

Likelihood Maximization on Phylogenetic Trees



James Wood
Hertford College
University of Oxford

Submitted in partial fulfilment of the
requirements for the degree of

*M.Sc. in Mathematical
Modelling and Scientific Computing*

September 2010

Abstract

We consider the problem of reconstructing the root ancestral state for a binary character on a fixed-topology binary phylogenetic tree, and compare the methods of maximum parsimony and maximum likelihood with the goal of checking if the methods are in agreement. For the likelihood method, we consider the problem under the Neyman N_2 model with a molecular clock constraint.

We show that finding the maximum likelihood assignment requires solving a constrained polynomial optimization problem. We review recently developed techniques of polynomial optimization, exploring some theoretical and practical aspects. Using these optimization techniques, we are able to find, within a small numerical error, the optimal likelihood values and certify the global optimality, thereby allowing us to reconstruct the ancestral state.

Using both methods, we solve the reconstruction problem on all four-leaf and five-leaf topologies. We find that the two methods agree on all four-leaf topologies, but there are five-leaf topologies on which they do not. It has previously been shown that without a molecular clock constraint, the methods are in agreement, but our example contributes to the knowledge of these methods by showing this is not the case with a molecular clock. This settles, by example, a problem in the field of phylogenetics.

Acknowledgements

I would like to thank Dr Raphael Hauser and Dr Bhalchandra Thatte for their help and supervision. Dr Thatte provided me with the problem for this dissertation, and Dr Hauser provided guidance in solving the optimization problems. Both were very patient with me starting out completely green when it came to the topic of optimization. I've learned a great deal in the process, and could not have done this without their help. I would also like to thank my friends and family for all their support, encouragement, and help that has come in various forms.

Contents

1	Introduction	1
2	Background	3
2.1	Trees and Phylogenesis	3
2.2	Definitions	5
2.3	Our Problem	5
3	Reconstruction of Ancestral States	7
3.1	Maximum Parsimony Method	7
3.1.1	Maximum Parsimony Algorithm	8
3.2	Maximum Likelihood Method	9
3.2.1	Markov Processes on Trees	9
3.2.1.1	Symmetric N_r Model	10
3.2.2	Molecular Clock Constraint	11
3.2.3	Maximum Likelihood for Ancestral States	13
3.2.4	Likelihood Maximization as an Optimization Problem	13
4	Polynomial Optimization	16
4.1	Polynomials and Sum of Squares	16
4.2	Semidefinite Programming	17
4.3	SDP and SOS Polynomials	19
4.4	The Generalized Problem of Moments	21
4.4.1	Unconstrained Programs	24
4.4.2	Inequality Constrained Case	27
5	Computing Solutions to Polynomial Optimization Problems	32
5.1	Computational Complexity	32
5.1.1	Sparsity Techniques	33
5.2	SDP Solvers	34
5.2.1	Interior Point Solvers	34
5.2.1.1	Newton's Method	35

5.2.1.2	Barrier/Interior-Point Methods	36
5.3	Computing Solutions to POPs	38
5.3.1	Measuring Solution Accuracy	38
5.3.2	Understanding Suboptimal Solutions	39
5.3.3	Equality Constraints	40
5.3.4	Multiple Optima	41
5.3.5	Improving The Estimator	43
6	ML Ancestral State Reconstruction	44
6.1	Results	45
6.1.1	How Results Are Presented	45
6.1.2	Computed Results	45
6.1.3	Discussion	46
6.1.4	Optimizing for Other Models and Trees	47
7	Conclusion	56
	Bibliography	57
	A Complete Results	61
	B Detailed SeDuMi Output For Examples on Which ML and MP Dis-	
	agree	66

Chapter 1

Introduction

Phylogenies are the evolutionary relations used to understand development and diversification of species, and serve purpose in biology, epidemiology, vaccine development, and disease treatment. Phylogenetics, the construction of these trees and inference of these relationships, has been a widely research field, with applications and many methods studied.

Evolution can be regarded as a branching, or speciation, process over time, where there is commonality in ancestry, and where new species are created and eliminated. Phylogenies help the understanding of when and what these branchings were, how similar species are, what changes have occurred, and various characteristics of the species.

A common problem in phylogenetics is inferring a characteristic of common ancestors, sometimes called reconstruction of the ancestral state. In other words, the problem is to decide, given a set of species that we observe, what properties a common ancestor would have had. This problem is solved in practice to discover various characteristics of past species and to test evolutionary hypotheses.

Uses of these ancestral reconstructions are many. Some examples include testing taxonomic classifications [21], recreation of ancestral proteins for lab testing [2], and understanding the development of antiretroviral proteins [7]. Certainly, the uses are varied — but the inferences from these reconstructions are limited by the methods used to perform these reconstructions, and there is therefore interest in the theoretical performance of the methods.

In a reconstruction problem data will generally only exist for extant species, and since we lack the data for ancestors it is a challenge to infer a complete and true phylogenesis. There are different models and methods for inferring phylogenetic information, and every method has a way to search among possible the possible relations and some way to judge the most fitting or in some sense best phylogenetic tree.

Two popular classes of methods used on phylogenetic trees are maximum parsimony (MP) and maximum likelihood (ML) methods. MP methods attempt to find the most parsimonious relationship with respect to evolution, or, in other words, simply minimize the number of evolutionary changes that would be required over time in order to obtain the observed data. MP methods are conceptually simple, computationally tractable for the reconstruction problem, and is a widely used method.

ML methods assume a probabilistic model for evolution, and attempt to find the relationships and parameters which maximize the probability of observing the data under the model, and reconstructs the ancestral state in a way which maximizes this probability.

In general, however, finding the parameters which maximize the likelihood is almost always a difficult problem for which there are no known methods to find the global maximizers. Commonly for these methods, heuristic searches are used to find the relationships which might maximize the likelihood, but are not guaranteed to find the globally optimal solution and cannot be certain if the optimum has been found.

Often, small trees of only a few species are used as conceptual examples to understand the methods, but even these simple examples can elucidate the performance of the methods. We examine single characteristics evolving on a phylogeny in our analysis. We apply a recently developed set of polynomial optimization techniques to solve the for ML ancestral reconstruction. We have the goal of checking if the methods of MP and ML will be in agreement, and this global optimization method allows us to do this.

Overview We begin with a brief introduction to phylogenies, and explain the methods of MP and ML. In order to solve the ML problem, we will introduce it as a polynomial optimization problem. We will present tools for polynomial optimization, and explore the theory behind them and some aspects of using them in practice. Finally, we will apply them to solve the maximization problem on four and five-leaf topologies, and present examples where the MP and ML methods do not agree.

Chapter 2

Background

2.1 Trees and Phylogenesis

A basis for evolution is that there is commonality in the ancestry of species and taxa, and that the characteristics of living organisms are developed from earlier forms. A phylogenetic tree is a way of representing the relationships we infer about the organisms and their ancestry.

On a phylogenetic tree we can represent things like speciation events, commonality of ancestry, and evolutionary similarity.

To illustrate some of these ideas, we will present some simple examples on primates, using phylogenetic relationships from [18]. If we are interested in showing a basic relationship between five species of primates and wish to represent how close they are evolutionarily, but are not interested in representing details of ancestral states, we might end up with an unrooted tree such as that shown in Figure 2.1. In

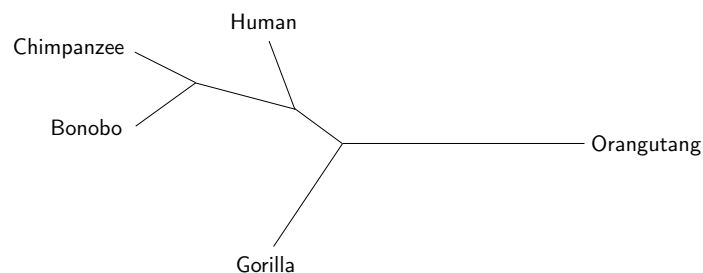


Figure 2.1: An unrooted phylogenetic tree showing a simple relationship of similarity, but not showing ancestry. Colloquial species names are shown here, but the correct binominal nomenclature is shown in Figure 2.2.

Figure 2.1, we can see that there is neither a direction of evolution and time, nor is a specific common ancestor explicitly implied.

If we wish to also represent ancestry and evolutionary time, we can use a rooted tree. In this case, we root the tree at ρ , and think of the edges directed away from the root of our tree, with the root being an earlier time period.

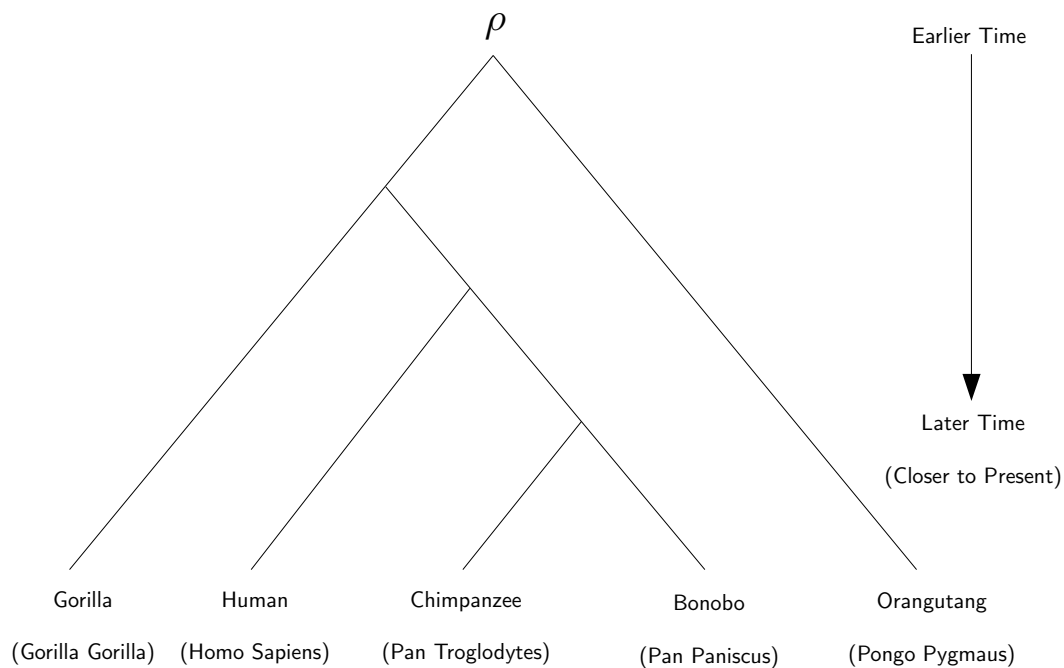


Figure 2.2: A rooted phylogenetic tree showing a relationship of similarity, and implying a single common ancestor ρ .

2.2 Definitions

We will begin with some definitions to formalize our phylogenetic trees mathematically. Note that we do not make extensive use of the definitions for graphs and a reader is not likely to find them enlightening, but they are provided for clarity of the model presented and the discussion in Chapter 3. Further, we make use of natural concepts, for example a “left” and “right” subtree on a binary tree, for which we do not provide a mathematical definition, and instead we assume is clear for our purposes.

Graphs We will only consider directed graphs. A graph G is an ordered pair (V, E) consisting of a non-empty set of vertices (or nodes) V and set of edges E . Each element of E is defined for $u, v \in V$ as the ordered pairs (u, v) representing a directed edge from vertices u to $v \in V$.

A path of length k on a tree is a sequence of vertices v_1, \dots, v_k such that for $1 \leq i \leq k - 1$, and the edges along the path are $(v_i, v_{i+1}) \in E$. A tree is a graph for which no path exists with the same beginning and ending vertex.

The degree of a vertex $u \in V$ is the cardinality of the set $\{(u, v) : (u, v) \in E, \forall v \in V\}$. A rooted binary tree is a tree with a single node designated as the root, for which there exists a path beginning at ρ and ending at v for any $v \in V$, and for which every node has degree 0 or degree 2. On a rooted binary tree with root ρ , we call ρ the common ancestor.

Characters For our purposes, a character χ on a tree \mathcal{T} is a map $\chi : X \rightarrow \mathcal{C}$, where $X \subset V$ is the set of leaves, and where \mathcal{C} is the set of character states. It is an r -state character if $|\chi(X)| = r$. When $r = 2$ we call it a binary character.

An extension of a character $\chi : X \rightarrow \mathcal{C}$ is defined as $\bar{\chi} : V \rightarrow \mathcal{C}$ satisfying $\chi(u) = \bar{\chi}(u)$ for $u \in X$. Put simply, a character is defined for the leaves of a tree and an extension of this character extends the mapping from all interior vertices of the tree onto the character set \mathcal{C} , but preserves the mapping on the leaves of the tree.

We will write the notation $\bar{\chi}_{\rho=\alpha}$ to mean an $\bar{\chi}$ is some extension of a character χ that assigns to the root vertex ρ the state $\alpha \in \mathcal{C}$.

2.3 Our Problem

Characters represent the properties of the species that correspond to the nodes on a phylogenetic tree. The leaves of the tree represent extant species, and internal nodes

represent ancestors. For example, characters may be anything from properties about diet to whether a particular protein is produced by the species.

The problem we are considering is the inference, or reconstruction, of the character state of a common ancestor on a rooted tree. We take some characteristic we wish to examine and map this as an extended character onto the tree. In doing so, we estimate the state of this character for a common ancestor.

We illustrate the concept on a simple hypothetical example. Consider the tree shown in Figure 2.3. Suppose we wish to find the diets, and we assume species have either frugivorous or folivorous diets. We can observe the diet for three extant taxa, and we then wish to infer what the diet of the common ancestor ρ is.

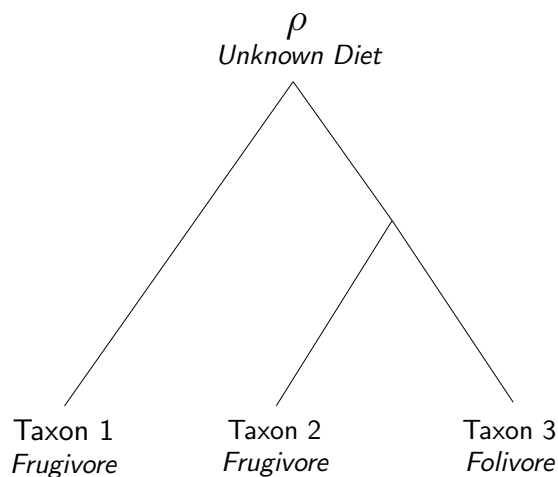


Figure 2.3: A rooted phylogenetic tree showing a common ancestor ρ , three extant species, and a character representing diet.

The idea of reconstruction is itself conceptually simple, but it is not immediately clear what the best way to do this is. One needs to formalize and understand a method to perform this reconstruction.

We will be concerned with two methods to solve for these ancestral states under a particular constraint known as the *molecular clock*, which we will explain in a later section. We make simplifying assumptions for the problems when we consider the reconstruction. First, we assume that our phylogenetic relationship is known, or, in other words, that the tree being used is the true tree (the topology is correct), and that we are considering all relevant taxa in the reconstruction. Second, we assume the phylogenetic tree is a rooted binary tree with root ρ , and we are reconstructing a full character. These assumptions mean that we are solving for ancestral states on a tree which is given and complete, and we wish to see how the methods compare under this assumption.

Chapter 3

Reconstruction of Ancestral States

3.1 Maximum Parsimony Method

Parsimony methods were presented by Fitch in [5], and described as a way to “account for the descent of characteristics in a manner requiring a minimum number of evolutionary steps.” This reconstruction method is simple on a fixed topology, and as we will see, easy to compute.

We will begin by formalizing the definition. For a graph G and a function f on V , the changing set of f is the set

$$\text{Ch}(f) = \{(u, v) \in E : f(u) \neq f(v)\}.$$

The cardinality of the changing set is denoted $\text{ch}(f)$.

Now, let χ be a character on a tree \mathcal{T} . Recall that an extension preserves the original map χ on the leaves of \mathcal{T} and makes some assignment of the states to the interior nodes.

We will denote $\text{ch}(\bar{\chi})$ as the parsimony score of the extension $\bar{\chi}$ of χ onto \mathcal{T} . A map $\bar{\chi}$ which has a minimum parsimony score among all possible extensions of χ is known as a minimum extension of χ onto \mathcal{T} .

If the minimum extensions of χ all assign a particular state $\alpha \in \mathcal{C}$ to the root, then the parsimony method chooses α as the root state. It is possible that there is more than one minimal extension of a character χ with different assignments to the state at the root ρ of the tree. We then say the reconstruction is ambiguous.

The motivating idea of maximum parsimony is that if some characteristic is observed in extant species, then it is probable that it would be observed in common ancestors, so we pick this state for the ancestor.

3.1.1 Maximum Parsimony Algorithm

We provide part of Fitch's algorithm, which we can use to reconstruct the state of the root ancestor for the MP method on a rooted binary tree. We let $\mathcal{S}(v)$ be a set of character states at node v . The algorithm is simple and runs in linear time proportional to the number of leaves on the tree. We call this algorithm using the root ρ as the starting point, and it runs recursively through the tree.

```

Input: Node  $v$ 
if  $v$  is a leaf then
  |  $\mathcal{S}(v) \leftarrow \chi(v)$ ;
else
  | call algorithm for left subtree with root  $v_l$ 
  | call algorithm for right subtree with root  $v_r$ 
  | if  $\mathcal{S}(v_l) \cap \mathcal{S}(v_r) = \emptyset$  then
  | |  $\mathcal{S}(v) = \mathcal{S}(v_l) \cup \mathcal{S}(v_r)$ 
  | else
  | |  $\mathcal{S}(v) = \mathcal{S}(v_l) \cap \mathcal{S}(v_r)$ 
  | end
end

```

The value of the set $\mathcal{S}(\rho)$ is the set of all possible assignments to the root node which can achieve the lowest maximum parsimony score, and we provide Theorem 1 to show this.

Theorem 1. *The Fitch algorithm initialized with the root ρ of a binary tree produces the set $\mathcal{S}(\rho)$ if and only if there exists a minimum extension of χ which assigns a value $\alpha \in \mathcal{S}(\rho)$ to the node ρ .*

Proof. We prove this by induction. Consider a two-leaf tree with height 1 and root ρ . Then at the leaves u and v , $\chi(v)$ and $\chi(u)$ either match or do not match. If they match with value $\alpha \in \mathcal{C}$, then the unique minimum extension $\bar{\chi}$ of the character clearly assigns the value α to the root, as does the fitch algorithm. If they do not match and $\chi(v) = \alpha$, $\chi(u) = \beta$, then two minimum extensions exist with either α or β at the root state and a change on one of the edges directed from ρ . The Fitch algorithm also makes the assignment $\mathcal{S}(\rho) = \{\alpha, \beta\}$ at the root.

Now, assume the algorithm works for a tree with height h . Then, on a tree with height $h + 1$ with root ρ having left and right subtrees l and r , the elements of $\mathcal{S}(l)$ and $\mathcal{S}(r)$ generated by the algorithm can provide minimum extensions on the left and right subtrees respectively. If $\mathcal{S}(l) \cap \mathcal{S}(r) = \emptyset$, then any member of $\mathcal{S}(l) \cup \mathcal{S}(r)$ provides a minimum extension with a change needed on only one of the edges directed

from ρ . If $\mathcal{S}(l) \cap \mathcal{S}(r) \neq \emptyset$, then any member of $\mathcal{S}(l) \cap \mathcal{S}(r)$ provides a minimum extension with no change on either edge directed from ρ . \square

Note that the value of $\mathcal{S}(v)$ for an internal node other than the root need not, in general, be its assignment under a maximum parsimony character extension because its value would depend on the possible states at other nodes closer to the root ρ . However, we are only concerned about the value at the root node, and we can therefore use this algorithm.

3.2 Maximum Likelihood Method

3.2.1 Markov Processes on Trees

Here we will introduce a Markov process on a tree, as this will be the model for our likelihood method. Let \mathcal{T} be a rooted binary tree, and for each $u \in V$, let ξ_u be a random variable defined for each $u \in V$ taking values in the state set \mathcal{C} . A Markov process on a tree associates with each edge $e_i = (u, v)$ a transition probability matrix $\mathbf{P}(e_i)$ indexed by the values of \mathcal{C} , such that $P(e_i)_{\alpha\beta} = \mathbb{P}(\xi_v = \beta | \xi_u = \alpha)$ for $\alpha \in \mathcal{C}$. Let $<$ be an ordering on the tree such that if $(u, v) \in E$ then $u < v$. Then if $(u, v) \in E$ we have the Markov property on the tree

$$\mathbb{P}\left(\xi_v = \alpha \mid \bigcap_{w < v} \xi_w\right) = \mathbb{P}(\xi_v = \alpha | \xi_u), \forall \alpha \in \mathcal{C}.$$

We consider a continuous-time Markov process with transition rate matrix \mathbf{Q}_i indexed by the states the process can take on and with entries representing the rate of mutation from one state to another.

If we assign edge lengths on our tree (which we consider simply as times), then the rate matrix \mathbf{Q}_i induces a transition probability matrix \mathbf{P}_i on edge e_i with length t_i according to

$$\mathbf{P}_i = \exp(t_i \mathbf{Q}_i).$$

This model allows multiple transitions on to occur on an edge, but the probability of starting at one state and finishing in another state are still given by the entries of \mathbf{P}_i .

We use this random model for the evolution of characters on a tree, supposing the states of a character χ on our tree transition based on this model. Let $\bar{\chi}$ be an extension of a character on our tree. For a given set of transition probability matrices and a given character extension, we now have an associated probability of observing this character extension on a tree

$$\mathbb{P}(\bar{\chi}) = \mathbb{P}\left(\bigcap_{v \in V} (\xi_v = \bar{\chi}(v))\right).$$

Then, we can write the associated probability of observing the character χ as

$$\mathbb{P}(\chi) = \sum_{\bar{\chi}} \mathbb{P}(\bar{\chi}) = \sum_{\bar{\chi}} \mathbb{P}\left(\bigcap_{v \in V} (\xi_v = \bar{\chi}(v))\right),$$

where the summation over $\bar{\chi}$ denotes the summation over all possible extensions $\bar{\chi}$ of χ . For the reconstruction of the state of the common ancestor with this model, we will be interested in the probability of extensions conditioned on certain values at the root ρ , given for some $\alpha \in \mathcal{C}$ by

$$\mathbb{P}(\chi|\rho = \alpha) = \sum_{\bar{\chi}_{\rho=\alpha}} \mathbb{P}(\bar{\chi}_{\rho=\alpha}) = \sum_{\bar{\chi}_{\rho=\alpha}} \mathbb{P}\left(\bigcap_{v \in V} (\xi_v = \bar{\chi}_{\rho=\alpha}(v))\right). \quad (3.1)$$

We will show an example of the calculation in the next section, as this seemingly overbearing equation is in fact extremely simple to evaluate for the model we adopt.

3.2.1.1 Symmetric N_r Model

Here we introduce the Neyman N_r model [16], which is the model we will be most interested in. In the N_r model we consider r possible states that the model can take on, and have an $r \times r$ rate matrix \mathbf{Q}_i for each $i \in E$ with the property that all off-diagonal entries are equal and nonzero. We will further assume the rate matrices \mathbf{Q} are equal on all edges of the tree, so that the process governing the transitions is constant across the tree.

From this it follows that for any edge of nonzero length t_i , we can write the associated $r \times r$ transition probability matrix as

$$\mathbf{P}(e_i) = \begin{bmatrix} 1 - p_i & \frac{p_i}{r-1} & \cdots & \frac{p_i}{r-1} \\ \frac{p_i}{r-1} & 1 - p_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{p_i}{r-1} \\ \frac{p_i}{r-1} & \cdots & \frac{p_i}{r-1} & 1 - p_i \end{bmatrix}$$

for all $\mathbf{P} \in \theta$, where $0 < p_i < (r - 1)/r$.

When $r = 2$ this is known as the Neyman N_2 model. While the model seems simple, it does have biological significance, as it is used to represent the evolution of purines and pyrimidines in a DNA (deoxyribonucleic acid) sequence.

We show how we can use this to calculate the probability of a character using the transition matrices under an N_2 model, conditioned on a fixed value of the character

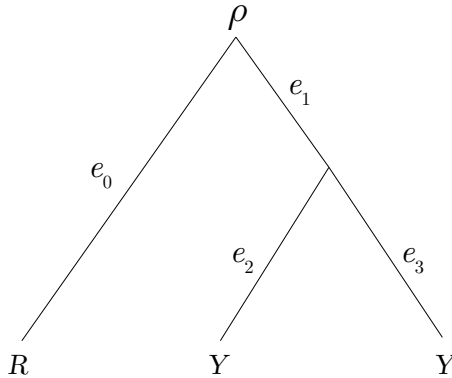


Figure 3.1: A simple tree with the character RYY at the edges.

at ρ . We consider the tree shown in Figure 3.1. The single character we observe has states $\mathcal{C} = \{R, Y\}$, and on this tree gives us the data pattern RYY.

In this case, we can construct the equation associated with the probability of observing this character, as shown in Example 1.

Example 1 (Constructing the Probability Equation for a Character). *We condition on the fixed value of $\rho = R$, or, in other words, only evaluate the sum in (3.1) over extensions of our character that assign this fixed value to ρ . We have*

$$\mathbb{P}(\chi|\rho = R) = (1 - p_0) [(1 - p_1)p_2p_3 + p_1(1 - p_2)(1 - p_3)]. \quad (3.2)$$

The equation is factored for convenient understanding, and we can explain this equation very simply. Since we are conditioning on $\rho = R$, on the leftmost edge of the tree we must have no change of state on this edge, as the character produces the state R at the leaf, and we obtain the term $(1 - p_0)$. The two terms that follow can be thought of in a simple way as well; there must be either a change from the root state on edge e_1 followed by no change on edges e_2 and e_3 , or alternatively no change on edge e_1 followed by changes on both of the edges e_2 and e_3 .

3.2.2 Molecular Clock Constraint

The concept of the molecular clock refers to a clocklike behaviour of change happening in evolution, so that, from the root state, there is a measure of time on the branches towards the leaves. We say that on a path from the root to any leaf (extant species), an equal amount of time would have elapsed. Therefore the sum of the edge lengths on along path from the root to a leaf must be the same. This restriction provides a constraint for the the transition probabilities.

To demonstrate what this constraint gives us, we will examine the tree shown in Figure 3.2, where the transition matrix P_i is labelled for each edge. The first thing we note about this tree is that two branches have been given the label P_4 ; this is because we require that both leaves descended from branching point B will have any equal distance from this point, and this gives us that the transition probability on the branches must be the same, so we can handle this by immediately labelling the tree appropriately. In Example 2, we show the constraints on the probabilities.

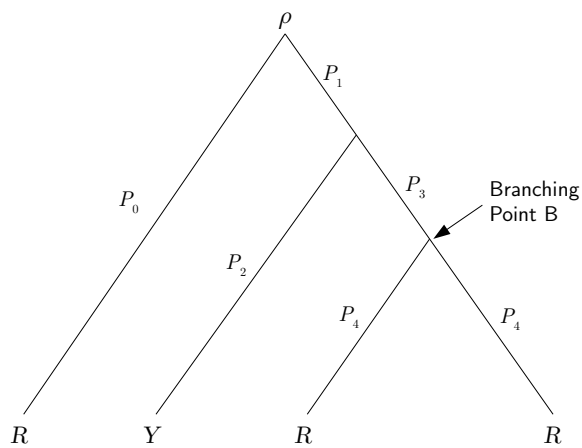


Figure 3.2: Transition probability matrices on a tree, with two branches given the same matrix.

Example 2 (Constraints from the Molecular Clock). *For the branch lengths t_i on each edge of the tree shown in Figure 3.2, the constraint gives us the following three equality equations*

$$\begin{aligned} t_0 &= t_1 + t_2, \\ t_0 &= t_1 + t_3 + t_4, \\ t_1 + t_2 &= t_1 + t_3 + t_4. \end{aligned}$$

It is easily seen using the matrix exponential that under the N_2 model and the assumptions we adopt that this translates into the equalities on the transition probability matrices as

$$\begin{aligned} P_0 &= P_1 P_2, \\ P_0 &= P_1 P_3 P_4, \\ P_1 P_2 &= P_1 P_3 P_4. \end{aligned} \tag{3.3}$$

3.2.3 Maximum Likelihood for Ancestral States

Maximum likelihood estimation involves searching the parameter space of a given model to find the values which maximize a likelihood function; it is a search to find parameters which maximize the probability of observing the data. We define the likelihood function of observing a character on our tree as

$$\mathbb{L}(\theta; \chi) = \mathbb{P}(\chi|\theta), \quad (3.4)$$

where θ parameterises the set of transition probability matrices on our tree. For any model, we look for parameters in some permitted set Θ , and we can formulate the likelihood problem of searching for the maximum likelihood \mathbb{L}^* as

$$\mathbb{L}^*(\chi) = \sup_{\theta \in \Theta} \mathbb{L}(\theta; \chi).$$

We are interested in using maximum likelihood to reconstruct the state of the common ancestor ρ on our tree. To do this, we have two steps. First, we find the value of the maximum likelihood \mathbb{L}^* conditioned on the possible values of the root state. That is, we find for each $\alpha \in \mathcal{C}$

$$\mathbb{L}^*(\chi|\rho = \alpha) = \sup_{\theta \in \Theta} \mathbb{L}(\theta; \chi|\rho = \alpha) = \sup_{\theta \in \Theta} \mathbb{P}(\chi|\theta, \rho = \alpha)$$

Finally, we assign to the root ρ the value α for which the likelihood $\mathbb{L}^*(\chi|\rho = \alpha)$ is maximized. If for two values α, β we have $\mathbb{L}^*(\chi|\rho = \alpha) = \mathbb{L}^*(\chi|\rho = \beta)$, then we say the reconstruction is ambiguous and we assign the state set $\{\alpha, \beta\}$.

The challenge of the method is that the first step in the process is difficult. For example, if we consider the example where we derived Example 1, the problem of finding $\mathbb{L}^*(\chi|\rho = R)$ is given by

$$\begin{aligned} \mathbb{L}^*(\chi|\rho = R) &= \sup_{\theta \in \Theta} \mathbb{P}(\chi|\rho = R) \\ &= \sup_{\theta \in \Theta} (1 - p_0) [(1 - p_1)p_2p_3 + p_1(1 - p_2)(1 - p_3)]. \end{aligned}$$

Searching the parameter space Θ involves searching over appropriate (constrained) values for p_i to find \mathbb{L}^* , which requires finding the global optimum of a multivariate a polynomial. In the case of the N_2 model we had the restriction $p_i \in (0, 1/2)$, and we will additionally wish to enforce the constraints of the molecular clock.

3.2.4 Likelihood Maximization as an Optimization Problem

We are interested in maximizing likelihood functions under particular constraints, and we wish to find a *global* maximum for this polynomial, and be able to certify its global optimality.

To solve the maximum likelihood problem, we will be concerned with optimization problems in the form

$$\begin{aligned} & \text{maximize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \geq 0 \quad i = 1, 2, \dots, m, \\ & && h_i(\mathbf{x}) = 0 \quad i = 1, 2, \dots, n. \end{aligned} \tag{3.5}$$

We refer to \mathbf{x} as the optimization variable, f_0 as the objective function, and f_i and h_i as the inequality and equality constraint functions respectively. We call the set $\{\mathbf{x} : f_i(\mathbf{x}) \geq 0, h_i(\mathbf{x}) = 0\}$ the feasible set, A vector \mathbf{x}^* in the feasible set an optimal value if $f_0(\mathbf{x}^*) \geq f_0(\mathbf{z})$ for all \mathbf{z} in the feasible set.

The maximum likelihood problems we will look at can be formulated in the form of (3.5). The likelihood function will be our objective function, and the constraint functions will come from our molecular clock constraints.

Consider the tree shown in Figure 3.1. We will have a two state character model with $\mathcal{C} = \{R, Y\}$, and adopt the N_2 model with a molecular clock constraint. Our objective function is the the likelihood function conditioned on $\rho = R$

$$\mathbb{L}(\chi; \theta | \rho = R) = \mathbb{P}(\chi | \rho = R) = (1 - p_0) [(1 - p_1)p_2p_3 + p_1(1 - p_2)(1 - p_3)].$$

Additionally, we require for the N_2 model that

$$p_i \in (0, 1/2),$$

which we can simply write as two inequality constraints for each variable. We will allow limiting cases of the values of p_i so that $p_i \in [0, 1/2]$. Hence our optimization problem contains inequality constraints.

Finally, we have the constraints given by our molecular clock, which in this case is the single matrix equality

$$P_0 = P_1P_2,$$

which when written out gives us

$$\begin{aligned} \begin{bmatrix} 1 - p_0 & p_0 \\ p_0 & 1 - p_0 \end{bmatrix} &= \begin{bmatrix} 1 - p_1 & p_1 \\ p_1 & 1 - p_1 \end{bmatrix} \begin{bmatrix} 1 - p_2 & p_2 \\ p_2 & 1 - p_2 \end{bmatrix} \\ \implies \begin{bmatrix} 1 - p_0 & p_0 \\ p_0 & 1 - p_0 \end{bmatrix} &= \begin{bmatrix} (1 - p_1)(1 - p_2) + p_1p_2 & (1 - p_1)p_2 + p_1(1 - p_2) \\ (1 - p_1)p_2 + p_1(1 - p_2) & (1 - p_1)(1 - p_2) + p_1p_2 \end{bmatrix}. \end{aligned}$$

From this matrix equation, we have four polynomial equalities, though only one of these is independent; part of this is due to the symmetry of the model, and part of this is because the two independent equations represent probabilities of complimentary events of a change occurring and for a change not occurring. We will always choose

the “upper-right” entry corresponding to the first row and second column. Thus we have

$$p_0 = (1 - p_1)p_2 + p_1(1 - p_2),$$

from which we form the equality constraint on our optimization problem

$$h_1(\mathbf{p}) = p_0 - [(1 - p_1)p_2 + p_1(1 - p_2)] = 0.$$

In Example 2, it is not difficult to see that (3.3) contains linearly dependent equations. To form our polynomial optimization problems, we choose a linearly independent set of equations. For all trees we examine, we can describe this in a natural way, but it is dependent on a fixed left-to-right interpretations of the trees as shown in the diagrams.

To form the equality constraints, we take the leftmost path and equate it with the path for the leaf immediately to the right. For the next equation, we take the leftmost path and equate it with the path to the leaf to immediately to the right of the previous one used. We repeat this until we have a set of n equality constraints, where n is the number of leaves on the tree.

We have shown that it is possible to formulate our maximum likelihood problem in terms of an optimization problem of the form (3.5), where our objective function and all of our constraint functions are polynomials. In the next chapter, we will consider how to solve optimization problems of this form and hence a method to solve our maximum likelihood problem.

Chapter 4

Polynomial Optimization

Optimization problems for which our objective and constraint functions are real-valued polynomials are known as polynomial optimization problems (POPs). They present challenging problems.

In the last decade there has been significant interest and development in the area of polynomial optimization based upon sum-of-squares (SOS) representations for polynomials and its relation to a type of convex optimization problem known as a semidefinite program (SDP) [11, 12, 14, 17, 28]. This is useful from a practical perspective because of there has concurrently been development of theory and software for the efficient computation of solutions to SDPs [27, 25].

In this chapter, we will discuss how POPs can be solved using theories of SOS and SDP, and we will present methods which allow us to optimize polynomials on certain sets. We will closely follow the results of Lasserre [12, 14].

Roughly, the theory provides a way to represent the polynomial optimization problem as a sequence of SDPs, which are constrained so as to guarantee that the optimal value of these SDPs will converge to the globally optimal value of the POP. The sequence of programs can be seen as a sequence looking over a larger and larger space of SOS polynomials to find an optimal bound. The result of convergence are due to some results on special representations of polynomials.

4.1 Polynomials and Sum of Squares

We will write a monomial of $\mathbf{x} \in \mathbb{R}^n$ as $\mathbf{x}^\alpha = \prod_{i=1}^n x_i^{\alpha_i}$ where $\alpha \in \mathbb{Z}_+^n$. A polynomial p is a sum of monomials which can be written as $p(\mathbf{x}) = \sum_{\alpha \in \mathcal{F}} c_\alpha \mathbf{x}^\alpha$ where $c_\alpha \neq 0$, and and $\mathcal{F} \subset \mathbb{Z}_{\geq 0}^n$ is some finite set. We will write the space of these polynomials $\mathbb{R}[\mathbf{x}]$. The set \mathcal{F} is called the support of the polynomial p , and the degree of the polynomial is $\max\{\sum_{i=1}^n \alpha_i : \alpha \in \mathcal{F}\}$. We denote the space of polynomials of degree less than

or equal to ω as $\mathbb{R}[\mathbf{x}, \omega]$. To write the individual linear terms x_i , we may sometimes write \mathbf{x}^{e_i} , where \mathbf{e}_i represents the i^{th} canonical basis vector in \mathbb{Z}^n .

We say a polynomial $p(\mathbf{x})$ is SOS (a sum-of-squares) if there exist polynomials $q_0, q_1, \dots, q_m \in \mathbb{R}[\mathbf{x}]$ such that

$$p(\mathbf{x}) = \sum_{k=0}^m q_k^2(\mathbf{x}). \quad (4.1)$$

We denote the space of these polynomials by $\mathbb{R}[\mathbf{x}]^2$. We will use the notation $\mathbb{R}[\mathbf{x}, \omega]^2$ to denote the space of SOS polynomials of degree at most 2ω . That is,

$$\mathbb{R}[\mathbf{x}, \omega]^2 = \left\{ \sum_{k=0}^m q_k^2(\mathbf{x}) : \forall m, k \ q_k \in \mathbb{R}[\mathbf{x}, \omega] \right\}.$$

We note immediately that if p is SOS, then it must have even degree, and also that if p is SOS it must be globally nonnegative.

Hilbert proved that there are exactly three cases where the spaces of nonnegative polynomials and SOS polynomials coincide: bivariate, quadratic, and trivariate quartic polynomials of homogenous degree. So, there are cases of nonnegative polynomials which are not SOS. Surprisingly, an example of a nonnegative polynomial that is not a SOS polynomial did not appear in literature until 1967 in a publication by Motzkin. For these details, see [20].

However, there has been research on special representations of nonnegative polynomials functions on compact sets in terms of SOS polynomials [19, 10]. We will see that because of these results, we will be able to construct SDPs that allow us solve many polynomial optimization problems.

We make use of a vector of all monomials in \mathbf{x} of degree at most d under the canonical basis for monomials. We use the notation $\mathbf{z}(\mathbf{x}, d)$ to denote this vector. We can think of having some ordering of the monomials. For example, when $\mathbf{x} = (x_1, x_2)$ and $d = 2$, then we could write $\mathbf{z}(\mathbf{x}, d) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)$. Orderings are also indispensable in computing with polynomials; for details on monomial orderings, see [3].

4.2 Semidefinite Programming

We will introduce some notation for semidefinite matrices and briefly remind the reader of some properties of positive semidefinite (PSD) matrices. We say a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is PSD if $\langle \mathbf{x}, \mathbf{A}\mathbf{x} \rangle = \mathbf{x}^T \mathbf{A}\mathbf{x} \geq 0$ for all nonzero vectors \mathbf{x} , and positive definite if the strict inequality holds.

We will use the notation $\mathbf{A} \succeq \mathbf{B}$ (resp. $\mathbf{A} \succ \mathbf{B}$) to denote that $\mathbf{A} - \mathbf{B}$ is PSD (resp. positive definite). We will restrict ourselves to the space of symmetric PSD matrices unless otherwise specified, and we will use \mathcal{S}^n (resp. $\mathcal{S}_+^n, \mathcal{S}_{++}^n$) to denote the space of $n \times n$ symmetric matrices (resp. symmetric positive semidefinite, symmetric positive definite), and $\mathbf{A} \bullet \mathbf{B}$ to denote $\text{tr}(\mathbf{A}\mathbf{B})$.

We will now introduce a type of optimization problem known as a semidefinite program (SDP). An optimization problem is a SDP if it can be formulated as

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{A}_0 + \sum_{i=1}^m x_i \mathbf{A}_i \succeq 0, \end{aligned} \tag{4.2}$$

where $\mathbf{x} \in \mathbb{R}^m$ is our programming variable, $\mathbf{c} \in \mathbb{R}^n$, and $\mathbf{A}_i \in \mathcal{S}^n$ are given problem parameters. We say that the problem (4.2) is feasible if $\exists \mathbf{x}$ satisfying $\sum_{i=1}^m x_i \mathbf{A}_i \succeq -\mathbf{A}_0$, and strictly feasible if $\exists \mathbf{x}$ satisfying $\sum_{i=1}^m x_i \mathbf{A}_i \succ -\mathbf{A}_0$.

This SDP problem has a related form which is

$$\begin{aligned} & \text{maximize} && -\mathbf{A}_0 \bullet \mathbf{X} \\ & \text{subject to} && \mathbf{A}_i \bullet \mathbf{X} = c_i, \quad i = 1, 2, \dots, m, \\ & && \mathbf{X} \succeq 0, \end{aligned} \tag{4.3}$$

where $\mathbf{X} \in \mathcal{S}_+^n$ is now our programming variable, and $\mathbf{A}_1, \dots, \mathbf{A}_m \in \mathcal{S}^n$ are the given problem parameters. We say that (4.3) is feasible if $\exists \mathbf{X} \succeq 0$ such that $\mathbf{A}_i \bullet \mathbf{X} = c_i, i = 1, \dots, m$, and strictly feasible if $\exists \mathbf{X} \succ 0$ satisfying this constraint. We say this program is solvable if a solution exists. The program takes the value $-\infty$ otherwise.

There is an important relationship of duality between the SDP forms (4.2) and (4.3). We will call (4.2) our primal program, and (4.3) our dual program. We will always use these forms for our primal and dual programs, and call both SDPs.

We denote the infimum of the objective function in the primal program as v_{primal} , and the supremum of the objective function in the dual program v_{dual} . When value of the primal and dual programs is not equal, we refer to a duality gap between the problems, defined by $\eta = \mathbf{c}^T \mathbf{x} - \mathbf{A}_0 \bullet \mathbf{X}$ for the optimal solutions \mathbf{x} and \mathbf{X} . The key property of the duality relationship is that feasible points to each problem provides a bound to the solution of the other. We have the following theorem about the duality relationship.

Theorem 2. *If both the primal and dual programs have feasible solutions, then the following hold:*

- i. (Weak duality) For any feasible \mathbf{x} to the primal problem and \mathbf{X} to the dual problem, the duality gap η is always nonnegative. That is, $\mathbf{c}^T \mathbf{x} \geq \mathbf{A}_0 \bullet \mathbf{X}$, hence $v_{\text{primal}} \geq v_{\text{dual}}$.*

ii. (Strong duality) If the primal program or the dual program has a strictly feasible solution, then there is no duality gap between the primal and dual programs so that $v_{\text{primal}} = v_{\text{dual}}$.

Proof. i. We have feasible solutions \mathbf{x} and \mathbf{X} to our primal and dual programs. We consider the duality gap, and we have

$$\begin{aligned}\eta &= \mathbf{c}^T \mathbf{x} - \mathbf{A}_0 \bullet \mathbf{X} \\ &= \sum_{i=1}^m x_i (\mathbf{A}_i \bullet \mathbf{X}) - \mathbf{A}_0 \bullet \mathbf{X} \\ &= \left(\sum_{i=1}^m x_i \mathbf{A}_i + \mathbf{A}_0 \right) \bullet \mathbf{X}.\end{aligned}$$

Since \mathbf{X} is feasible for the dual program, $\mathbf{X} \succeq 0$, and since \mathbf{x} is feasible for the primal program we have that $\sum_{i=1}^m x_i \mathbf{A}_i + \mathbf{A}_0 \succeq 0$. Since the trace of two PSD matrices is nonnegative, we have $\eta \geq 0$, the desired result.

ii. We refer to [24] for this result. □

We can see a pair of solutions to primal and dual programs as bound on each other. Further, when the duality gap is zero the dual can act as a certificate of optimality on the primal program, and vice versa. However, a feature of SDPs is that it is possible to have nonzero duality gaps at the optimum values of the programs. For solving the programs in practice, we desire these strictly feasible solutions.

We will also call a program an SDP if it is clear that it can be transformed to either primal or dual form, though we may not always make the transformation explicit. For example, we may write multiple semidefinite constraints in the program (4.2). To make the transformation, we could simply augment the matrices \mathbf{A}_i appropriately.

4.3 SDP and SOS Polynomials

Here we will begin to demonstrate a relationship between SDP and SOS polynomials by a discussion about the relationship between PSD matrices and SOS polynomials. Suppose we have some polynomial $p(\mathbf{x})$ of degree $2d$. We use \mathbf{z} as a simpler notation for $\mathbf{z}(\mathbf{x}, d)$ in this section. We can then write $p(\mathbf{x})$ as

$$p(\mathbf{x}) = \mathbf{z}^T \mathbf{Q} \mathbf{z}. \tag{4.4}$$

It is simple to construct a constant \mathbf{Q} to satisfy (4.4), but it will not be unique. If \mathbf{Q} is not symmetric, there are an infinite number of possible entries which give, for example, the degree-1 monomial x_1 . There is also a dependence between the variables,

since, for example we could construct the term $x_1^2x_2^2$ as the product of the monomials x_1^2 and x_2^2 , or as the square of x_1x_2 .

However, if \mathbf{Q} is a PSD matrix in (4.4), we have some important results. First, it is clear that $p(\mathbf{x})$ must be a nonnegative function (even if \mathbf{Q} is not symmetric). Further, if $\mathbf{Q} \in \mathcal{S}_+^n$ then the polynomial will in fact be SOS, since \mathbf{Q} will have the eigen decomposition $\mathbf{Q} = \mathbf{U}^T \mathbf{D} \mathbf{U}$, where \mathbf{D} is diagonal with $d_{ii} \geq 0$, which gives us

$$p(\mathbf{x}) = \mathbf{z}^T \mathbf{U}^T \mathbf{D} \mathbf{U} \mathbf{z} \quad (4.5)$$

$$= \sum_{i=1}^n d_{ii} [(\mathbf{U} \mathbf{z})^T (\mathbf{U} \mathbf{z})]_{ii}. \quad (4.6)$$

This is a SOS polynomial in the scaled entries of the vector given by $\mathbf{U} \mathbf{z}$, with the scaling being the eigenvalues of \mathbf{Q} . Therefore, finding $\mathbf{Q} \in \mathcal{S}_+^n$ which satisfies (4.4) means we know p to be SOS.

The converse relationship is also true, since if we have any polynomial in the SOS representation (4.1), we can immediately construct an appropriate matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, where n is the size of our vector \mathbf{z} , so that $\mathbf{Q} = \mathbf{A}^T \mathbf{A}$. We simply take the coefficients of each q_k for the rows of \mathbf{A} so that $p(\mathbf{x}) = \mathbf{z}^T \mathbf{A}^T \mathbf{A} \mathbf{z}$, and $\mathbf{Q} = \mathbf{A}^T \mathbf{A}$ must be PSD. Thus we have the relationship

$$p(\mathbf{x}) \text{ is SOS} \iff \exists \mathbf{Q} \succeq 0, p(\mathbf{x}) = \mathbf{z}^T \mathbf{Q} \mathbf{z}. \quad (4.7)$$

Example 3 (An SOS Polynomial Represented Using a PSD Matrix). *We consider the bivariate polynomial $p(\mathbf{x}) = x_1^4 + 2x_2^4 - 2x_1^3x_2 - 6x_1x_2^3 + 8x_1^2x_2^2$, and suppose we are interested in an SOS representation, if it exists, for this polynomial. For simplicity, this example is constructed so that we can use the basis $\mathbf{z} = (x_1^2, x_2^2, x_1x_2)$, although this is not a complete basis for bivariate quartic polynomials. We wish to find $\mathbf{Q} \in \mathcal{S}_+$ so that we satisfy the equality*

$$x_1^4 + 2x_2^4 - 2x_1^3x_2 - 6x_1x_2^3 + 8x_1^2x_2^2 = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1x_2 \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1x_2 \end{bmatrix}. \quad (4.8)$$

From this, we equate the coefficients of monomials on either side of the equation. The solution for \mathbf{Q} here is not unique, but one set of solutions for symmetric \mathbf{Q} is given by

$$\mathbf{Q} = \begin{bmatrix} 1 & \lambda & -1 \\ \lambda & 2 & -3 \\ -1 & -3 & 8 - 2\lambda \end{bmatrix},$$

for $\lambda \in \mathbb{R}$. The particular choice $\lambda = 1$ gives us a positive definite matrix for \mathbf{Q} , which submits to the Cholesky factorization

$$\mathbf{Q} = \mathbf{L}\mathbf{L}^T, \quad \mathbf{L}^T = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}.$$

From this, we can immediately write p in SOS form as follows:

$$p(\mathbf{x}) = (x_1^2 + x_2^2 - x_1x_2)^2 + (x_2^2 - 2x_1x_2)^2 + (x_1x_2)^2.$$

We can reformulate the problem of testing if a polynomial is SOS through an equivalent feasibility SDP¹, where instead of being concerned with the value of the program, we are concerned about whether the feasible set is nonempty and wish to find any \mathbf{Q} in the feasible set of the program

$$\begin{aligned} & \text{maximize} && 0 \\ & \text{subject to} && \mathbf{A}_i \bullet \mathbf{Q} = b_i, \quad i = 1, 2, \dots, p, \\ & && \mathbf{Q} \succeq 0. \end{aligned} \tag{4.9}$$

In this case, the matrices \mathbf{A}_i are formed from the equality constraints of the polynomial coefficients. For example, in Equation (4.8) of our example, one constraint is

$$q_{12} + q_{21} + q_{33} = 8,$$

which we can rewrite in the desired form as

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} = 8.$$

The entries of the matrices \mathbf{A}_i are simply 1 in the appropriate places, and these places correspond to elements of \mathbf{Q} which give algebraically dependent terms in $\mathbf{z}^T \mathbf{Q} \mathbf{z}$.

4.4 The Generalized Problem of Moments

We can also consider a POP as a generalized problem of moments. We introduce this approach here, and will see that this is an alternate way of looking at SOS polynomials. If we are interested in minimizing $p(\mathbf{x})$, consider the problem defined by

$$\min_{\mu \in \mathcal{P}(\mathbb{R}^n)} \int p(\mathbf{x}) \mu(dx), \tag{4.10}$$

¹The standard form for feasibility problem is, by convention, to maximize or minimize 0.

where $\mathcal{P}(\mathbb{R}^n)$ is the set of probability measures on \mathbb{R}^n . When solving on the set K , we have the same formulation but instead over the set of measures with support in the feasible set K of the problem.

Of course, if \mathbf{x}^* is a minimizer of $p(\mathbf{x})$ with minimum value p^* , then the Dirac probability measure $\delta_{\mathbf{x}^*}$ achieves the global minimum p^* . And if p^* is the minimum value of $p(\mathbf{x})$, then $\int p(\mathbf{x})\mu(dx) \geq p^*$ since μ is a probability measure, so the problems are equivalent.

We can write (4.10) as a linear function of the moments of \mathbf{x} with respect to a measure μ . Let \mathbf{y} be these moments, where

$$y_\alpha = \int x^\alpha \mu(dx), \quad (4.11)$$

When a measure μ exists so that a sequence $\mathbf{y} = (y_\alpha)$ can be defined by (4.11) we say \mathbf{y} has a representing measure.

We define a function $L_y(f)$ for $f = \sum_\alpha f_\alpha \mathbf{x}^\alpha \in \mathbb{R}[\mathbf{x}]$ such that

$$L_y(f) = \sum_\alpha f_\alpha y_\alpha,$$

which linearizes f in the moments y_α .

The set of moments \mathbf{y} up to order $2m$ has an associated moment matrix, indexed by $\alpha, \beta \in \mathbb{Z}_{\geq 0}^n$, and defined by

$$\mathbf{M}_m(\mathbf{y}) = [y_{\alpha+\beta}]_{\alpha\beta}, \quad \alpha, \beta \in \mathbb{Z}_+^n, \quad \|\alpha\|_1, \|\beta\|_1 \leq m.$$

For our purposes, we will require that $y_{\mathbf{0}} = 1$. We can also write this as

$$\mathbf{M}_m(\mathbf{y}) = L_y(\mathbf{z}(\mathbf{x}, m)\mathbf{z}(\mathbf{x}, m)^T),$$

where the function $L_y(\cdot)$ is applied element-wise.

We also define a localizing matrix, for any polynomial θ with support \mathcal{F} , where $\theta = \sum_{\gamma \in \mathcal{F}} \theta_\gamma \mathbf{x}^\gamma$,

$$M_m(\theta\mathbf{y}) = \left[\sum_\gamma \theta_\gamma y_{\alpha+\beta+\gamma} \right]_{\alpha\beta}, \quad \alpha, \beta \in \mathbb{Z}_+^n, \quad \|\alpha\|_1, \|\beta\|_1 \leq m,$$

which we can also write using our linearizing function L_y as

$$\mathbf{M}_m(\theta\mathbf{y}) = L_y(\mathbf{z}(\mathbf{x}, m)\mathbf{z}(\mathbf{x}, m)^T \theta(\mathbf{x})).$$

Example 4 (A Localizing Matrix). *Suppose we are interested in constructing the localizing matrix $\mathbf{M}_1(\theta\mathbf{y})$ for $\mathbf{x} \in \mathbb{R}^2$ with $\theta(\mathbf{x}) = x_1$. First, we examine the matrix $\mathbf{z}(\mathbf{x}, m)\mathbf{z}(\mathbf{x}, m)^T\theta(\mathbf{x})$, and we have*

$$\begin{aligned}\mathbf{z}(\mathbf{x}, m)\mathbf{z}(\mathbf{x}, m)^T\theta(\mathbf{x}) &= \begin{bmatrix} 1 & x_1 & x_2 \\ x_1 & x_1^2 & x_1x_2 \\ x_2 & x_1x_2 & x_2^2 \end{bmatrix} x_1 \\ &= \begin{bmatrix} x_1 & x_1^2 & x_1x_2 \\ x_1^2 & x_1^3 & x_1^2x_2 \\ x_1x_2 & x_1^2x_2 & x_1x_2^2 \end{bmatrix}.\end{aligned}$$

Therefore, for $\mathbf{M}_1(\theta\mathbf{y})$ we have

$$\mathbf{M}_1(\theta\mathbf{y}) = \begin{bmatrix} y_{(1,0)} & y_{(2,0)} & y_{(1,1)} \\ y_{(2,0)} & y_{(3,0)} & y_{(2,1)} \\ y_{(1,1)} & y_{(2,0)} & y_{(1,2)} \end{bmatrix},$$

which is the desired localizing matrix.

Now, for some polynomials p and q of degree m with coefficient vectors \mathbf{p} and \mathbf{q} , then if y is generated by the measure μ_y ,

$$\begin{aligned}\mathbf{p}^T\mathbf{M}_m(\mathbf{y})\mathbf{q} &= \sum_{\alpha} (pq)_{\alpha}y_{\alpha} \\ &= L_y(p(\mathbf{x})q(\mathbf{x})) \\ &= \int p(\mathbf{x})q(\mathbf{x})\mu_y(dx),\end{aligned}$$

and here $\alpha \in \mathbb{Z}_+$ is such that $\|\alpha\|_1 \leq 2m$, and we use it to index the moments and polynomial coefficients. With this, we are able to provide a necessary condition for \mathbf{y} to have a representing measure.

Lemma 1. *If a vector \mathbf{y} has a representing measure μ_y , then $\mathbf{M}_m(\mathbf{y}) \succeq 0$.*

Proof. For any polynomial p of degree less than or equal to m with coefficient vector \mathbf{p} ,

$$\mathbf{p}^T\mathbf{M}_m(\mathbf{y})\mathbf{p} = L_y(p(\mathbf{x})p(\mathbf{x})) = \int p(\mathbf{x})^2 d\mu_y \geq 0,$$

and, since this holds for all p , we have $\mathbf{M}_m(\mathbf{y}) \succeq 0$. \square

We have a very similar result for the localizing matrix on a semialgebraic set defined by $\theta(\mathbf{x})$.

Lemma 2. *Let K be the set defined by $K = \{\mathbf{x} : \theta(\mathbf{x}) \geq 0\}$. If a vector \mathbf{y} has a representing measure μ_y with support strictly on the set K , then $\mathbf{M}_m(\theta\mathbf{y}) \succeq 0$.*

Proof. For any polynomial p of degree less than or equal to m with coefficient vector \mathbf{p} ,

$$\begin{aligned}\mathbf{p}^T \mathbf{M}_m(\mathbf{y}\theta) \mathbf{p} &= L_y(p(\mathbf{x})p(\mathbf{x})\theta(\mathbf{x})) \\ &= \int p(\mathbf{x})[p(\mathbf{x})\theta(\mathbf{x})]d\mu_y \\ &= \int p(\mathbf{x})^2\theta(\mathbf{x})d\mu_y \geq 0,\end{aligned}$$

and we have $\mathbf{M}_m(\theta\mathbf{y}) \succeq 0$. □

A particular consequence of this is that if we have any point \mathbf{x}_0 , Lemma 1 gives us that the matrix generated by the Dirac measure $\delta_{\mathbf{x}_0}$ will be PSD. This holds similarly for the localizing matrices when $\mathbf{x}_0 \in K$ in Lemma 2.

In Section 4.3, we noted that a polynomial f is SOS if $\exists \mathbf{Q} \in \mathcal{S}_+$ such that $f = \mathbf{z}^T \mathbf{Q} \mathbf{z} = \mathbf{Q} \bullet (\mathbf{z}\mathbf{z}^T)$, where \mathbf{z} was the appropriate basis vector of polynomials. We see why the formulation in terms of these sequences \mathbf{y} is similar, since

$$L_y(\mathbf{Q} \bullet (\mathbf{z}\mathbf{z}^T)) = \mathbf{Q} \bullet \mathbf{M}(y).$$

4.4.1 Unconstrained Programs

We will start our examination of optimization by presenting how an unconstrained POP of the form

$$\text{minimize } f(\mathbf{x}) \tag{4.12}$$

can be solved as an SDP.

Let our polynomial to be optimized be $f \in \mathbb{R}[\mathbf{x}]$, and be of degree $2d$ with corresponding coefficient vector \mathbf{c} indexed by α so that $f(\mathbf{x}) = \sum_{\alpha} c_{\alpha} \mathbf{x}^{\alpha} = \mathbf{c}^T \mathbf{z}(\mathbf{x}, 2d)$, and assume $f(\mathbf{0}) = 0$.

Now, we consider the program, where $\omega = d$,

$$\begin{aligned}\text{minimize } & \mathbf{c}^T \mathbf{y} \\ \text{subject to } & \mathbf{M}_{\omega}(\mathbf{y}) \in \mathcal{S}_+^{\omega}.\end{aligned}$$

We can write this for some appropriate matrices \mathbf{B}_{α} as

$$\begin{aligned}\text{minimize } & \mathbf{c}^T \mathbf{y} \\ \text{subject to } & \mathbf{B}_0 + \sum_{\alpha \neq 0} \mathbf{y}_{\alpha} \mathbf{B}_{\alpha} \succeq 0,\end{aligned} \tag{4.13}$$

which we call our primal program. The dual to this problem is

$$\begin{aligned}\text{maximize } & -\mathbf{X} \bullet \mathbf{B}_0 \\ \text{subject to } & \mathbf{X} \bullet \mathbf{B}_{\alpha} = c_{\alpha}, \alpha \neq 0 \\ & \mathbf{X} \succeq 0.\end{aligned} \tag{4.14}$$

To make the formulation concrete, we provide Example 5 for constructing the primal program.

Example 5 (Formulation of a Program). *Suppose we have the (unconstrained) polynomial optimization problem*

$$\text{minimize } x_1^2 - 3x_1x_2 + 2x_2^2.$$

Let \mathbf{y} , indexed by integer entries in this case, correspond with the vector $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$. In this case, we want to minimize $\mathbf{c}^T\mathbf{y}$ where $\mathbf{c} = (0, 0, 0, 1, -3, 2)$. Now, note that

$$\mathbf{z}(\mathbf{x}, 1)^T\mathbf{z}(\mathbf{x}, 1) = \begin{bmatrix} 1 & x_1 & x_2 \\ x_1 & x_1^2 & x_1x_2 \\ x_2 & x_1x_2 & x_2^2 \end{bmatrix},$$

and recall that we can think of $\mathbf{M}_m(\mathbf{y})$ as a linearization of this in the variables y_α . In this case, we can therefore formulate our problem in the form (4.13) as

$$\begin{aligned} &\text{minimize } \mathbf{c}^T\mathbf{y} \\ &\text{subject to } \begin{bmatrix} y_1 & y_2 & y_3 \\ y_2 & y_4 & y_5 \\ y_3 & y_5 & y_6 \end{bmatrix} \succeq 0. \end{aligned} \tag{4.15}$$

The definitions of \mathbf{B}_α are easily understood from this. Notice that, similar to Example 3, the entries of the matrices \mathbf{B}_α are 1 in the places where we find the algebraically dependent terms in $\mathbf{z}(\mathbf{x}, 1)^T\mathbf{z}(\mathbf{x}, 1)$.

We provide the several results about these programs. We begin by showing that the duality gap can close if we solve the dual program.

Theorem 3. *If the dual program (4.14) has a feasible solution, then there is no duality gap between the primal (4.13) and dual (4.14) programs.*

Proof. Choose any measure μ with strictly positive density and so that all moments exist (for example, the Gaussian measure). For $f \neq 0$, $\int f(\mathbf{x})^2\mu(dx) > 0$ and is defined since all moments exist. Thus in Lemma 1 we achieve a strict inequality, and the result follows from Theorem 2(ii) on SDP duality. \square

Next, we have the following theorem about using these programs to solve our POP. We will see the relationship between these programming problems and our POP in the proof.

Theorem 4 (Theorem 3.2 of [12]). *Suppose polynomial f in (4.12) is of degree $2d$ and has global minimizer \mathbf{x}^* and takes on the minimum value f^* . Then*

i. If $f(\mathbf{x}) - f^*$ is SOS, the POP (4.12) is equivalent to the program (4.13). That is, the problems have the same optimal value, and

$$\mathbf{y}^* = (1, x_1^*, \dots, x_n^*, (x_1^*)^2, (x_2^*)^2, \dots)$$

(generated by the Dirac measure $\delta_{\mathbf{x}^*}$) is a minimizer of (4.13).

ii. If the program (4.14) has a feasible solution and the value of the solution to (4.13) is equal to f^* then $f(\mathbf{x}) - f^*$ is SOS.

Proof. Recall that we assumed the constant part of f is 0.

i. We have that

$$f(\mathbf{x}) - f^* = \sum_i^r q_i(\mathbf{x})^2$$

for some $q_1, \dots, q_r \in \mathbb{R}[\mathbf{x}, d]$. Let $\mathbf{q}_1, \dots, \mathbf{q}_r$ be the corresponding coefficient vectors of these polynomials. Note that $L_y(q_i(\mathbf{x})^2) = \mathbf{q}_i \mathbf{q}_i^T \bullet \mathbf{M}_d(y)$. So, we set $\mathbf{X} = \sum_i^r \mathbf{q}_i \mathbf{q}_i^T$, which will be PSD, and we have

$$\begin{aligned} L_y(f(\mathbf{x}) - f^*) &= L_y\left(\sum_i^r q_i(\mathbf{x})^2\right) \\ &= \mathbf{X} \bullet \mathbf{M}_d(y) \\ &= \mathbf{X} \bullet \mathbf{B}_0 + \sum_{\alpha \neq 0} y_\alpha \mathbf{X} \bullet \mathbf{B}_\alpha. \end{aligned}$$

However, we also have

$$L_y(f(\mathbf{x}) - f^*) = -f^* + \sum_{\alpha \neq 0} c_\alpha y_\alpha,$$

where c_α is the coefficient on \mathbf{x}^α in our polynomial.

By looking at the coefficients on the y_α terms in these equations, we can see that $\mathbf{X} \bullet \mathbf{B}_0 = X_{11} = -f^*$ and $\mathbf{X} \bullet \mathbf{B}_\alpha = c_\alpha$, therefore \mathbf{X} is feasible for our dual program (4.14) with optimal value f^* . Next, from Lemma 1 we know that $\mathbf{y}^* = (1, x_1^*, \dots, x_n^*, (x_1^*)^2, (x_2^*)^2, \dots)$ is feasible for the primal program, and the objective of the primal program at this point takes value f^* , so we have our desired result.

ii. We refer to [12] for the details of this result. \square

We can notice from this that the solution \mathbf{X} to the dual program actually implies an SOS decomposition. This proof used the fact that $f(\mathbf{x}) - f^*$ was SOS, but if we solve the programs and obtain \mathbf{X} , then, through an eigen decomposition, we can form an SOS representation of $f(\mathbf{x})$.

A question that is raised is what happens in the general case when $f(\mathbf{x}) - f^*$ is not SOS, since, as previously noted, there is, in general, a gap between nonnegative and SOS polynomials. In fact, it may still be the case that $f(\mathbf{x}) - f^* + v$ is SOS for some v , in which case the dual program will be solvable with value $-f^* + v$ and the dual program gives a SOS decomposition for the polynomial $f - f^* + v$. However, we can solve the problem if we restrict ourselves to solving for the minimum on certain compact sets. We will be concerned with solving constrained problems, and we present these related results next.

4.4.2 Inequality Constrained Case

Consider an optimization problem of the form

$$\begin{aligned} & \text{minimize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \geq 0, \quad i = 1, \dots, r. \end{aligned} \tag{4.16}$$

where $f_0(\mathbf{x}) \in \mathbb{R}[\mathbf{x}, d]$, $f_i(\mathbf{x}) \in \mathbb{R}[\mathbf{x}, \omega_i]$. Again, we will assume $f_0(\mathbf{0}) = 0$. *Nota bene* we now assume our objective polynomial function is of degree d not necessarily even. In this case, the feasible set K is a semialgebraic set, defined by the polynomials f_i , $K = \{\mathbf{x} : f_i(\mathbf{x}) \geq 0, i = 1, \dots, r\}$.

We assume that the following condition on the set K and holds.

Condition 1 (Putinar's Condition [19]). *The set $K = \{\mathbf{x} : f_i(\mathbf{x}) \geq 0, i = 1, \dots, r\}$ is compact, and there exists a function u*

$$u(\mathbf{x}) = u_0(\mathbf{x}) + \sum_{i=1}^r f_i(\mathbf{x})u_i(\mathbf{x}) \tag{4.17}$$

for some u_0, \dots, u_i where $u_i \in \mathbb{R}[\mathbf{x}, \psi_i]^2$ for some ψ_i , such that the set $\{\mathbf{x} : u(\mathbf{x}) \geq 0\}$ is compact.

From [19] (Lemma 4.1), we have that Condition 1 holds if and only if there exist polynomials q_1, \dots, q_i , where $q_i \in \mathbb{R}[\mathbf{x}, \Omega_i]^2$, such that for every polynomial p strictly positive on K , p has a representation

$$p(\mathbf{x}) = q_0(\mathbf{x}) + \sum_{i=1}^r f_i(\mathbf{x})q_i(\mathbf{x}). \tag{4.18}$$

While Condition 1 ensures the existence of this representation, we do not know *a priori* the degrees of the polynomials q_i . We have some important cases when this will be satisfied.

A case of where this turns out to be satisfied is when all the f_i ($i = 1, \dots, r$) are linear [10] (Theorem 4.1). Another case where this is satisfied is when we can add to

the definition of K the (redundant) constraint $f_{r+1}(\mathbf{x}) = a^2 - \|\mathbf{x}\|^2 \geq 0$, as pointed out in [12]. To see this, we simply set all the q_0 equal to 0 and q_{r+1} equal to 1, and the result is obvious. This reasoning would also generalize if for some fixed i the set $\{\mathbf{x} : f_i(\mathbf{x}) \geq 0\}$ is compact.

We now introduce our programs to solve the constrained POP. First, we let

$$\omega_{\max} = \max\{\lceil d/2 \rceil, \lceil \omega_1/2 \rceil, \dots, \lceil \omega_r/2 \rceil\}.$$

We will call ω our *relaxation order*. We now consider the program of order $\omega \geq \omega_{\max}$,

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T \mathbf{y} \\ & \text{subject to} && \mathbf{M}_\omega(\mathbf{y}) \in \mathcal{S}_+^m, \\ & && \mathbf{M}_{\omega-\omega_i}(f_i \mathbf{y}) \in \mathcal{S}_+^m, \quad i = 1, \dots, r. \end{aligned}$$

As before, we write this program as

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T \mathbf{y} \\ & \text{subject to} && \mathbf{B}_0 + \sum_{\alpha \neq 0} y_\alpha \mathbf{B}_\alpha \succeq 0, \\ & && \mathbf{C}_{i0} + \sum_{\alpha \neq 0} y_\alpha \mathbf{C}_{i\alpha} \succeq 0, \quad i = 1, \dots, r, \end{aligned} \tag{4.19}$$

for appropriate matrices $B_\alpha, C_{i\alpha}$. This problem has dual

$$\begin{aligned} & \text{maximize} && -\mathbf{X} \bullet \mathbf{B}_0 - \sum_{i=1}^r \mathbf{C}_{i0} \bullet \mathbf{Z}_i \\ & \text{subject to} && \mathbf{X} \bullet \mathbf{B}_\alpha + \sum_{i=1}^r \mathbf{Z}_i \bullet \mathbf{C}_{i\alpha} = c_\alpha, \quad \alpha \neq 0 \\ & && \mathbf{X}, \mathbf{Z}_i \succeq 0, \quad i = 1, \dots, m. \end{aligned} \tag{4.20}$$

The matrices $\mathbf{C}_{i\alpha}$ are constructed for each f_i as in Example 4, and the matrices \mathbf{B}_α are constructed as in Example 5.

We will be able to use the programs (4.19)-(4.20) to converge to the optimal solution of our polynomial on the set K by increasing the relaxation order ω . We will begin with a lemma to show that the solutions to the programs of increasing relaxation order ω have the property of monotonicity.

Lemma 3. *The sequence of solutions $\{v_{\text{primal}}(\omega)\}$ of increasing relaxation order ω for the program (4.19) is monotone nondecreasing.*

Proof. Suppose we have programs of order ω and $\tilde{\omega}$, $\omega \geq \tilde{\omega}$, and that a solution to the program of order ω is given by \mathbf{y} . Take the truncated vector $\tilde{\mathbf{y}}$ of \mathbf{y} so that it only includes y_α up to degree $\tilde{\omega}$. Since $\mathbf{z}(\mathbf{x}, \tilde{\omega})^T \mathbf{z}(\mathbf{x}, \tilde{\omega})$ is a principal submatrix of $\mathbf{z}(\mathbf{x}, \omega)^T \mathbf{z}(\mathbf{x}, \omega)$ and thus is a submatrix in the linearization $L_y(\cdot)$, and similarly for $\mathbf{z}(\mathbf{x}, \tilde{\omega})^T \mathbf{z}(\mathbf{x}, \tilde{\omega}) f_i(\mathbf{x})$ and $\mathbf{z}(\mathbf{x}, \omega)^T \mathbf{z}(\mathbf{x}, \omega) f_i(\mathbf{x})$, it follows that $\tilde{\mathbf{y}}$ is a feasible solution for the program of order $\tilde{\omega}$. Since \mathbf{y} provided the *optimal* solution to the program of order ω , the solution to the program of order $\tilde{\omega}$ can be no greater than the solution to the program of order ω . \square

As we did for the unconstrained case, we will show that, under some conditions on K , the duality gap for our programs will close. As previously mentioned, this is a strongly desirable property to have on SDPs.

Theorem 5 (A version of this is noted in Theorem 4.2(a) of [12]). *Assume the dual program (4.20) is solvable at order ω , $K = \{\mathbf{x} : f_i(\mathbf{x}) \geq 0, i = 1, \dots, r\}$ is compact with nonempty interior, and for $i = 1, \dots, r$, f_i is not identically 0. Then there is no duality gap between (4.19) and (4.20).*

Proof. Choose any measure μ with strictly positive density and support contained in K , such that all moments exist up to order 2ω (for example, the uniform probability measure). For f of degree less than 2ω and $f \neq 0$, $\int f(\mathbf{x})^2 \mu(dx) > 0$ and is defined since all moments up to order 2ω exist. Similarly, for $f_i(\mathbf{x}) \geq 0$ on K and some $f \neq 0$ with $2\omega \geq 2\deg(f) + \deg(f_i)$, we have $\int f_i(\mathbf{x})f(\mathbf{x})^2 \mu(dx) > 0$. Note that since K has nonempty interior, $f_i(\mathbf{x})$ must have support on a set of nonzero measure if f_i is not identically 0. Thus in Lemma 1 and 2 we achieve a strict inequality, and the result follows from Theorem 2(ii) on SDP duality. \square

In Theorem 5, the assumption that f_i is not identically 0 is fair, since such a constraint would be entirely vacuous and could be removed from the definition of K .

Theorem 6 (Part of Theorem 4.2(a) in [12], and a particular case of Theorem 3.2 in [14]). *For the POP (4.16), let $f_0(\mathbf{x})$ be a degree d polynomial, and K be the compact set $K = \{\mathbf{x} : f_i(\mathbf{x}) \geq 0, i = 1, \dots, r\}$ for which Condition 1 holds. Let f^* be the minimum value of $f_0(\mathbf{x})$ on K , occurring at point \mathbf{x}^* . Then as $\omega \rightarrow \infty$, then the solutions to (4.19) converge to f^* from below.*

Proof. Take any $\epsilon > 0$. Then $f_0(\mathbf{x}) - f^* + \epsilon$ is strictly positive on K , and so we know that

$$f_0(\mathbf{x}) - f^* + \epsilon = \sum_{i=1}^{r_1} q_i(\mathbf{x})^2 + \sum_{i=1}^r f_i(\mathbf{x}) \sum_{j=1}^{r_i} s_{ij}(\mathbf{x})^2 \quad (4.21)$$

for some polynomials $q_i \in \mathbb{R}[\mathbf{x}, \Omega_0]$, and $s_{ij} \in \mathbb{R}[\mathbf{x}, \Omega_i]$. So, for for some N_0 such that $2N_0 \geq \max\{\Omega_0, \Omega_i + \omega_i (i = 1, \dots, r)\}$, we can show that we have a feasible solution to the dual program (4.20) of order $\omega = N_0$. We do this in a similar fashion to the proof of Theorem 4(i).

Let $\mathbf{q}_i, \mathbf{s}_{ij}$ be the vectors of coefficients corresponding to the polynomials. Now, let $\mathbf{X} = \sum_{i=1}^{r_1} \mathbf{q}_i \mathbf{q}_i^T$, and $\mathbf{Z}_i = \sum_{j=1}^{r_i} \mathbf{s}_{ij} \mathbf{s}_{ij}^T$. Thus the representation (4.21) means we

can write

$$\begin{aligned}
L_{\mathbf{y}}(f_0(\mathbf{x}) - f^* + \epsilon) &= L_{\mathbf{y}}\left(\sum_{i=1}^{r_1} q_i(\mathbf{x})^2 + \sum_{i=1}^r f_i(\mathbf{x}) \sum_{j=1}^{r_i} s_{ij}(\mathbf{x})^2\right) \\
&= \mathbf{X} \bullet \mathbf{M}_{\omega}(\mathbf{y}) + \sum_{i=1}^r \mathbf{Z}_i \bullet \mathbf{M}_{\omega-\omega_i}(f_i \mathbf{y}) \\
&= \mathbf{X} \bullet \mathbf{B}_0 + \sum_{i=1}^r \mathbf{Z}_i \bullet \mathbf{C}_{i0} \\
&\quad + \sum_{\alpha \neq 0} y_{\alpha} \left[\mathbf{X} \bullet \mathbf{B}_{\alpha} + \sum_{i=1}^r \mathbf{Z}_i \bullet \mathbf{C}_{i\alpha} \right].
\end{aligned}$$

But we also have

$$L_{\mathbf{y}}(f_0(\mathbf{x}) - f^* + \epsilon) = -f^* + \epsilon + \sum_{\alpha \neq 0} c_{\alpha} y_{\alpha}.$$

Therefore it follows that the dual program is admissible with \mathbf{X} and \mathbf{Z}_i ($i = 1, \dots, r$) with value $f^* - \epsilon$, and we therefore have that $v_{\text{dual}} \geq f^* - \epsilon$.

Also, using Lemmas 1 and 2, we also know that \mathbf{y} generated by δ_{x^*} is feasible for the primal program (4.19), with value f^* for this \mathbf{y} . Thus $v_{\text{primal}} \leq f^*$. Using Theorem 2(i), we know $v_{\text{dual}} \leq v_{\text{primal}}$, so we have $f^* - \epsilon \leq v_{\text{primal}} \leq f^*$, for all $\epsilon > 0$ at some relaxation order N_0 . By Lemma 3, we know that the sequence of solutions to the primal program in increasing relaxation order ω is monotone nondecreasing. Therefore we have convergence. \square

Theorem 6 has given us a powerful convergence property for our SDPs towards the minimum of our polynomial. However, when we solve these programs in practice, we will use ideas similar to those in [11] (Lemma 5.1), as a way to check optimality. Briefly, the idea is to extract a point $\hat{\mathbf{x}}$ from our solution vector \mathbf{y} to the primal program (4.19), and have it can act as an upper bound on the minimum or, if it is the optimal point, a certificate of optimality. First, we consider the following corollary.

Corollary 1. *Assume the POP (4.16) has optimal value f^* , and the dual program (4.20) is solvable at order ω . If \mathbf{x} is feasible for the POP (4.16) and v_{primal} is the optimal value of (4.19), then $v_{\text{primal}} \leq f_0(\mathbf{x})$, and f^* lies in the closed interval $[v_{\text{primal}}, f_0(\mathbf{x})]$.*

Proof. By definition of the optimal value of the POP (4.16), $f^* \leq f_0(\mathbf{x})$ for all feasible points \mathbf{x} . As in the proof of Theorem 6, we know that $v_{\text{primal}} \leq f^*$. Using these inequalities, we obtain the desired results. \square

Importantly Corollary 1 gives us a way to certify global optimality. If \mathbf{x} is some feasible point for the POP (4.20) and $f_0(\mathbf{x}) = v_{\text{primal}}$, we are assured that \mathbf{x} is an optimal point for the POP, and the global optimum is v_{primal} . Even if $f_0(\mathbf{x}) \neq v_{\text{primal}}$, we are then assured of an interval in which the optimal value must lie. Of course, it may not be easy to find even a single point in the feasible set, let alone the optimal point to certify optimality.

However, we now provide a useful result that our solutions converge to the global minimizer of the problem. In fact, it is the case that if there is a unique global minimizer, the solution vector \mathbf{y} for programs of increasing relaxation orders converges to a sequence generated by $\delta_{\mathbf{x}^*}$ for the optimal point \mathbf{x}^* . In particular, we use that there is convergence of $\mathbf{y}_{\mathbf{e}_i}^j$ to x_i^* . We provide Theorem 7 to formalize this result.

We will say a sequence of solutions is a *nearly optimal* sequence of solutions to the POP (4.16) if, as $r \rightarrow \infty$, $L_{y^r}(f) \rightarrow f^*$. In the case of nearly optimal solutions, we have the following theorem.

Theorem 7 (Theorem 12 of [22] and Theorem 3.6 (c) of [13]). *Assume the POP (4.16) has a unique global minimizer $\mathbf{x}^* \in \mathbb{R}^n$ and let the feasible set K be compact. Let $\{\mathbf{y}^j\}_j$ be nearly optimal solutions to the primal program (4.19) of increasing relaxation orders ω_j . Let $\hat{\mathbf{y}}^j = \mathbf{y}_{\mathbf{e}_i}^j$ for $i = 1, \dots, n$. Then, as $j \rightarrow \infty$, \mathbf{y}^j converges to a sequence generated by the dirac measure $\delta_{\mathbf{x}^*}$, and, in particular*

$$\hat{\mathbf{y}}^j \rightarrow \mathbf{x}^*.$$

Proof. We refer to §3 of [22] for this result and exact details of the convergence. An alternative proof is provided in §4 of [13]. \square

We will use this result in the next section along with Corollary 1 to certify global optimality of our solutions.

Chapter 5

Computing Solutions to Polynomial Optimization Problems

We have demonstrated that through a sequence of SDPs we can converge to the globally optimal value for some POPs. This is a theoretical result, and it is not clear how practicable the method, since we do not know how fast the convergence will happen. In other words, we do not know *a priori* at what relaxation order we would require to achieve a given precision. In practice, however, we find there is very good convergence at low relaxation orders. In this chapter we discuss some issues of computing these in practice.

Note All computations are done on 32-bit machines with 4GB of memory. We use `cvx` [8] to model our programs, and use SeDuMi [23] as our solving software.

5.1 Computational Complexity

In practice we have always found that near-optimal values are found at low relaxation orders for the problems we tried, but as we wish to increase the relaxation in order to approach the optimal value, the biggest burden in the practical computation of a sequence of solutions is the growth of the size of the related SDP. The size of the basis vector $\mathbf{z}(\mathbf{x}, \omega)$ is $\binom{n+\omega}{\omega}$. When we linearize a primal program in the variables y_α , we have $\binom{n+2\omega}{2\omega}$ variables, and the matrices are of size $\binom{n+\omega}{\omega} \times \binom{n+\omega}{\omega}$.

Obviously, this very fast growth rate is a severely limiting factor on our method. In practice, it means we can only solve small to medium sized problems, and we can neither increase the number of variables nor the degree of the polynomials by much before we are unable to solve the problem in practice. However, for problems where the size is tractable, we find that we do not need to increase the relaxation order ω to achieve very good numerical results.

5.1.1 Sparsity Techniques

Since the size of the associated SDPs grow extremely fast, this means that the method is prohibitive for large problems. Sparsity techniques are designed to exploit specific structure in larger problems to reduce the size of the relaxations. We will briefly outline when this technique can be applied, and discuss why it can *not* be used effectively for our ML problem. The idea of sparsity is to use only some of the y_α terms to form the semidefinite matrix inequalities, thereby having a significant savings in the size of the corresponding SDP.

To describe when we can use sparsity techniques for the problem (4.16) where $\mathbf{x} \in \mathbb{R}^n$, we define sets I_k such that

$$I_k \subset \{1, \dots, n\}, \quad (5.1)$$

$$\bigcup_{k=1}^p I_k = \{1, \dots, n\}, \quad (5.2)$$

and where the running intersection property holds, so that

$$\forall k \in \{1, \dots, p-1\}, \exists s \leq k, I_{k+1} \cap (I_1 \cup \dots \cup I_k) \subseteq I_s. \quad (5.3)$$

Let \mathbf{x}_I be the vector of variables indicated by the index set I . Suppose that for each i , we know that polynomial f_i belongs to $\mathbb{R}[\mathbf{x}_{I_j}]$ for some j , and that $f_0 \in \sum_{j=1}^p \mathbb{R}[\mathbf{x}_{I_p}]$. In this case, we can create special “sparse” relaxations of our POP with a significantly smaller size. In these cases, the semidefinite constraints are generated from the y_α terms using only α which correspond to variables indicated by the sets I_k . When these sets I_k are small, this significantly reduces the size of the corresponding programs.

First, we try to understand precisely what kind of sparsity this takes advantage of. In [13], the author describes two types of variable coupling where (5.3) is naturally satisfied, which are *strong* and *weak* coupling.

We say there is strong coupling if, for all $j > 1$, $I_k \cap I_{k+j} = \emptyset$ holds. In this case, the lack of common elements between I_k and I_{k+j} would mean the sets I_k are pairwise disjoint. This would happen in the case where, for $j \neq k$, the variables x_i and x_j never appear as a product in the objective function and do not simultaneously appear in any of the constraint functions.

There is weak coupling if there is a set of coupling variables $\hat{I}_0 \subset \{1, \dots, n\}$, and the set $\{1, \dots, n\} \setminus \hat{I}_0$ can be partitioned into disjoint sets $\hat{I}_1, \dots, \hat{I}_k$ such that $I_k = \hat{I}_0 \cup \hat{I}_k$. This has some similarity to the previous coupling, except that we only require that the variables indicated by two of the sets \hat{I}_k and \hat{I}_j , where $j \neq k$ and $j, k > 0$, do not appear as a product in the objective function and do not simultaneously appear in the constraint function.

Unfortunately, a case where we cannot take advantage of these sparsity techniques is when products between all of the elements of \mathbf{x} appear in the objective function. Since we require $f_0 \in \sum_{j=1}^p \mathbb{R}[\mathbf{x}_{I_j}]$, then some I_j must be equal to $\{1, \dots, n\}$. If this is the case, we have the problem that we need still need semidefinite constraints of size $\binom{n+\omega}{\omega} \times \binom{n+\omega}{\omega}$.

For ML problems, the objective function in our problem has cross terms between all variables. This was seen in Example 1. When calculating the probability conditioned on each root state, cross terms appear between *all* edges of the tree, and since our variables are given on the edges, this limits our use of the technique.

5.2 SDP Solvers

Semidefinite programs are an important type of optimization problem because there are many optimization problems can be framed as a SDPs, and, more importantly, it has been discovered that the programs can be solved efficiently in theory and in practice; indeed in practice we are able to solve programs with thousands of variables on a modern personal computer.

Because of the interest and theory developed for SDPs, excellent software exists for the numerical solution to these programs, for example the solvers `SeDuMi` [23] and `sdpt3` [25]. Because of the cumbersome formatting, there are modelling languages to interface with these solvers; two popular examples are `cvx` [8] and `Yalmip` [15].

Another powerful aspect of this theory applied to POPs is that we do not need to provide the solver with an initial guess to the optimal value of our POP, as we will discuss.

5.2.1 Interior Point Solvers

We will briefly discuss interior point solvers for the computation of solutions to convex programs, although this is itself an actively researched and many variations exist these solvers exists. The concept of these solvers is to generate a sequence of solutions $\mathbf{x}^{(k)}$ and $\mathbf{X}^{(k)}$ to the primal and dual programs, which will approach the optimal solution. From a computational point of view, we can consider the program solved at step k when the duality gap is below a desired precision ϵ_{sol} , so that

$$\eta = \mathbf{c}^T \mathbf{x}^{(k)} - \mathbf{A}_0 \bullet \mathbf{X}^{(k)} \leq \epsilon_{\text{sol}}.$$

A sequence of feasible solutions can essentially “close-in” on the optimal value of the program, since we know the optimal value of the programs must lie in the interval defined by the duality gap, though we do not know where. We visualize this with Figure 5.1.

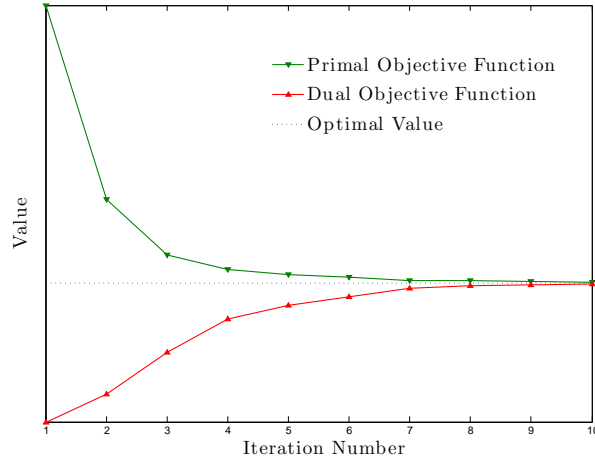


Figure 5.1: A sequence of solutions using a primal-dual SDP solver (hypothetical).

In the following sections we provide an informal description of how general interior point methods work, showing some basic concepts and briefly describing some others. We intend to outline the main features of why they work.

5.2.1.1 Newton's Method

To begin our understanding, we will consider the unconstrained minimization problem

$$\text{minimize } f(\mathbf{x})$$

where $f \in C^2$ with domain $\mathbf{x} \in \mathbb{R}^n$. We will present a method to find a minimum of f using Newton's method. We write the Hessian of f as $H(f)$, and we assume that $H(f) \in \mathcal{S}_{++}^n$. If we take any starting point $\mathbf{x}^{(0)}$ in the domain of f , then the iterates for the Newton steps are

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - H^{-1}(f(\mathbf{x}^{(k)}))\nabla f(\mathbf{x}^{(k)}). \quad (5.4)$$

There are two main reasons for why this method is useful. First, the method displays transform invariance in the coordinates, unlike a gradient method. Second, if we start at a point $x^{(0)}$ sufficiently close to a minimum of f , and certain conditions on f are satisfied, then it is possible for the method to display a quadratic convergence rate in a neighbourhood of the minimum, therefore limiting the number of iterations needed for very good convergence. If f is convex and has a global minimum, then this method will also display global convergence.

The Newton method, of course, can not be directly applied to all minimization problems and clearly does not account for any constraints we might add to the minimization. However, we will soon see why it is useful because of the properties of convex functions and the introduction of special a barrier function.

5.2.1.2 Barrier/Interior-Point Methods

We will now introduce the the concept of a barrier function. The idea of introducing this barrier function will be to transform the constrained optimization problem into a sequence of unconstrained problems we *can* solve, and for which the minimizer converges to the minimum of our original program. The barrier function can be seen as way to ensure we respect the constraint functions, so that we stay on the interior of the feasible set — hence the name for the interior-point method.

A barrier function ϕ is defined on the interior of the feasible set K of our program, which we assume is convex. We will require that our barrier function ϕ is convex and C^2 , that as $x_0 \rightarrow \partial K$, $\phi(\mathbf{x}) \rightarrow \infty$, and that the barrier function is extended outside of the feasible set K so that $\phi(K^C) = \infty$.

Now, for the problem

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{x} \in K \end{aligned} \tag{5.5}$$

which we assume is convex, we consider a related unconstrained optimization problem parameterized in ν ,

$$\text{minimize} \quad \nu \mathbf{c}^T \mathbf{x} + \phi(\mathbf{x}). \tag{5.6}$$

Because the barrier function ϕ is convex, the objective function for the problem (5.6) is also convex. And since local minimization of a convex function implies global minimization, if we can successfully apply Newton’s method to the problem in this case, then we have found the global minimum for the problem (5.6).

For fixed ν , neither the minimum nor the minimizing point of the new problem (5.6) is the same as the original problem (5.5). However, as the parameter ν is increased, the effect of the barrier function relative to the original function is decreased, and the problem we are solving seems to become closer to the problem (5.5).

Indeed, provided the sublevel sets $\{\mathbf{x} : \mathbf{c}^T \mathbf{x} \leq l\}$ are bounded for some $l \geq 0$, the optimal solutions $\mathbf{x}(\nu) = \operatorname{argmin}_{\mathbf{x}}(\nu \mathbf{c}^T \mathbf{x} + \phi(\mathbf{x}))$ exist for (5.5) for some ν_0 for $\nu > \nu_0$. Further, it traces out a continuous path in the feasible set K such that, as $t \rightarrow \infty$, $x(\nu) \rightarrow x^*$, where x^* is the optimal value of the original program (5.5). This is known the *central path*.

If we start at a point on this central path and increase ν by a small enough amount, then we know the next optimal point will be close to the starting point.

We can therefore use Newton's method to solve (5.5) for this increased value of ν , and ideally be close enough that we will obtain a rapid convergence rate. Then, we repeat the process and continue following the central path. We provide the informal description of the algorithm for a program of the form (5.6) in the following steps, which we have taken from [9]:

1. Take \mathbf{x}_0 near some point $\mathbf{x}(\nu_0)$ on the central path. Set $k = 0$.
2. If \mathbf{x}_k is sufficiently close to the boundary, then stop. Otherwise, set $\nu_{k+1} = \delta\nu_k$, where $\delta > 1$.
3. Apply a Newton step to $\nu\mathbf{c}^T\mathbf{x} + \phi(\mathbf{x})$. Increment k and return to Step 2.

To illustrate the concept concretely, we provide Example 6. The example in this case is trivial, but it is useful because we can easily visualize the method in two dimensions. We use a logarithmic barrier in this case.

Example 6 (Using a Barrier Function). *Consider the function $f(\mathbf{x}) = 5x_1 + 3x_2$, and suppose we wish to minimize this, constrained by $x_1, x_2 \geq 0$, $x_2 \leq 1$, and $x_2 \leq -x_1 - 2$. The minimum value is at $(0, 0)$. Let f_i represent the strict constraints for the interior of our feasible set such that $f_i(\mathbf{x}) > 0$. We form the barrier function $\phi(\mathbf{x}) = \log(f_1(\mathbf{x})f_2(\mathbf{x})f_3(\mathbf{x})f_4(\mathbf{x}))$.*

We can use an interior-point type method to solve this problem. We estimate an initial point $(0.82, 0.36)$, and minimize $\nu f(\mathbf{x}) + \phi(\mathbf{x})$ for successive values of ν , and find that we are able to approach the optimum point $(0, 0)$.

In Figure 5.2, we can see how the level sets of the function follow the boundary of the domain. The central path found is shown as the dashed curve, and level curves for $\nu = 0, 1, 5$ are shown. We can also see that the minimum value of the function is tracing out the central path.

In Example 6 the initial point was easy to estimate. In general there is something known as the *Phase I* method to find a feasible solution. For this method, we apply a barrier method to an auxiliary problem. For this auxiliary problem, we always have an initial point which is feasible. This auxiliary problem has the feature that its optimal solution is guaranteed to be feasible for the original problem, unless the original problem is infeasible, in which case the Phase I method provides us with a certificate of infeasibility.

We also note that there is a special class of C^3 barrier functions on convex sets known as *self-concordant* barriers which provide further useful properties. In fact, these allow us to bound the complexity of the process in polynomial time if we wish to solve the original problem to some predetermined precision.

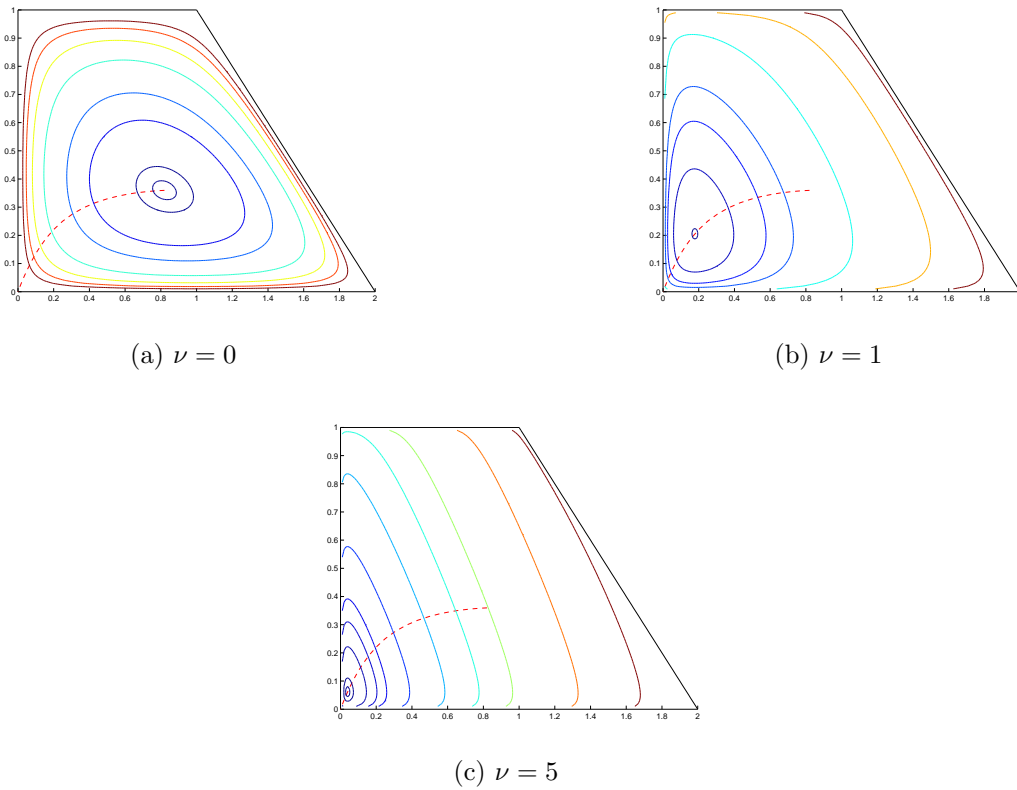


Figure 5.2: Level curves of $\nu f(\mathbf{x}) + \phi(\mathbf{x})$ for varying values of ν . The whole central path is shown as the dashed curve in each subfigure.

The other notable feature is that points on the central path can yield dual feasible solutions. Remarkably, however, we can bound sub-optimality of the value of the program at a point $\mathbf{x}(\nu)$ on the central path for either the primal or dual program, using only the parameter ν and a parameter derived from the barrier function ϕ .

Finally, we mention a self-concordant function ϕ on the cone \mathcal{S}_{++} for $\mathbf{X} \in \mathcal{S}_{++}$ is given by

$$\phi(\mathbf{X}) = \log \det(\mathbf{X}),$$

which can be used to apply these barrier methods to SDPs.

5.3 Computing Solutions to POPs

5.3.1 Measuring Solution Accuracy

We define two measurements for the solution accuracy for the problem

$$\begin{aligned} & \text{minimize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \geq 0 \quad i = 1, 2, \dots, m, \\ & && h_i(\mathbf{x}) = 0 \quad i = 1, 2, \dots, n. \end{aligned}$$

When we use the approach described and find a solution \mathbf{y} to the SDP, we consider the linear variables $y_{\mathbf{e}_i}$ as a candidate solution $\hat{\mathbf{x}}$, so that $\hat{x}_i = y_{\mathbf{e}_i}$. We assume here that f_0 has a unique minimum on the feasible set.

The following are only slightly modified versions of the definitions used by the authors in [28] when solving POPs.

Our first measurement is on the objective function f_0 of our POP.

$$\epsilon_{\text{obj}} = \frac{|\mathbf{c}^T \mathbf{y} - f_0(\hat{\mathbf{x}})|}{\max\{1, |f_0(\hat{\mathbf{x}})|\}}. \quad (5.7)$$

When the program reaches exactly the optimal value of our POP, and the estimator $\hat{\mathbf{x}}$ is an optimal point, then ϵ_{obj} will achieve the value 0. Otherwise, we see it as a measurement of the accuracy of our solution; when the exact evaluation of f_0 approaches the bound given by $\mathbf{c}^T \mathbf{y}$, ϵ_{obj} is small. Notice that the definition ϵ_{obj} is such that it will be defined even for $f_0(\hat{\mathbf{x}}) = 0$, but this means that for optimal solutions which are very small, one must pay close attention to the orders of magnitude on ϵ_{obj} and compare magnitudes to those of $\mathbf{c}^T \mathbf{y}$ and $f_0(\hat{\mathbf{x}})$.

This measure can be used for an unconstrained POP, but in the constrained case it only makes sense if the value \mathbf{x} satisfies the constraints. We define a second measurement which we use for the constraint equations. It is defined by

$$\epsilon_{\text{feas}} = \max\{-f_i(\hat{\mathbf{x}}) \ (i = 1, \dots, m), |h_j(\hat{\mathbf{x}})| \ (j = 1, \dots, n), 0\}. \quad (5.8)$$

Whenever we have a feasible point \mathbf{x} for our POP, ϵ_{feas} will achieve the value 0. Otherwise, we interpret it as a measure of how well we the constraint equations have been met.

5.3.2 Understanding Suboptimal Solutions

When we are solving a sequence of programs of increasing relaxation order for a constrained POP, then, if the programs are solvable, we will find a lower bound on the minimum value of the objective function. However, it is not guaranteed that the value $\hat{\mathbf{x}}$ extracted from \mathbf{y} corresponds to a point \mathbf{x} in the feasible set, but we know it will converge to the optimum value. When $\hat{\mathbf{x}}$ yields ϵ_{feas} or $\epsilon_{\text{obj}} \gg 0$, this is an indicator we are not close to the optimal solution.

When find a suboptimal solution, then we can simply increase the relaxation order ω to $\omega + 1$ and solve the corresponding programs. By increasing the relaxation order, we can converge to an optimal solution, and, additionally, the value $\hat{\mathbf{x}}$ converges to the global optimum, assuming it is unique.

We summarize how we can interpret the optimal values with respect to the measures in Table 5.1, and provide an example of a program converging at increasing relaxation orders.

Feasibility ϵ_{feas}	Objective ϵ_{obj}	Lower Bound $\mathbf{c}^T \mathbf{y}$	Upper Bound $f(\hat{\mathbf{x}})$	Information About Optimal Value
Large ($\gg 0$)	Large ($\gg 0$)	Valid	Invalid	Lower Bound Only
Large ($\gg 0$)	Small (≈ 0)	Valid	Invalid	Lower Bound Only
Small (≈ 0)	Large ($\gg 0$)	Valid	Valid	Certified Range
Small (≈ 0)	Small (≈ 0)	Valid	Valid	Certified Value

Table 5.1: Interpretations of the measures ϵ_{feas} and ϵ_{obj} when solving a POP, assuming the programs are solvable.

Example 7 (A Sequence of Programs to Solve a POP. Problem 4.6 in [6], and also an example in [12]). *Consider the POP*

$$\begin{aligned}
& \text{minimize} && -x_1 - x_2 \\
& \text{subject to} && 2x_1^4 - 8x_1^3 + 8x_1^2 - x_2 + 2 \geq 0, \\
& && 4x_1^4 - 32x_1^3 + 88x_1^2 - 96x_1^2 - x_2 + 36 \geq 0, \\
& && 0 \leq x_1 \leq 3, \\
& && 0 \leq x_2 \leq 4.
\end{aligned} \tag{5.9}$$

We solve this problem with relaxation orders $\omega = 2, \dots, 5$. The results are shown in Table 5.2. In this case, ϵ_{obj} is always 0 because our objective function is linear in

Relaxation Order ω	ϵ_{feas}	ϵ_{obj}	Lower Bound $\mathbf{c}^T \mathbf{y}$
2	4.000×10^0	0	-7
3	4.000×10^0	0	-7
4	2.695×10^0	0	-6.653
5	2.400×10^{-7}	0	-5.508

Table 5.2: Computed solutions to the problem (5.9).

each x_i , but since this is a constrained problem we are concerned with the value of ϵ_{feas} . Notice that for relaxation orders $\omega < 5$, the bound on our polynomial is quite good but our candidate solutions are well outside of the feasible set, until at $\omega = 5$ we find a good solution $x_1 \approx 2.3295$, $x_2 \approx 3.1785$.

5.3.3 Equality Constraints

Equality constraints will introduce new problems into computing our solutions, because they introduce an empty interior on the feasible set of our problem.

One way around this is to relax the equality constraints slightly. For example, the equality constraint $h_i(\mathbf{x}) = 0$ can be relaxed to $-\epsilon \leq h_i(\mathbf{x}) \leq \epsilon$, and then the relaxed equality constraints can be treated as two inequality constraints. This relaxation can

help us to find a point in a neighbourhood around the optimum value. In the next section when we use this technique, we choose the value $\epsilon = 5 \times 10^{-8}$.

However, there is another way we can formulate the program and allow the solver to handle the equality constraints in practice, because the solver **SeDuMi** allows us to use a program formulation with free variables which we find can provide better numerical results. To develop this formulation, we can first notice that the constraint

$$L_y (\mathbf{z}(\mathbf{x}, m)\mathbf{z}(\mathbf{x}, m)^T\theta(\mathbf{x})) = \mathbf{O},$$

generated from a constraint $\theta(\mathbf{x}) = 0$ implies that every entry in this matrix must be zero. Therefore, since the entries of the matrix $\mathbf{z}(\mathbf{x}, m)\mathbf{z}(\mathbf{x}, m)^T$ are given by $\mathbf{z}(\mathbf{x}, 2m)$, an equivalent formulation of this is

$$\mathbf{z}(\mathbf{x}, 2m)\theta(\mathbf{x}) = 0.$$

When we linearize this, it becomes a set of linear equalities in our primal program, for each entry of this vector. So, in the primal program, we have an equality of the form

$$\mathbf{G}\mathbf{y} = 0.$$

When we model these directly in our primal program, we therefore obtain

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T\mathbf{y} \\ & \text{subject to} && \mathbf{A}_0 + \sum_{i=1}^m y_i\mathbf{A}_i \succeq 0, \\ & && \mathbf{G}\mathbf{y} = 0. \end{aligned} \tag{5.10}$$

If $\mathbf{G} \in \mathbb{R}^{q \times r}$, where r is the size of \mathbf{y} and q is the number of equality constraints, then the dual associated with this program is

$$\begin{aligned} & \text{maximize} && -\mathbf{A}_0 \bullet \mathbf{X} \\ & \text{subject to} && \mathbf{A}_i \bullet \mathbf{X} - \mathbf{g}_i^T \mathbf{f} = c_i, \quad i = 1, 2, \dots, m, \\ & && \mathbf{X} \succeq 0, \\ & && \mathbf{f} \in \mathbb{R}^q, \end{aligned} \tag{5.11}$$

where \mathbf{g}_i^T denotes the i^{th} column of the matrix \mathbf{G} , and \mathbf{f} is the free variable. We will not show this conversion to dual form in detail, but it is a special case of taking the Lagrangian dual.

5.3.4 Multiple Optima

We will be interested in not just finding a bound on the optimal value of our POP, but also in an optimal point \mathbf{x}^* . As before, we consider the linear variables $y_{\mathbf{e}_i}$ as a candidate solution $\hat{\mathbf{x}}$, so that $\hat{x}_i = y_{\mathbf{e}_i}$.

When our objective function has multiple optima, we are presented with the problem that the solution vector \mathbf{y} will not necessarily correspond to any particular optimum point \mathbf{x}^* .

However, we can ensure that our objective function f_0 has a unique optimal solution by replacing it with $\hat{f}_0(\mathbf{x}) = f_0(\mathbf{x}) + l_\epsilon(\mathbf{x})$, where $l_\epsilon(\mathbf{x}) = \mathbf{I}_\epsilon^T \mathbf{x}$ is a small random linear perturbation. Note that this shifts very slightly both the optimal value and *where* the optimal value occurs, but, on the feasible set, we will still converge to a very near optimal solution.

When we have added a perturbation, we use the measure for the objective function

$$\epsilon_{\text{obj}} = \frac{|\mathbf{c}^T \mathbf{y} - \hat{f}_0(\hat{\mathbf{x}})|}{\max\{1, |\hat{f}_0(\hat{\mathbf{x}})|\}}. \quad (5.12)$$

Example 8 (Solution Extraction With Multiple Optima). *Consider the POP*

$$\begin{aligned} \text{minimize} \quad & f_0(\mathbf{x}) = (x_1^2 - 2)^2 + (x_2^2 - 2)^2 - x_1 x_2 \\ \text{subject to} \quad & f_1(\mathbf{x}) = \|\mathbf{x}\|^2 - 1 \geq 0. \end{aligned} \quad (5.13)$$

The objective function in this problem has nonunique optima at $\mathbf{x}_1^ = (1.5, 1.5)$ and $\mathbf{x}_2^* = (-1.5, -1.5)$, and takes on the optimum value -2.125 . The function also has two local minima away from the global minima, and one local maximum. The feasible set is \mathbb{R}^2 take away the unit ball centered at the origin, and is not convex.*

When we solve the corresponding SDP relaxation problem without a perturbation, we find the lower bound -2.125 at the relaxation order $\omega = 2$, with corresponding candidate solution $\mathbf{x} \approx (0.205 \times 10^{-7}, 0.205 \times 10^{-7})$, which is not in the feasible set. In this case, the point corresponds to an approximately equal convex combination of the optimal points. Evaluated at this point, the function takes on value ≈ 8 .

We add a random linear perturbation function to our objective function. For the perturbation we choose elements of \mathbf{I}_ϵ as random numbers selected from a uniform distribution on the interval $[-\epsilon, \epsilon]$. We find an appropriate ϵ through some experimentation, and choose $\epsilon = 1 \times 10^{-6}$.

With the random perturbation added, we solve the SDP relaxation of order $\omega = 2$, and find the bound -2.125 and candidate solution $\hat{\mathbf{x}} \approx (-1.500, -1.500)$, with $\epsilon_{\text{obj}} = 1.645 \times 10^{-11}$, and $\epsilon_{\text{feas}} = 0$. Solving with a different random perturbation, we find $\epsilon_{\text{obj}} = 2.550 \times 10^{-8}$ with candidate solution $\hat{\mathbf{x}} \approx (1.500, 1.500)$, corresponding to the other optimum.

We require this technique for some cases in the next section, and used a perturbation of $\mathcal{O}(10^{-6})$ in each case.

5.3.5 Improving The Estimator

In Example 8, we provided an example of how adding a perturbation may allow us to find a good estimator of an optimal point for the problem (5.13). In this case, the optimum values of the POP also happen to be minima of the objective function, and in this problem we could try using an appropriate gradient descent method in an attempt to improve on the first estimator found at relaxation order $\omega = 2$.

Note that if the SDP can be solved, then value of the SDP, given by $\mathbf{c}^T \mathbf{y}$, always provides a lower bound on the optimum value of the POP. But no matter how an estimator $\hat{\mathbf{x}}$ is found, if we are able to reduce ϵ_{feas} and ϵ_{obj} to 0 (or near 0) we are still provided with a certificate of optimality. This justifies the use of a technique to improve the estimator $\hat{\mathbf{x}}$, and one could use *any* heuristic that works well for a given problem. Note that on a constrained problem, an optimal point \mathbf{x}^* need not be at a minimum of the objective function.

We are interested in moving our estimator $\hat{\mathbf{x}}$ to be in (or nearly in) the feasible point to the feasible set. To improve our candidate solution $\hat{\mathbf{x}}$, we find several algorithms are effective, but use MATLABs nonlinear optimization tool `fmincon` with the active-set algorithm. Of course, the tool requires a very good initial guess that we provide it, and, in contrast to our method, provides no information about global optimality.

Using this method, we are able to reduce ϵ_{feas} to desired thresholds when solving our problems in the next section. We only need this technique in a small number of cases — but in such cases we are able to reduce our measures by several orders of magnitude, and it is important to reduce ϵ_{feas} to near 0 if we want the value $f_0(\hat{\mathbf{x}})$ to act as either an upper bound on the found minimum, or a certificate of optimality.

Chapter 6

ML Ancestral State Reconstruction

The motivation for using these polynomial optimization methods was to solve the ML problem for assignment of an ancestral state on a phylogenetic tree. Recall that to assign the ancestral state, we compute $\mathbb{L}^*(\chi|\rho = \alpha)$ for each value of α , and assign to the root the state α which achieves a highest value of $\mathbb{L}^*(\chi|\rho = \alpha)$.

Our ML method requires a maximization, but we have so far discussed POPs as minimization problems. Of course, we can very easily transform a problem of maximizing f into a minimization one by simply forming a minimization problem where we minimize $-f$. Then, we transform the optimal value appropriately. Note that we previously formed a lower bound on the minimum, but under this transformation we form an upper bound on the maximum, and, similarly, the value of $f(\hat{\mathbf{x}})$ is now treated as a lower bound on the maximum.

Our polynomial optimization method allows us to compute the value of $\mathbb{L}^*(\chi|\rho = \alpha)$ for each root state α , by finding a lower bound with $f(\hat{\mathbf{x}})$ and upper bound \mathbb{L}_{\max} from the optimization method. In every case, we have been able to certify that this bound is optimal within a small numerical error, or to find a very small range in which it lies. If we can only find a range in which the value lies, this still means we can select the value, since if $\mathbb{L}^*(\chi|\rho = \alpha_1)$ has an upper bound which is lower than the lower bound on $\mathbb{L}^*(\chi|\rho = \alpha_2)$, then we know the assignment must be α_2 . In only a small number of cases, the intervals overlapped, and we consider these to be equal if they overlap on an interval of size $\mathcal{O}(10^{-5})$ or less. This allows us to use these values to infer the root state.

Fortunately, all the problems we solved with our optimization methods demonstrated very good convergence, and could be solved to very high accuracy at low orders.

6.1 Results

6.1.1 How Results Are Presented

We present results of the calculations for each tree topology considered under the N_2 model with the character states $\mathcal{C} = \{R, Y\}$. The topologies on which we solve the problem are shown in Figures 6.2 and 6.3. The corresponding site pattern can be represented as a string sequence for each tree, where each element in the sequence represents the character state at the leaf nodes, assigned the reader's left to right.

We only show results for trees that are certain to be distinct. For example, in Figure 6.1, the maximum likelihood for the site pattern $RRRY$ will clearly be equivalent to the site pattern $YRRR$, since the trees are simply isomorphisms of each other. We therefore only need to compute the result for the pattern $RRRY$.

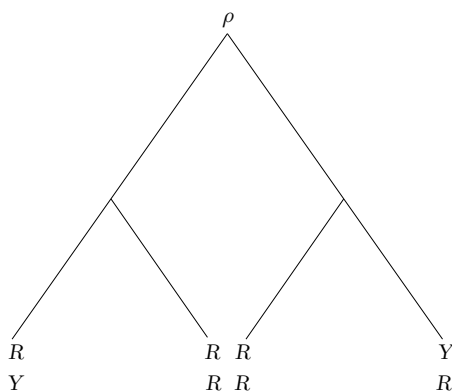
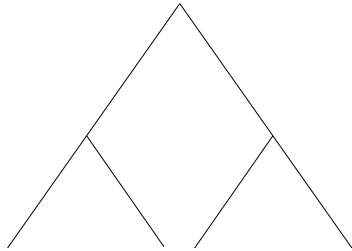


Figure 6.1: Two different characters that must have the same maximum likelihood value.

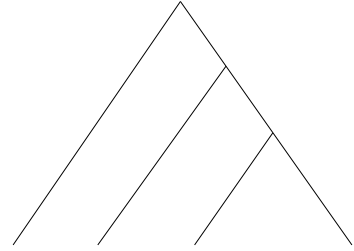
All calculations of the likelihood are presented for the computation conditioned on $\rho = R$. To determine the value conditioned on $\rho = Y$, we need only to look at the value of the likelihood when condition on $\rho = R$ but with every element of the character pattern at the leaves reversed, and this is due to the symmetry of the N_2 model.

6.1.2 Computed Results

The results we have computed are shown in Tables 6.1 through 6.5. The implied assignments are shown in Tables 6.6 through 6.10. The equality constraints needed to be relaxed in only one case for each tree, but in these cases the ML value is easily seen analytically, and in these cases our results are still quite good.

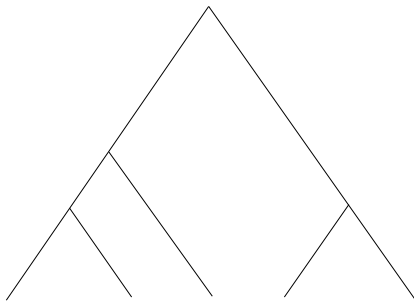


(a) Four-leaf $[[,],[,]]$ topology

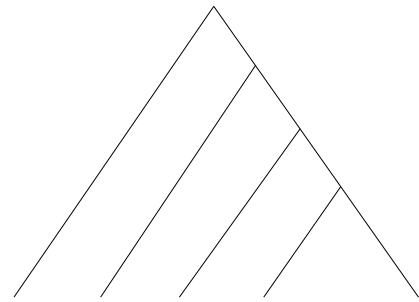


(b) Four-leaf “comb” $[,[[,]]]$ topology

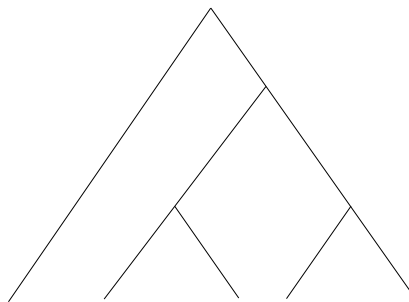
Figure 6.2: Four-leaf topologies on which we solve the ML problem.



(a) Five-leaf $[[[,],[,]]]$ topology



(b) Five-leaf “comb” $[,[[,[,]]]$ topology



(c) Five-leaf $[,[[,[,]]]$ topology

Figure 6.3: Five-leaf topologies on which we solve the ML problem.

6.1.3 Discussion

We have been able to solve for the maximum likelihood values for all characters and on all four and five leaf topologies. In all of the four leaf cases, the assignments from the MP and ML method agree. This was also the case for the five-leaf trees in all but

two cases where we were able to find characters where the methods do not agree on the ancestral state reconstruction. Both of these examples occurred on the five-leaf “comb” topology.

The result where the reconstructions disagree is counterintuitive and surprising. For example, in Figure 6.4b, while we only observe the character Y once, the ML method makes an ambiguous selection for the ancestral root state. Intuition suggests that it would not be the case that both root states were equally likely and that this is an unrealistic reconstruction.

We also notice that in these cases, the likelihood has been maximized at limiting values of 0 and 1/2 on each edge. From our model, we can see that values of 1/2 correspond to edge lengths which are increasing in length t , and values of 0 correspond to edge lengths which are decreasing to length 0. Thus, there are trees with long branch lengths for which the ML method makes a seemingly unrealistic reconstruction if the N_2 model is adopted with a molecular clock constraint.

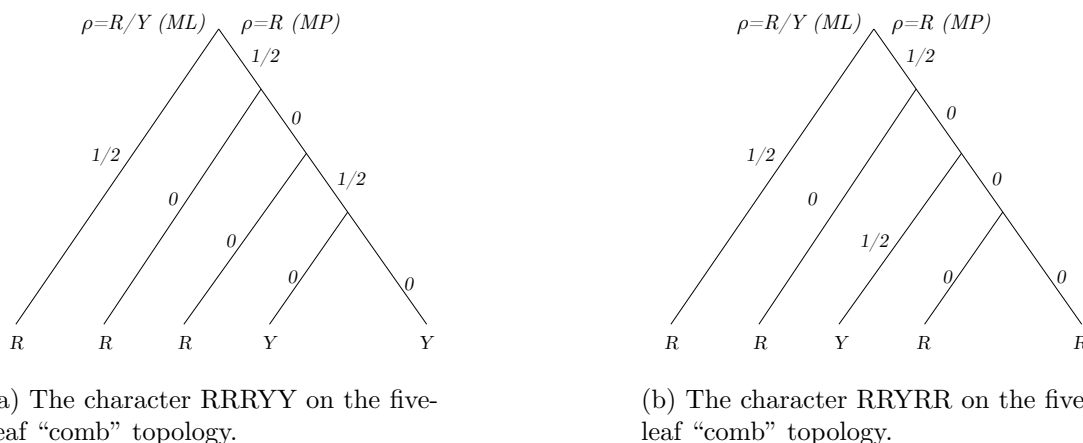


Figure 6.4: Examples where the MP and ML method do not make the same reconstruction. The computed values have been rounded from the computation on this labelling.

6.1.4 Optimizing for Other Models and Trees

In Section 5.1.1, we examined that the techniques of sparsity are not well-suited to our problem on phylogenetic trees. In practice, we find that only trees of up to five leaves can reasonably be solved. The most computationally difficult polynomial is the five-leaf “comb” tree. On this tree, there are 7 distinct edge length lengths, and the longest path in the tree is of length 5. Therefore, in the associated polynomial we have 7 variables and a degree 5 polynomial; this would require a relaxation order of $\omega \geq 3$ in our method. However, we required a relaxation order of $\omega = 4$ to solve the

problem. So, in this case our primal program has $\binom{n+2\omega}{2\omega} = 6435$ variables. Also, for each inequality in the POP we require a semidefinite matrix of size $\binom{n+\omega}{\omega} \times \binom{n+\omega}{\omega}$.

Clearly it is a challenge to increase the number of leaves. For example, with 6 leaves, a similar “comb” topology and under the same model would have 9 variables and a degree 6 polynomial, requiring a relaxation order of $\omega \geq 3$. However, as in the case of the five-leaf tree, we would expect to require the relaxation order of at least 4. This would require $\binom{n+2\omega}{2\omega} = 24310$ variables in the primal program.

This growth size also limits the use of the method for asymmetric models on our tree. Because we applied this to the N_2 model with symmetric transition rates between states, the transition probability matrices on each edge had the form

$$\mathbf{P}(e_i) = \begin{bmatrix} 1 - p_i & p_i \\ p_i & 1 - p_i \end{bmatrix},$$

and, as we can see, this has only one probability variable p_i associated with all possible changes on the edge. However, models of transitions with asymmetric transitions between the states are also of interest, but this immediately doubles the number of variables in our problem, since the transition probability matrices in a two state model now have the form

$$\mathbf{P}(e_i) = \begin{bmatrix} 1 - p_i & p_i \\ q_i & 1 - q_i \end{bmatrix}.$$

On the positive side, this method may work well to examine the symmetric N_4 model, which is used as a model to represent the evolution of full DNA sequences on phylogenies. This is because we still only obtain a single variable associated with the transition probability on each edge. Therefore, the number of variables in our the objective function for the associated POP would not increase. In general, however, the optimization method for the ML method seems to be limited to a small number of problems and on topologies with only a small number of leaves.

Site Pattern				Lower Bound	Upper Bound	Order	Measures	
1	2	3	4	$f(\hat{\mathbf{x}})$	\mathbb{L}_{\max}	ω	ϵ_{feas}	ϵ_{obj}
R	R	R	R	1.000	1.000‡	3	1.473×10^{-13}	2.126×10^{-12}
R	R	R	Y	0.1055	0.1055	3	1.692×10^{-10}	3.761×10^{-10}
R	R	Y	Y	0.1481	0.1481	3	4.821×10^{-12}	1.310×10^{-11}
R	Y	R	R	0.1481	0.1481	3	3.945×10^{-12}	1.495×10^{-11}
R	Y	R	Y	0.06250	0.06250	3	3.165×10^{-09}	3.431×10^{-10}
R	Y	Y	Y	0.2500	0.2500	3	1.129×10^{-10}	1.702×10^{-10}
Y	R	R	R	0.2500	0.2500	3	1.431×10^{-10}	2.924×10^{-11}
Y	R	R	Y	0.06250	0.06250	3	7.581×10^{-09}	6.441×10^{-10}
Y	R	Y	Y	0.1250†	0.1251	3	5.551×10^{-17}	6.007×10^{-5}
Y	Y	R	R	0.1250†	0.1251	3	0×10^0	6.008×10^{-5}
Y	Y	R	Y	0.06250†	0.06250	3	8.455×10^{-11}	9.942×10^{-12}
Y	Y	Y	Y	0.2500	0.2500	3	7.802×10^{-12}	1.225×10^{-11}

Table 6.1: Results for computation of the likelihood function on the four-leaf “comb” topology $[[:,[::]]]$, conditioned on $\rho = R$. The dagger † indicates values where we used a local optimization technique to improve the estimator $\hat{\mathbf{x}}$, and the double dagger ‡ indicates a point where we relaxed the equality constraint.

Site Pattern				Lower Bound	Upper Bound	Order	Measures	
1	2	3	4	$f(\hat{\mathbf{x}})$	\mathbb{L}_{\max}	ω	ϵ_{feas}	ϵ_{obj}
R	R	R	R	1.000	1.000‡	3	1.110×10^{-16}	3.974×10^{-14}
R	R	R	Y	0.1481	0.1481	3	5.560×10^{-09}	4.911×10^{-10}
R	R	Y	Y	0.2500	0.2500	3	3.806×10^{-10}	1.562×10^{-10}
R	Y	R	Y	0.06250	0.06250	3	6.351×10^{-11}	1.672×10^{-10}
R	Y	Y	Y	0.1250†	0.1250	3	1.337×10^{-33}	7.224×10^{-11}
Y	Y	Y	Y	0.2500	0.2500	3	4.136×10^{-16}	2.363×10^{-09}

Table 6.2: Results for computation of the likelihood function on the four-leaf topology $[[:,[::]]]$, conditioned on $\rho = R$. The dagger † indicates values where we used a local optimization technique to improve the estimator $\hat{\mathbf{x}}$, and the double dagger ‡ indicates a point where we relaxed the equality constraint.

Site Pattern					Lower Bound	Upper Bound	Order	Measures	
1	2	3	4	5	$f(\hat{\mathbf{x}})$	\mathbb{L}_{\max}	ω	ϵ_{feas}	ϵ_{obj}
R	R	R	R	R	1.000	1.000‡	4	6.785×10^{-11}	4.187×10^{-10}
R	R	R	R	Y	0.1481	0.1481	4	3.281×10^{-12}	2.138×10^{-12}
R	R	R	Y	Y	0.2500	0.2500	4	6.518×10^{-12}	4.227×10^{-13}
R	R	Y	R	R	0.1481	0.1481	4	4.335×10^{-13}	7.129×10^{-13}
R	R	Y	R	Y	0.06250†	0.06251	4	5.551×10^{-17}	6.963×10^{-6}
R	R	Y	Y	Y	0.1250†	0.1251	4	4.804×10^{-35}	1.317×10^{-4}
R	Y	R	R	R	0.1055	0.1055	4	2.929×10^{-10}	1.999×10^{-10}
R	Y	R	R	Y	0.03456	0.03456	4	4.590×10^{-12}	1.956×10^{-13}
R	Y	R	Y	Y	0.06250	0.06250	4	1.934×10^{-10}	2.417×10^{-12}
R	Y	Y	R	R	0.06250†	0.06250	4	8.616×10^{-17}	2.320×10^{-13}
R	Y	Y	R	Y	0.03125†	0.03125	4	6.938×10^{-17}	1.489×10^{-7}
R	Y	Y	Y	Y	0.06250	0.06250	4	3.426×10^{-12}	7.736×10^{-13}
Y	Y	R	R	R	0.1481	0.1481	4	1.563×10^{-12}	2.000×10^{-12}
Y	Y	R	R	Y	0.06250†	0.06251	4	5.551×10^{-17}	6.443×10^{-6}
Y	Y	R	Y	Y	0.1250†	0.1250	4	4.593×10^{-33}	1.445×10^{-5}
Y	Y	Y	R	R	0.2500	0.2500	4	4.587×10^{-14}	6.054×10^{-13}
Y	Y	Y	R	Y	0.1250	0.1250	4	1.558×10^{-11}	1.764×10^{-13}
Y	Y	Y	Y	Y	0.2500	0.2500	4	1.137×10^{-14}	3.335×10^{-13}

Table 6.3: Results for computation of the likelihood function on the five-leaf topology $[[[[],[]],[]],[]]$, conditioned on $\rho = R$. Each problem here is solved in approximately 15 minutes. The dagger † indicates values where we used a local optimization technique to improve the estimator $\hat{\mathbf{x}}$, and the double dagger ‡ indicates a point where we relaxed the equality constraint.

Site Pattern					Lower Bound	Upper Bound	Order	Measures	
1	2	3	4	5	$f(\hat{\mathbf{x}})$	\mathbb{L}_{\max}	ω	ϵ_{feas}	ϵ_{obj}
R	R	R	R	R	1.000	1.000‡	4	2.830×10^{-11}	2.335×10^{-9}
R	R	R	R	Y	0.1055	0.1055	4	5.770×10^{-10}	4.970×10^{-11}
R	R	R	Y	Y	0.1481	0.1481	4	1.267×10^{-10}	7.190×10^{-10}
R	R	Y	R	Y	0.03455	0.03455	4	4.204×10^{-10}	1.137×10^{-11}
R	R	Y	Y	Y	0.06250†	0.06250	4	5.551×10^{-17}	1.204×10^{-10}
R	Y	Y	Y	Y	0.2500	0.2500	4	4.549×10^{-11}	8.525×10^{-11}
Y	R	R	R	R	0.2500	0.2500	4	4.548×10^{-11}	8.525×10^{-11}
Y	R	R	R	Y	0.06250†	0.06250	4	3.673×10^{-40}	1.454×10^{-11}
Y	R	R	Y	Y	0.1250†	0.1250	4	5.551×10^{-17}	1.392×10^{-5}
Y	R	Y	R	Y	0.03125†	0.3125	4	5.551×10^{-17}	9.776×10^{-12}
Y	R	Y	Y	Y	0.06250†	0.06250	4	0×10^0	1.685×10^{-9}
Y	Y	Y	Y	Y	0.2500	0.2500	4	8.504×10^{-11}	1.582×10^{-10}

Table 6.4: Results for computation of the likelihood function on the five-leaf topology $[[[[],[]],[]],[]]$, conditioned on $\rho = R$. Each problem here is solved in approximately 15 minutes. The dagger † indicates values where we used a local optimization technique to improve the estimator $\hat{\mathbf{x}}$, and the double dagger ‡ indicates a point where we relaxed the equality constraint.

Site Pattern					Lower Bound	Upper Bound	Order	Measures	
1	2	3	4	5	$f(\hat{\mathbf{x}})$	\mathbb{L}_{\max}	ω	ϵ_{feas}	ϵ_{obj}
R	R	R	R	R	1.000	1.000‡	4	2.772×10^{-12}	1.009×10^{-12}
R	R	R	R	Y	0.08192	0.08192	4	9.416×10^{-11}	3.100×10^{-12}
R	R	R	Y	Y	0.1250	0.1250	4	9.622×10^{-11}	2.944×10^{-11}
R	R	Y	R	R	0.1250	0.1250	4	3.042×10^{-11}	9.940×10^{-12}
R	R	Y	R	Y	0.03456	0.03456	4	1.0467×10^{-9}	3.625×10^{-12}
R	R	Y	Y	Y	0.1481	0.1481	4	3.770×10^{-12}	5.241×10^{-12}
R	Y	R	R	R	0.1481	0.1481	4	4.907×10^{-12}	7.989×10^{-12}
R	Y	R	R	Y	0.03456	0.03456	4	2.751×10^{-10}	4.725×10^{-12}
R	Y	R	Y	Y	0.1250	0.1250	4	2.727×10^{-11}	3.076×10^{-11}
R	Y	Y	R	R	0.1250	0.1250	4	2.668×10^{-11}	3.184×10^{-11}
R	Y	Y	R	Y	0.06250	0.06250	4	1.117×10^{-10}	2.829×10^{-11}
R	Y	Y	Y	Y	0.2500	0.2500	4	5.153×10^{-12}	1.017×10^{-11}
Y	R	R	R	R	0.2500	0.2500	4	4.584×10^{-12}	3.363×10^{-12}
Y	R	R	R	Y	0.06250	0.06250	4	1.777×10^{-10}	5.173×10^{-13}
Y	R	R	Y	Y	0.1250	0.1250	4	9.745×10^{-12}	6.507×10^{-12}
Y	R	Y	R	R	0.1250	0.1250	4	1.001×10^{-11}	6.0577×10^{-12}
Y	R	Y	R	Y	0.03125†	0.03125	4	5.551×10^{-17}	1.035×10^{-5}
Y	R	Y	Y	Y	0.1250†	0.1251	4	5.551×10^{-17}	6.052×10^{-4}
Y	Y	R	R	R	0.1250†	0.1250	4	2.106×10^{-29}	1.436×10^{-5}
Y	Y	R	R	Y	0.03125†	0.03125	4	1.110×10^{-16}	1.035×10^{-5}
Y	Y	R	Y	Y	0.1250	0.1250	4	2.695×10^{-11}	3.196×10^{-11}
Y	Y	Y	R	R	0.1250	0.1250	4	1.232×10^{-11}	1.544×10^{-11}
Y	Y	Y	R	Y	0.06250	0.06250	4	1.808×10^{-10}	4.676×10^{-11}
Y	Y	Y	Y	Y	0.2500	0.2500	4	7.625×10^{-12}	1.566×10^{-11}

Table 6.5: Results for computation of the likelihood function on the five-leaf “comb” topology $[[[,[,[,[,]]]]]$, conditioned on $\rho = R$. Each problem here is solved in approximately 250 minutes. The dagger † indicates values where we used a local optimization technique to improve the estimator $\hat{\mathbf{x}}$, and the double dagger ‡ indicates a point where we relaxed the equality constraint.

Site Pattern				Max. Likelihood	Max. Parsimony
1	2	3	4	Root ρ	Root ρ
R	R	R	R	R	R
R	R	R	Y	R	R
R	R	Y	Y	R	R
R	Y	R	R	R	R
R	Y	R	Y	R/Y	R/Y
R	Y	Y	Y	R/Y	R/Y
Y	R	R	R	R/Y	R/Y
Y	R	R	Y	R/Y	R/Y
Y	R	Y	Y	Y	Y
Y	Y	R	R	Y	Y
Y	Y	R	Y	Y	Y
Y	Y	Y	Y	Y	Y

Table 6.6: Implied Assignments for the tree $[[,],[,]]$.

Site Pattern				Max. Likelihood	Max. Parsimony
1	2	3	4	Root ρ	Root ρ
R	R	R	R	R	R
R	R	R	Y	R	R
R	R	Y	Y	R/Y	R/Y
R	Y	R	Y	R/Y	R/Y
R	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y

Table 6.7: Implied assignments for the tree $[[,],[,]]$.

Site Pattern					Max. Likelihood	Max. Parsimony
1	2	3	4	5	Root ρ	Root ρ
R	R	R	R	R	R	R
R	R	R	R	Y	R	R
R	R	R	Y	Y	R/Y	R/Y
R	R	Y	R	R	R	R
R	R	Y	R	Y	R/Y	R/Y
R	R	Y	Y	Y	Y	Y
R	Y	R	R	R	R	R
R	Y	R	R	Y	R	R
R	Y	R	Y	Y	R/Y	R/Y
R	Y	Y	R	R	R/Y	R/Y
R	Y	Y	R	Y	Y	Y
R	Y	Y	Y	Y	Y	Y
Y	Y	R	R	R	R	R
Y	Y	R	R	Y	R/Y	R/Y
Y	Y	R	Y	Y	Y	Y
Y	Y	Y	R	R	R/Y	R/Y
Y	Y	Y	R	Y	Y	Y
Y	Y	Y	Y	Y	Y	Y

Table 6.8: Implied assignments for the five-leaf topology $[[[[],],],[],[]]$.

Site Pattern					Max. Likelihood	Max. Parsimony
1	2	3	4	5	Root ρ	Root ρ
R	R	R	R	R	R	R
R	R	R	R	Y	R	R
R	R	R	Y	Y	R	R
R	R	Y	R	Y	R	R
R	R	Y	Y	Y	R/Y	R/Y
R	Y	Y	Y	Y	R/Y	R/Y
Y	R	R	R	R	R/Y	R/Y
Y	R	R	R	Y	R/Y	R/Y
Y	R	R	Y	Y	Y	Y
Y	R	Y	R	Y	Y	Y
Y	R	Y	Y	Y	Y	Y
Y	Y	Y	Y	Y	Y	Y

Table 6.9: Implied assignments for the topology $[,[[[],],],[]]$.

Site Pattern					Max. Likelihood	Max. Parsimony
1	2	3	4	5	Root ρ	Root ρ
R	R	R	R	R	R	R
R	R	R	R	Y	R	R
R	R	R	Y	Y	R/Y	R
R	R	Y	R	R	R/Y	R
R	R	Y	R	Y	R	R
R	R	Y	Y	Y	R	R
R	Y	R	R	R	R	R
R	Y	R	R	Y	R	R
R	Y	R	Y	Y	R/Y	R/Y
R	Y	Y	R	R	R/Y	R/Y
R	Y	Y	R	Y	R/Y	R/Y
R	Y	Y	Y	Y	R/Y	R/Y
Y	R	R	R	R	R/Y	R/Y
Y	R	R	R	Y	R/Y	R/Y
Y	R	R	Y	Y	R/Y	R/Y
Y	R	Y	R	R	R/Y	R/Y
Y	R	Y	R	Y	Y	Y
Y	R	Y	Y	Y	Y	Y
Y	Y	R	R	R	Y	Y
Y	Y	R	R	Y	Y	Y
Y	Y	R	Y	Y	R/Y	Y
Y	Y	Y	R	R	R/Y	Y
Y	Y	Y	R	Y	Y	Y
Y	Y	Y	Y	Y	Y	Y

Table 6.10: Implied assignments for the five-leaf “comb” topology $[[[,[,]]]]$.

Chapter 7

Conclusion

We have presented the problem of ancestral state reconstruction in phylogenetics, and showed that the solving the ML problem under a N_2 model requires solving a constrained polynomial optimization problem. We presented techniques of polynomial optimization which allowed us to solve the ML problem.

The method of polynomial optimization presented is powerful tool to solve POPs in general. Because of tools available to solve semidefinite programming problems numerically, the theory can be applied successfully in practice to solve the related optimization problems to global optimality. Unfortunately, the method can only be applied in practice to small size problems, due to the size of the corresponding semidefinite programs which need to be solved, but we find it works extremely well for small problems.

The method used is therefore not generally suitable to solve large likelihood maximization problems in phylogenetics. In practice, one might reconstruct trees with dozens of leaves, and this would not be a tractable to solve with our method. However, this method can be used on small trees to help provide an understanding of the tools used in phylogenetics.

In addition to the other models this method could be used for that were highlighted in Section 6.1.4, a related problem this optimization method may be applicable to is using maximum likelihood to reconstruct the tree topology itself. A four-leaf tree was used to show that the maximum parsimony and maximum likelihood methods do not necessarily agree on reconstructing tree topologies under a molecular clock [4], but this was done under the N_3 model and required a large number of characters on the tree. It is still unknown if the methods will agree for tree reconstruction under an N_2 model with a single character, and this method could be used to test this on four and five leaf topologies.

For the reconstruction problem, we have been able to find an example where, for the problem of ancestral state reconstruction under the N_2 model, the methods

of maximum likelihood and maximum parsimony do not agree. The methods do not make conflicting choices, but in our examples we found cases where maximum likelihood makes an ambiguous reconstruction, whereas maximum parsimony does not. It was known that without the molecular clock constraint, the two methods will agree [26], but we extend this knowledge to show with an example that this result does not hold under the molecular clock constraint.

Bibliography

- [1] S.P. BOYD AND L. VANDENBERGHE. *Convex optimization*. Cambridge Univ Pr, 2004.
- [2] B.S.W. CHANG AND M.J. DONOGHUE. Recreating ancestral proteins. *Trends In Ecology & Evolution*, **15**(3):109–114, 2000.
- [3] D. COX, J. LITTLE, AND D. O’SHEA. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 3rd edition, 2007.
- [4] M. FISCHER AND B. THATTE. Revisiting an equivalence between maximum parsimony and maximum likelihood methods in phylogenetics. *Bulletin of mathematical biology*, **72**(1):208–220, 2010.
- [5] W.M. FITCH. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic zoology*, pages 406–416, 1971.
- [6] C.A. FLOUDAS AND P.M. PARDALOS. *A collection of test problems for constrained global optimization algorithms*. Springer, 1990.
- [7] V. GOLDSCHMIDT, A. CIUFFI, M. ORTIZ, D. BRAWAND, M. MUNOZ, H. KAESSMANN, AND A. TELENTI. Antiretroviral Activity of Ancestral TRIM5 $\{\alpha\}$. *Journal of virology*, **82**(5):2089, 2008.
- [8] M. GRANT AND S. BOYD. CVX: Matlab software for disciplined convex programming, version 1.21 (code build 795). <http://cvxr.com/cvx>, May 2010.
- [9] R. HAUSER. A brief encounter with interior-point methods and semidefinite programming. Lecture Notes From a Short Course at Santa Clara University.
- [10] THOMAS JACOBI AND ALEXANDER PRESTEL. On special representations of strictly positive polynomials. *Journal for Pure and Applied Mathematics*, **2001**:223–235, 2001.

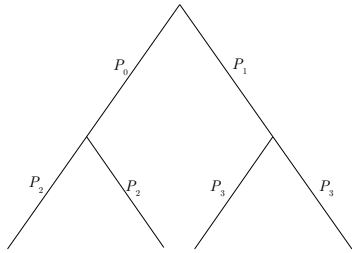
- [11] S. KIM, M. KOJIMA, AND H. WAKI. Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. *SIAM J. on Optimization*, **15**(3):697–719, 2005.
- [12] J. LASSERRE. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, **11**:796–817, 2001.
- [13] J. LASSERRE. Convergent SDP-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, **17**(3):822–843, 2006.
- [14] J. LASSERRE. Moments and sums of squares for polynomial optimization and related problems. *Journal of Global Optimization*, **45**(1):39–61, 2009.
- [15] J. LOFBERG. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, 2004.
- [16] J. NEYMAN. Molecular studies of evolution: a source of novel statistical problems. *Statistical decision theory and related topics*, pages 1–27, 1971.
- [17] P. A. PARILLO. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, **96**:293–320, 2003.
- [18] A. PURVIS. A composite estimate of primate phylogeny. *Philosophical Transactions: Biological Sciences*, **348**(1326):405–421, 1995.
- [19] M. PUTINAR. Positive polynomials on compact semi-algebraic sets. *Indiana Univ. Math. J.*, **42**:969–984, 1993.
- [20] B. REZNICK. Some concrete aspects of Hilbert’s 17th problem. In *Contemporary Mathematics*, **253**, pages 251–272. American Mathematical Society, 2000.
- [21] S. SCHAFFER, S. KOBLMULLER, T. PFINGSTL, C. STURMBAUER, AND G. KRISPER. Ancestral state reconstruction reveals multiple independent evolution of diagnostic morphological characters in the “Higher Oribatida” (Acari), conflicting with current classification schemes. *BMC Evolutionary Biology*, **10**(1):246, 2010.
- [22] M. SCHWEIGHOFER. Optimization of polynomials on compact semialgebraic sets. *SIAM Journal on Optimization*, **15**(3):805–825, 2005.
- [23] J. STURM. Sedumi 1.21, a matlab toolbox for optimization over symmetric cones, 2008.

- [24] M.J. TODD. Semidefinite optimization. *Acta Numerica*, **10**:515–560, 2003.
- [25] K. C. TOH, R. H. TÜTÜNCÜ, AND M. J. TODD. Sdpt3 version 4.0. <http://www.math.nus.edu.sg/~mattohk/sdpt3.html>, 2006.
- [26] C. TUFFLEY AND M. STEEL. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. *Bulletin of Mathematical Biology*, **59**(3):581–607, 1997.
- [27] L. VANDENBERGHE AND S. BOYD. Semidefinite programming. *SIAM review*, **38**(1):49–95, 1996.
- [28] H. WAKI, S. KIM, M. KOJIMA, AND M. MURAMATSU. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, **17**:218–242, 2006.

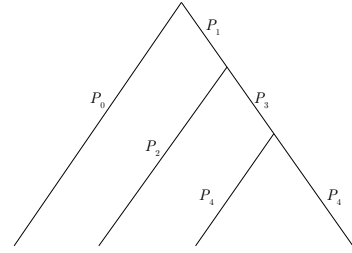
Appendix A

Complete Results

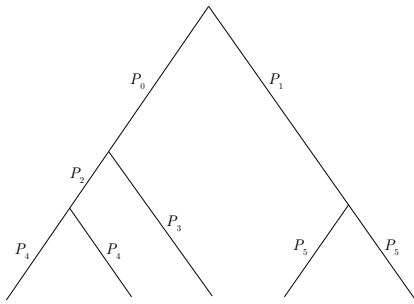
We provide in this chapter complete results for the calculated values. Complete labelings of the solved trees are shown in Figure [A.1](#). The values of the probabilities p_i under the N_2 model are shown in Tables [A.1](#) through [A.4](#).



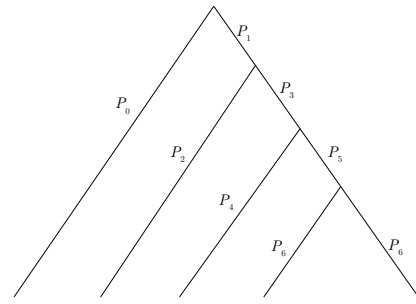
(a) Four-leaf $[[,],[,]]$ topology



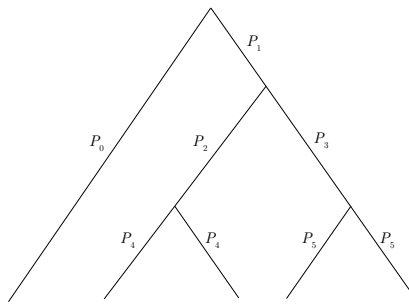
(b) Four-leaf “comb” $[,[,],[,]]$ topology



(c) Five-leaf $[[[,],[,],[,]]$ topology



(d) Five-leaf “comb” $[,[,[,],[,]]$ topology



(e) Five-leaf $[,[,[,],[,]]$ topology

Figure A.1: Tree topologies on which we solve the ML problem, with edges labelled.

Site Pattern				P0	P1	P2	P3	P4
1	2	3	4					
R	R	R	R	0	0	0	0	0
R	R	R	Y	0.2500	0	0.2500	0	0.2500
R	R	Y	Y	0.3333	0	0.3333	0.3333	0
R	Y	R	R	0.3333	0	0.3333	0.3333	0
R	Y	R	Y	0.5000	0	0.5000	0	0.5000
R	Y	Y	Y	0.5000	0.5000	0	0	0
Y	R	R	R	0.5000	0.5000	0	0	0
Y	R	R	Y	0.5000	0	0.5000	0.4998	0.5000
Y	R	Y	Y	0.5000	0.5000	0.4364	0.4364	0
Y	Y	R	R	0.5000	0.5000	0.4362	0.4354	0
Y	Y	R	Y	0.5000	0.5000	0.5000	0.5000	0.5000
Y	Y	Y	Y	0.5000	0.5000	0.5000	0.5000	0.5000

Table A.1: Optimizing values four-leaf comb topology $[[, [, []]]$, conditioned on $\rho = R$.

Site Pattern				P0	P1	P2	P3
1	2	3	4				
R	R	R	R	0	0	0	0
R	R	R	Y	0.3333	0	0	0.3333
R	R	Y	Y	0.5000	0.5000	0	0
R	Y	R	Y	0.2269	0.2269	0.5000	0.5000
R	Y	Y	Y	0.0039	0.5000	0.5000	0
Y	Y	Y	Y	0.5000	0.5000	0	0

Table A.2: Results for computation of the likelihood function on the four-leaf topology $[[, [, []]]$, conditioned on $\rho = R$.

Site Pattern					P Values						
1	2	3	4	5	P0	P1	P2	P3	P4	P5	P6
R	R	R	R	R	0	0	0	0	0	0	0
R	R	R	R	Y	0.2000	0	0.2000	0	0.2000	0	0.2000
R	R	R	Y	Y	0.5000	0.5000	0	0	0	0.5000	0
R	R	Y	R	R	0.5000	0.5000	0	0	0.5000	0	0
R	R	Y	R	Y	0.4000	0	0.4000	0	0.4000	0	0.4000
R	R	Y	Y	Y	0.3333	0	0.3333	0.3333	0	0	0
R	Y	R	R	R	0.3333	0	0.3333	0.3333	0	0	0
R	Y	R	R	Y	0.4000	0	0.4000	0	0.4000	0	0.4000
R	Y	R	Y	Y	0.5000	0.5000	0	0	0.5000	0	0
R	Y	Y	R	R	0.5000	0.5000	0	0	0	0.5000	0
R	Y	Y	R	Y	0.5000	0.5000	0	0	0	0.5000	0.5000
R	Y	Y	Y	Y	0.5000	0.5000	0	0	0	0	0
Y	R	R	R	R	0.5000	0.5000	0	0	0	0	0
Y	R	R	R	Y	0.5000	0.5000	0	0	0	0.5000	0.5000
Y	R	R	Y	Y	0.5000	0.5000	0	0	0	0.5000	0
Y	R	Y	R	R	0.5000	0.5000	0	0	0.5000	0	0
Y	R	Y	R	Y	0.5000	0.5000	0.5000	0.4246	0.4143	0.0202	0.5000
Y	R	Y	Y	Y	0.5000	0.4981	0.5000	0.5000	0	0	0
Y	Y	R	R	R	0.4998	0.4981	0.4531	0.4531	0	0	0
Y	Y	R	R	Y	0.5000	0.5000	0.5000	0.4119	0.4150	0.2406	0.4959
Y	Y	R	Y	Y	0.5000	0.5000	0	0	0.5000	0	0
Y	Y	Y	R	R	0.5000	0.5000	0	0	0	0.5000	0
Y	Y	Y	R	Y	0.5000	0.5000	0	0	0	0	0.5000
Y	Y	Y	Y	Y	0.5000	0.5000	0	0	0	0	0

Table A.3: Optimizing points on the five-leaf “comb” topology $[[,[,[,]]]]$, conditioned on $\rho = R$.

Site Pattern					P0	P1	P2	P3	P4	P5
1	2	3	4	5						
R	R	R	R	R	0	0	0	0	0	0
R	R	R	R	Y	0.3333	0	0	0	0	0.3333
R	R	R	Y	Y	0.5000	0.5000	0	0	0	0
R	R	Y	R	R	0	0.3333	0.3333	0.3333	0	0
R	R	Y	R	Y	0.4958	0.2627	0.5000	0.5000	0	0.5000
R	R	Y	Y	Y		0.5000	0.5000	0.5000	0	0
R	Y	R	R	R	0	0.2500	0	0.2500	0.2500	0
R	Y	R	R	Y	0	0	0	0.4000	0.4000	0.4000
R	Y	R	Y	Y	0.5000	0.5000	0.5000	0.5000	0.5000	0
R	Y	Y	R	R	0.5000	0.5000	0	0	0.5000	0
R	Y	Y	R	Y	0.4711	0.4677	0.0153	0.2234	0.4998	0.4998
R	Y	Y	Y	Y	0.5000	0.5000	0.5000	0.5000	0.5000	0
Y	Y	R	R	R	0	0.3333	0.3333	0.3333	0	0
Y	Y	R	R	Y	0.4966	0.3911	0.4999	0.5000	0	0.5000
Y	Y	R	Y	Y	0.5000	0.5000	0.5000	0.5000	0	0
Y	Y	Y	R	R	0.5000	0.5000	0	0	0	0
Y	Y	Y	R	Y	0.5000	0	0	0	0	0.5000
Y	Y	Y	Y	Y	0.5000	0.5000	0	0	0	0

Table A.4: Optimizing values of the likelihood function on the five-leaf topology $[[[[],[]],[]],[]]$, conditioned on $\rho = R$.

Site Pattern					P0	P1	P2	P3	P4	P5
1	2	3	4	5						
R	R	R	R	R	0	0	0	0	0	0
R	R	R	R	Y	0.2500	0	0.2500	0	0	0.2500
R	R	R	Y	Y	0.3333	0	0.3333	0.3333	0	0
R	R	Y	R	Y	0.4000	0	0	0	0.4000	0.4000
R	R	Y	Y	Y	0.5000	0.5000	0.4997	0.5000	0.5000	0
R	Y	Y	Y	Y	0.5000	0.5000	0	0	0	0
Y	R	R	R	R	0.5000	0.5000	0	0	0	0
Y	R	R	R	Y	0.5000	0.5000	0	0.5000	0	0.5000
Y	R	R	Y	Y	0.5000	0.4934	0.5000	0.5000	0	0
Y	R	Y	R	Y	0.5000	0.1405	0.4993	0.0001	0.5000	0.5000
Y	R	Y	Y	Y	0.5000	0.0001	0.0001	0.5000	0.5000	0.5000
Y	Y	Y	Y	Y	0.5000	0.5000	0	0	0	0

Table A.5: Optimizing values of the likelihood function on the five-leaf topology $[[[[],[]],[]],[]]$, conditioned on $\rho = R$.

Appendix B

Detailed SeDuMi Output For Examples on Which ML and MP Disagree

We provide detailed output from from SeDuMi for the four calculations on the examples where ML and MP disagree. Thus the following information is for the patterns computed on the “comb” topology $[[,[[,[[,]]]]$. We notice the solver reaches a high degree of precision, and indicates feasibility for the problem. The solver precision in cvx here is set to $[\epsilon^{0.65}, \epsilon^{0.5}, \epsilon^{0.5}]$, where ϵ is the machine precision. This sets the SeDuMi solver precision to $\epsilon^{0.8} \approx 7 \times 10^{-11}$, but accepts solutions as valid at $\epsilon^{0.5} \approx 1.5 \times 10^{-9}$.

Pattern RRRYY

```
SeDuMi 1.21 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 6425, order n = 2013, dim = 310829, blocks = 17
nnz(A) = 210423 + 0, nnz(ADA) = 41280625, nnz(L) = 20643525
it :   b*y      gap      delta      rate      t/tP*      t/tD*      feas cg cg prec
0 :           9.94E-02 0.000
1 : 4.57E-01 6.87E-02 0.000 0.6914 0.9000 0.9000 2.88 1 1 5.5E+00
2 : 3.99E-01 3.19E-02 0.000 0.4643 0.9000 0.9000 3.62 1 1 1.1E+00
3 : 2.19E-01 1.25E-02 0.000 0.3917 0.9000 0.9000 2.50 1 1 2.9E-01
4 : 1.23E-01 4.11E-03 0.000 0.3288 0.9000 0.9000 2.34 1 1 7.0E-02
5 : 1.00E-01 1.79E-03 0.000 0.4344 0.9000 0.9000 1.76 1 1 2.6E-02
6 : 1.02E-01 9.09E-04 0.000 0.5089 0.9000 0.9000 1.41 1 1 1.2E-02
7 : 1.13E-01 5.94E-04 0.000 0.6541 0.9000 0.9000 1.08 1 1 8.0E-03
8 : 1.22E-01 1.63E-04 0.000 0.2737 0.9000 0.9000 1.10 1 1 2.0E-03
9 : 1.24E-01 4.86E-05 0.000 0.2990 0.9000 0.9000 1.03 1 1 6.0E-04
10 : 1.25E-01 1.52E-05 0.000 0.3128 0.9011 0.9000 1.01 1 1 1.9E-04
11 : 1.25E-01 5.17E-06 0.000 0.3395 0.0000 0.9000 1.00 1 1 9.8E-05
12 : 1.25E-01 2.24E-06 0.000 0.4330 0.9000 0.9020 1.00 1 1 4.4E-05
13 : 1.25E-01 4.87E-07 0.000 0.2177 0.6806 0.9000 1.00 1 1 1.6E-05
14 : 1.25E-01 1.92E-07 0.000 0.3937 0.9000 0.9000 1.00 1 1 6.4E-06
15 : 1.25E-01 6.10E-08 0.000 0.3180 0.9000 0.9000 1.00 1 1 2.0E-06
16 : 1.25E-01 2.13E-08 0.000 0.3495 0.9000 0.9000 1.00 1 1 7.2E-07
17 : 1.25E-01 7.22E-09 0.000 0.3387 0.9000 0.9000 1.00 1 1 2.4E-07
18 : 1.25E-01 2.67E-09 0.000 0.3700 0.9000 0.9000 1.00 1 1 9.1E-08
19 : 1.25E-01 9.22E-10 0.000 0.3454 0.9000 0.9000 1.00 1 1 3.2E-08
20 : 1.25E-01 3.56E-10 0.000 0.3861 0.9000 0.9000 1.00 1 2 1.2E-08
21 : 1.25E-01 1.18E-10 0.000 0.3300 0.9000 0.9000 1.00 2 2 4.1E-09
22 : 1.25E-01 4.32E-11 0.000 0.3680 0.9000 0.9000 1.00 2 2 1.5E-09
23 : 1.25E-01 1.40E-11 0.000 0.3236 0.9000 0.9000 1.00 2 2 4.8E-10
24 : 1.25E-01 4.99E-12 0.000 0.3564 0.9000 0.9000 1.00 7 7 1.7E-10
25 : 1.25E-01 1.63E-12 0.000 0.3276 0.9000 0.9000 1.00 17 17 5.7E-11

iter seconds digits      c*x      b*y
25 3306.1 Inf 1.2499999984e-01 1.2499999984e-01
|Ax-b| = 1.4e-09, [Ay-c]_+ = 2.5E-12, |x| = 1.6e+01, |y| = 1.5e+00

Detailed timing (sec)
```

```

Pre          IPM          Post
3.210E+00   3.306E+03   6.900E-01
Max-norms: ||b||=4, ||c|| = 1,
Cholesky ladd|=15, |skip|= 0, ||L.L|| = 6.06096e+09.

```

Pattern RRYRR

```

SeDuMi 1.21 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 6425, order n = 2013, dim = 310829, blocks = 17
nnz(A) = 210423 + 0, nnz(ADA) = 41280625, nnz(L) = 20643525
it :   b*y      gap      delta      rate      t/tP*      t/tD*      feas cg cg prec
0 :           9.76E-02 0.000
1 :   3.97E-01 6.75E-02 0.000 0.6920 0.9000 0.9000 2.93 1 1 5.3E+00
2 :   3.76E-01 3.13E-02 0.000 0.4634 0.9000 0.9000 3.63 1 1 1.1E+00
3 :   2.13E-01 1.22E-02 0.000 0.3903 0.9000 0.9000 2.51 1 1 2.8E-01
4 :   1.26E-01 4.12E-03 0.000 0.3369 0.9000 0.9000 2.34 1 1 6.8E-02
5 :   9.99E-02 1.78E-03 0.000 0.4318 0.9000 0.9000 1.73 1 1 2.6E-02
6 :   1.01E-01 9.18E-04 0.000 0.5163 0.9000 0.9000 1.43 1 1 1.2E-02
7 :   1.12E-01 6.10E-04 0.000 0.6643 0.9000 0.9000 1.08 1 1 8.2E-03
8 :   1.22E-01 1.76E-04 0.000 0.2889 0.9000 0.9000 1.11 1 1 2.2E-03
9 :   1.24E-01 5.09E-05 0.000 0.2889 0.9000 0.9000 1.03 1 1 6.3E-04
10 :  1.25E-01 1.60E-05 0.000 0.3139 0.9002 0.9000 1.01 1 1 2.0E-04
11 :  1.25E-01 5.50E-06 0.000 0.3446 0.9000 0.9005 1.00 1 1 7.0E-05
12 :  1.25E-01 1.90E-06 0.000 0.3450 0.9000 0.9047 1.00 1 1 2.5E-05
13 :  1.25E-01 6.46E-07 0.000 0.3403 0.9000 0.9073 1.00 1 1 9.3E-06
14 :  1.25E-01 1.90E-07 0.000 0.2940 0.9000 0.9143 1.00 1 1 3.4E-06
15 :  1.25E-01 3.31E-08 0.000 0.1745 0.9000 0.9238 1.00 1 1 1.2E-06
16 :  1.25E-01 1.29E-08 0.000 0.3878 0.9000 0.6795 1.00 1 1 5.6E-07
17 :  1.25E-01 5.00E-09 0.000 0.3893 0.9000 0.9000 1.00 1 1 2.2E-07
18 :  1.25E-01 2.24E-09 0.000 0.4474 0.9000 0.9000 1.00 1 1 9.7E-08
19 :  1.25E-01 7.43E-10 0.000 0.3319 0.9000 0.9000 1.00 1 1 3.2E-08
20 :  1.25E-01 2.78E-10 0.000 0.3739 0.9000 0.9000 1.00 1 1 1.2E-08
21 :  1.25E-01 9.13E-11 0.000 0.3285 0.9000 0.9000 1.00 2 2 4.0E-09
22 :  1.25E-01 3.31E-11 0.000 0.3629 0.9000 0.9000 1.00 2 2 1.4E-09
23 :  1.25E-01 1.12E-11 0.000 0.3375 0.9000 0.9000 1.00 2 2 4.8E-10
24 :  1.25E-01 4.17E-12 0.000 0.3734 0.9000 0.9000 1.00 5 5 1.8E-10
25 :  1.25E-01 1.39E-12 0.000 0.3323 0.9000 0.9000 1.00 10 10 6.0E-11

```

```

iter seconds digits      c*x          b*y
25 3302.0 Inf 1.2499999984e-01 1.2499999984e-01
|Ax-b| = 1.5e-09, [Ay-c]_+ = 4.2E-12, |x|= 1.3e+01, |y|= 1.6e+00

```

```

Detailed timing (sec)
Pre          IPM          Post
3.210E+00   3.302E+03   6.300E-01
Max-norms: ||b||=4, ||c|| = 1,
Cholesky ladd|=8, |skip|= 0, ||L.L|| = 4.68055e+09.

```

Pattern YYRY

```

SeDuMi 1.21 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 6425, order n = 2013, dim = 310829, blocks = 17
nnz(A) = 210423 + 0, nnz(ADA) = 41280625, nnz(L) = 20643525
it :   b*y      gap      delta      rate      t/tP*      t/tD*      feas cg cg prec
0 :           9.76E-02 0.000
1 :   8.22E-02 6.78E-02 0.000 0.6946 0.9000 0.9000 3.09 1 1 5.2E+00
2 :   8.20E-02 3.28E-02 0.000 0.4837 0.9000 0.9000 3.82 1 1 1.1E+00
3 :   5.91E-02 1.36E-02 0.000 0.4137 0.9000 0.9000 2.64 1 1 3.0E-01
4 :   4.93E-02 4.75E-03 0.000 0.3504 0.9000 0.9000 2.48 1 1 7.9E-02
5 :   6.28E-02 2.53E-03 0.000 0.5314 0.9000 0.9000 1.79 1 1 3.6E-02
6 :   1.03E-01 1.34E-03 0.000 0.5314 0.9000 0.9000 1.35 1 1 1.6E-02
7 :   1.20E-01 3.34E-04 0.000 0.2484 0.9000 0.9000 1.25 1 1 3.4E-03
8 :   1.23E-01 9.21E-05 0.000 0.2763 0.9000 0.9000 1.07 1 1 9.1E-04
9 :   1.24E-01 3.02E-05 0.000 0.3274 0.9000 0.9000 1.02 1 1 3.0E-04
10 :  1.25E-01 7.52E-06 0.000 0.2493 0.0000 0.9000 1.00 1 1 1.5E-04
11 :  1.25E-01 3.02E-06 0.000 0.4017 0.9000 0.9015 1.00 1 1 6.2E-05
12 :  1.25E-01 7.39E-07 0.000 0.2447 0.7643 0.9000 1.00 1 1 2.3E-05
13 :  1.25E-01 3.06E-07 0.000 0.4134 0.9000 0.7640 1.00 1 1 9.7E-06
14 :  1.25E-01 1.01E-07 0.000 0.3299 0.9000 0.9000 1.00 1 1 3.2E-06
15 :  1.25E-01 3.80E-08 0.000 0.3764 0.9000 0.9000 1.00 1 1 1.2E-06
16 :  1.25E-01 1.30E-08 0.000 0.3417 0.9000 0.9000 1.00 1 1 4.2E-07
17 :  1.25E-01 4.98E-09 0.000 0.3841 0.9000 0.9000 1.00 1 1 1.6E-07
18 :  1.25E-01 1.70E-09 0.000 0.3410 0.9000 0.9000 1.00 1 1 5.6E-08
19 :  1.25E-01 6.55E-10 0.000 0.3854 0.9000 0.9000 1.00 1 1 2.2E-08
20 :  1.25E-01 2.16E-10 0.000 0.3295 0.9000 0.9000 1.00 1 1 7.1E-09
21 :  1.25E-01 7.95E-11 0.000 0.3683 0.9000 0.9000 1.00 1 2 2.6E-09
22 :  1.25E-01 2.59E-11 0.000 0.3264 0.9000 0.9000 1.00 1 1 8.6E-10
23 :  1.25E-01 9.37E-12 0.000 0.3612 0.9000 0.9000 1.00 2 3 3.1E-10
24 :  1.25E-01 3.08E-12 0.000 0.3286 0.9000 0.9000 1.00 7 7 1.0E-10
25 :  1.25E-01 1.12E-12 0.000 0.3630 0.9000 0.9000 1.00 18 18 3.7E-11

```

```

iter seconds digits      c*x          b*y
25 3355.2  Inf  1.2499999989e-01 1.2499999989e-01
|Ax-b| = 9.8e-10, [Ay-c]_+ = 1.8E-12, |x|= 1.3e+01, |y|= 1.5e+00

```

```

Detailed timing (sec)
Pre      IPM      Post
3.240E+00 3.355E+03 7.200E-01
Max-norms: ||b||=4, ||c|| = 1,
Cholesky |add|=16, |skip|= 0, ||L.L|| = 8.89997e+09.

```

Pattern YYRR

SeDuMi 1.21 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
eqs m = 6425, order n = 2013, dim = 310829, blocks = 17
nnz(A) = 210423 + 0, nnz(ADA) = 41280625, nnz(L) = 20643525

it	b*y	gap	delta	rate	t/tP*	t/tD*	feas	cg	cg	prec
0		9.76E-02	0.000							
1	7.96E-02	6.78E-02	0.000	0.6947	0.9000	0.9000	3.09	1	1	5.2E+00
2	8.19E-02	3.28E-02	0.000	0.4840	0.9000	0.9000	3.82	1	1	1.1E+00
3	6.12E-02	1.36E-02	0.000	0.4140	0.9000	0.9000	2.64	1	1	3.0E-01
4	5.28E-02	4.90E-03	0.000	0.3605	0.9000	0.9000	2.48	1	1	8.0E-02
5	6.37E-02	2.54E-03	0.000	0.5176	0.9000	0.9000	1.80	1	1	3.5E-02
6	1.03E-01	1.38E-03	0.000	0.5445	0.9000	0.9000	1.37	1	1	1.6E-02
7	1.20E-01	3.42E-04	0.000	0.2480	0.9000	0.9000	1.26	1	1	3.4E-03
8	1.23E-01	9.38E-05	0.000	0.2738	0.9000	0.9000	1.07	1	1	9.1E-04
9	1.24E-01	3.09E-05	0.000	0.3300	0.9000	0.9000	1.02	1	1	3.0E-04
10	1.25E-01	7.63E-06	0.000	0.2465	0.0000	0.9000	1.00	1	1	1.5E-04
11	1.25E-01	3.04E-06	0.000	0.3982	0.9000	0.9016	1.00	1	1	6.2E-05
12	1.25E-01	7.51E-07	0.000	0.2472	0.7691	0.9000	1.00	1	1	2.3E-05
13	1.25E-01	3.11E-07	0.000	0.4144	0.9000	0.7626	1.00	1	1	9.7E-06
14	1.25E-01	1.03E-07	0.000	0.3300	0.9000	0.9000	1.00	1	1	3.2E-06
15	1.25E-01	3.87E-08	0.000	0.3771	0.9000	0.9000	1.00	1	1	1.2E-06
16	1.25E-01	1.32E-08	0.000	0.3412	0.9000	0.9000	1.00	1	1	4.2E-07
17	1.25E-01	5.07E-09	0.000	0.3837	0.9000	0.9000	1.00	1	1	1.6E-07
18	1.25E-01	1.73E-09	0.000	0.3409	0.9000	0.9000	1.00	1	1	5.6E-08
19	1.25E-01	6.66E-10	0.000	0.3854	0.9000	0.9000	1.00	1	1	2.2E-08
20	1.25E-01	2.20E-10	0.000	0.3298	0.9000	0.9000	1.00	1	1	7.2E-09
21	1.25E-01	8.13E-11	0.000	0.3700	0.9000	0.9000	1.00	1	1	2.7E-09
22	1.25E-01	2.65E-11	0.000	0.3263	0.9000	0.9000	1.00	2	2	8.7E-10
23	1.25E-01	9.58E-12	0.000	0.3611	0.9000	0.9000	1.00	3	3	3.2E-10
24	1.25E-01	3.15E-12	0.000	0.3286	0.9000	0.9000	1.00	7	7	1.0E-10
25	1.25E-01	1.14E-12	0.000	0.3630	0.9000	0.9000	1.00	17	17	3.8E-11

```

iter seconds digits      c*x          b*y
25 3342.6  Inf  1.2499999989e-01 1.2499999989e-01
|Ax-b| = 9.9e-10, [Ay-c]_+ = 1.8E-12, |x|= 1.3e+01, |y|= 1.5e+00

```

```

Detailed timing (sec)
Pre      IPM      Post
3.260E+00 3.343E+03 6.800E-01
Max-norms: ||b||=4, ||c|| = 1,
Cholesky |add|=15, |skip|= 0, ||L.L|| = 8.73946e+09.

```