquad \text{and} \qquad R_\chi(n) = n - \big(c'_\chi + o(1)\big)\sqrt{n}\,?$$

- If so, does c_χ or c'_χ depend on χ? If the constants exist, for which χ is the inequality $c_\chi \leq c'_\chi$ strict?

There are of course similar questions arising from Theorem 2.0.2 on the average behaviour of $X = d_\chi(A, B)$ where $A, T \in \mathcal{B}(n)$ are fixed trees and B is obtained from T by a random permutation of the non-root leaf-labels of T (chosen uniformly from S_{n-1}).

- Does there exists a constant $\lambda_\chi > 0$ such that $\mathbb{E}[X] = n - (\lambda_\chi + o(1))n^{2/3}$?
- If so, does λ_χ depend on the tree-shapes of A and T? Does λ_χ depend on χ?

These questions are left as open problems, for now. In chapter 3, the $\mathcal{NNI}$-diameter of $\mathcal{B}(n)$ was bounded by

$$\frac{1}{4}n\log_2 n - n < D_{\mathcal{NNI}}(n) < n\log_2 n + 3n.$$

Does there exist a constant c such that $d_{\mathcal{NNI}}(n) = (c + o(1))n\log n$? For a pair of trees $A, B \in \mathcal{B}(n)$ chosen independently, uniformly at random, does there exist a constant $\lambda > 0$ such that $\mathbb{E}[d_{\mathcal{NNI}}(A, B)] = (\lambda + o(1))n\log n$? If both constants exist, do we have $\lambda = c$ or $\lambda < c$?
For two arbitrary unlabelled binary trees T_1 and T_2, we showed that the $\mathcal{NNI}$-distance between T_1 and T_2 is at most $2n$ (Lemma 3.1.5) by showing that both T_1 and T_2 must be less than n NNI moves away from the caterpillar tree (Lemma 3.1.4). By comparing an almost balanced tree to the caterpillar, it is clear that sometimes $n - \mathcal{O}(\log n)$ NNI moves might be needed to transform T_1 into T_2. However, is n NNI moves always sufficient to transform T_1 into T_2?

Subtree transfer operations on phylogenies

In chapter 4, we showed that $\mathcal{P}(l, r)$ is traversable fby type-$\mathbf{t}$ $p\mathcal{SPR}$s whenever $l \geq 3r + 7$. It would be nice to know that this result still holds, for arbitrary positive integers l and r?
We might also consider the traversability of particular subclass $\mathcal{C}$ of $\mathcal{P}(l, r)$. For example the class of tree-child networks [15], or the class of normal networks [51]. The traversability of $\mathcal{C}$ is not guaranteed by the traversability of $\mathcal{P}(l, r)$. Theoretically, it might not be possible to transform a phylogeny $A \in \mathcal{C}$ to a phylogeny $B \in \mathcal{C}$ unless some of the intermediate phylogenies are in $\mathcal{P}(l, r)\backslash\mathcal{C}$.
It also might be interesting to search for generalisations of the $\mathcal{TBR}$ or the $\mathcal{SPR}$

moves which act on phylogenetic networks with recombination. Some complications arises when trying to deal with the edge orientations. For example, if the root is to be pruned off and then regrafted onto the edge emanating from a recombination vertex, then how do you decide on the new orientation of the edges?

The guessing number of cycles graphs

In chapter 6, for any s we determined $\mathrm{gn}(C_n, s)$ exactly, for all large enough n. Let C_n^r be the graph with vertex set $\{v_1, v_2, \ldots, v_n\}$ such that, each vertex v_i is joined to v_j for all $|i-j| < r$. This is a generalisation of the cycle graph since $C_n \equiv C_n^2$. Entropy techniques similar to those used in the proof of Theorem 6.3.4, can be used to show that

$$\mathrm{gn}(C_n^r, s) \leq \left(1 - \frac{1}{r}\right) n.$$

For all integers $s \geq 2$, $r > 1$ and $n > 2r - 1$. However, just as we found in the $r = 2$ case, it is likely that the optimal protocols will be quite varied depending on whether s is large or n is very large.

The information flow problem

In chapter 7, we established Corollary 7.4.7 for $n_0 = 3r^3$. It would be nice to strengthen this result to some $n_0 = \mathcal{O}(r^2)$. However, by the final example in that chapter, we know that the result fails to hold for $n_0 \leq r^2 - r$, so quadratic is the smallest growth rate we can wish for. I conjecture that there exists some $n_0 = \Theta(r^2)$, such that for all $n \geq n_0$, if R is a finite set of positive integers with $\gcd(R) = 1$ and $\max(R) = r$, then the information flow network, $N_n(R)$, is linearly s-solvable for all s.

Appendix A

Subtree transfer operation distance calculations

Let χ be a subtree transfer operation and let $A \in \mathcal{B}(9)$ be a fixed tree. Let $f(x)$ be the number of trees $B \in \mathcal{B}(9)$ such that $d_\chi(A, B) = x$. We compute $f(x)$ for all x, by the following brute force computation (written in python 2.7). This code iterates for all subtree transfer operations $\chi \in \{\mathcal{TBR}, \mathcal{SPR}, r\mathcal{SPR}, \mathcal{NNI}\}$, and for each of the distinct tree shapes that A might have. The computed values $R_\chi(n)$ and $D_\chi(n)$ for $n \leq 9$ were presented in Figure 2.1. More detailed results are presented in Figure A.1.

The Code

```
def getRightBracket(S,n):
    if S[n]=="(":
        depth=1
        i=n+1
        while depth>0:
            if S[i]=="(":
                depth+=1
            elif S[i]==")":
                depth-=1
            i+=1
        return i-1
    return n

def sf(T):
    T = rootAtZero(T)
    T = putRootedTreeInSF(T)
    return T
```

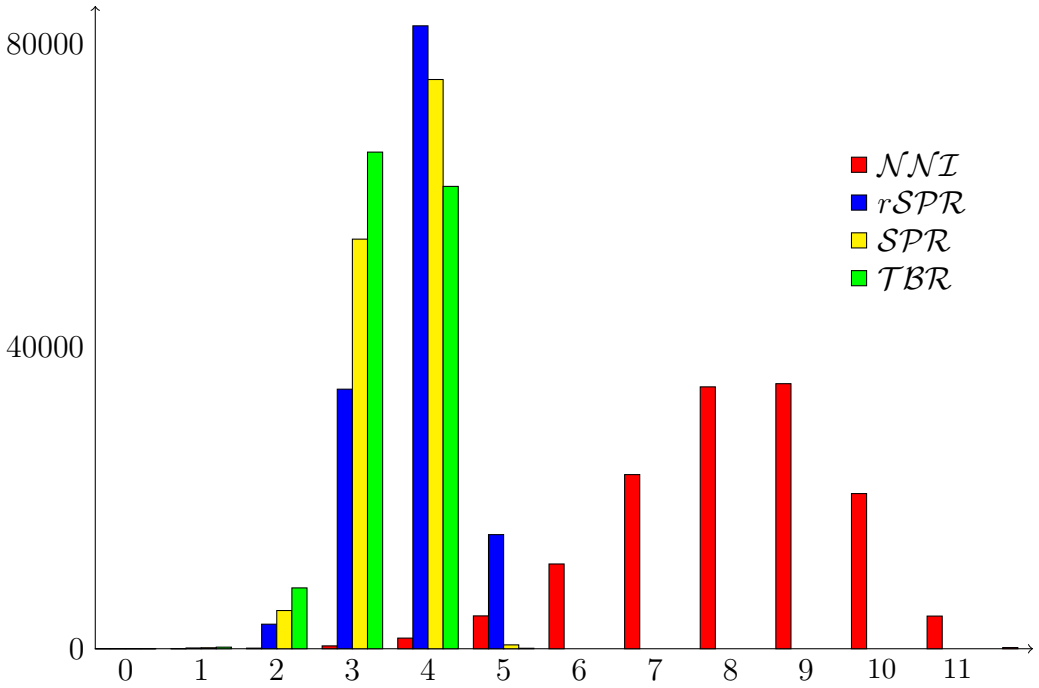


Figure A.1: The number of trees in $\mathcal{B}(9)$ vs. distance from $\text{cat}(1, 2, \ldots, 8)$.

```
def putRootedTreeInSF(T):
    if len(T)==1:
        return T
    if len(T)==4:
        L = T[1:2]
        R = T[2:3]
        if int(R)<int(L):
            return "("+R+L+")"
        return T
    x = getRightBracket(T,1)
    L = putRootedTreeInSF(T[1:x+1])
    R = putRootedTreeInSF(T[x+1:-1])
    if subtreeCompare(L,R):
        return "("+L+R+")"
    return "("+R+L+")"

def characterScore(C):
        if C=="(":
                return 10
        elif C==")":
                return 11
        else:
                return int(C)
```

```
def subtreeCompare(A,B):
        n = len(A)
        if n<len(B):
                return True
        if n>len(B):
                return False
        i=0
        while A[i]==B[i]:
        i+=1
        return characterScore(A[i])<characterScore(B[i])

def rootAtZero(T):
    if T[1]=="0":
        return T
    leftBrace = []
    i = 0
    while T[i]!="0":
        if T[i]=="(":
            leftBrace.append(i)
        elif T[i]==")":
            leftBrace.pop()
        i+=1
    depth = len(leftBrace)
    rightBrace = [0]*depth
    leftBrace.append(i)
    rightBrace.append(i)
    i+=1
    todo = depth-1
    while todo>=0:
        if T[i]=="(":
            depth+=1
        elif T[i]==")":
            depth-=1
            if depth==todo:
                rightBrace[todo]=i
                todo-=1
        i+=1
    if leftBrace[0]==leftBrace[1]-1:
        S = T[rightBrace[1]+1:rightBrace[0]]
    else:
        S = T[leftBrace[0]+1:leftBrace[1]]
    for n in range(1,len(leftBrace)-1):
        if leftBrace[n]==leftBrace[n+1]-1:
            S = "("+T[rightBrace[n+1]+1:rightBrace[n]]+S+")"
```

```
        else:
            S = "("+T[leftBrace[n]+1:leftBrace[n+1]]+S+")"
    return "(0"+S+")"

def getTreeBisections(T):
    ret = [["0",T[2:-1]]]
    for i in range(2,len(T)):
        if T[i]=="(":
            j = getRightBracket(T,i+1)
            k = getRightBracket(T,i)
            A = T[:i]
            B = T[i+1:j+1]
            C = T[j+1:k]
            D = T[k+1:]
            ret.append([A+B+D,C,len(A),len(B)])
            ret.append([A+C+D,B,len(A),len(C)])
    return ret

def getAllRerootings(T):
    if len(T)<=4:
        return [T]
    ret = getRightRerootings(T)
    x = getRightBracket(T,1)
    S = "("+T[x+1:-1]+T[1:x+1]+")"
    ret += getRightRerootings(S)
    return ret[:-1]

def getRightRerootings(T):
    x = getRightBracket(T,1)+1
    if x==len(T)-2:
        return [T]
    y = getRightBracket(T,x+1)+1
    A = T[1:x]
    B = T[x+1:y]
    C = T[y:-2]
    ret = getRightRerootings("(("+A+B+")"+C+")")
    ret += getRightRerootings("(("+A+C+")"+B+")")
    ret.append(T)
    return ret

def getNNIrerootings(T):
    ret = [T]
    if len(T)<=4:
        return ret
    x = getRightBracket(T,1)+1
```

```
    y = getRightBracket(T,2)+1
    z = getRightBracket(T,x+1)+1
    if x>2:
        A = T[2:y]
        B = T[y:x-1]
        C = T[x:-1]
        ret.append("("+A+"("+B+C+"))")
        ret.append("(("+A+C+")"+B+")")
    if x<len(T)-2:
        A = T[1:x]
        B = T[x+1:z]
        C = T[z:-2]
        ret.append("(("+A+B+")"+C+")")
        ret.append("(("+A+C+")"+B+")")
    return ret

def getTBRneighbourhood(T):
    ret = set()
    S = getTreeBisections(T)
    for s in S:
        A = getAllRerootings(s[0])
        B = getAllRerootings(s[1])
        for a in A:
            for b in B:
                ret.add(sf("("+a+b+")"))
    return ret

def getRSPRneighbourhood(T):
    ret = set()
    S = getTreeBisections(T)
    for s in S:
        for a in getAllRerootings(s[0]):
            ret.add(sf("("+a+s[1]+")"))
    return ret

def getNNIneighbourhood(T):
    ret = set()
    S = getTreeBisections(T)
    for s in S:
        if len(s[0])>1:
            A = s[0][:s[2]]
            B = s[0][s[2]:s[2]+s[3]]
            D = s[0][s[2]+s[3]:]
            for C in getNNIrerootings(s[1]):
                ret.add(sf(A+"("+B+C+")"+D))
```

```
        else:
            for C in getNNIrerootings(s[1]):
                ret.add(sf("(0"+C+")"))
    return ret

def getSPRneighbourhood(T):
    ret = set()
    S = getTreeBisections(T)
    for s in S:
        for a in getAllRerootings(s[0]):
            ret.add(sf("("+a+s[1]+")"))
        if len(s[0])>1:
            A = s[0][:s[2]]
            B = s[0][s[2]:s[2]+s[3]]
            D = s[0][s[2]+s[3]:]
            for C in getAllRerootings(s[1]):
                ret.add(sf(A+"("+B+C+")"+D))
        else:
            for C in getAllRerootings(s[1]):
                ret.add(sf("(0"+C+")"))
    return ret

def getDistribution(T,move):
    NNI = False
    TBR = False
    SPR = False
    if move=="NNI":
        NNI = True
    elif move=="TBR":
        TBR = True
    elif move=="SPR":
        SPR = True
    nTrees = 1
        for i in range(len(T)//3-1):
            nTrees *= 2*i+1
    interval = 3600
    found = set()
    found.add(T)
    ret = []
    N = list(found)
    while len(N)>0:
        oldFound = found.copy()
        ret.append(len(N))
        for t in N:
            if NNI:
```

```
                found = found.union(getNNIneighbourhood(t))
            elif TBR:
                found = found.union(getTBRneighbourhood(t))
            elif SPR:
                found = found.union(getSPRneighbourhood(t))
            else:
                found = found.union(getRSPRneighbourhood(t))
            if len(found)==nTrees:
                break
            N = list(found.difference(oldFound))
    return ret

def getAllDistributions(T):
    print("Tree = "+T)
    dist = getDistribution(T,"NNI")
    print("  NNI-Distribution = "+str(dist))
    dist = getDistribution(T,"rSPR")
    print(" rSPR-Distribution = "+str(dist))
    dist = getDistribution(T,"SPR")
    print("  SPR-Distribution = "+str(dist))
    dist = getDistribution(T,"TBR")
    print("  TBR-Distribution = "+str(dist)+"\n")

def getRootedDistributions(T):
    print("Tree = "+T)
    dist = getDistribution(T,"rSPR")
    print(" rSPR-Distribution = "+str(dist)+"\n")

def getUnrootedDistributions(T):
    print("Tree = "+T)
    dist = getDistribution(T,"NNI")
    print("  NNI-Distribution = "+str(dist))
    dist = getDistribution(T,"SPR")
    print("  SPR-Distribution = "+str(dist))
    dist = getDistribution(T,"TBR")
    print("  TBR-Distribution = "+str(dist)+"\n")

CAT4 = "(0(1(23)))"
CAT5 = "(0(1(2(34))))"
CAT6 = "(0(1(2(3(45)))))"
CAT7 = "(0(1(2(3(4(56))))))"
CAT8 = "(0(1(2(3(4(5(67)))))))"
CAT9 = "(0(1(2(3(4(5(6(78))))))))"
CAT10 = "(0(1(2(3(4(5(6(7(89)))))))))"
```

```
CATERPILLARS = [CAT4,CAT5,CAT6,CAT7,CAT8]

BAL4 = "(0(1(23)))"
BAL5 = "(0((12)(34)))"
BAL8 = "(0(1((23)((45)(67)))))"
BAL9 = "(0(((12)(34))((56)(78))))"

BALANCED_TREES = [BAL4,BAL5,BAL8]

T9a = "(0(1(2(3(4((56)(78)))))))"
T9b = "(0(1(2(3((45)(6(78)))))))"
T9c = "(0(1(2((34)((56)(78))))))"
T9d = "(0(1(2((3(45))(6(78))))))"

U_NINE = [CAT9,T9a,T9b,T9c,T9d,BAL9]

CAT9A = sf("(0(1(2(3(4(5(6(78)))))))")
CAT9B = sf("(2(1(0(3(4(5(6(78)))))))")
CAT9C = sf("(3(1(2(0(4(5(6(78)))))))")
CAT9D = sf("(4(1(2(3(0(5(6(78)))))))")
T9aA = sf("(0(1(2(3(4((56)(78)))))))")
T9aB = sf("(2(1(0(3(4((56)(78)))))))")
T9aC = sf("(3(1(2(0(4((56)(78)))))))")
T9aD = sf("(4(1(2(3(0((56)(78)))))))")
T9aE = sf("(5(1(2(3(4((06)(78)))))))")
T9bA = sf("(0(1(2(3((45)(6(78)))))))")
T9bB = sf("(2(1(0(3((45)(6(78)))))))")
T9bC = sf("(3(1(2(0((45)(6(78)))))))")
T9bD = sf("(4(1(2(3((05)(6(78)))))))")
T9bE = sf("(6(1(2(3((45)(0(78)))))))")
T9bF = sf("(7(1(2(3((45)(6(08)))))))")
T9cA = sf("(0(1(2((34)((56)(78))))))")
T9cB = sf("(2(1(0((34)((56)(78))))))")
T9cC = sf("(3(1(2((04)((56)(78))))))")
T9cD = sf("(5(1(2((34)((06)(78))))))")
T9dA = sf("(0(1(2((3(45))(6(78))))))")
T9dB = sf("(2(1(0((3(45))(6(78))))))")
BAL9A = sf("(0(((12)(34))((56)(78))))")
BAL9B = sf("(1(((02)(34))((56)(78))))")

R_NINE = [CAT9A,CAT9B,CAT9C,CAT9D,T9aA,T9aB,T9aC,T9aD]
R_NINE += [T9aE,T9bA,T9bB,T9bC,T9bD,T9bE,T9bF]
R_NINE += [T9cA,T9cB,T9cC,T9cD,T9dA,T9dB,BAL9A,BAL9B]

def main():
```

```
print("\n ------- BALANCED TREES ------- \n")
for t in BALANCED_TREES:
    getAllDistributions(t)
print("\n ------- CATERPILLAR TREES ------- \n")
for t in CATERPILLARS:
    getAllDistributions(t)
print("\n ------- UNROOTED NINE LEAFED TREES ------- \n")
for t in U_NINE:
    getUnrootedDistributions(t)
print("\n ------- ROOTED NINE LEAFED TREES ------- \n")
for t in R_NINE:
    getRootedDistributions(t)
```

The Output

```
 ------- BALANCED TREES -------

Tree = (0(1(23)))
  NNI-Distribution = [1, 2]
 rSPR-Distribution = [1, 2]
  SPR-Distribution = [1, 2]
  TBR-Distribution = [1, 2]

Tree = (0((12)(34)))
  NNI-Distribution = [1, 4, 8, 2]
 rSPR-Distribution = [1, 12, 2]
  SPR-Distribution = [1, 12, 2]
  TBR-Distribution = [1, 12, 2]

Tree = (0(1((23)((45)(67)))))
  NNI-Distribution = [1, 10, 64, 268, 752, 1648, 2516,
                      2672, 2192, 272]
 rSPR-Distribution = [1, 78, 1496, 6356, 2464]
  SPR-Distribution = [1, 90, 1868, 6976, 1460]
  TBR-Distribution = [1, 106, 2060, 6836, 1392]

 ------- CATERPILLAR TREES -------

Tree = (0(1(23)))
  NNI-Distribution = [1, 2]
 rSPR-Distribution = [1, 2]
  SPR-Distribution = [1, 2]
  TBR-Distribution = [1, 2]
```

```
Tree = (0(1(2(34))))
  NNI-Distribution = [1, 4, 8, 2]
 rSPR-Distribution = [1, 10, 4]
  SPR-Distribution = [1, 12, 2]
  TBR-Distribution = [1, 12, 2]

Tree = (0(1(2(3(45)))))
  NNI-Distribution = [1, 6, 20, 40, 34, 4]
 rSPR-Distribution = [1, 24, 72, 8]
  SPR-Distribution = [1, 30, 74]
  TBR-Distribution = [1, 34, 70]

Tree = (0(1(2(3(4(56))))))
  NNI-Distribution = [1, 8, 36, 110, 244, 328, 198, 20]
 rSPR-Distribution = [1, 44, 410, 478, 12]
  SPR-Distribution = [1, 56, 558, 330]
  TBR-Distribution = [1, 72, 598, 274]

Tree = (0(1(2(3(4(5(67)))))))
  NNI-Distribution = [1, 10, 56, 220, 670, 1616, 2810,
                      3064, 1708, 236, 4]
 rSPR-Distribution = [1, 70, 1328, 6082, 2908, 6]
  SPR-Distribution = [1, 90, 1978, 7486, 840]
  TBR-Distribution = [1, 130, 2598, 7218, 448]

------- UNROOTED NINE LEAFED TREES -------

Tree = (0(1(2(3(4(5(6(78))))))))
  NNI-Distribution = [1, 12, 80, 378, 1408, 4344, 11210,
                      23028, 34630, 35050, 20518, 4320, 156]
  SPR-Distribution = [1, 132, 5054, 54164, 75272, 512]
  TBR-Distribution = [1, 212, 8046, 65672, 61140, 64]

Tree = (0(1(2(3(4((56)(78)))))))
  NNI-Distribution = [1, 12, 84, 404, 1520, 4688, 11546,
                      22420, 33584, 34748, 20832, 5104, 192]
  SPR-Distribution = [1, 132, 4980, 52090, 77244, 688]
  TBR-Distribution = [1, 196, 7376, 62222, 65132, 208]

Tree = (0(1(2(3((45)(6(78)))))))
  NNI-Distribution = [1, 12, 84, 416, 1586, 4796, 11548,
                      22188, 32688, 34588, 21936, 5100, 192]
  SPR-Distribution = [1, 132, 4930, 51404, 77620, 1048]
  TBR-Distribution = [1, 188, 7218, 60284, 66924, 520]
```

```
Tree = (0(1(2((34)((56)(78))))))
  NNI-Distribution = [1, 12, 88, 454, 1724, 5096, 11864,
                      21808, 30520, 33200, 24624, 5712, 32]
  SPR-Distribution = [1, 132, 4850, 49068, 78556, 2528]
  TBR-Distribution = [1, 172, 6386, 55660, 70900, 2016]

Tree = (0(1(2((3(45))(6(78))))))
  NNI-Distribution = [1, 12, 84, 428, 1652, 4920, 11550,
                      21752, 32088, 34372, 22804, 5320, 152]
  SPR-Distribution = [1, 132, 4880, 50730, 77216, 2176]
  TBR-Distribution = [1, 180, 6964, 57562, 68396, 2032]

Tree = (0(((12)(34))((56)(78))))
  NNI-Distribution = [1, 12, 88, 430, 1688, 4912, 11844,
                      22352, 31616, 34224, 22368, 5248, 352]
  SPR-Distribution = [1, 132, 4910, 50424, 78644, 1024]
  TBR-Distribution = [1, 180, 6742, 59312, 68516, 384]
```

——— ROOTED NINE LEAFED TREES ———

```
Tree = (0(1(2(3(4(5(6(78))))))))
 rSPR-Distribution = [1, 102, 3248, 34324, 82370, 15090]

Tree = (0((12)(3(4(5(6(78)))))))
 rSPR-Distribution = [1, 112, 3744, 39232, 82494, 9552]

Tree = (0((2(13))(4(5(6(78))))))
 rSPR-Distribution = [1, 118, 4024, 41782, 82370, 6840]

Tree = (0((3(2(14)))(5(6(78)))))
 rSPR-Distribution = [1, 120, 4118, 42568, 80920, 7408]

Tree = (0(1(2(3(4((56)(78)))))))
 rSPR-Distribution = [1, 104, 3300, 34386, 82408, 14936]

Tree = (0((12)(3(4((56)(78))))))
 rSPR-Distribution = [1, 114, 3822, 39414, 82352, 9432]

Tree = (0((2(13))(4((56)(78)))))
 rSPR-Distribution = [1, 120, 4128, 42366, 82856, 5664]

Tree = (0((3(2(14)))((56)(78))))
 rSPR-Distribution = [1, 122, 4246, 44054, 82920, 3792]

Tree = (0(6((78)(4(3(2(15)))))))
```

```
 rSPR-Distribution = [1, 110, 3592, 36948, 82334, 12150]

Tree = (0(1(2(3((45)(6(78)))))))
 rSPR-Distribution = [1, 106, 3362, 34612, 81464, 15590]

Tree = (0((12)(3((45)(6(78))))))
 rSPR-Distribution = [1, 116, 3910, 39974, 81870, 9264]

Tree = (0((2(13))((45)(6(78)))))
 rSPR-Distribution = [1, 122, 4240, 43712, 82972, 4088]

Tree = (0(5((6(78))(3(2(14))))))
 rSPR-Distribution = [1, 114, 3744, 38008, 81980, 11288]

Tree = (0((78)((45)(3(2(16))))))
 rSPR-Distribution = [1, 118, 4032, 41474, 81298, 8212]

Tree = (0(8(6((45)(3(2(17)))))))
 rSPR-Distribution = [1, 108, 3460, 35388, 81444, 14734]

Tree = (0(1(2((34)((56)(78))))))
 rSPR-Distribution = [1, 110, 3528, 35508, 81484, 14504]

Tree = (0((12)((34)((56)(78)))))
 rSPR-Distribution = [1, 120, 4126, 42048, 80760, 8080]

Tree = (0(4((2(13))((56)(78)))))
 rSPR-Distribution = [1, 116, 3838, 38668, 83344, 9168]

Tree = (0(6((78)((34)(2(15))))))
 rSPR-Distribution = [1, 114, 3748, 37696, 82176, 11400]

Tree = (0(1(2((3(45))(6(78))))))
 rSPR-Distribution = [1, 110, 3518, 35734, 80860, 14912]

Tree = (0((12)((3(45))(6(78)))))
 rSPR-Distribution = [1, 120, 4116, 42190, 82212, 6496]

Tree = (0(((12)(34))((56)(78))))
 rSPR-Distribution = [1, 124, 4374, 45704, 83396, 1536]

Tree = (0(2((34)(1((56)(78))))))
 rSPR-Distribution = [1, 112, 3670, 37160, 82608, 11584]
```

Bibliography

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, 2000.

[2] David Aldous. Probability distributions on cladograms. *Springer (IMA Volumes Math Appl.*, 76:1–18, 1996.

[3] Benjamin Allen and Mike Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Ann. Comb.*, 5:1–15, 2001.

[4] Noga Alon, Avinatan Hasidim, Eyal Lubetzky, Uri Stav, and Amit Weinstein. Broadcasting with side information. *arXiv preprint arXiv:0806.3246*, 2008.

[5] Ross Atkins. The information flow problem on clock networks. *arXiv:1605.05391*, 2016.

[6] Ross Atkins and Colin McDiarmid. Extremal distances for subtree transfer operations in binary treess. *arXiv:1509.00669*, 2016.

[7] Ross Atkins, Puck Rombach, and Fiona Skerman. Guessing numbers of odd cycles. *arXiv:1602.03586*, 2016.

[8] Rahil Baber, Demetres Christofides, Anh N. Dang, Søren Riis, and Emil R. Vaughan. Graph guessing games and non-shannon information inequalities. *arXiv:1410.8349*, 2014.

[9] Zic Bar-Yossef, Yitzhak Birk, T.S. Jayram, and Tomer Kol. Index coding with side information. *Information Theory, IEEE Transactions on*, 57(3):1479–1494, 2011.

[10] Daniel Irving Bernstein, Lam Si Tung Ho, Colby Long, Mike Steel, Katherine st. John, and Seth Sullivant. Bounds on the expected size of the maximum agreement subtree. *SIAM J. Discrete Math.*, 29:2065–2074, 2014.

[11] Magnus Bordewich and Charles Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Ann. Comb.*, 8:409–423, 2005.

[12] Steve Butler, Mohammad T. Hajiaghayi, Robert D. Kleinberg, and Tom Leighton. Hat guessing games. *SIAM Review*, 51(2):399–413, 2009.

[13] Peter J. Cameron, Anh N. Dang, and Søren Riis. Guessing games on triangle-free graphs. *arXiv preprint arXiv:1410.2405*, 2014.

[14] Jullian Cannons, Randall Dougherty, Chris Freiling, and Kenneth Zeger. Network routing capacity. *IEEE Trans. Inform. Theory*, 52:777–788, 2006.

[15] Gabriel Cardona, Francesc Rossello, and Gabriel G. Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):552–569, 2009.

[16] Gerard Jennhwa Chang, Keqin Feng, Liang-Hao Huang, and Mei Lu. The linear guessing number of undirected graphs. *Linear Algebra and its Applications*, 449:119–131, 2014.

[17] Demetres Christofides and Klas Markström. The guessing number of undirected graphs. *The Electronic Journal of Combinatorics*, 18(1):P192, 2011.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, Hoboken, NJ, 2006.

[19] Karel Culik and Derick Wood. A note on some tree similarity measures. *Inf. Proc. Letts.*, 15:39–42, 1982.

[20] William H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2(1):7–28, 1985.

[21] Yang Ding, Stefan Grünewald, and Peter J Humphries. On agreement forests. *J. Comb. Theory*, 7:2059–2065, 2011.

[22] Randall Dougherty and Kenneth Zeger. Nonreversibility and equivalent constructions of multiple unicast networks. *IEEE Trans. Inf. Theory*, 52:1287–1291, 2006.

[23] Todd Ebert. *Applications of recursive operators to randomness and complexity*. PhD thesis, University of California Santa Barbara, 1998.

[24] Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Composition Mathematica 2*, pages 463–470, 1935.

[25] L.R. Foulds and D.F. Robinson. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981.

[26] Maximilien Gadouleau and Søren Riis. Graph-theoretical constructions for graph entropy and network coding based communications. *Information Theory, IEEE Transactions on*, 57(10):6703–6717, 2011.

[27] Pablo Golobof. Calculating spr distances between trees. *Cladistics*, 24(4):591–597, 2008.

[28] A. D. Gordon. On the assessment and comparison of classifications. *Analyse de Donndes et lnformatique*, pages 149–160, 1980.

[29] Dan Gusfield. *Recombinatorics*. The MIT Press, Cambridge, Massachusetts, 2014.

[30] Jotun Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.*, 98:185–200, 1990.

[31] Jotun Hein, Tao Jiang, Lusheng Wang, and Kaizhong Zhang. On the complexity of comparing evolutionary trees,. *Discrete Appl. Math.*, 71:153–169, 1996.

[32] Peter J Humphreys and Charles Semple. Note on the hybridization number and subtree distance in phylogenetics. *Applied Mathematics Letters*, 22:611–615, 2009.

[33] Leo Van Iersel, Steven Kelk, Nela Lekic, and Simone Linz. Satisfying ternary permutation constraints by multiple linear orders or phylogenetic trees. *arXiv:1503.00368*, 2014.

[34] Ming Li, John Tromp, and Louxin Zhang. On the nearest neighbour interchange distance between evolutionary trees. *Journal of Theoretical Biology*, 182(4):463–467, 1996.

[35] Shuo-Yen Robert Li. Unachievability of network coding capacity. *IEEE Trans. Inf. Theory*, 52:2365–2372, 2006.

[36] Daniel M. Martin and Bhalchandra D. Thatte. The maximum agreement subtree problem. *Disc. Appl. Math.*, 161:1805–1817, 2013.

[37] Colin McDiarmid. On the method of bounded differences. *Surveys in Combinatorics, J. Siemons ed., London Mathematical Society Lecture Note Series 141, Cambridge University Press*, pages 148–188, 1989.

[38] Maura B. Paterson and Douglas R. Stinson. Yet another hat game. *The Electronic Journal of Combinatorics 17*, 2010.

[39] Nicholas D. Pattengale, Eric J. Gottlieb, and Bernard M.E. Moret. Efficiently computing the robinson-foulds metric. *Journal of Computational Biology*, 14(6):724–735, 2007.

[40] David Penny and Mike Steel. Distributions of tree comparison metrics – some new results. *Syst. Biol.*, 42:126–141, 1993.

[41] Søren Riis. Utilising public information in network coding. *General Theory of Information Transfer and Combinatorics*, pages 861–897, 2006.

[42] Søren Riis. Graph entropy, network coding and guessing games. *arXiv preprint arXiv:0711.4175*, 2007.

[43] Søren Riis. Information flows, graphs and their guessing numbers. *Electronic Journal of Combinatorics. 14*, pages 44–61, 2007.

[44] Donn Erik Rosen. Vicariant patterns and historical explanation in biogeography. *Systematic Zoology*, 27:159–188, 1978.

[45] Charles Semple and Mike A. Steel. *Phylogenetics*, volume 24. Oxford University Press on Demand, 2003.

[46] Daniel Sleator, Robert Tarjan, and William Thurston. Short encodings of evolving structures. *SIAM J. Discrete Math.*, 5:428–450, 1992.

[47] Mike Steel and Andy McKenzie. Properties of phylogenetic trees generated by yule-type speciation models. *Mathematical Biosciences*, 170:91–112, 2001.

[48] Mike Steel and László A. Székely. An improved bound on the maximum agreement subtree problem. *Applied Mathematics Letters*, 22(11):1778–1780, 2009.

[49] Mike Steel and Tandy Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. *Information Processing Letters*, 48(2):77–82, 1993.

[50] David Swofford. Paup*, phylogenetic analysis using parsimony (*and other methods). *Sinauer Associates, Sunderland MA*, 1998.

[51] Stephen J. Willson. Properties of normal phylogenetic networks. *Bulletin of Mathematical Biology*, 72(2):340–358, 2010.

[52] Taoyang Wu, Peter Cameron, and Søren Riis. On the guessing number of shift graphs. *Journal of Discrete Algorithms*, 7(2):220–226, 2009.

[53] Derrick Zwickl. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion, 2006.