LIBRARY
Te Tumu Herenga
THE UNIVERSITY OF AUCKLAND

RESEARCHSPACE@AUCKLAND

Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand). This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage.
http://researchspace.auckland.ac.nz/feedback

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form

Stochastic Geometry, Data Structures and Applications of Ancestral Selection Graphs

Nicoleen Cloete

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Mathematics at the University of Auckland, 2005

To Danie, Renier and Nicola

May all your dreams come true

Contents

Notation **v**

Abstract **xxiii**

Acknowledgment **xxv**

1 Introduction **1**

2 Stochastic geometry **17**

2.1 The model 19

2.2 Ancestral graphs 21

2.3 Space of ancestral graphs 24

2.3.1 Unlabeled graphs 24

2.3.2 Type-labeled graphs 27

2.4 The full posterior density 32

3 Markov Chain Monte Carlo **35**

3.1 Metropolis-Hastings algorithm 36

3.1.1 Moves 38

3.2 Example 51

4 Implementation **55**

4.1 Data type 56

4.2 Data structure 57

4.2.1 Example 62

4.3 Labeling 64

4.4 The process of adding an edge 68

4.5 Other moves 69

4.6 Special cases 71

4.6.1 Above the root 71

4.6.2 Bubbles 71

4.6.3 Illegal states 73

4.6.4 Genealogy 75

5 Debugging **81**

5.1 Structural testing 82

5.2 Analytic tests 87

6 Experiments and Results **101**

6.1 Ladders 104

6.1.1 $k(B_Y, g|\sigma, B_I)$ and $f_G(g|\sigma)$ 104

6.1.2 $|V_L| = 5$ and $|V_L| = 10$ 117

6.1.3 Continuous versus separate Markov chains 122

6.2 Integrated Autocorrelation Time 126

6.3 The model parameter σ 136

6.3.1 Predetermined genealogy Tr 136

6.3.2 Multiple loci 153

7 Conclusion **157**

7.1 Future work 159

A Data structures **163**

B Theorems and proofs **165**

Notation

n : Size of sample taken from a population of organisms, 19

N : Size of the population of organisms, 19

$\boldsymbol{\lambda_0}$ **:** Type 0 birth rate, 19

$\boldsymbol{\lambda_1}$ **:** Type 1 birth rate, 19

UA : Ultimate ancestor, 19, 22

MRCA : Most recent common ancestor, 3, 29, 75, 77

$\boldsymbol{t_{UA}}$ **:** Time to ultimate ancestor, 20, 22

$\boldsymbol{Z_0(t_0)}$ **:** Number of allele type 0 at time t, 20

$\boldsymbol{Z_1(t_0)}$ **:** Number of allele type 1 at time t, 20

$\boldsymbol{\theta}$ **:** Mutation parameter, 20

$\boldsymbol{\sigma}$ **:** Selection parameter, 20

z : Frequency of allele type 0, 20

$\boldsymbol{\eta(z)}$ **:** Density of the diffusion process, 20

K : Normalizing constant, 20

$\boldsymbol{\mathcal{G}_n(t)}$ **:** Ancestral selection graph process with n leaves, 21

$\boldsymbol{\mathcal{A}}$ **:** Set of labels assigned to the sample, 21

$\boldsymbol{G = (V, E)}$ **:** Directed graph with vertex labels V and edge labels E, 21

M : Total number of events in one realization of a graph, 21

$\boldsymbol{t_v}$ **:** Time at vertex v, 24

$\boldsymbol{\mathcal{A}(t)}$ **:** Set-valued process denoting the set of particles at time t, 21

$\boldsymbol{\mathcal{R}_m}$ **:** Jump process of times at which events occur, 21

$\boldsymbol{|\mathcal{A}(t)|}$ **:** Cardinality of the set $\mathcal{A}$ at time t, 22

k : Number of edges between two events, 22, 26

$\boldsymbol{B_v}$ **:** Particle type at vertex $v \in V$ where $B \in \{0, 1\}$, 23

$\boldsymbol{V_L}$ **:** Set of leaf vertex labels, 23

$\boldsymbol{V_A}$ **:** Set of internal or ancestral vertex labels, 23

$\boldsymbol{V_c}$ **:** Set of vertex labels representing coalescent events, 24

$\boldsymbol{V_b}$ **:** Set of vertex labels representing branching events, 24

$\boldsymbol{E_g}$: Set of edge labels, 24

$\boldsymbol{E_c}$: Set of edge labels with coalescent events at the top, 24

$\boldsymbol{E_b}$: Set of edge labels with branching events at the top, 24

$\boldsymbol{N_c}$: Total number of coalescent events in a graph g, 24

$\boldsymbol{N_b}$: Total number of branching events in a graph g, 24

$\boldsymbol{d_{in}}$: In-degree of a vertex, 25

$\boldsymbol{d_{out}}$: Out-degree of a vertex, 25

$\boldsymbol{\chi_{V,E}}$: Space of admissible vertex times for a given topology V, E, 25

$\boldsymbol{S_i}$: Gating value of edge i, 25

$\boldsymbol{\mathcal{S}_{V,E}}$: Set of admissible assignments of selection gating values, 25

$\boldsymbol{\Gamma^{\star}_{N_b,n}}$: Set of admissible ancestral graph topologies V, E on N_b branching events and n leaves, 25

$\boldsymbol{(\Gamma, \mathcal{F}, P)}$: Space of unlabeled ancestral graphs with σ algebra of subsets $\mathcal{F}$ and probability measure P, 26

$\boldsymbol{\rho_m}$: Total rate of coalescent and branching events in the interval t_{m-a}, t_m, 26

$\boldsymbol{v^{(i)}}$: Vertex at the top of edge i, 28

$\boldsymbol{v_{(i)}}$: Vertex at the bottom of edge i, 28

$\langle \boldsymbol{v_{(i)}}, \boldsymbol{v^{(i)}} \rangle$ **:** Edge connecting vertex $v_{(i)}$ with vertex $v^{(i)}$, 28

$\boldsymbol{B_{UA}}$ **:** Ancestral type of the ultimate ancestor, 31

$\boldsymbol{p_0}$ **:** Probability that ultimate ancestor is of type 0, 31

$\boldsymbol{p_1}$ **:** Probability that ultimate ancestor is of type 1, 31

$\boldsymbol{\tau_i}$ **:** Length of edge i, 31

$\boldsymbol{P_{B_{v_{(i)}}, B_{v^{(i)}}}}$ **:** 2×2 transition matrix associated with edge $\langle v_{(i)}, v^{(i)} \rangle$, 31

$\boldsymbol{B_I}$ **:** Set of allele types of the individuals at the leaf tips, 32

$\boldsymbol{B_Y}$ **:** Set of allele types at the internal vertices, 32

$\boldsymbol{f_G(g|\sigma)}$ **:** Population density on the space Γ of unlabeled graphs, 27

$\boldsymbol{P(B|\theta, g)}$ **:** Likelihood of the leaf types given θ and g, 31

$\boldsymbol{h(g, B_Y, \theta, \sigma|B_I)}$ **:** Posterior density, 33

$\boldsymbol{\Phi_n = (g, B_Y, \theta, \sigma)}$ **:** The nth update of the Markov Chain, 36

$\boldsymbol{p^\dagger}$ **:** Probability of deleting an edge, 38

$\boldsymbol{p^\star}$ **:** Probability of adding an edge, 38

$\boldsymbol{|E|}$ **:** Total number of edges in a graph, 41

$\boldsymbol{|E_D|}$ **:** Total number of deletable edges in a graph, 42

$\boldsymbol{V_{AR}}$ **:** Label of the vertex above the root, 56

$\boldsymbol{Tr_{(V,E)}}$ **:** Genealogy of an ancestral selection graph with topology (V, E), 79

$\boldsymbol{\mathcal{E}}$ **:** Set of events in a graph, 87

$(\boldsymbol{\beta_m}, \boldsymbol{\gamma_m})$ **:** Label process describing the topology of the graph, 89

$\boldsymbol{\beta_m}$ **:** The label of the edge branching if the mth event is a branching event, 89

$\boldsymbol{\gamma_m}$ **:** The labels of the two coalescent edges if the mth event is a coalescent event, 89

$\boldsymbol{\Sigma}$ **:** The set of selection parameter values used for experiments, 102

$\boldsymbol{w(g, B_Y, \sigma, \theta | B_I, Tr)}$, 102

$\boldsymbol{k(g, B_Y | \sigma \in \Sigma, B_I)}$, 102

LI : Ladder index, 107

$\boldsymbol{\zeta}$ **:** Statistic of interest, 126

$\boldsymbol{\tau_\zeta}$ **:** Integrated autocorrelation time, 126

$\boldsymbol{v_\zeta(s)}$ **:** Auto-covariance at lag s, 126

$\boldsymbol{v\hat{a}r_\zeta}$ **:** Estimated variance of statistic ζ, 126

$\boldsymbol{t_{FE}}$ **:** Time to the first event, 127

D : Data at the leaf tips, 136

$\boldsymbol{Tr_{0.5}}$ **:** Genealogy when $\sigma = 0.5$, 138

$\boldsymbol{Tr_{exp}}$ **:** Genealogy for exponential growth, 148

$\boldsymbol{Tr_{0.5}^{(15)}}$ **:** Genealogy when true $\sigma = 0.5$ and $\sigma \sim U(1.5)$, 152

$\boldsymbol{D_{0.5}}$ **:** Data simulated when $\sigma = 0.5$, 138

$\boldsymbol{w(\sigma)}$ **:** Marginal posterior density of σ, 140

se : Standard error, 152

acct : Acceptance rate, 152

List of Figures

1.1 Binary tree representing the coalescent process 4

1.2 Parent vertex i with two children vertices j and k 8

2.1 Ancestral selection graph 22

2.2 The t_{UA} and t_{MRCA} need not coincide. The edge labeled I is replacing and the edge labeled II is continuing 29

3.1 Add an edge 39

3.2 Bubble 43

3.3 Add edge with one vertex above the root 43

3.4 The deletion of an edge results in an illegal state 45

3.5 The next and previous events of a vertex v need not be connected by an edge 45

3.6 Coalescing event at vertex 47

3.7 Branching event 48

3.8 A realization of an actual Markov chain, which was created by adding and deleting edges. The dashed lines indicate the proposed new states. 52

4.1 Convention of labeling the edges 58

4.2 Vertex labels before and after adding an edge 62

4.3 Edge labels before and after adding an edge 64

4.4 Process of adding an edge with edge labels in bold 69

4.5 Bubble after deleting an edge 73

4.6 Bubble above a root 74

4.7 Creation of illegal states 74

5.1 $\mathcal{E}_{g_1} = \{c\}$ and $\mathcal{E}_{g_2} = \{b, c, c\}$ 88

5.2 (A): Label process, (B): Labels for the code 90

5.3 Two graphs with the same $\mathcal{E} = \{b, c, c\}$ and different label processes 91

5.4 Diagram of the three different graphs used for debugging. The number of leaves is 2. The Markov chain generated for debugging is $\{\Phi_i\} \sim f_G(g|\sigma)$ 92

5.5 Diagram of the three different configurations of the selection graph with zero branching events used for debugging purposes. The number of leaves is 2. The Markov chain generated for debugging is $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$ 94

5.6 Diagram of the configurations of the selection graph with one branching event and zero bubbles used for debugging purposes. The number of leaves is 2. The Markov chain generated for debugging is $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$ 97

5.7 The histogram of $P(\sigma)$ where the data set consists of 10 individuals of type 1 99

5.8 The histogram of $P(\sigma)$, where all of the 10 leaves are of type 0 . 100

5.9 The histogram of $P(\sigma)$, where 4 of the 10 leaves are of type 0 . . 100

6.1 The trace of t_{UA} of the Markov chain $\{\Phi_{if}\} \sim f_G(g|\sigma = 1.8, B_I)$. The spikes show the presence of ladders. 105

6.2 Part of a graph showing 10 branching events and 12 deletable edges. The dashed lines indicate the deletable edges. 106

6.3 The trace of the ladder index,(LI), of the Markov chain $\{\Phi_{if}\} \sim f_G(g|\sigma = 1.8, |V_L| = 10)$ 108

6.4 Part of a graph with 4 bubbles and $LI = 0.875$ 109

6.5 The mean number of branching events $\hat{N}_{bk}$ denoted by $(+)$, the mean number of deletable edges, $\hat{nde}_k$ denoted $(.)$ and the upper limit of the number of deletable edges denoted by a dashed line where $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ 110

6.6 The mean ladder index, $\hat{LI}_k$ of $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, |V_L| = 5)$ denoted by $(+)$ and the mean ladder index $\hat{LI}_f$ of $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, |V_L| = 5)$ denoted by $(.)$ 112

6.7 The frequency of graphs with branching events. $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, |V_L| = 5)$ is denoted by $(.)$ and of $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma)$ is denoted by $(+)$. 114

6.8 The acceptance rate $(\%)$ of $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(+)$ and the acceptance rate, $(\%)$ of $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(.)$ 116

6.9 The mean ladder index, $\hat{LI}_k^{(5)}$ of $\{\Phi_{ik}^{(5)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(.)$ and the mean ladder index $\hat{LI}_k^{(10)}$ where $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ is denoted by $(+)$ 117

6.10 The mean number of branching events $\hat{N}_{b_k}^{(10)}$ denoted by $(+)$, the mean number of deletable edges, $\hat{nde}_k^{(10)}$ denoted by $(.)$ and the upper limit of the number of deletable edges by $(--)$ of the Markov chain $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$. 118

6.11 Acceptance rates $(\%)$ of $\{\Phi_{ik}^{(10)}\}$ denoted by $(+)$ and $\{\Phi_{ik}^{(5)}\}$ denoted by $(.)$ 120

6.12 Proportion of graphs showing branching events, $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ denoted by $(+)$ and $\{\Phi_{ik}^{(5)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(.)$ 122

6.13 The mean ladder index $\hat{LI}_f$ of continuous Markov chain $\{\Phi_{if}^{(c)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (.) and the separate Markov chains $\{\Phi_{if}^{(s)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (+) 123

6.14 Frequency of graphs showing branching events where the continuous Markov chain $\{\Phi_{if}^{(c)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ is denoted by (.) and the separate Markov chains $\{\Phi_{if}^{(s)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ is denoted by (+) 124

6.15 The acceptance rates, (%) of the continuous Markov chain $\{\Phi_{if}^{(c)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (.) and of the separate Markov chains $\{\Phi_{if}^{(s)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (+) 125

6.16 The IACT, τ of the ladder index on the y-axis and $\sigma \in \Sigma$ on the x-axis. 129

6.17 The IACT on y-axis and σ on x-axis of t_{UA}. 130

6.18 The IACT on the y-axis, with σ on x-axis of t_{FE}. 131

6.19 IACT on y-axis and σ on x-axis of t_{MRCA}. 132

6.20 The IACT on the y-axis and σ on the x-axis of N_b 133

6.21 The IACT of t_{UA} of the Markov chain $\{\Phi_{ik}^{(10)}\}$ denoted by (+) and of the Markov chain $\{\Phi_{ik}^{(5)}\}$ denoted by (.) 134

6.22 The IACT of the ladder index, τ_{LI}, of the Markov chain $\{\Phi_{ik}^{(10)}\}$ denoted by $(+)$ is more correlated than τ_{LI} of the Markov chain $\{\Phi_{ik}^{(5)}\}$ denoted by $(.)$ 135

6.23 The ancestral selection graph, $g_{0.5}$. The ancestry, $Tr_{0.5}$ is denoted in a thicker line. The labels of the leaves are in brackets. 139

6.24 Trace of σ where the embedded ancestry is $Tr_{0.5}$ 140

6.25 Trace of t_{UA} where the embedded ancestry is $Tr_{0.5}$. 141

6.26 Histogram of the marginal posterior density, $w(\sigma)$, where the ancestry is $Tr_{0.5}$. 141

6.27 Ancestral selection graph, $g_{1.5}$ where the ancestry, $Tr_{1.5}$ is indicated by a thicker line 143

6.28 The ancestral trees $Tr_{0.5}$ (A) and $Tr_{1.5}$ (B) of the selection graphs in Figure 6.23 and Figure 6.27 144

6.29 Trace of the parameter σ, $P(\sigma) \sim U(0, 2.5)$ and the ancestry is $Tr_{1.5}$ 145

6.30 Trace of t_{UA},$P(\sigma) \sim U(0, 2.5)$ and the ancestry is $Tr_{1.5}$ 145

6.31 Histogram of $w(\sigma)$,$P(\sigma) \sim U(0, 2.5)$ and the ancestry is $Tr_{1.5}$ . . 146

6.32 Trace of the parameter σ where $P(\sigma) \sim U(0, 1.5)$ and the ancestry is $Tr_{1.5}$ 146

6.33 Trace of t_{UA} where $P(\sigma) \sim U(0, 1.5)$ and the ancestry is $Tr_{1.5}$ . . 147

6.34 Histogram of $w(\sigma)$ where $P(\sigma) \sim U(0, 1.5)$ and the ancestry is $Tr_{1.5}$ 147

6.35 Ancestry, Tr_{exp}, of a sample taken from an exponential growing population. All the events happen close together at the top of the tree. The time of each internal event is followed by the allele type of the vertex 149

6.36 Trace of the parameter σ,where $P(\sigma) \sim U(0, 2.5)$. The ancestry is Tr_{exp} 151

6.37 Histogram of $w(\sigma)$, where $P(\sigma) \sim U(0, 2.5)$. The ancestral tree Tr_{exp} 151

6.38 Trace of t_{UA} where $P(\sigma) \sim U(0, 2.5)$. The ancestral tree is Tr_{exp}. . 152

B.1 The dark edges are deletable edges. Graph A has initially three deletable edges. A branching event is added at the top and one deletable edge is taken away while two deletable edges are added. The resulting graph has 4 deletable edges. Graph B has three deletable edges. After a branching event is added between two branching events, only one deletable edge is added and the graph has 4 deletable edges. 167

List of Tables

5.1 Analytic and debugging results of the MCMC $\{\Phi_i\} \sim f_G(g|\sigma)$. $P(B)$ is the probability of one branching event, but not a bubble, Figure 5.4 (B). $P(C)$ is the probability of one bubble, Figure 5.4 (C). t_{UA}, time to ultimate ancestor and t_{FE} is time to first event. 93

5.2 Time to ultimate ancestor and probability that the ultimate ancestor is of type 0 for different values of σ when $\mathcal{E} = \{c\}$ and type configuration as in diagram 5.5 96

5.3 Analytic and debugging results of the MCMC $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$ where the graphs have event set $\mathcal{E} = \{c\}$. $P(A)$ is the probability of $B_{UA} = 0, B_{V_{L(1)}} = 0$ and $B_{V_{L(2)}} = 0$, Figure 5.5 (A). $P(B)$ is the probability of $B_{UA} = 0$ and $B_{V_{L(1)}} \neq B_{V_{L(2)}}$, Figure 5.5 (B). $P(C)$ is the probability $B_{UA} = 0$ and $B_{V_{L(1)}} = B_{V_{L(2)}} = 1$, Figure 5.5 (C). 96

5.4 The time t_0, t_1, t_2 and t_3 for $\sigma = 0.5$ and $\sigma = 1.5$ when the graph has event set $\mathcal{E} = \{b, c, c\}$ but no bubbles as can be seen in Figure 5.6 97

5.5 Analytic and debugging results of the MCMC $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$. $P(A)$ is the probability of the configuration as in Figure 5.6 . . . 98

6.1 The mean ladder index $\hat{LI}$, the mean number of branching $\hat{N}_b$, the mean number of deletable edges $\hat{nde}$ and the mean number of bubbles $\hat{nbl}$ of separate Markov chains, $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, |V_L| = 5)$ and $\{\Phi_{ik}\} \sim k(g, B_Y|\sigma \in \Sigma, |V_L| = 5)$ 111

6.2 The number of graphs having a ladder index $LI \geq 0.75$ and a ladder index $LI \geq 0.85$. The maximum ladder index $\max(LI)$ and the frequency of graphs showing branching events of the Markov chains $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, |V_L| = 5)$ and $\{\Phi_{ik}\} \sim k(g, B_Y|\sigma \in \Sigma, |V_L| = 5)$ 113

6.3 Acceptance rate,(%) of $\{\Phi_{if}\} \sim f_G(g, \sigma \in \Sigma, |V_L| = 5)$ and $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, |V_L| = 5)$ 115

6.4 The mean ladder index, $\hat{LI}$, the mean number of branching, $\hat{N}_b$ and the mean number of deletable edges $\hat{nde}$ of Markov chain $\{\Phi_{ik}\}$ where $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ 119

6.5 The acceptance rate,(%), of the Markov chain $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ broken in the different moves used for th MCMC120

6.6 The number of graphs, showing a ladder index $LI \geq 0.75$ and $LI \geq 0.85$. The maximum ladder index $\max(LI)$ and the frequency of graphs showing branching events where $\{\Phi_{ik}\}$ is sampled form $k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ and $k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ respectively. 121

6.7 The integrated autocorrelation time τ, the variance of IACT var(τ) and the cutting off value M of the ladder index, LI. The variance is $\times 1.0e + 008$ 128

6.8 The IACT τ, the variance of IACT var(τ) and the cutting off value M of t_{UA}. The variance is $\times 1.0e + 0.010$ 130

6.9 The IACT, (τ), the variance of IACT, (var(t_{FE})) and the value M of t_{FE}. The variance is $\times 1.0e + 007$ 131

6.10 The IACT, τ, the variance of IACT, var(τ) and M of t_{MRCA}. The variance is $\times 1.0e + 009$ 132

6.11 The IACT, τ, the variance of IACT, var(τ) and M of N_b. The variance is $\times 1.0e + 007$ 133

6.12 Summary of the statistics of $w(\sigma)$. τ is the IACT, acct is the overall acceptance percentage and cpu time is the run time of the MCMC in seconds. $Tr_{1.5}$ is the ancestral tree simulated for $\sigma = 1.5$ and $P(\sigma) \sim U(0, 2.5)$. $Tr_{1.5}^{(15)}$ indicate the ancestral tree simulated when $\sigma = 1.5$ and $P(\sigma) \sim U(0, 1.5)$. 152

Abstract

The genealogy of a random sample of a population of organisms can be represented as a rooted binary tree. Population dynamics determine a distribution over sample genealogies. For large populations of constant size and in the absence of selection effects, the coalescent process of Kingman determines a suitable distribution. Neuhauser and Krone gave a stochastic model generalising the Kingman coalescent in a natural way to include the effects of selection.

The model of Neuhauser and Krone, determines a distribution over a class of graphs of randomly variable vertex number, known as ancestral selection graphs. Because vertices have associated scalar ages, realisations of the ancestral selection graph process have randomly variable dimensions.

A Markov chain Monte Carlo method is used to simulate the posterior distribution for population parameters of interest. The state of the Markov chain Monte Carlo is a random graph, with random dimension and equilibrium distribution equal to the posterior distribution.

The aim of the project is to determine if the data is informative of the selection parameter by fitting the model to synthetic data.

Acknowledgment

I want to acknowledge my supervisors Dr. Geoff Nicholls from the Department of Mathematics and Assoc. Prof. David Scott of the Department of Statistics for their help, suggestions and encouragement throughout the project. Our weekly discussions will be missed. Dr Nicholls helped me in formulating the project and spent hours helping me to find bugs in the code. Assoc. Prof Scott was very helpful especially at the end when Dr. Nicholls left the University of Auckland. His suggestions on the content of the thesis are greatly appreciated.

I wish to thank Prof. Allen Rodrigo of the Bio-informatics Institute of the University of Auckland for his involvement during the initial stages of this project.

As the recipient of a Bright Future Top Achiever Doctoral Scholarship, I want to thank the Foundation of Science and Industry of New Zealand for the financial support. Without the support this thesis would not have been possible.

I also want to thank all my friends - there are too many to name - but espe-

cially Prof. John Butcher for the moral support throughout my PhD.

The last but certainly not the least, thank you to the most important people in my life, my husband Danie and my two children Renier and Nicola. You guys are better than A1. Sometimes the going was tough, but still you supported me and were always there with smiles.

Chapter 1

Introduction

Evolution is an ongoing process driven by mutation, natural selection, recombination, population structure, random genetic drift and migration. The result of this process as seen today, is the foundation for the ongoing process of the future. To be able to predict the future, one needs to understand the past, by analyzing the data of today.

The field of evolutionary genetics can be roughly divided into two main areas, phylogenetics and population genetics. Phylogenetics involves the study of relationships between and among species, genes, nucleotides or proteins. The field of population genetics encompasses the methods to investigate the influence of evolutionary forces on an entire population.

The aim of both these fields is to develop suitable models and computational methods to reconstruct the evolutionary history from the information available. Since the evolutionary history is an inverse problem, evolutionary

genetics is home to extremely challenging computational problems.

One of the earliest models is the Wright-Fisher model, which was introduced by Sewall Wright [28] in 1931 and R. A. Fisher [8] in 1930. The Wright-Fisher model is a discrete-time stochastic process. It assumes discrete non-overlapping generations, a constant population of size N, no natural selection, no mutation and no recombination. The genes of generation $t+1$ are sampled with replacement from generation t.

Suppose the population consists of two types of genes, $\{0, 1\}$. Let $X(t)$ denote the number of genes of type 0 in generation t. Since the genes in generation $t+1$ are sampled with replacement from generation t, $X(t+1)$ has a binomial distribution with index N and parameter $\frac{X(t)}{N}$, i.e $X(t+1) \sim \text{Bi}(N, \frac{X(t)}{N})$, Ewens [5].

The n-coalescent on the other hand, is a continuous-time stochastic process which was introduced by Kingman [16]. The coalescent can be derived as the diffusion limit of the Wright-Fisher model. It consists of two processes, a death process, $\mathcal{D}$ and a jump process $\mathcal{J}$. Let $\mathcal{A}_n(0) = \{1, 2, 3, ...n\}$ be the labeled elements of a sample of n particles. An equivalence relation is defined on the set $\mathcal{A}_n(0) = \{1, 2, ...n\}$ by $i \sim j$ if i and j are the descendant particles of a particle, c, which was present at a time t in the past. The particle c is called the common ancestor of the particles i and j. If $i \sim j$ let (i, j) denote the equivalence class containing the elements i and j. The number of equivalence classes at time t is denoted by $\mathcal{D}_t$ and the set containing the equivalence classes

at time t is denoted by $\mathcal{J}_t$. For instance at time $t = 0$ each element of the sample is an equivalence class by itself such that $\mathcal{J}_{t=0} = \{(1), (2), ...(n)\}$ and $\mathcal{D}_{t=0} = \{n\}$.

Suppose at time $t = t_1$ two of the elements of $\mathcal{A}_n(0) = \{1, 2, 3, ...n\}$ labeled 1 and 3 have a common ancestor. The set of equivalence classes is $\mathcal{J}_{t_1} = \{(1, 3), (2), (4), ...(n)\}$ and the number of equivalence classes changes to $n - 1$. The process $\mathcal{D} = \{n, n - 1, n - 2, ...1\}$ is a pure death process, with absorbing state 1. Let $T = \inf\{t \geq 0; D_t = 1\}$. Since the elements of the sample at time T are all in the same equivalence class $\mathcal{J}_T = (1, 2, ...n)$, the time T is known as the time to the most recent common ancestor and is denoted by t_{MRCA}.

Let $\tau_k = t_k - t_{k-1}$ be the length of time the death process spends in the state $D_{t_k} = k$. Each τ_k is exponentially distributed with rate $\binom{k}{2}$, where k is the number of equivalence classes in the jump process at time t_k. The times τ_k are independent of each other for each $k = 2, 3, ...n$ and therefore $t_{MRCA} = \sum_{k=2}^{n} \tau_k$.

By letting $N \to \infty$ the coalescent is the limiting approximation of the genealogy of a sample of n genes in a large population. The genealogy can be represented by a binary tree where the leaves of the tree represent the sample n individuals as can be seen in Figure 1.1. The binary tree is depicted with the root of the tree at the top and the leaves at the bottom. The time of sampling is assumed to be zero. By going up the tree from the leaves to the root of the tree,

time is increasing from zero to t_{MRCA}. If the tree is applied to evolutionary biology, going up the tree is going backward in time.

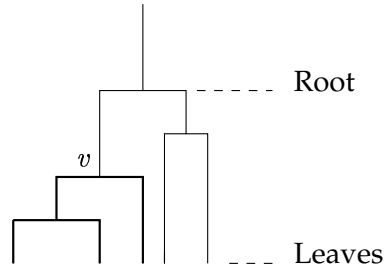


Figure 1.1: Binary tree representing the coalescent process

It is easily seen that the representation of the genealogy by a binary tree is in agreement with the death and the jump processes of the coalescent. The lineages are represented by the edges of the tree. By going up the tree, two lineages coalesce at a vertex and the number of lineages decreases by one. The death process $\mathcal{D}$ is on the number of lineages in the tree, where the process starts with the size of the sample and going up the tree, ending with the root.

When starting with any vertex v in the tree and following the edges connected to the vertex down toward the leaves, each internal vertex connects to two or more elements in the sample. The original vertex, v is the common ancestor of the elements it connects to in the sample. The elements in the sample, which are connected by a path to the vertex v, are in the same equivalence class. An example of a path connecting a vertex v to the elements in the sample is illustrated in Figure 1.1 by a thicker line.

The time between two events is exponentially distributed with rate $\binom{k}{2}$ where

k is the number of lineages between the two events. Let V_L denote the set of leaf vertices. The leaf vertices represent a sample from a population. The sample can be individuals, genes or proteins. If the sample is of size n, let $V_L = \{1, 2, 3, ..., n\}$.

The number of lineages between the leaves and the first coalescent event is equal to the size of the sample. A binary tree can easily be simulated from the set of leaf labels V_L of size n by the following algorithm.

```
Step 1:  while k ≠ 1
Step 2:    draw t ~ exponential(binom(k,2))
Step 3:    set t_i = t_{i-1} + t
Step 4:    set k = k - 1
Step 5:  go to step 1
```

The Wright-Fisher model can be extended to incorporate mutation. Let μ be the probability that a gene in generation $t+1$ is different from its parent in generation t.

Since the mechanism of transferring a gene to the next generation and the mutation process are independent of each other, the coalescent or binary tree is created first. The mutation process is then superimposed on the tree.

Different mutation models have been developed. These processes can be divided into three groups, the infinite allele model, the infinite sites model and the finite sites model [14]. All these models assume neutrality, constant population size, no recombination and no population structure.

In the infinite allele model each mutation produces an allele not previously seen in the population. Under the Wright-Fisher model, with probability μ a gene in generation t mutates to a gene not previously seen.

The infinite sites model specifies that the mutation happens at a site where the mutation has not happened before.

The finite sites models are more realistic. These models include the Jukes-Cantor model, $J69$, the $K80$ model developed by Kimura and the $F81$ by Felsenstein to mention just a few of the models. For a more detailed description of these models see Swofford *et al* [24].

To keep things simple assume the population consists only of two types of alleles denoted by 0 and 1. Suppose the rate of mutation from a type 0 to a type 1 is μ and the rate of mutation from a type 1 to a type 0 is ν.

Let

$$Q = \begin{pmatrix} -\mu & \mu \\ \nu & -\nu \end{pmatrix}$$

be the rate matrix associated with the mutation process. Let i and j be the two vertices of an edge $\langle i, j \rangle$ where i is at an earlier time than the vertex j. Let $\tau = t_j - t_i$ be the length of the edge $\langle i, j \rangle$. The number of mutations on each edge is a function of the length of the edge. Let $P(\tau) = e^{Q\tau}$ be the probability

matrix associated with the rate matrix Q over an edge of length τ, then

$$P(\tau) = \begin{pmatrix} \frac{\nu}{\nu+\mu} + \frac{\mu}{\nu+\mu}e^{-(\mu+\nu)\tau} & \frac{\mu}{\nu+\mu} - \frac{\mu}{\nu+\mu}e^{-(\mu+\nu)\tau} \\ \frac{\nu}{\nu+\mu} - \frac{\mu}{\nu+\mu}e^{-(\mu+\nu)\tau} & \frac{\mu}{\nu+\mu} + \frac{\mu}{\nu+\mu}e^{-(\mu+\nu)\tau} \end{pmatrix}$$

The entries on the diagonal of $P(\tau)$ are the probabilities that no change has occurred over the edge. Since each edge has a unique length, a different probability matrix $P(\tau)$ is associated with each edge.

Since the process of transferring genes from one generation to the next is independent of the mutation process, the tree can be simulated using the algorithm above. The mutation process can be superimposed on the tree. The type of the root is sampled from a unique stationary distribution π. The mutation process propagates independently along the edges toward the leaves of the tree using the transition matrix $P(\tau)$ associated with each edge.

Let $\mathcal{B}$ be the sample space of all possible allele types. Each vertex v of the tree has a time value t and an allele type $B \in \mathcal{B}$.

When estimating a parameter,Υ, such as the mutation rate, the idea is to calculate the likelihood, $\mathcal{L}(\Upsilon) = \mathbf{P}(D|\Upsilon)$, which is the probability of the data given the parameter Υ for a range of values for the parameter in question. In a maximum likelihood framework the idea is to find the tree together with any other parameters in the parameter space which has the highest likelihood to match the given data. Let Tr be the genealogy of a sample of size n and let μ

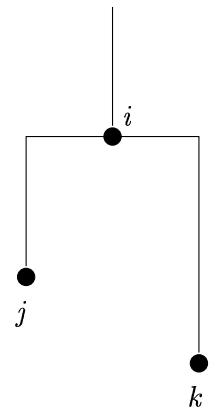


Figure 1.2: Parent vertex i with two children vertices j and k

be the mutation rate. Under the coalescent model, the likelihood $\mathcal{L}(\Upsilon)$, is the expression

$$\mathcal{L}(\Upsilon) = P(D|\Upsilon) = \int_{(Tr,\mu)} \mathbf{P}(D|Tr,\mu)\mathbf{P}(Tr,\mu)d(Tr,\mu) \tag{1.1}$$

where $\mathbf{P}(Tr,\mu)$ is the joint prior density of Tr and μ.

The probability of the data D at the leaf tips, given the genealogy Tr and the mutation rate μ, $\mathbf{P}(D|Tr,\mu)$ is a function of the topology of the coalescent tree and the lengths of the edges of the tree.

One of the challenges is to find an explicit formula for the expression $\mathbf{P}(D|Tr,\mu)$ in Equation 1.1.

The pruning algorithm of Felsenstein [7] can be used to estimate $\mathbf{P}(D|Tr,\mu)$ in Equation 1.1. The algorithm uses the following principle recursively.

When moving from the leaves of the tree towards the root of the tree, each vertex is of in-degree two and of out-degree one. This means that each vertex i has two children vertices k and j as can be seen in Figure 1.2. Let the edges connecting vertex i with vertex j and vertex i with vertex k be of lengths $\tau_{\langle j,i \rangle}$ and $\tau_{\langle k,i \rangle}$ respectively. Let $\mathcal{B}$ be the space of all possible types and let B_i, B_j, $B_k \in \mathcal{B}$ be the types at vertices i, j and k respectively. Then

$$\mathbf{P}(B_i|Tr,\mu) = \sum_{B_j \in \mathcal{B}} e^{\mu Q(\tau_{\langle j,i \rangle})} \mathbf{P}(B_j|Tr,\mu) \quad \times \sum_{B_k \in \mathcal{B}} e^{\mu Q(\tau_{\langle k,i \rangle})} \mathbf{P}(B_i|Tr,\mu)$$

for all type $B_i, B_j \in \mathcal{B}$.

By starting at the root of the tree, where the ancestral type of the root has distribution π, and working recursively down to the leaves of the tree.

$$\mathbf{P}(D|Tr,\mu) = \sum_{B_i \in \mathcal{B}} \pi \mathbf{P}(B_i|Tr,\mu)$$

Normally sophisticated Mont Carlo methods are used to calculate the likelihood $\mathcal{L}(\Upsilon)$ in Equation 1.1. The two main classes of Monte Carlo methods applied are importance sampling methods and Markov chain Monte Carlo methods.

The idea of importance sampling is to find a function Q known as the importance function, which allows the samples to be drawn from a part of the parameter space where there is a higher probability for the parameters to match

the data.

The Equation 1.1 is rewritten in the form

$$\mathcal{L}(\Upsilon) = \mathbf{P}(D|\Upsilon) = \int_{Tr,\mu} \mathbf{P}(D|Tr,\mu)\mathbf{P}(Tr,\mu)\frac{Q(Tr,\mu)}{Q(Tr,\mu)}d(Tr,\mu) \tag{1.2}$$

Griffiths and Tavaré [13] derived recursively the joint distribution of the number of alleles, K_n and the time to the most recent common ancestor by using the structure of the coalescent. The recursive equation is applied to create a Markov chain to estimate the mutation rate and the time to the most recent common ancestor of a sample of DNA sequences. This recursion is derived in the following manner.

For a sample of size n, let $\mathbf{n} = \{n_1, n_2, ...n_K\}$ be the configuration of a sample where n_i is the number of type i alleles and $n = n_1 + n_2 + ... + n_K$. By going from the leaves of the tree to the root of the tree, the first event is a mutation with probability $\frac{\theta}{n-1+\theta}$ or a coalescence with probability $\frac{n-1}{n-1+\theta}$.

Let $\mathbf{e}_j = (0, ..., 0, 1, ...0)$ be a vector of length K, with the jth entry equal to 1. If the first event is a mutation then for the configuration to be $\mathbf{n}$ at the leaves, either a type i changed into a type i or a type i changed into a type j. If a type i changed into a type i the configuration before the mutation was $\mathbf{n}$ otherwise the configuration was $\mathbf{n} + \mathbf{e}_i - \mathbf{e}_j$.

In the case when the first event is a coalescence involving a type j, then the

configuration before the event must have been $\mathbf{n} - \mathbf{e}_j$.

By using the argument above the following recursion can be derived for the probability of a sample configuration $\mathbf{n}$.

$$\begin{aligned} q(\mathbf{n}) &= \frac{\theta}{n+\theta-1}\left(\sum_{i=1}^{K}\frac{n_i}{n}p_{ii}q(\mathbf{n}) + \sum_{i,j\in\{1,2,\ldots K\},n_j>0,i\neq j}\frac{n_i+1}{n}p_{ij}q(\mathbf{n}+\mathbf{e}_i-\mathbf{e}_j)\right) \\ &\quad + \frac{n-1}{n+\theta-1}\sum_{j\in\{1,2,\ldots K\},n_j>0}\frac{n_j-1}{n-1}q(\mathbf{n}-\mathbf{e}_j) \end{aligned} \tag{1.3}$$

The recursion in Equation 1.3 above is used by Griffiths and Tavaré [13] as the importance function in Equation 1.2.

All these models assume neutrality. The Neutral Theory of Kimura, [15] states that the variation observed at the molecular level is selectively neutral and that deleterious alleles are removed from the population. Only a few advantageous mutations become fixed in the population. The fixation is because of random genetic drift and not natural selection as was previously believed.

As more information becomes available in the form of DNA sequences, high levels of variation in regions of some sequences is incompatible with the neutral theory. These sequences can be separated into groups where there is little variation within a group but large variation between groups. This can be explained by balancing selection where the frequency of different alleles in a population is kept constant. The mechanism of balancing selection is overdominance or negative frequency dependent selection. The n-coalescent can

be adapted to incorporate balancing selection (Hein [14]).

The mutation of a codon which does not change the amino acid of a protein is known as synonymous mutation. Non-synonymous mutations on the other hand cause a change in the amino acid of a protein which can be detrimental to the organism. However, if this change is to the advantage of the population, the rate of fixation of non-synonymous mutations can be higher than the fixation of synonymous mutations. This is known as positive or directional selection.

One of several examples of directional selection is the high level of variation in amino acids caused by the env gene of HIV (Nielsen [19]).

Under directional selection the frequency of an allele is changed over time. The coalescent can not incorporate directional selection. Therefore the challenge was and still is to incorporate natural selection in a model and use the model to infer the time to the most recent common ancestor, the mutation rate and the selection parameter from a given sample.

The Moran model with natural selection and mutation was used by Neuhauser and Krone [17] to extend the idea of a binary tree representing a genealogy without selection to a graph representing a stochastic process incorporating directional selection.

The Moran model assumes a haploid population of constant size, N and non-overlapping generations. At discrete time points one individual in the population is chosen at random to give birth to a single offspring. As soon as

an individual is born, an individual in the population is chosen at random to die. This individual may be the parent of the offspring but not the offspring itself. Assume two types of individuals in the population. When incorporating mutation in the model, the offspring of an individual is of a different type than the parent with probability μ. The Moran model can be used to model natural selection. Natural selection is defined by letting the individuals of one type have a higher birth rate than the individuals of another type. A mutation process is required for natural selection to take place. Neuhauser and Krone [17] extended the graphically representation of the coalescent by a binary tree to the Moran model with mutation and natural selection by means of the biased voter model. When $N \to \infty$ the model can be represented by an ancestral selection graph. In the neutral case where natural selection is not present the ancestral selection graph collapses to the binary tree of the coalescent. This is discussed in more detail in Chapter 2.

An ancestral selection graph need not be a tree. However its degree-one vertices correspond to individuals sampled from a population, and it does contain the tree-like genealogy of those individuals as a sub-graph. Each graph vertex carries a real scalar time parameter and a binary type parameter.

Statistical information about genealogies is summarized by a probability distribution on a space of augmented-tree graphs.

Suppose DNA data is gathered from individuals in the present and used to infer their common genealogy. The data determine a measure on the space of

genealogies. This measure, which is related to the Bayesian posterior probability distribution for genealogies, is defined in such a way that the probability that the unknown true genealogy lies in any particular set is given as the measure of that set. Recent work treating probability distributions on genealogies and related but more general graph spaces include studies of the genealogy of spatially unstructured (Stephens and Donnelly [23]) and spatially structured (Bahlo, Griffiths and Ewing *et al* [1, 6]) populations, of recombining populations (Donnelly and Kurtz [3]), studies of the intertwined genealogies of a host and virus (Welch *et al* [26]) and genealogies for individuals subject to natural selection (Slade [21]). The analysis of these probability distributions (for example, the estimation of moments of parameters) is in general intractable. So called "full posterior" studies of genealogies determined from biological data have been made for just a few population models.

For some problems (Stephens and Donnelly, Bahlo and Griffiths, Slade [23, 1, 21]) it has been possible to find efficient importance-sampling distributions. However these methods do not seem to be practical for large collections of DNA sequence data taken from rapidly evolving species such as HIV. Markov chain Monte Carlo methods (Wilson *et al*, Drummond *et al*, Ewing *et al* [27, 4, 6] make use of a class of transitions in the target space of graphs which includes any importance-sampling distribution as a special case. It is hoped that efficient Markov chain Monte Carlo (MCMC) algorithms can be found to treat those "full posterior" problems which are not amenable to analysis by

importance sampling.

This is done by presenting a MCMC algorithm which samples the posterior probability distribution for the Krone-Neuhauser model [17] of a single-sex population evolving subject to selection.

The state space of the Markov chain is formed by taking the union, over all admissible graphs, of the parameter space associated with each graph as discussed in Chapter 2. Because the number of vertices is allowed to vary from one graph to another, and each vertex carries a scalar parameter, this state space is a union of spaces of unequal dimension. This Monte Carlo algorithm is defined in Chapter 3 so that it determines a random graph process on this target space with equilibrium equal to the desired posterior probability distribution over ancestral selection graphs.

An efficient importance sampling algorithm by Slade [21] has been given for this problem by extending the recursive algorithm (Equation 1.3) given by Griffiths and Tavaré to ancestral selection graphs. However, the MCMC algorithm presented here generalizes far more easily. For example, it is straightforward to condition the probability distribution on exact knowledge of the embedded genealogy, if one simply rejects MCMC updates which modify the genealogy one obtains the correct target equilibrium distribution as can be seen in Chapter 6. It is not clear how to condition an importance sampling algorithm in this way.

The data structures, the arrays and the implementation are discussed in

Chapter 4, followed by a discussion on the debugging procedures in Chapter 5.

The aim of this thesis is to determine if the data available is informative of a selection parameter by taking a sample of genetic material from a population of organisms. The experiments and the results are presented in Chapter 6.

Chapter 2

Stochastic geometry

Stochastic geometry is the study of randomly evolving geometric objects. The geometric object under discussion in this chapter is the ancestral selection graph. The ancestral selection graph, as introduced by Neuhauser-Krone [17] is a graphical representation of the Moran model in evolutionary biology.

The Moran model describes a haploid population of constant size N. One individual in generation t is chosen at random to give birth to an offspring in generation $t+1$. Under the assumption that the population consists of individuals of two different allele types, natural selection is incorporated in the model by letting individuals of one allele type give birth to offspring at a higher rate than the individuals of the other type. In generation $t+1$, one individual is chosen at random to die. The offspring of the individual giving birth is of a different allele type than the parent with probability μ.

Neuhauser-Krone [17] used the biased-voter model as an equivalent to the

Moran model. The biased-voter model can be approximated in the diffusion process by a graph process known as the ancestral selection graph. This chapter is started by a discussion on the biased-voter model, followed by the introduction of the ancestral selection graph, which leads to the definition of the probability space $(\Gamma, \mathcal{F}, \mathcal{P})$ of all admissible ancestral selection graphs.

The probability space $(\Gamma, \mathcal{F}, \mathbf{P})$ of ancestral selection graphs can be thought of a space of unlabeled selection graphs, which can be extended to a space of type-labeled graphs $(\tilde{\Gamma}, \mathcal{F}, \mathbf{P})$.

Unlabeled ancestral graphs are graphs with only a time variable associated with each vertex. The space of unlabeled graphs is discussed to describe the density of a process realizing graphs.

The space of unlabeled graphs is then extended to the space of type-labeled graphs. The type-labeled graphs have a time variable and a type variable associated with each vertex. The type label is the result of the mutation process. The mutation process is realized by an appropriate transition matrix associated with each edge and is discussed in Section 2.3.

These discussions lead to the definition of the posterior probability distribution on the probability space $(\tilde{\Gamma}, \mathcal{F}, \mathcal{P})$ of type-labeled graphs. This posterior distribution is the target distribution of a Markov chain Monte Carlo sampling scheme as discussed in Chapter 3.

2.1 The model

Neuhauser and Krone [17] introduced a model for an evolutionary process incorporating natural selection by using the biased-voter model. The biased-voter model [17] describes a microscopic stochastic process to construct a genealogy of a small sample n of a population of constant size N, incorporating natural selection. Suppose that N is the number of individuals in a haploid population and suppose that there are only two types of individuals in the population. Denote the two types by 0 and 1 and suppose type 1 has a selective advantage over type 0. This advantage is represented in the model by having a higher birth rate for the favored type. Let type 0 have a birth rate λ_0 and type 1 have a birth rate $\lambda_1 = \lambda_0(1 + s_N)$, with $s_N \geq 0$ so that s_N is the selective advantage of allele 1. The population size N, is constant. This is maintained by the rule that at a birth event, the offspring replaces a randomly chosen member of the population. An individual may replace its own parent.

In the model, symmetric mutation between the two allele types may occur at birth. The offspring of a parent of type $0(1)$ has type $1(0)$ with probability μ_N. A graphical representation of this model is known as the biased-voter model with selection and mutation on a complete graph. Suppose, as in the Kingman coalescent, n individuals are sampled from the present population of size N. Using the biased-voter model, one can, by reversing the time, follow the ancestral history of these n individuals. Let UA denote the ultimate ances-

tor and let the time to the ultimate ancestor be t_{UA}^N. This reversed-time process is called the dual or ancestral process $\{\mathcal{G}_n^N(t) : 0 \leq t \leq t_{UA}^N\}$. An independent mutation process $\{\mathcal{Y}_n^N : 0 \leq t \leq t_{UA}^N\}$ evolves along the edges of the ancestral process. The random graph realized by this process is called the ancestral selection graph. For a more detailed description of the microscopic process see the paper by Krone and Neuhauser [17].

The biased-voter model is equivalent to the Moran model with selection and mutation. Let $Z_0(t)$ and $Z_1(t)$ be the number of alleles 0 and 1 respectively in a population of size N. The process $Z_0(t)$ satisfies the Markov property $P(Z(t_1) = j|Z(t \leq t_0)) = P(Z(t_1) = j|Z(t_0)), \text{if}\, t_0 \leq t_1$ The behaviour of this Markov chain can be approximated by a limiting diffusion process on $[0, 1]$ by rescaling the parameters and by letting $N \to \infty$, $N\mu_N \to \theta$ and $Ns_N \to \sigma$. Let z denote the frequency of allele 0 in the population at time t. Allowed values of z are $\{0, \frac{1}{N}, ...1\}$. The diffusion process on $[0, 1]$ has drift $a(z) = -(\frac{\sigma}{2})z(1-z) + \frac{\theta}{2}(1-2z)$ and diffusion parameter $b(z) = z(1-z)$. The process has a unique stationary distribution with density

$$\eta(z) = Kz^{\theta-1}(1-z)^{\theta-1}e^{-\sigma z}, z \in [0, 1]$$

where K is a normalizing constant ([17]).

As $N \to \infty$ the time process converges in distribution to a stochastic process $\{\mathcal{G}_n(t), t \geq 0\}$, which is called the ancestral selection graph. As in the

Kingman coalescent only the global structure of the ancestral graph is of interest which consists only of the coalescing and branching events.

2.2 Ancestral graphs

The ancestral selection graph can be thought of as a list of coalescing and branching events. The graph is generated from the present back into the past (so time t increases into the past). Let the elements of $\mathcal{A} = \{1, 2, ..., n\}$ label a set of n particles. At a coalescent event, two particles merge resulting in one particle, while at a branching event one particle splits into two. In the process defined below, coalescent events dominate. The process stops the moment the number of particles drop to one. It realizes a directed, connected graph $G = (V, E)$. The branching and coalescing process itself is called the ancestral selection graph process, $\{\mathcal{G}(t)\}$. Let M denote the total number of events in one realization. A specimen ancestral selection graph is depicted in Figure 2.1 where $M = 5, n = 4$ and at t_3 is a branching event.

The graph process $\{\mathcal{G}(t)\}$ consists of two sub-processes, a set-valued process $\{\mathcal{A}(t) : t \geq 0\}$, denoting the set of particles at time t and a jump process $\{\mathcal{R}_m : m \geq 1\}$, giving the times at which events occur.

Let $t = R_0$ denote the time associated with the leaves. At time $t = R_0$, $\{\mathcal{A}(t)\} = \mathcal{A}$, the set of labels associated with the terminal nodes of the graph.

As branching and coalescing events occur the number of lineage labels

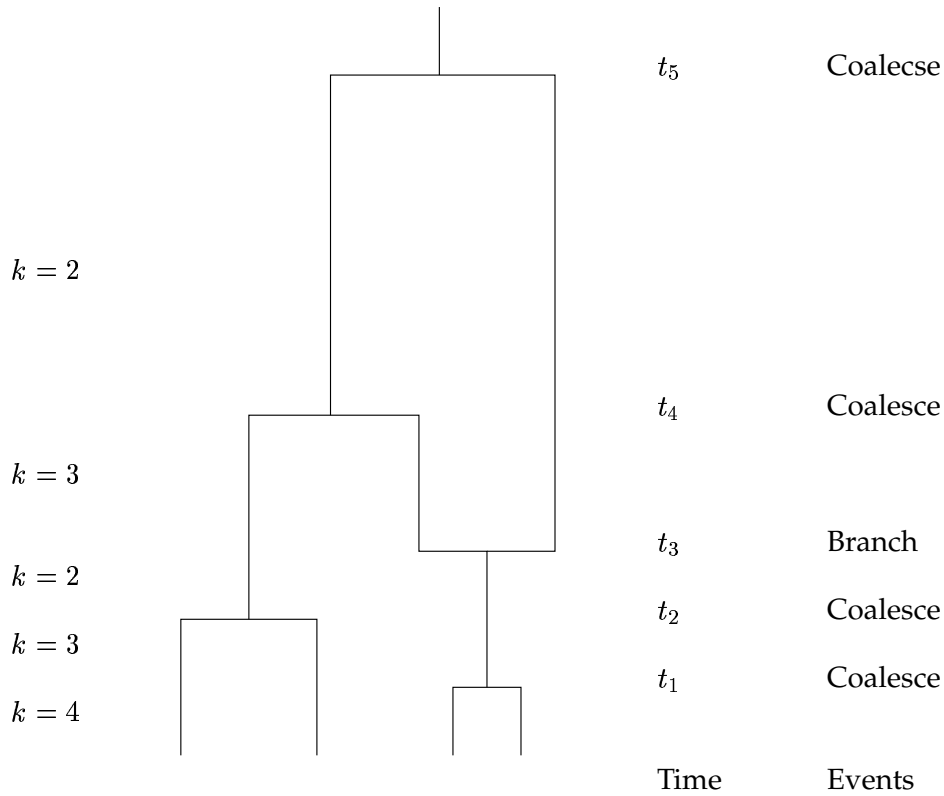


Figure 2.1: Ancestral selection graph

in this set may increase or decrease. For each $t \geq R_0$, $\{\mathcal{A}(t)\}$ is a subset of $\{1, 2, 3, 4...\}$. Let $|\mathcal{A}(t)|$ denote the cardinality of the set $\mathcal{A}(t)$. The process stops when the final coalescence occurs and $|\mathcal{A}(t)| = 1$.

The vertex at this time is called the ultimate ancestor (UA). The time of the UA vertex is denoted t_{UA} with $t_{UA} = \inf\{t : |\mathcal{A}(t)| = 1\}$.

Let $R_m, m \geq 1$ be a time at which either a coalescing or a branching event occurs. Let $\sigma \geq 0$ be a real positive constant. While $|\mathcal{A}(R_{m-1})| = k$, coalescing happens at rate $\binom{k}{2}$ and branching happens at a rate $\frac{\sigma k}{2}$. The time intervals $\{R_m - R_{m-1}\}$ are independent and exponentially distributed with rate param-

eter $\binom{k}{2} + \frac{\sigma k}{2}$.

If a branching event happens at time R_m a random particle j branches resulting in an extra particle with label $n+m$, so that $\mathcal{A}(R_m) = \mathcal{A}(R_m^-) \cup \{n+m\}$. When two random ancestors i, j coalesce with $i < j$, then $\mathcal{A}(R_m) = \mathcal{A}(R_m^-) \backslash \{j\}$ and the resulting ancestor is labeled i.

The selection parameter σ determines the amount of branching in the ancestral graph. If $\sigma = 0$ the ancestral selection graph collapses to the n-coalescent of Kingman. Large values of σ result in many branching events.

As soon as the ancestral graph is created, the process $\{\mathcal{Y}_n(t), t \geq 0\}$ of determining the types of the particles at time t is run from the root of the graph to the terminal or leaf vertices of the graph.

At the scale of the ancestral selection process, the process $\{\mathcal{Y}_n(t), t \geq 0\}$ evolves independently along the edges of the selection graph according to a Poisson process with rate $\frac{\theta}{2}$. This means that associated with each vertex $v \in \mathcal{G}$ is a particle type and a time t.

In the ancestral selection graph for each $v \in V$, let B_v denote the particle types $B_v \in \{0, 1\}$ at the vertex. If V_L and V_A respectively denote the sets of leaf and the ancestral (internal) vertex labels, so that $V = V_L \cup V_A$, then $B_v, v \in V_L$ will typically be the given data whilst the ancestral particle types $B_v, v \in V_A$ will be the unknowns.

At a branching event one of the particles represents the original particle and continues on a straight line through the vertex. This is called the continuing

edge. The other edge, called the replacing edge only represent extra births of the selective advantageous particle. (Neuhauser and Krone call the replacing edge the incoming edge.)

2.3 Space of ancestral graphs

The probability space is the non-empty measure space $(\Gamma, \mathcal{F}, \mathbf{P})$, where Γ is the sample space of admissible ancestral selection graphs, $\mathcal{F}$ is a suitable σ-algebra of subsets of Γ and $\mathbf{P}$ a probability measure on Γ. An example of a set $A \in \mathcal{F}$ is the set of all ancestral graphs where the UA is less than 10 generations old.

2.3.1 Unlabeled graphs

For a graph $g \in \Gamma$, let $V_c \subset V_A$ be the set of vertices where coalescing of two edges occur, $V_b = V_A \setminus V_c$ the set of vertices where an edge branches and V_L the set of leaf tips, i.e the set of vertices representing the data, $V_g = V_c \cup V_b \cup V_L$. We define corresponding sets of edges, $E_g = E_b \cup E_c$, with E_b the set of edges branching at the top and E_c the set of edges coalescing at the top ("up" is into the past, increasing t). Let N_c denote the number of coalescing events, N_b the number of branching events and n the number of leaves, then $N_c = n + N_b - 1$.

An ancestral selection graph g is a directed graph. Associated with each vertex $v \in V_g$ is a time $t_v \in \Re$, increasing from the leaves to the root vertex of the graph (UA). Edge $\langle v, w \rangle \in E_g$ is directed from the "child" v to w, the

"parent", so our convention is $t_v \leq t_w$.

For each vertex, $v \in V_g$, let d_{in} and d_{out} denote the in-degree and the out-degree respectively. Vertices $v \in V_g$ of out-degree one and in-degree zero, are leaf vertices and are located at time zero, $t_v = 0$. Vertices of out-degree one and in-degree two are coalescent vertices, $v \in V_c$. Branching vertices $v \in V_b$ are of in-degree one and out-degree two.

The set $V_A = V_c \cup V_b$ denotes all ancestral vertices. Let t_A denote the ordered set of times associated with the vertices $v \in V_A$. Let

$$\chi_{V,E} = \{t_A; t_A \in [R_0, \infty)^{N_b+N_c} \text{ and for } \langle i, j\rangle \in E_g, t_i < t_j\},$$

denote the space of admissible vertex times for a given topology.

Edges $i \in E_g$ at each branching vertex carry a label $S_i \in \{0, 1\}$ indicating whether the edge is continuing (1) or a replacing (0). At any branching vertex exactly one outgoing edge is continuing and one is replacing. These values act as gating values on selective advantage passing through the branching vertex. If $E_{bt} = \{\langle u, v\rangle \in E_g : u \in V_b\}$ is the set of edges out-going from branching vertices then $S_b = (S_i)_{i \in E_{bt}}$ is the set of gating values. Denote by $\mathcal{S}_{V,E}$ the set of admissible assignments of selection gating values S to edges in the graph topology (V, E).

The set $\Gamma^{\star}_{N_{b,n}}$ of admissible ancestral selection graph topologies (V, E) on N_b branching events and n leaves is determined by the requirement that $|V_L| =$

$n, |V_b| = N_b, V_g = V_L \cup V_A$ and the total number of edges below an event is not less than two. The space Γ of all selection graphs with exactly n leaves is then

$$\Gamma = \bigcup_{N_b=0}^{\infty} \bigcup_{\{V,E\}\in\Gamma^{\star}_{N_b,n}} \bigcup_{t_A\in\chi_{V,E}} \bigcup_{S\in\mathcal{S}_{V,E}} \{(V, E, t_A, S)\}$$

For $g \in \Gamma$, let $g = (V_g, E_g, t_A(g))$ be an ancestral selection graph with $M = N_b + N_c$ ancestral vertices. Let $\nu(t_A)$ denote the element of volume in χ_{V_g,E_g} so that

$$d\nu(t_A) = dt_1 dt_2 dt_3 ... dt_M$$

The element of measure dg is $d\nu(t_A(g))$ with counting measure over distinct topologies and selection gating values.

Denote by $\mathcal{F}$ the Borel σ-algebra of sets in Γ. For each set $B \in \mathcal{F}$ let $\mathbf{P}(B) = Pr\{\mathcal{G} \in B\}$ be a distribution given in terms of a certain density $f_G(g|\sigma)$ on Γ, so that $\mathbf{P}(B) = \int_B f_G(g|\sigma)dg$. The density f_G has a free parameter σ which controls the shape of the distribution in Γ. As σ is increased the distribution is weighted more heavily towards a larger number of branching events.

Vertex labels are ordered by time and are consecutive. Informally, any line drawn across the graph at time $t \in (t_v, t_{v+1})$ intersects a constant number $k_v = |\{\langle i, j\rangle \in E_g : t_i \leq t_v \leq t_{v+1} \leq t_j\}|$ of edges. See Figure 2.1. Let $\rho_m = \binom{k_m}{2} + \frac{\sigma k_m}{2}$ and

$$f_G(g|\sigma) = \left(\frac{\sigma}{2}\right)^{N_b} \prod_{m=1}^{2N_b+2n-2} e^{-\rho_m(t_m - t_{m-1})}. \tag{2.1}$$

This arises as the density of a process realizing graphs by starting with n particles at time $t = 0$ and generating coalescing and branching events as the particles randomly merge and split. The process stops at the first time t_{UA} when the number of particles equals one. If each particle branches independently at instantaneous rate σ and each of the $\binom{k_m}{2}$ pairs of particles alive in the interval (t_{m-1}, t_m) coalesce at instantaneous rate equal one, we arrive at the density f_G above.

In order to compute the expectation of a state function $\pi : \Gamma \to \Re$, for example $\pi(g) = t_{UA}(g)$, the time to ultimate coalescence, we compute $\mathsf{E}_{\mathcal{G}}\{\pi(g)\} = \int_\Gamma \pi(g) f(g) dg$, that is,

$$\int_\Gamma \pi(g) f(g) dg = \sum_{N_b=0}^{\infty} \sum_{\{V,E\} \in \Gamma^\star_{N_b,N_c}} \int_{\chi_{V,E}} \pi((V,E,t_A)) f((V,E,t_A)) dt_1 dt_2 ... dt_{N_b+N_c}.$$

2.3.2 Type-labeled graphs

Individuals in the population model of [18] are assumed to be of two types. Natural selection favors type 1 individuals over type 0 individuals. More complex non-selected or neutral types, indexed by entire DNA sequences may be included using now standard Monte Carlo methods [23, 27].

Type is propagated along an edge by a mutation process which acts from

the past towards the present, against the edge direction. Denote by $v^{(i)}$ and $v_{(i)}$ the vertices at the top and bottom of edge i. An edge $\langle v_{(i)}, v^{(i)} \rangle$ is associated with a real interval $(t_{v_{(i)}}, t_{v^{(i)}})$ so it is natural to think of type as a function $B_i(t)$ mapping this interval to $\{0, 1\}$. However it is more convenient (and equivalent) to record the type $B_{v^{(i)}} = \lim_{t \uparrow t_{v^{(i)}}} B_i(t)$ emitted onto edge i from the vertex at its top and the type $B_{v_{(i)}} = \lim_{t \downarrow t_{v_{(i)}}} B_i(t)$ passed into the vertex at the bottom.

Type is passed through a coalescent vertex in a simple way. If i and j are edges incoming at vertex v (so $v = v^{(i)} = v^{(j)}$ and i and j are below v) and k is out-going (so $v = v_{(k)}$ and k is above v) then type is passed straight through, $B_{v^{(i)}} = B_{v_{(k)}}$ and $B_{v^{(j)}} = B_{v_{(k)}}$. The way type is passed through a branching vertex depends on the selection gating values S on the edges above that vertex. Suppose i and j are edges out-going at vertex v (so $v = v_{(i)} = v_{(j)}$ and i and j are above v) with selection gating values $S_i = 1$ and $S_j = 0$, so i is the continuing edge and j is the replacing edge. Suppose edge k is in-coming at v (so $v = v^{(k)}$ and k is below v). Edges correspond to ancestral lineages. Just one of the edges i or j is ancestral to edge k and provides the type $B_{v^{(k)}}$ at the top of edge k. The rule passing ancestry through a branching vertex therefore determines the rule passing type and is as follows. The replacing edge (edge j satisfying $S_j = 0$) is ancestral if it passes the favored type, $B_{v_{(j)}} = 1$, into vertex $v = v_{(j)}$. In this case $B_{v^{(k)}} = B_{v_{(j)}}$ and both equal one. Otherwise, if the replacing edge passes the unfavored type, $B_{v_{(j)}} = 0$, into $v_{(j)}$, the continuing

edge (edge i satisfying $S_i = 1$) is ancestral. In this case $B_{v^{(k)}} = B_{v_{(i)}}$ and $B_{v^{(k)}}$ could be zero or one depending on the type $B_{v_{(i)}}$.

The ancestry rule allows us to follow the ancestry of the leaf individuals up the graph through branching vertices, and determine the genealogy of the leaf individuals, which is a tree-like subgraph of g. In Figure 2.2 the edge labeled I is replacing, the edge labeled II is continuing. In the graph on the left, the type at the bottom of edge I is 0, so the dark line indicating the ancestral sub-tree follows continuing edge II. In the graph on the right, the type at the bottom of edge I is 1, so the dark line indicating the ancestral sub-tree follows the replacing edge I. The time t_{MRCA} of the root vertex of the ancestral subgraph is of interest, since it is the time of the most recent common ancestor of the individuals represented at the leaves. Notice that t_{MRCA} and t_{UA} need not coincide.

Figure 2.2: The t_{UA} and t_{MRCA} need not coincide. The edge labeled I is replacing and the edge labeled II is continuing

Denote by $B = (B_{v_{(i)}}, B_{v^{(i)}})_{i \in E_g}$ the ordered set of all edge type label pairs

and by $\mathcal{B}_g$ the set of all admissible assignments of type label pairs B to edges of g. Admissible assignments $\mathcal{B}_g$ satisfy the rules given in the previous paragraph constraining the passage of type through vertex, and are otherwise unconstrained. The probability space $(\Gamma, \mathcal{F}, \mathbf{P})$ of unlabeled graphs is extended to a space of states for type-labeled graphs, $\tilde{\Gamma}$, where

$$\tilde{\Gamma} = \bigcup_{g \in \Gamma} \bigcup_{B \in \mathcal{B}_g} (g, B).$$

A labeled ancestral selection graph $(g, B) \in \tilde{\Gamma}$ is determined by specifying a topology (V, E), a vector of ancestral vertex times t_A (leaf vertex times t_L are identically zero), a vector S of selection gating values for edges attached above branching vertices, and a set B of pairs of edge type values.

Let B_{UA} denote the type of the root vertex $UA \in V_g$. This individual is drawn at random from the population, and its type is determined by the proportions of type 0 and 1 individuals in the population. The proportion $z \in [0, 1]$ of type 0 individuals is itself a random variable with density function $\eta(z|\theta, \sigma)$ controlled by the parameters of the mutation and branching processes,

$$\eta(z|\theta, \sigma) = K z^{\theta-1}(1-z)^{\theta-1} e^{-\sigma z}, \tag{2.2}$$

If p_0 is the probability that $B_{UA} = 0$ then

$$\begin{aligned} p_0 &= \int Pr(B_{UA} = 0|z)\eta(z|\theta,\sigma)dz \\ &= \int_0^1 Kz^\theta(1-z)^{\theta-1}e^{-\sigma z}dz \end{aligned}$$

and $p_1 = 1 - p_0$.

The evolution of selection type along an edge is modelled via a symmetric continuous time Markov process (the raw mutation process is ignorant of selective advantage). Individuals swap type at instantaneous rate $\frac{\theta}{2}$. Summing over all mutation histories along an edge, we obtain the probability for a transition between types along an edge as a 2×2 transition matrix between types, depending on the length τ_i of the edge

$$P_{B_{v_{(i)}},B_{v^{(i)}}} = \begin{cases} \frac{1}{2} + \frac{e^{-\theta\tau_i}}{2}, & \text{if } B_{v_{(i)}} = B_{v^{(i)}} \\ \frac{1}{2} - \frac{e^{-\theta\tau_i}}{2}, & \text{if } B_{v_{(i)}} \neq B_{v^{(i)}} \end{cases} \tag{2.3}$$

Taken together the graph and mutation processes determine the conditional probability $P(B|\theta, g)$ to realize an admissible set of type labels $B \in \mathcal{B}_g$ on the edges of ancestral selection graph g,

$$P(B|\theta, g) = p_{B_{UA}} \prod_{i \in E_g} P_{B_{v_{(i)}},B_{v^{(i)}}}. \tag{2.4}$$

The joint probability distribution $P(g, B)$ of the ancestral selection graph and its type labels is the probability distribution on $\tilde{\Gamma}$ given by the product

$$P(g, B) = P(B|\theta, g) f_G(g|\sigma).$$

Denote by B_I the set of types passed into leaf vertices, $B_I = (B_{v_{(i)}} : i \in E_g, v_{(i)} \in V_L)$ ordered by leaf label. These types are observable properties of the sample individuals and are our data. Other data such as DNA sequences for the leaf individuals will be usually be available. If this information is unrelated to selection type it may be incorporated in the MCMC analysis using now-standard methods [27, 4] which we do not discuss here. Suppose the B_I are observed and take values $B_I = D$. Denote by B_Y the remaining ancestral type values so that $B = (B_Y, B_I)$.

2.4 The full posterior density

Bayesian analysis requires the definition of a posterior probability distribution, which contains all the information available of the known as well as the unknown variables. Although the aim of this project is to determine how informative the data is of the selection parameter σ, knowledge of other parameters, known as nuisance parameters, such as the ancestral selection graph g, the ancestral allele types B_Y, and the mutation parameter θ is necessary for the

understanding and realism of the model.

We now define the probability distribution which we actually wish to sample using Markov chain Monte Carlo.

The joint probability distribution $P(g, B, \theta, \sigma)$ is given by the product

$$P(g, B, \theta, \sigma) = P(B|\theta, g) f_G(g|\sigma) P(\theta) P(\sigma)$$

The full posterior probability density for the missing information g, B_Y, θ and σ is the density we obtain by conditioning on these leaf types,

$$h(g, B_Y, \theta, \sigma | B_I) \propto P(g, (B_Y, B_I), \theta, \sigma) \mathbb{I}_{B_I = D}$$

with $\mathbb{I}_{B_I=D}$ the indicator function for $B_I = D$. From a Monte Carlo point of view we might sample $(g, B_Y, \theta, \sigma) \sim h(g, B_Y, \theta, \sigma | B_I)$ by sampling $(g, (B_Y, B_I), \theta, \sigma) \sim P(g, B)$ and rejecting samples with $B_I \neq D$ however this would be hopelessly inefficient. Instead we construct a Markov chain with equilibrium $h(g, B_Y, \theta, \sigma | B_I)$ on the subspace of $\tilde{\Gamma}$ in which $B_I = D$. This can be thought of as a random walk on the space of labeled ancestral selection graphs with $B_I = D$ which does not update the leaf labels.

Chapter 3

Markov Chain Monte Carlo

The unknown variables of interest are the graph g, the ancestral types at the internal vertices B_Y, the mutation parameter μ and the selection parameter σ. The mutation parameter μ as well as the selection parameter σ are scalars and pose no problems. However, the graph g is a high dimensional geometric object of variable dimension. Therefore, sophisticated sampling techniques, such as importance sampling or MCMC are needed to be able to infer the parameters in question.

A MCMC approach was chosen because of the ease with which a random walk of graphs can be created in the probability space $(\tilde{\Gamma}, \mathcal{F}, \mathbf{P})$. The general form of the Metropolis-Hastings algorithm is discussed in the first section. This is followed by the an explanation of the different moves or operators used to move around in the space of graphs.

The creation of the Markov chain can be thought of as the movement among

spaces of different dimensions. This fact is accommodated for by the inclusion of a term called the Jacobian as introduced by Peter Green [12]. A full discussion on the Jacobian can be seen in Section 3.1.

Therefore the aim of this chapter is to define a Metropolis-Hastings Markov chain Monte Carlo (MCMC) algorithm, which simulates the posterior probability density h defined in Chapter 2 associated with the Krone-Neuhauser selection model [17]. The algorithm realizes a sequence of random variables $\Phi_n = (g, B_Y, \theta, \sigma) \in \tilde{\Gamma}$ converging in probability density to $h(g, B_Y, \theta, \sigma | B_I)$ on the probability space $\tilde{\Gamma}$. Attention is also given to some special cases and an example of a realization of the Markov chain can be seen at the end of this chapter.

3.1 Metropolis-Hastings algorithm

Suppose the chain is in some state $\Phi_n = (g, B_Y, \theta, \sigma) \in \tilde{\Gamma}$. A candidate state $(g', B'_Y, \theta', \sigma')$ is generated in the following way. Draw uniform random variates $u = (u_0, u_1, u_2, ..)$ according to some fixed simple density q and compute $(g', B'_Y, \theta', \sigma') = \psi(g, B_Y, \theta, \sigma, u)$, where ψ is a fixed mapping. We are to some extent (details below) free to choose q and ψ. We suppose there is $u' = (u'_1, u'_2, ..)$ such that $(g, B_Y, \theta, \sigma) = \psi(g', B'_Y, \theta, \sigma, u')$. The new state is ac-

cepted with probability $\alpha(g', B'_Y, \theta', \sigma'|g, B_Y, \theta, \sigma)$, with

$$\alpha(g', B'_Y, \theta', \sigma'|g, B_Y, \theta, \sigma) = \min\left[1, \frac{h(g', B'_Y, \theta', \sigma'|B_I)}{h(g, B_Y, \theta, \sigma|B_I)} \frac{q(u')}{q(u)} \left|\frac{\partial(g', u')}{\partial(g, u)}\right|\right] \quad (3.1)$$

The variables B_Y, θ and σ are absent from the Jacobian term, $\left|\frac{\partial(g',u')}{\partial(g,u)}\right|$. If the new state is accepted, $\Phi_{n+1} = (g', B'_Y, \theta', \sigma')$, and otherwise $\Phi_{n+1} = (g, B_Y, \theta, \sigma)$.

The above expression for the acceptance probability is needed in the variable dimension setting (but is convenient in any case). It is due to Green [12]. Some details of the algorithm may be clarified in the example below and in the paper by Cloete, Nicholls and Scott [2].

Roughly speaking, the density q and the mapping ψ must be chosen so that the Markov chain is irreducible on $\tilde{\Gamma}$. Since $\tilde{\Gamma}$ is continuous we need the chain to be π-irreducible (in the sense of [25]). A proof that this property holds for the chain below is straightforward and can be found in Appendix B. The above form for the acceptance probability ensures the Markov chain is reversible with respect to $h(g, B_Y, \theta, \sigma|B_I)$. It follows that the sequence $\{\Phi_n\}$, $n = 0, 1, 2 \ldots$ is ergodic with unique equilibrium density $h(g, B_Y, \theta, \sigma|B_I)$. The Metropolis-Hastings birth-death process we describe below is similar to that presented in [9], one of the first instances of this type of Markov chain Monte Carlo.

3.1.1 Moves

The application of a collection of operators which act on the labeled ancestral selection graph (g, B_Y) creates a random walk in the space $\tilde{\Gamma}$.

The addition and deletion of an edge (and associated vertices) are two key operations (also called 'moves' or 'updates') needed to get irreducibility in $\tilde{\Gamma}$. Because the edge addition move adds two new vertices, one of which is branching, it adds associated type labels $(B_{v_{(i)}}, B_{v^{(i)}})$ and selection gating values S_i, S_j. The addition-deletion pair give irreducibility on their own, since irreducibility on $\mathcal{B}_g$ and $\mathcal{S}_g$ is obtained by sequences of addition and deletion operations. Extra moves, which do not alter the graph topology, were implemented to make the MCMC more efficient, in the sense of exploring $\tilde{\Gamma}$ more rapidly, and thereby reducing the variance of any estimate formed from the output. These moves include changing the selection types $(B_{v_{(i)}}, B_{v^{(i)}})$ and selection gating values S_i, S_j on edges, and changing the time t_v of a vertex in a range between its youngest parent and eldest child.

At each update one of these operators is chosen (at random, with a probability we are free to choose) and applied to the state. With probability p^* we attempt to add an edge. With probability $p^\dagger = p^*$ we attempt to delete an edge from a graph g. With probability $1 - p^* - p^\dagger$ one of the other operators is chosen. A random variate $u_0 \sim U(0, 1)$ is used to simulate this decision. The role of variates $u_1, u_2 \ldots$ depends on the outcome of choice of move type.

Add an edge

Suppose state $(g, B_Y) = (V, E, t_A, S, B_Y)$ has $M = N_b + N_c$ ancestral vertices. It is convenient from this point to drop the requirement that vertex labels be ordered by time and consecutive from 1 to $n + N_b + N_c$. Refer to Figure 3.1. Choose two edges $e, e' \in E$ uniformly at random with replacement. The

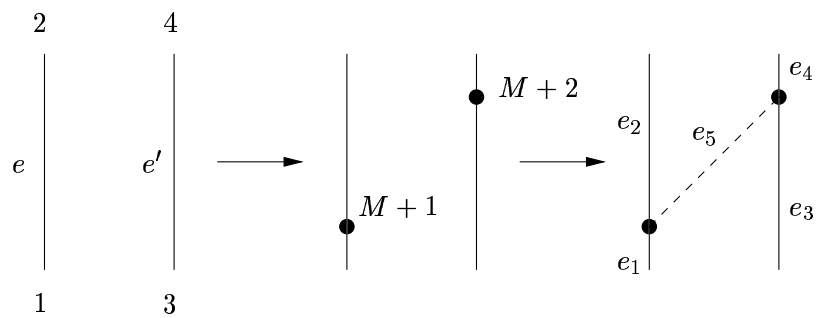


Figure 3.1: Add an edge

edge above the root can be chosen (so for the purpose of this algorithm it is convenient to regard E as containing such an edge) and it is possible that $e = e'$ (these must be treated as special cases). Suppose for exposition that $v_{(e)} = 1, v^{(e)} = 2, v_{(e')} = 3$ and $v^{(e')} = 4$. Points corresponding to new vertices with labels $M + 1$ and $M + 2$ are chosen on $\langle 1, 2\rangle$ and $\langle 3, 4\rangle$ using uniform random variates u_1 and u_2, respectively. Let $t'_{M+1} = t_1 + (t_2 - t_1)u_1$ and $t'_{M+2} = t_3 + (t_4 - t_3)u_2$ and suppose $t'_{M+1} < t'_{M+2}$ so that vertex $M + 1$ is branching. Connect vertices $M + 1$ and $M + 2$ with an edge e_5. Selection gating values S_{e_2} and $S_{e_5} = 1 - S_{e_2}$ are assigned to the edges out-going at $M + 1$ by a coin toss. Let $\tau_1 = t_2 - t_1$ and $\tau_2 = t_4 - t_3$ be the lengths of edges e

and e' respectively. The new ancestral selection graph is $g' = (V', E', t'_A, S') = (V \cup \{M+1, M+2\}, E_g \cup \{e_5\} \cup \{e_2\} \cup \{e_4\}, (t_A, t'_{M+1}, t'_{M+2}), (S, S'_{e_2}, S'_{e_5}))$.

Notice that $t_A \in [0, \infty)^{N_b+N_c}$ while $t'_A \in [0, \infty)^{N_b+N_c+2}$ so the state dimension changes. The Jacobian in Equation 3.2 for the transformation is the determinant of an $M+2 \times M+2$ matrix,

$$\frac{\partial(g, u')}{\partial(g, u)} = \begin{array}{c} \\ \mathbf{t'_1} \\ \mathbf{t'_2} \\ \mathbf{t'_3} \\ \mathbf{t'_4} \\ \vdots \\ \mathbf{t'_{M+1}} \\ \mathbf{t'_{M+2}} \end{array} \begin{array}{c} \begin{array}{ccccccccc} \mathbf{t_1} & \mathbf{t_2} & \mathbf{t_3} & \mathbf{t_4} & \mathbf{t_5} & \dots & \mathbf{t_M} & \mathbf{u_1} & \mathbf{u_2} \end{array} \\ \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 \\ 1-u_1 & u_1 & 0 & 0 & 0 & \dots & 0 & \tau_1 & 0 \\ 0 & 0 & 1-u_2 & u_2 & 0 & \dots & 0 & 0 & \tau_2 \end{vmatrix} \end{array}$$

that is $\partial(g, u')/\partial(g, u) = \tau_1 \tau_2$.

In order to complete the candidate state three new selection types $B'_{v_{(e_2)}}, B'_{v_{(e_5)}}$ and $B'_{v_{(e_4)}}$ must be generated. The two additional new selection types $B'_{v^{(e_5)}}$ and $B'_{v^{(e_3)}}$ around vertex $M+2$ are locked to $B'_{v_{(e_4)}}$ by the type-passing rule at a coalescent vertex, and the type of $B'_{v^{(e_1)}}$ is determined by type-passing at branching vertices and the values of $B'_{v_{(e_2)}}, B'_{v_{(e_5)}}, S'_{e_2}$ and S'_{e_5}. We sample $B'_{v_{(e_2)}}, B'_{v_{(e_5)}}$ and $B'_{v_{(e_4)}}$ conditional on the types $B_{v^{(e_2)}}, B_{v_{(e_1)}}, B_{v^{(e_4)}}$ and $B_{v_{(e_3)}}$

which surround them. The conditional distribution, q say, is

$$q(B'_Y|B_Y) = q(B'_{v_{(e_2)}}, B'_{v_{(e_5)}}, B'_{v_{(e_4)}}|B_{v^{(e_2)}}, B_{v_{(e_1)}}, B_{v^{(e_4)}}, B_{v_{(e_3)}})$$

with

$$q(B'_Y|B_Y) = Z(B)P_{B_{v_{(1)}},B'_{v^{(1)}}}P_{B'_{v_{(2)}},B_{v^{(2)}}}P_{B'_{v_{(5)}},B'_{v^{(5)}}}P_{B_{v_{(3)}},B'_{v^{(3)}}}P_{B'_{v_{(4)}},B_{v^{(4)}}}.$$

over all admissible type-assignments. The normalizing constant $Z(B)$ is the sum over 8 possibilities for three binary variables of the product of transition probabilities. The selection types B'_Y in the candidate state (g', B'_Y) are those of B_Y plus the six new selection types around $M+1$ and $M+2$.

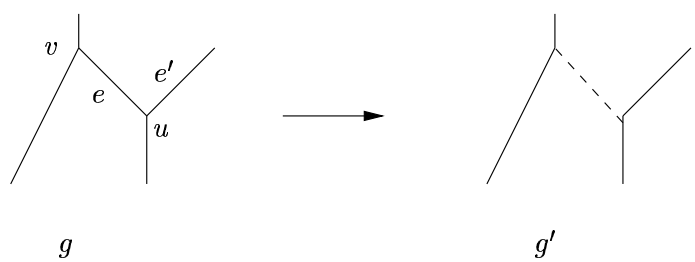
The probability to propose this update is $q(u) = p^* q(B'|B)/|E|^2$ where $|E|$ is the number of edges in g and a factor of two, coming from the fact that e and e' can be chosen in either order, has canceled a factor of a half, for the choice of S'_{e_2} and S'_{e_5}. Since this is the general case of adding an edge which does not involve the root of the graph or the parameters θ and σ, $p_0 = p'_0$, $B_{UA} = B'_{UA}$, $\theta = \theta'$ and $\sigma = \sigma'$. Therefore the acceptance probability for the new state is

$$\alpha(\Phi', \Phi) = \min\left[1, \frac{P(B'_Y|g,\theta)}{P(B_Y|g,\theta)}\frac{f_G(g'|\sigma)}{f_G(g|\sigma)}\frac{p^\dagger}{p^* q(B'_Y|B_Y)}\frac{|E|^2}{|E'_D|} \times \tau_1\tau_2\right]. \tag{3.2}$$

The extra factor $q(u') = p^\dagger/|E'_D|$ comes from the delete edge operation discussed below. The expression $P(B'_Y|g,\theta)/P(B_Y|g,\theta)q(B'_Y|B_Y)$ simplifies a great

deal, as both the numerator and denominator are long products of transition probabilities, and all edges sufficiently removed from the update-neighborhood contribute the same factor to numerator and denominator.

Delete an edge



Let E_D be the set of "deletable" edges in g. Directed edge $e = \langle u, v \rangle$ is in E_D iff $v \in V_c$ and $u \in V_b$. An edge e is drawn from E_D. The new state g' is generated by deleting edge e from g. A new ancestral selection graph $g' = (V', E', t'_A, S') = (V_g \backslash \{u, v\}, E_g \backslash \{e\}, (t_A, \backslash \{t_u, t_v\}), (S \backslash \{S_e, S_{e'}\}))$ is generated with probability $q(u) = p^\dagger \frac{1}{|E_D|}$ and acceptance probability

$$\alpha(\Phi', \Phi) = \min \left[1, \frac{P(B'_Y|g, \theta)}{P(B_Y|g, \theta)} \frac{f_G(g'|\sigma)}{f_G(g|\sigma)} \frac{p^* q(B_Y|B'_Y)}{p^\dagger} \frac{|E_D|}{E'^2} \times \frac{1}{\tau_1 \tau_2} \right]. \tag{3.3}$$

Special cases

Since two edges are chosen at random with replacement, a new edge can be added between two new vertices on the same edge. This creates a bubble as can be seen in Figure 3.2.

The probability to propose a bubble is $q(u) = \frac{p^* q(B'|B)}{|E|^2} \frac{1}{2}$. The factor one

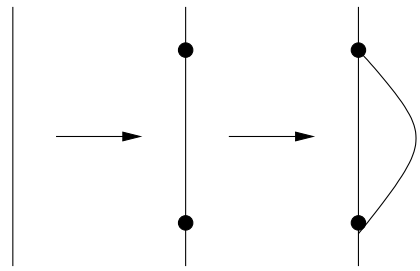


Figure 3.2: Bubble

half comes from the choice of S'_{e_2} and S'_{e_5}. This factor is not canceled as was discussed in Section 3.1.1 because there is only one way to choose the same edge twice. In this case the acceptance probability is

$$\alpha(\Phi', \Phi) = \min\left[1, \frac{P(B'_Y|g,\theta)}{P(B_Y|g,\theta)} \frac{f_G(g'|\sigma)}{f_G(g|\sigma)} \frac{p^\dagger}{p^* q(B'_Y|B_Y)} \frac{2|E|^2}{|E'_D|} \times \tau_1\tau_2\right]. \tag{3.4}$$

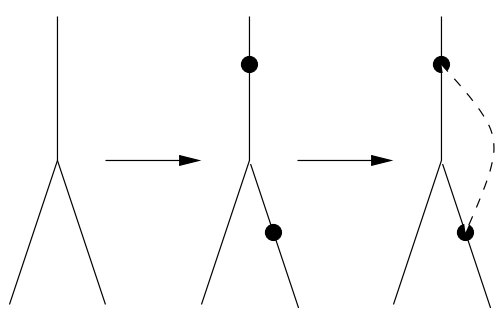


Figure 3.3: Add edge with one vertex above the root

The other special case is when a new root is proposed. The new edge is added between the edge above the root of the graph and another edge below the root as can be seen in Figure 3.3. Since one of the new vertices $M+2$ is at a

time above the root, the time, t'_{M+2} is simulated from the exponential density with rate $1+\sigma+\theta$. One of the factors $\tau_i, i = 1, 2$ in Equation 3.2 is replaced by a factor $(\rho)e^{-\tau\rho}$ where τ is the difference in time between the time of the old root and the new root t'_{M+2}. The new type B'_{UA} of the new root of the graph is determined by drawing the probability p'_0 from the density $\eta(z|\theta,\sigma)$, (Chapter 2, Equation 2.2) and with probability p_0 the selection type of the new ultimate ancestor (new root) is 0.

The conditional distribution $q(B'_{UA} = 0|B_{UA}) = \eta(p'_0|\theta,\sigma)p'_0$ and $q(B'_{UA} = 1|B_{UA}) = \eta(p'_0|\theta,\sigma)(1-p'_0)$ while $q(B_{UA} = 0|B'_{UA}) = \eta(p_0|\theta,\sigma)p_0$ and $q(B_{UA} = 1|B'_{UA}) = \eta(p_0|\theta,\sigma)(1-p_0)$.

The acceptance probability for a new root is

$$\begin{aligned}\alpha(\Phi',\Phi) &= \min\left[1, \frac{\eta(p'_0|\theta,\sigma)}{\eta(p_0|\theta,\sigma)}\frac{P(B'_Y|g,\theta)}{P(B_Y|g,\theta)}\frac{f_G(g'|\sigma)}{f_G(g|\sigma)}\frac{p^\dagger}{p^* q(B'_Y|B_Y)}\frac{|E|^2}{|E'_D|}\right] \times \\ &\quad \left[((\rho)e^{-\tau\rho})^{\tau_2}\frac{q(B_{UA}|B'_{UA})}{q(B'_{UA}|B_{UA})}\right]. \end{aligned} \tag{3.5}$$

A legal state is a state where the number of lineages below each ancestral vertex is at least two. When deleting an edge, the proposed graph has to be checked to determine if the graph is admissible. As can be seen in Figure 3.4, when the edge e is deleted the number of lineages between vertex u and vertex v is one. The proposal of this state will be rejected.

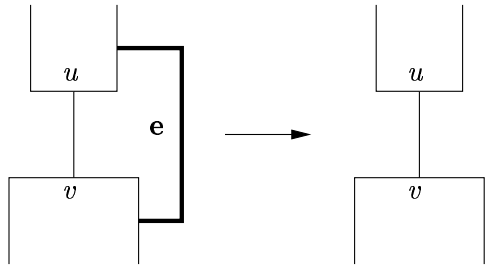


Figure 3.4: The deletion of an edge results in an illegal state

Changing the time t_v

Each vertex $v \in V_A$ except the root vertex is between two events. Without loss of generality let the vertex at the previous event of vertex v have label w and the vertex of the next event have label u. The previous event is at a later time than the current event. As can be seen in Figure 3.5 the vertices u, v and w are not necessarily connected by an edge.

The leaf vertices represent the data and a new time can not be assigned to these vertices. As before the root vertex is a special case and is discussed later.

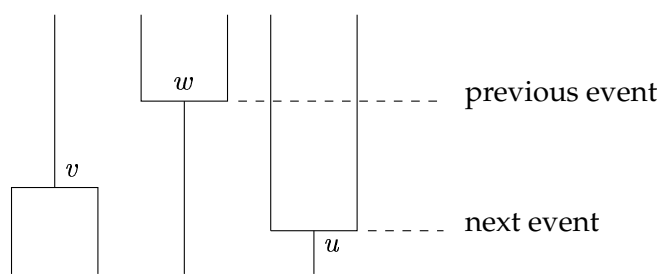


Figure 3.5: The next and previous events of a vertex v need not be connected by an edge

Choose a vertex $v \in V_A$ at random. Suppose the vertex is not the root

vertex. Let t_w and t_u be the time at the previous and the next event respectively. The new position of the vertex v is chosen by using a random variate u_1. Let $t'_v = (t_w - t_u)u_1$ where $t_w - t_u$ is the time interval between the previous and the next event of the vertex v.

The selection type of the vertex v stays unchanged during this move. An update of only the selection types at vertices is done during a separate move and is discussed in more detail in the next subsection.

Since the proposal involves only the time of one vertex and not the selection type or any other parameter, all the terms in the acceptance ratio cancel to give the acceptance probability for the new state as $\alpha(\Phi'|\Phi) = 1$.

Suppose the vertex, v is the root vertex. As before let t_v be the time of the root vertex v and t_u be the time of the vertex of the next event. Let $\tau = t_v - t_u$. The new time t'_v is determined randomly on the interval $(\frac{1}{2}\tau, 2\tau)$. Let $\tau' = t'_v - t_u$. The acceptance ratio for the proposed move involving the root, is $\frac{q(t_v|t'_v)}{q(t'_v|t_v)} = \frac{\tau}{\tau'}$.

During the movement of the root to a new position, the selection type of the root is not changed. This means that the probability p_0 and the type B_{UA} stay unchanged.

The acceptance probability of the special case involving the root is therefore

$$\alpha(\Phi', \Phi) = \min\left[1, \frac{q(t_v|t'_v)}{q(t'_v|t_v)}\right] = \min\left[1, \frac{\tau}{\tau'}\right]. \tag{3.6}$$

Changing the selection type B_v

This move involves the random choice of a vertex $v \in V_A$ and the proposal to assign a selection type $B_v \in \{0, 1\}$ to the vertex v. The vertex v can not be a leaf vertex, since the leaf vertices represent the data sampled from a population of organisms. Since the root of the graph is a special case, assume v is not the root.

Suppose v is the result of two edges merging. Without loss of generality let e_1 and e_2 denote the in-going edges below the vertex v and e_3 the out-going edge above v. See Figure 3.6.

The selection types at the vertices $1, 2$ and 3 are kept fixed. The probabilities of the 2 possible selection types of the vertex v are determined by the transition matrices associated with the edges e_1, e_2 and e_3 conditioned on the selection types at vertices $1, 2$ and 3.

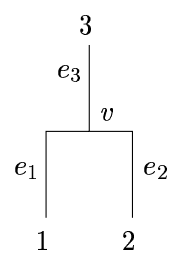


Figure 3.6: Coalescing event at vertex

The conditional distribution q is

$$q(B_v'|B_v) = P_{B_1,B_v'} P_{B_2,B_v'} P_{B_v',B_3}$$

Since the root of the graph is not involved the acceptance probability of the new state is

$$\alpha(\Phi', \Phi) = \min\left[1, \frac{P(B_v'|g,\theta)}{P(B_v|g,\theta)} \frac{f_G(g|\sigma)}{f_G(g|\sigma)} \frac{q(B_v|B_v')}{q(B_v'|B_v)}\right] = 1. \tag{3.7}$$

If v is a branching vertex, let e_1 be the edge below the vertex v and e_2 and e_3 be the edges above the branching vertex. Let the label of the vertex at the bottom of edge e_1 be 1 and the labels of the vertices at the top of the edges e_2 and e_3 be 2 and 3 respectively. An example of a branching event at vertex v can be seen in Figure 3.7.

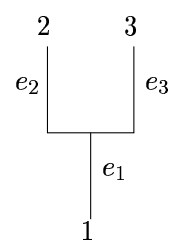


Figure 3.7: Branching event

If v is a branching vertex, the type of particle at the vertex is determined by the type of the particles on the edges e_2 and e_3 just above the vertex v. Let the vector $x = [x_1, x_2]$ represent the type of particles just above the vertex, where the entry in the first position is the particle type on the continuing edge and the entry in the second position is the type on the replacing edge. The type of the vertex v is determined by the rule as explained in Section 2.3.2. The vertex v will always have a type 1 particle if the combinations of the particle types

above the vertex v are $x = [1, 0], x = [0, 1]$ or $x = [1, 1]$. During the mutation process the types of the particles on edges e_2 and e_3 depend on the particle types at the vertices 2 and 3. The transition matrices over the edges e_2 and e_3 determine the types of the two particles. As in the case with a coalescing event, during the Nodetype move the types at the vertices $1, 2$ and 3 are kept fixed. Therefore the proposal involves the transition matrices over edges e_1, e_2 and e_3 given the types at vertices $1, 2$ and 3. For instance if the combination is $x = [0, 1]$ and the bottom vertex of the edge below the vertex v is of type 1, the vertex v is of type 1 and mutation has not occurred over the bottom edge. Therefore this must be taken in consideration when multiplying the entries of the relevant transition matrices.

Let τ_{e_3}, τ_{e_2} and τ_{e_1} denote the lengths of the edges $e_3 = \langle 3, v\rangle, e_2 = \langle v, 2\rangle$ and $e_1 = \langle v, 1\rangle$ respectively and let

$$P_{\tau_{e_i}} = \begin{pmatrix} \frac{1}{2} + \frac{1}{2}e^{-\theta\tau_{e_i}} & \frac{1}{2} - \frac{1}{2}e^{-\theta\tau_{e_i}} \\ \frac{1}{2} - \frac{1}{2}e^{-\theta\tau_{e_i}} & \frac{1}{2} + \frac{1}{2}e^{-\theta\tau_{e_i}} \end{pmatrix}$$

be the transition matrices over the edges e_1, e_2 and e_3.

The probability that a particle does not change over an edge $e_i, i = 1, 2, 3$ is the entry in the first column and first row of the transition matrix $P_{\tau_{e_i}}, i = 1, 2, 3$, while the probability of a change is in the second column and the first row. The probability of a particle of type j is calculated by multiplying the en-

tries of the three matrices and dividing each of the products by the normalizing constant $Z(B)$.

For example at a coalescing vertex v, suppose the types of the particles at all three vertices $1, 2$ and 3 are 0. The probability that the type of particle at vertex v is 0, is calculated by multiplying the first row first column of all three transition matrices $P_{e_1}(1,1)P_{e_2}(1,1)P_{e_3}(1,1)$ and dividing by the normalizing constant $Z(B)$. The probability for a particle type 1, requires that the entries of the first row, second column of the transition matrices $P_{e_i}, i = 1, 2, 3$ over the three edges are multiplied and divided by the normalizing constant $Z(B)$.

It is clear that in the case of a branching vertex v, the probabilities of four combinations of the vector $x = [x_1, x_2]$ are calculated.

Since the numbers of vertices and edges stay unchanged during the Node-type move, the probability of choosing a vertex in the graph before the proposal is the same as the probability of choosing a vertex after the proposal. These probabilities cancel in the acceptance ratio.

By using a random variate u the type of the vertex is proposed and accepted with probability

$$\alpha(\Phi', \Phi) = \min\left[1, \frac{P(B'_Y|g,\theta)}{P(B_Y|g,\theta)} \frac{f_G(g'|\sigma)}{f_G(g|\sigma)} \frac{q(B_Y|B'Y)}{q(B'_Y|B_Y)}\right]. \tag{3.8}$$

Changing gating values S_i

At a branching event, v, the gating values S_i and S_j of edges i and j outgoing from vertex v, determines the selection type B_v at the vertex v as discussed in Chapter 2. A value $S_i = 0$ indicates that the edge i is a replacing edge while a value $S_i = 1$ means that edge i is a continuing edge.

Therefore, this move involves the random choice of a branching event in a graph g. The proposal is to assign gating values to the resulting edges, i and j of the branching event. With probability $\frac{1}{2}$, a gating value $S_i \in \{0, 1\}$ is assigned to edge i. The edge j has a gating value $S_j = 1 - S_i$.

The proposal of the new state is independent of the assignment of gating values in the current state. Therefore is can happen that the proposed state is the same as the current state.

The acceptance probability is

$$\alpha(\Phi', \Phi) = 1$$

3.2 Example

As an example in Figure 3.8 is a realization of a Markov chain created by the add/delete move.

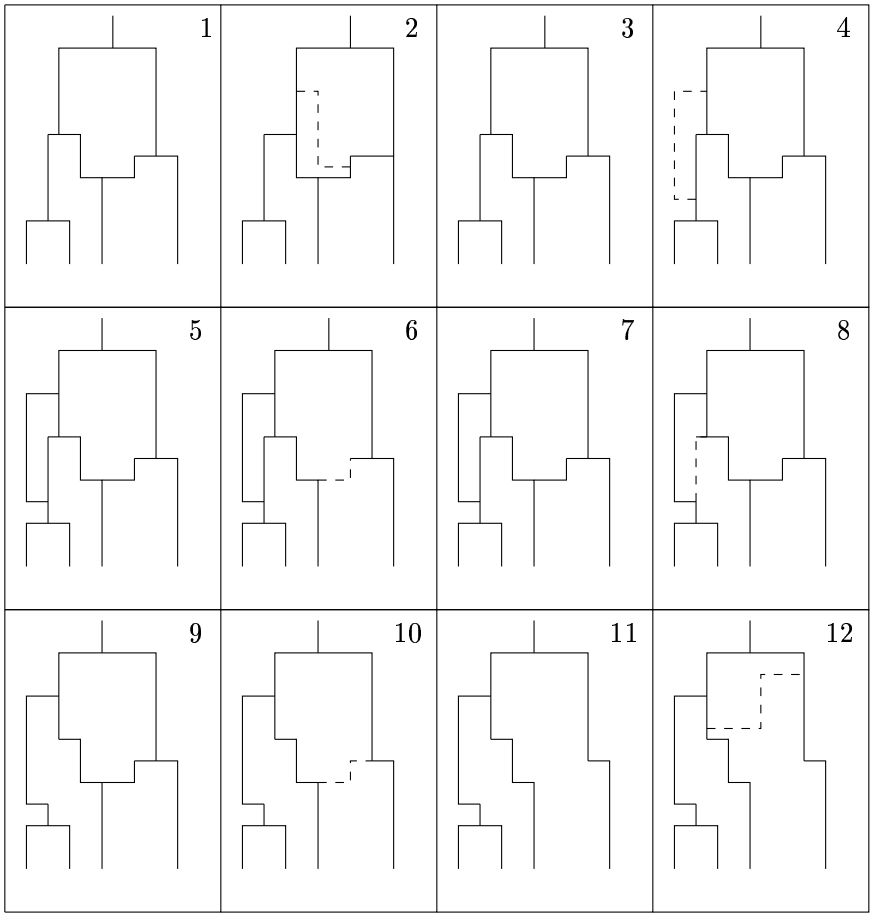


Figure 3.8: A realization of an actual Markov chain, which was created by adding and deleting edges. The dashed lines indicate the proposed new states.

1. Initial graph with 1 branching and 4 coalescent events.
2. Propose to add an edge.
3. Proposal rejected.
4. Propose to add an edge.
5. Proposal accepted. The new graph has 2 branching and 5 coalescent events.
6. Propose to delete an edge.

7. Proposal rejected.
8. Propose to delete an edge.
9. The proposal is accepted. The new graph has 1 branching and 4 coalescing events.
10. Propose to delete an edge.
11. Proposal accepted. The new graph has 0 branching events.
12. Propose to add an edge....

Chapter 4

Implementation

The ancestral selection graph is a complex geometric structure, evolving over time. Therefore, the code is written in Matlab to give attention to the creation of the Markov chain and not to write functions already available. The other advantage of Matlab is the ease of creating graphical output of statistical results such as histograms, line plots and error bars.

In this chapter the creation of data structures, the use of different arrays and the general implementation of the code is discussed. The first section is a discussion of the language Matlab for the sole purpose of getting an understanding of the data types available in Matlab.

4.1 Data type

The data type, structure, in Matlab describes a collection of information within a single variable. The structure consists of different fields which contain the information. The information can be retrieved and manipulated from the structure by addressing the relevant fields.

Formally an ancestral graph, g, consists of a set of points or vertices, V, together with a set of pairs of distinct vertices, E.

To be able to retrieve information in a systematic way, a graph g is represented in a natural way by two arrays of structures `V` and `E`. The structures, `V` and `E` with corresponding fields can be seen in Appendix A. The representation is maintained throughout and supports queries in an effective way. These two data structures were chosen in such a way that the graph is described in a consistent manner. All internal vertices, V_A, have degree three, while the external vertices are all of degree one. The root vertex is also considered an internal vertex, with an additional vertex above the root. Let V_{AR} denote the vertex above the root.

The two operations of adding and deleting edges involve different queries. For instance only certain edges can be legally deleted from a graph without collapsing the structure of the graph. These edges are the result of a branching event but the event at the top vertices of these edges can not be a branching event. A typical query is to check if the edge in itself is branching at the top

vertex. If it is not branching, a second query is done by checking if the edge is a result of a branching event.

```
Step 1:  if edge branches
Step 2:   RETURN
Step 3:  else
Step 4:   if vertex at bottom of edge is branching
Step 5:   delete edge
Step 6:  else
Step 7:   RETURN
Step 8:  end
```

4.2 Data structure

For each vertex $v \in V_L \cup V_A$, (Chapter 2), the structure $\mathsf{V}(v)$ contains fields describing the vertex v. The convention for a vertex v of degree three is to assign, the three edges e_1, e_2 and e_3 as in Figure 4.1. If the vertex v is a coalescing vertex, then the edge e_3 is out-going from the top of the vertex v and edges e_1 and e_2 are incoming from the bottom of vertex v. In the case where the vertex is a branching vertex, the edge e_1 is incoming from the bottom and the edges e_2 and e_3 are outgoing from the top of vertex v. The field $\mathsf{V}(v)\mathtt{.e} = (e_1, e_2, e_3)$ is a pointer to the corresponding edge structures $\mathsf{E}(e_1)$, $\mathsf{E}(e_2)$ and $\mathsf{E}(e_3)$. In the case of a leaf vertex $v \in V_L$ of degree one, the field $\mathsf{V}(v)\mathtt{.e}$ contains only one inscription e_1 pointing to the edge structure $\mathsf{E}(e_1)$. The vertex above the ultimate ancestor is denoted by V_{AR} and is above the root of the graph. This vertex is of

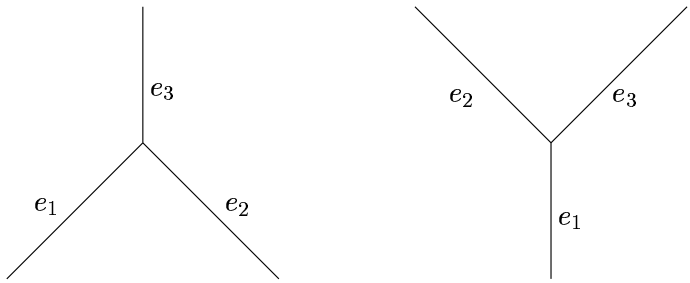


Figure 4.1: Convention of labeling the edges

degree 1 and the field $\mathsf{V}(v_{AR})\mathtt{.e}$ contains only one inscription e_1.

Since g is a directed graph, by going from the leaves to the root of the graph, let i denote the initial vertex and j the terminal vertex of an edge $e \in E$, i.e $e = \langle i, j \rangle$ where $t_j > t_i$. The field $\mathsf{E}(e)\mathtt{.v}= (i, j)$ of the edge structure $\mathsf{E}(e)$, acts as a pointer to the vertex structures $\mathsf{V}(i)$ and $\mathsf{V}(j)$, respectively.

By convention for a vertex v, the inscription e_1 of the field $\mathsf{V}(v)\mathtt{.e}= (e_1, e_2, e_3)$ indicates if the event at the vertex is a branching event or not. The corresponding field $\mathsf{E}(e_1)\mathtt{.b}$ has a value 0 or 1, where a value of 1 indicates that the edge e_1 is branching at the top at vertex v.

A value of 0 in the field $\mathsf{E}(e_1)\mathtt{.b}$, indicates a coalescent event and the edges e_1 and e_2 are merging at the vertex v.

If the event at vertex v is a branching event, one of the resulting edges is a continuing edge. The corresponding field of the edge structure $\mathsf{E}(e)\mathtt{.cnt}$ has a value 1. If the field $\mathsf{E}(e)\mathtt{.cnt}$ has value 0, the edge is the replacing edge. The distinction between the continuous or the replacing edge is needed in the

application field of evolutionary biology. The replacing edge represents the extra births of the advantageous allele. For a more detailed explanation see Chapter 2.

Since the leaves, V_L are representing the data, the edges connected to the leaves are all continuing edges. The edge connecting the ultimate ancestor to the vertex above is not included in the genealogy and is neither continuous nor replacing. When two continuing edges coalesce, the resulting edge is continuing. All other combinations of coalescing edges result in non-continuing edges. For instance when a continuing edge coalesces with a replacing edge, the resulting edge is a replacing edge.

The field `V`(v)`.time` contains the time, t, at which each event occurs. The vertices v of a graph can be ordered according to these time values, where the root vertex has the largest time value and the leaves have the minimum time value of 0. By starting at the root vertex in this list, all vertices can be reached by using the field `V`(v)`.next` (next event) of each vertex v.

Similarly, the same can be done by starting at a leaf vertex and using the field `V`(v)`.prev` (previous event). Each vertex is not necessarily connected to its next or previous vertex by an edge.

The convention for the field containing the next vertex of the leaf vertices is one down from the label of the leaf vertex to the right. For example if there are three leaf vertices with labels $\{1, 2, 3\}$, the field `V`(3)`.next`$= 2$, `V`(2)`.next`$= 1$ and `V`(1)`.next`$= 1$. The field containing the label of the pre-

vious event is one up from the label to the left. In the example the fields `V`(1)`.prev`=2 and `V`(2)`.prev`$= 3$.

The number of edges in the graph changes between events according to the type of event. Starting with n leaf edges, the number of edges increases by one at a branching event and decreases by one at a coalescent event. The graph ends at the final coalescent event at the root at which time the number of edges below the root is two. To keep track of these changes, the field `V`(v)`.na` holds the number of edges in the graph between the event at vertex v and the next event at vertex u.

Four types of vertices can be identified, the leaf vertices, V_L, the ancestral vertices, V_A, the root vertex V_{UA} and the vertex above the root, V_{AR}. The data structure `V`(v) contains a field,`V`(v)`.type` indicating the type of the vertex. The value is either 1 for a leaf vertex, 2 for an internal vertex, 4 for the root of the graph and 5 for the vertex above the root.

When adding an edge, two new vertices are added to the existing graph. If one of the new vertices is added above the root, a new root is proposed to the graph. This is a special case. See Chapter 3. It is therefore essential to identify the type of vertex involved in the addition of an edge.

When proposing a new state care must be taken not to propose an illegal state. An illegal state is when the number of edges between events is less than two. This can happen when the addition of a bubble above the root is proposed. To avoid this illegal proposal the field `V`(v)`.type` is queried to make

sure that the bubble is added between two internal vertices, i.e. vertices of type 2.

During the change of time move, a new time can not be assigned to the leaf vertices. The field `V`(v)`.type` is queried to ensure that only ancestral vertices can be moved.

The other purpose of the `V`(v)`.type` field is for debugging purposes. The leaf vertices are connected to only one edge. Therefore the field `V`(v)`.e` contains only one inscription. The root vertex represent the final coalescent event and the inscription of the field `E`(e_1).b must have the value 0.

The field `V`(v)`.a` contains the value of the type of allele associated with the vertex v. If there are only two types of alleles in the population, the value of the field `V`(v)`.a` is either a 0 or a 1.

Corresponding to the field `V`(v)`.a` of the vertex v is the field `E`(e)`.allele`. The field `E`(e)`.allele` is a vector and contains the type of allele at the bottom and the top of the edge e. The convention is that the first inscription of the vector is the bottom allele and the second inscription is the top allele. The allele type at the bottom of an edge can be different from the allele type at the top.

Associated with each edge of the graph is a field `E.used`. This field has a value -1 which will change to either 0 or 1 during the determination of the ancestry of the sample. The field `E.used` is discussed in detail in Section 4.6.4 later in this Chapter.

4.2.1 Example

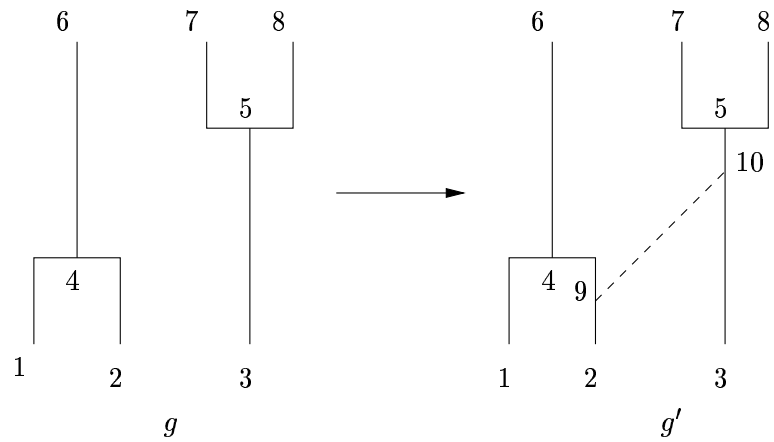


Figure 4.2: Vertex labels before and after adding an edge

As an example suppose a graph g has one coalescent and one branching event as can be seen in Figure 4.2. At vertex 4 two edges coalesce and at the vertex 5 one edge branches.

The vertex structure array for the graph g is $\mathsf{V}= [V(1), V(2), V(3), V(4), V(5), V(6), V(7), V(8)]$. For example the structure $\mathsf{V}(4)$ of the vertex with label 4 before adding an edge has the following values:

$\mathsf{V}(4)$`.na` $= 3$, $\mathsf{V}(5)$`.prev` $= 5$, $\mathsf{V}(4)$`.next` $= 3$.

After adding an edge, let g' denote the new graph as seen in Figure 4.2. The vertex structure array V' of the graph g is updated by adding two new vertex structures $\mathsf{V}'(9)$ and $\mathsf{V}'(10)$ to the array, while the vertex structure $\mathsf{V}(4)$ and $\mathsf{V}(5)$ is updated to take the new changes into account.

The new vertex structure array is $\mathsf{V}' = [V'(1), V'(2), V'(3), V'(4),$

$V'(5), V'(6), V'(7), V'(8), V'(9), V'(10)]$ and $nv' = nv + 2$. Since a new vertex with label 9 is added below vertex 4 the next event below vertex 4 is a branching event. The number of edges between the events at vertex 4 and vertex 9 is now one more than before the addition of an edge.

The next event below vertex 5 is a coalescing event. Therefore the number of edges in the graph below vertex 5 and above vertex 10 is still the same as before the addition.

The existing structures of vertex 4 and vertex 5 are updated as follows: $\mathsf{V}'(4)$`.na` $= 4$, $\mathsf{V}'(4)$`.next` $= 9$, $\mathsf{V}'(4)$`.prev` $= 10$, and $\mathsf{V}'(5)$`.na` $= 2$, $\mathsf{V}'(5)$`.next` $= 10$, $\mathsf{V}'(5)$`.prev` $= 6$. The new vertex structure $\mathsf{V}'(9)$ has the following fields $\mathsf{V}'(9)$`.na` $= 3$, $\mathsf{V}'(9)$`.next` $= 3$ and $\mathsf{V}'(9)$`.prev` $= 4$, while the vertex structure $\mathsf{V}'(10)$ has as fields $\mathsf{V}'(10)$`.na` $= 3$, $\mathsf{V}'(10)$`.next` $= 4$ and $\mathsf{V}'(10)$`.prev` $= 5$.

The same example is depicted in Figure 4.3 where the labels are the edge labels.

When adding an edge to a graph g, three new edge structures $\mathsf{E}'(7), \mathsf{E}'(8)$ and $\mathsf{E}'(9)$ are added to the edge structure array E.

The edge structure array for graph g changes from $\mathsf{E}= [E(1), E(2), E(3), E(4), E(5), E(6)]$ to $\mathsf{E}' = [E'(1), E'(2), E'(3), E'(4), E'(5), E'(6), E'(7), E'(8), E'(9)]$.

Before adding an edge, the first event in the graph g is a coalescent event and the field $\mathsf{E}(2)$`.b` has a value 0. After the edge is added, the first event is a branching event. The field $\mathsf{E}'(2)$`.b` changes to 1. The main changes to the

existing edge structures $\mathsf{E}(2)$ and $\mathsf{E}(3)$ are the fields containing the labels of the vertices incident to the edges 2 and 3. These fields change from $\mathsf{E}(2)\,.\,\mathsf{v}= (2,4)$ before adding an edge to $\mathsf{E}'(2)\,.\,\mathsf{v}= (2,9)$ after adding an edge. Similarly, $\mathsf{E}(3)\,.\,\mathsf{v}= (3,5)$ changes to $\mathsf{E}'(3)\,.\,\mathsf{v}= (3,10)$.

The new edge structures are $\mathsf{E}'(7)$ with $\mathsf{E}'(7)\,.\,\mathsf{v}$ = (9,4), $\mathsf{E}'(8)$ with $\mathsf{E}'(8)\,.\,\mathsf{v}= (10,5)$ and $\mathsf{E}'(9)$ with $\mathsf{E}'(9)\,.\,\mathsf{v}= (9,10)$.

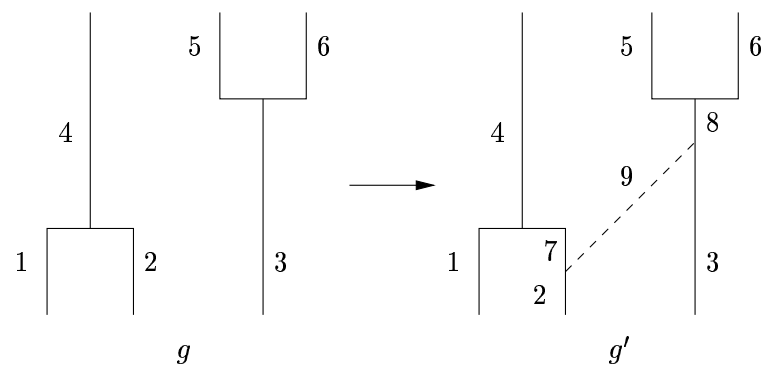


Figure 4.3: Edge labels before and after adding an edge

4.3 Labeling

Two arrays, `Nodes` and `Edges` contain the labels of the vertices and the edges respectively. Associated with each array is a marker, which indicates how many vertex and edge labels are in use. The variable nv is the marker for the array `Nodes`. It is also the number of vertices in a graph, g. Similarly, the variable ne is a marker for the array `Edges`.

The first n inscriptions of the `Nodes` array are the leaf labels $[1, 2, 3, ..., n]$

and since each leaf is of degree one, the first n inscriptions of the edge array `Edges` are also $[1, 2, ..., n]$. These n inscriptions stay unchanged throughout the Markov chain updates.

The first graph in the Markov chain is generated by growing a graph, g backwards from the leaf vertices, $V_L = \{1, 2, 3, ..., n\}$. The array `Nodes` consists of the first n labels and the variable nv is set to n. Corresponding to the vertex labels are the edge lables $E_L = \{1, 2, 3...n\}$. The edge array `Edges` contains the labels $1, ...n$ and the variable $ne = n$.

A variable k indicates the number of edges between two events. This variable is set to n.

Let $\rho = \binom{n}{2} + \frac{\sigma n}{2}$. Sample a time t_1 from an exponential distribution with rate ρ. With probability $\frac{\frac{\sigma n}{2}}{\rho}$ one of the n edges is chosen at random to branch. Otherwise with probability $1 - \frac{\frac{\sigma n}{2}}{\rho}$ two edges are chosen at random to coalesce. The event is represented by a vertex and a label $n + 1$ is added to the array `Nodes`. The variable nv is increased by 1.

If the event is a branching event two new edges are created and get the labels $n + 1$ and $n + 2$. These labels are added to the array `Edges`. The counter $nv = nv + 1$ and the counter $ne = ne + 2$. The rate variable ρ becomes $\binom{n+1}{2} + \frac{\sigma(n+1)}{2}$ and the variable $k = k + 1$.

If the event is a coalescing event the new edge label is $n + 1$ and is added to the array `Edges`. The rate variable ρ changes to $\binom{n-1}{2} + \frac{\sigma(n-1)}{2}$ and the variable $k = k - 1$.

The structures `V` and `E` are updated accordingly.

A second event occurs at time t_2 which is exponentially distributed with rate ρ. The label of the second event is $n+2$. If the first event is a coalescent event and the second event is a branching event, the labels of the two new edges are $n+2$ and $n+3$ respectively. However, if the first event is a branching event and the second event is a branching event, then the labels of the two new edges are $n+3$ and $n+4$ respectively. If the first event is a branching event and the second event is a coalescing event, the label of the new edge is $n+3$, while if both the events are coalescing events the label of the new edge is $n+2$. The entire process is repeated until the variable $k = 1$.

When adding an edge, two new vertices and three new edges are added to the existing graph g. Therefore two new vertex labels and three new edge labels are needed. The first nv of labels in the array `Nodes` are in use. If the array `Nodes` has more than nv inscriptions, the extra labels are available for the new vertices. In this case the marker nv is simply moved to a new position. If there are no extra labels, new ones are created and added to the array `Nodes` and $nv' = nv + 2$. A similar procedure is used for the edge labels.

The process is reversed when deleting an edge. When deleting an edge, two vertices and three edges are removed from the graph. The markers nv and ne are simply updated accordingly. Two vertex and three edge labels are then free for later use.

Before adding an edge, the graph g in the previous example as seen in Fig-

ure 4.2 corresponds to the array `Nodes` $= [1, 2, 3, 4, 5, 6, 7, 8]$ and $nv = 8$. After adding an edge, the node array becomes `Nodes'` $= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$, and $nv' = nv + 2 = 10$.

The corresponding edge array before adding an edge of the same graph, g in Figure 4.3 is `Edges`$= [1, 2, 3, 4, 5, 6]$ and $ne = 6$. After an edge is added, the array becomes `Edges'` $= [1, 2, 3, 4, 5, 6, 7, 8, 9]$ and $ne' = ne + 3 = 9$.

When an edge is added to a graph, one branching and one coalescent event are added to the graph. A variable nb keeps track of the number of branching events in a graph, since the number of branching events in a particular graph is important in the calculation of the prior density function. (Chapter 2 Equation 2.1).

An edge can be legally deleted from a graph if the vertex at the bottom of the edge represents a branching event and the vertex at the top of the edge is a coalescent event. An array `ee` contains the labels of the edges in a graph, which can be legally deleted. After an edge is deleted or added to the graph, the array `ee` is updated accordingly. A function is called to check if the array `ee` contains only edges which can be deleted. A quick check can be run to see if the number of labels in the array `ee` is equal to the variable nb.

In the example of Figure 4.2, after adding an edge, the edges with labels 7 and 9 can be deleted, since they are the result of a branching event and both have a coalescent event at the top. If the edge with label 9 is deleted from the graph at a later stage, the edges 7 and 8 are also deleted from the graph. The

number of deletable edges decreases by two and the two corresponding labels, 7 and 9 are removed from the array `ee` of deletable edges.

4.4 The process of adding an edge

The processes of adding and deleting an edge consist of well-defined steps. When adding an edge, two edge labels, i and j are chosen at random from the array `Edges`. A function `AddVertex` is called to add one vertex at a time to one of the chosen edges. The position of the new vertex is randomly determined and a label from the array `Nodes` is assigned to the new vertex. This is done twice, since the addition of an edge requires the addition of two vertices, one on each edge.

The new vertices divides the two edges, i and j into two parts. Labels for the new edges above the two new vertices are chosen from the array `Edges`. Another function is called to simply connect the two new vertices with a new edge. The label of the new edge is chosen from the array `Edges`.

For instance in the example in Figure 4.4 a new edge is added between the two edges $\mathbf{3}$ and $\mathbf{2}$. The edge $\mathbf{3}$ has vertices 3 and 5 and edge $\mathbf{2}$ has vertices 2 and 4. First, the new label 10 is assigned to the new vertex on edge $\mathbf{3}$. The label $\mathbf{8}$ is assigned to the new edge above the new vertex 10. A label 9 is assigned to the new vertex on edge $\mathbf{2}$ with the label of the new edge above the vertex 9 being $\mathbf{7}$. The last labeling step involves the labeling of the new edge $\mathbf{9}$.

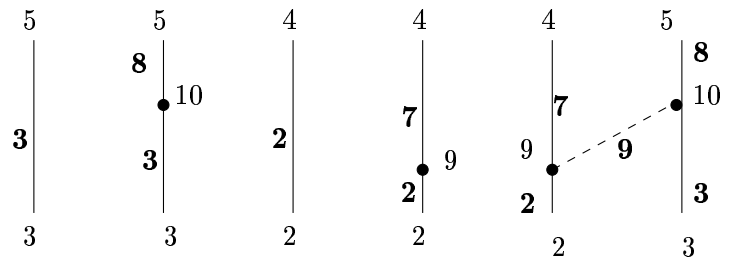


Figure 4.4: Process of adding an edge with edge labels in bold

After all the label assignments, a function `AddEdge` is called to make the relevant connections and updates of the new edge. The arrays `Nodes` and `Edges` are updated with the extra labels and the vertex and edge structures `V` and `E` are updated separately.

When deleting an edge the labels of the deleted vertices and edges are removed from their positions in the `Nodes` and `Edges` arrays and the `V` and `E` structures are updated as discussed in Section 4.3

4.5 Other moves

The other moves involve only minor updates. Since these moves do not involve the addition or deletion of vertices, the arrays `Nodes` and `Edges` are not involved and new data structures are not created. The changes to graphs are only local changes and the relevant fields of the vertex and edge data structures are updated. The numbers of branching and coalescing events stay unchanged and the list of deletable edges is not modified.

When proposing to change the time associated with a vertex, a vertex label, v is chosen at random from the array `Nodes`. Since the vertex cannot be a leaf vertex, the process of chosing a vertex is continued until an ancestral vertex label is chosen. The next and previous vertices of the vertex v are determined. A new time t is chosen between the next and the previous events of vertex v. This is to ensure that the previous and next fields of the vertex v need not to be updated, when the vertex v is assigned the new time t. The field `V`(v)`.time` is updated with the new time, t.

The selection type move involves the choice of a randomly selected vertex v from a graph g. This is done in exactly the same manner as for the move involving the change of time of a vertex.

A list of probabilities of possible allele types is maintained according to the transition matrices associated with each edge. A new type of allele is proposed for the vertex by randomly chosing a type according to the list of probabilities. If the new state is accepted the fields `V.a` and `E.allele` are updated, accordingly.

4.6 Special cases

4.6.1 Above the root

The vertex above the root, V_{AR} is only for cosmetic reasons. Only one edge connects the vertex V_{AR} with the root. The field `V`(V_{AR})`.time` contains the maximum floating number of the software being used. The label and the time of the vertex V_{AR} stay fixed throughout the Markov chain updates. The fields `V.prev` and `V.a` are empty.

Since the edge above the root connecting the root with the vertex above the root is neither branching nor coalescing, the field `E.b` is empty. The type of the allele above the root on the edge is the same as the root. Since the field `E.allele` must contain two inscriptions, the convention is to assign the particle type at the bottom of the edge to the particle type at the top of the edge of the first graph. The second inscription of the field `E.allele` stays fixed during the generation of the chain.

4.6.2 Bubbles

When adding a new edge to the graph g, the new edge is added between two edges chosen at random from the array `Edges`. When the same edge is chosen twice, a bubble is created since the two new vertices are on the same edge. The creation of a bubble is discussed in Chapter 3.

The variables $ae(1)$ and $ae(2)$ holds the labels of the two chosen edges. In the case of a bubble the variables $ae(1)$ and $ae(2)$ contain the same label.

Since the same edge is chosen twice, care must be taken in the assignment of the edge and the vertex labels. Let t_1 and t_2 denote the positions of the first and the second vertices on edges $ae(1)$ and $ae(2)$, respectively. Suppose $t_2 > t_1$. The first vertex is labeled and placed in position t_1. The label of the edge below the first vertex is unchanged and is the same as the label in $ae(1)$. A new label is assigned to the edge above the first vertex. All the connections with the first vertex is carried out.

If the second vertex is placed at position t_2 on the edge with label in $ae(2)$ an error occurs, since the label in $ae(2)$ is the same as the label in $ae(1)$. The second vertex is placed below the first vertex although the time of the second vertex is greater than the time of the first vertex. Therefore a temporary labeling is necassary. A logical operator `SAME` is switched on to indicate that a bubble is created and that $t_2 > t_1$. The label of $ae(2)$ is changed to the label of the new edge above the first vertex. The second vertex is placed on the same edge but with a different temporary label. As soon as all the connections are made, the label of $ae(2)$ is changed back to the original label. The variable ae is used later in the code.

Bubbles can be created by either adding or deleting an edge. When the edge e in Figure 4.5 is deleted, a bubble is created.

The deletion of an edge, which creates a bubble can eventually lead to an

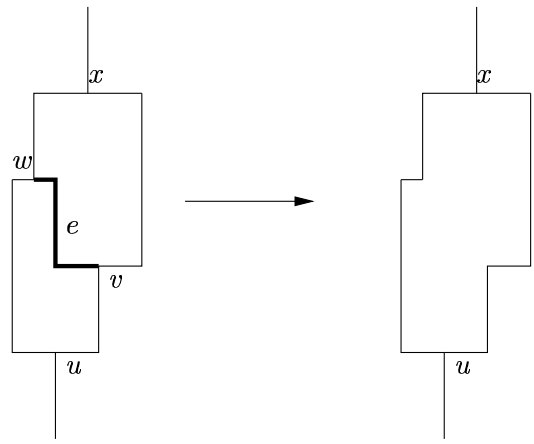


Figure 4.5: Bubble after deleting an edge

illegal state as explained in Subsection 4.6.3 below. When proposing to add a bubble to a graph, the acceptance probability of the new state contains a factor 2 as discussed in Chapter 3. Therefore, after each deletion or addition of an edge, the graph g is checked to find all bubbles in the graph by calling a function `chbubble`.

4.6.3 Illegal states

An illegal state is created when the number of edges between two events is less than two. If the edge above the root, e_1 is chosen twice during the addition of an edge, the creation of a bubble above the root is proposed as can be seen in Figure 4.6. This leads to an illegal state. Let t_1 and t_2 be the two positions of the new vertices on the edge e and let $t_{v_{UA}}$ be the time of the root. The following piece of code identifies this situation and the new illegal state is rejected.

```
Step 1:if min(t1, t2) > tUA
Step 2:  REJECT
Step 3:end
```

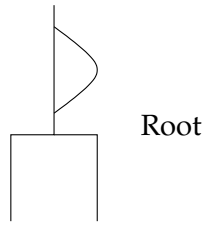


Figure 4.6: Bubble above a root

Another illegal state is proposed when the number of edges between two events u and v is equal to two and the proposal is to delete one of the two edges as can be seen in Figure 4.7. After the edge, e, is removed the number of edges in the graph below the vertex u is equal to one.

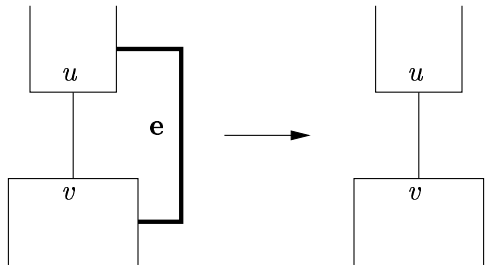


Figure 4.7: Creation of illegal states

This illegal state is much more difficult to pick up than the creation of a bubble above the root. For this reason a function is employed to check if the proposed graph is a legal state. This function focuses on the part of the graph

where a change is proposed. Each vertex is queried to see if it is an internal vertex, i.e. the field `V.type` has value 2. The field `V.na` contains the number of edges below a vertex. If this field contains the value 2 before a deletion of an edge, and changes to 1 after the deletion, the proposed state is illegal and the new state is rejected.

The algorithm for this function is as follows:

```
Step 1:  if V(v).type =2
Step 2:    if V(v).na = 2
Step 3:      set ILLEGAL equal to 1
Step 4:    end
Step 5:  end
```

4.6.4 Genealogy

The ancestry or genealogy of the sample at the leaves is embedded in the ancestral selection graph. The rules for determining the subtree are explained in Chapter 2. Since the most recent common ancestor, (MRCA) of the sample is of importance to evolutionary biologists, the ancestry of the sample is determined with each update.

This is done by traversing the graph from the leaves to the root of the graph by using the `V`(v)`.prev` field of each vertex v. Since the leaf edges are all connected to the leaf vertices, all the leaf edges are included in the genealogy. Therefore the fields `E.used` of the first n edges of the array `Edges` are changed from -1 to 1. The flag of -1 is used for debugging purposes and indicates

that the edge was not visited to determine the inclusion of the edges in the genealogical tree.

At each vertex v, the edge e_1 is queried through the edge structure `E`(e_1)`.b` for a varyfication of branching. If the vertex v is a branching vertex, one of the two edges e_2 or e_3 at the top of v is a continuous edge and the other edge is a replacing edge. The allele type at the vertex v as well as the allele types at the bottom of the two edges e_2 and e_3 determine the edge to be included in the genealogy.

If the value of the field `V`(v)`.a` is 0 and both the values of `E`(e_2)`.allele`(1) and `E`(e_3)`.allele`(1) are 0 then the continuous edge of the two outgoing edges is included in the genealogy. The continuous edge is determined by using the field `E`(e)`.cnt` of the edge. The inclusion is signalled by changing the field `E.used` of the continuous edge from -1 to 1. The field `E.used` of the replacing edge is changed from -1 to 0.

If the allele type of the branching vertex, v is 1 and both the allele types on bottom of the edges e_2 and e_3 are of type 1, the replacing edge is included in the genealogy. The field `E`(e)`.used` of the replacing edge is changed to 1 and the field `E`(e)`.used` of the continous edge is changed to 0.

If `V`(v)`.a`$= 1$ and the allele type at the bottom of one of the edges e_2 or e_3 are of type 1, the field `E.used` of the edge with the allele type 1 at the bottom, is changed to 1 and field `E.used` of the other edge is changed to 0. The following code determines the edges to be included in the genealogy if the vertex is a

branching vertex.

```
Step 1:  if V(v).a =0
Step 2:    if E(e2).cnt==1
Step 3:      set E(e2) used =1
Step 4:    else set E(e3).used=1
Step 5:  if both edges at top has allele type 1
Step 6:    if E(e2).cnt=0
Step 7:      set E(e2) .used=1
Step 8:  else set E(e3) .used=1
Step 9:    if E(e2).allele(1)=1 and E(e3).allele(1)=0
Step10:      set E(e2) used=1
Step11:  else set E(e3).used =1
```

At a coalescent vertex, if the fields `E.used` of at least one of the edges going into the vertex from the bottom have value 1, the field `E.used` of the edge e_3 at the top going out of the vertex, v , is changed from -1 to 1. If both the fields `E.used` of the edges at the bottom of a coalescent event have value 0, the field `E.used` of the edge e_3 at the top is changed to 0. The following code is used to determine the edges included in the tree when the vertex is a coalescent vertex.

```
Step 1:  if E(e1).used =0 and E(e2).used =0
Step 2:    set E(e3).used =0
Step 3:  else
Step 4:    set E(e3).used=1
Step 5:  end
```

After all the edges of the ancestral tree are determined and tagged, the search for the MRCA of the tree is carried out. The most recent common ancestor of the genealogy of the sample at the leaves is the first coalescent vertex

from the top of the ancestral graph, for which the two edges e_1 and e_2 below the vertex are tagged to be part of the tree. Therefore the search starts at the root vertex v_{UA} of the selection graph. Since the event at v_{UA} is a coalescent event, it is only necessary to determine if the two edges, e_1 and e_2 below the root are part of the binary tree. If at least one of these two edges has a value 0 in the field `V`(e)`.used`, the edge as well as the root vertex are not included in the ancestral tree. The search is continued by determining the label of the next event, v, using the field `V.next`. The next event down from the root of the graph cannot be a branching event, therefore both of the edges e_1 and e_2 below the vertex, v are checked to see if they are tagged to be part of the tree. If one of the edges has a value 0 in the field `E`(e)`.used`, the vertex as well as both the edges e_1 and e_2 below the vertex, v are not part of the tree. The process of finding the first coalescent event where both the edges e_1 and e_2 below the event are included in the tree, is carried out by using the field `V.next` of the vertex structure `V` and the field `E.used` of the edge structure `E`. All the branching events are ignored until the MRCA is found. Let cn denote the current vertex.

```
Step 1:  If cn is a coalescent vertex.
Step 2:  if E(e1)used=1 and E(e2) used=1
Step 3:    MRCA is equal cn
Step 4:  else find next event
Step 5:    if event is branching event
Step 6:      determine next event
Step 7:    else
Step 8:      go back to step 2
```

```
Step 9:  end
```

A variable `MRCA` and a variable `MRCATIME` keep track of the label of the most recent common ancestor and the time to the most recent ancestor respectively, during the Markov chain updates.

One of the experiments in Chapter 6 requires the knowledge of the ancestral tree for a group of organisms. The ancestral tree is kept fixed throughout the Markov chain updates by rejecting all ancestral selection graphs with a different genealogy embedded in the graph.

The process of finding a genealogy in the graph is carried out in the same manner as the process of finding the MRCA. After the MRCA is found the branching events can not be ignored. At each vertex, the edges which are tagged are included in the genealogy. The time of the vertices included in the genealogy is sorted in an array in ascending order.

The following definition is used to compare two genealogies.

Definition 1 *Two genealogies or trees $Tr_{V,E}$ and $Tr'_{V',E'}$ are isomorphic if the following two conditions hold:*

1. *A bijection $\phi : V \to V'$ exists such that if $\langle u, v\rangle \in E$ then $\langle \phi(u), \phi(v)\rangle \in E'$.*

2. *A graph isomorphism $\psi : Tr_{V,E} \to Tr_{V',E'}$ exists which preserves the edge lengths, i.e. $\mathcal{L}'(\psi(\langle u, v\rangle)) = \mathcal{L}(\langle u, v\rangle)$.*

Before the acceptance-rejection step of the algorithm the embedded tree of the selection graph is determined and the time of the vertices placed in an

array. This array is compared to the array containing the times of the vertices of the true or original tree. If the two arrays have the same size, condition 1 is satisfied. The second condition is satisfied if the inscriptions of the two arrays are the same.

If the embedded tree of the proposed graph agrees with the true tree, the acceptance step is carried out. All other selection graphs are rejected.

Chapter 5

Debugging

One of the challenges of a computational problem is to thoroughly debug the code. This is accomplished by carrying out a collection of different debugging tests on a regular basis. The data structures, associated arrays and variables as discussed in Chapter 4 are applied to the debugging process. The debugging process consists of two main procedures. The first procedure is to check if the graphs are structurally correct. This part is carried out at a regular basis throughout the implementation of the code. The second important set of tests are the statistical and analytic tests.

The structural tests can be divided into two main areas of which the first consists of tests to see if the vertices and the edges are correctly connected. The second set of structural tests consists of tests to see if the states being visited are legal states. If the states are not legal the proposal of the new state is rejected. After a new state is accepted the state is again checked to see if it is an

admissible graph. If the new state is accepted, but is not legal, an error occurs and the running of the program is stopped.

As discussed in the previous chapter by starting at the root of the graph every vertex in the graph can be reached by using the field `V.next` of the vertex data structure. The reverse direction can also be followed by starting at the leaf vertex with label 1 and using the `V.prev` field. This information is used extensively during the debugging process in the following algorithm.

```
Step 1:  cn=Root;
Step 2:  while V(cn).type ≠ 1
Step 3:    carry out check
Step 4:    cn = V(cn).next;
Step 5:  end
```

5.1 Structural testing

The set of external vertices consists of the leaf vertices and the vertex above the root. All other vertices are internal vertices.

If the degree of an internal state is not three or if the degree of an external vertex is not one, the graph is not admissible. The following algorithm determines if the degrees of the vertices of an accepted graph g are correct.

```
Step 1:  if V(v).type =1 or 5
Step 2:    if V(v).e has length 1
Step 3:      TRUE
Step 4:    else ERROR
```

```
Step 5:    end
Step 6:  else if V(v).type = 2 or 4
Step 7:    if V(v).e has length 3
Step 8:      TRUE
Step 9:    else ERROR
Step 10:  end
Step 11:end
```

After a structural change of a graph is accepted, a sorted list of the times associated with all the vertices is created. When starting at the root of the graph the list should be in a descending order with the times associated with the leaves all zero.

Another test is run to see if the fields of the data structures containing the next and previous events are correct. This is done using the following piece of code where the current vertex label is denoted by `cn`. The code is nested in a while loop which starts with the root vertex and ends with the last leaf vertex, labeled one.

```
Step 1:  if (cn == V(V(cn).next).prev and
         cn == V(V(cn).prev).next)
Step 2:   TRUE
Step 3:  else ERROR
Step 4:  end
```

The field `V.na` contains the number of edges between two events and below the older event. The purpose of this field is primarily for the calculation of the prior probabilities. This field can also be used as a debugging tool. When starting at the leaves going up to the root of the graph, the graph consists of a

series of coalescing and branching events. The number of edges between the leaves and the first event is equal to the number of leaves. If the event at a vertex is a branching event, the number of edges increases by one, while at a coalescing event the number of edges decreases by one. The root of the graph is the ultimate coalescing event. The number of lineages below the root and before the next event is equal to two.

The vertex above the root is for cosmetic purposes only and there is only one edge between the root and the vertex above the root.

The algorithm for determining if the number of edges between events is correctly updated consists of the `while` loop at the beginning of this chapter. Again the current vertex is chosen to be the root. The type of event at the current vertex is determined as is explained in Chapter 4. If the vertex is a coalescing vertex, the number in the field `V.(V(cn).prev).na` is one less than the number in the field `V(cn).na`. If the event at the current vertex is a branching vertex, the number in the field `V.(V(cn).prev).na` is one more than the number in the field `V(cn).na` of the current vertex.

During the same function if a branching event is encountered the edges in the field `V.e` are checked to see if the top vertices of the edges e_2 and e_3 are at a later time than the current vertex, and the bottom vertex of the edge e_1 is at an earlier time than the current vertex.

The same procedure is carried out to determine if the edges e_1 and e_2 are below a coalescing vertex. The bottom vertices of the two edges, e_1 and e_2

should be at an earlier time than the coalescing event. The top vertex of the third entry of the field `V.e` is at a later time than the coalescing vertex.

The field of the edge structure `E.v` contains the labels of the vertices incident to an edge. The convention is to have the label of the bottom vertex in the first position and the label of the top vertex in the second position. A check is run to make sure the labels are in the correct order.

One of the most important structural tests is to check if the labels of the edges are in agreement with the labels of the vertices. For instance suppose an edge i has vertex labels $(v_{(i)}, v^{(i)})$. The field `E`(i)`.v` contains the labels $(v_{(i)}, v^{(i)})$. One of the entries in the field `V.e` of the corresponding vertex data structures `V`$(v_{(i)})$ and `V`$(v^{(i)})$ must contain the label i.

As discussed in Chapter 4, the array `ee` containing the labels of the edges which can legally be deleted is updated after each acceptance of a new state. The edges in this array are not branching edges but are the result of a branching event. It is easily checked if the array contains only non-branching edges by checking the field `E.b` of the edges in the array `ee`. If the edges are not branching, the vertex at the bottom of each edge should represent a branching event.

By starting at the root of the graph and using a `while` loop, the number of branching events in a graph is counted by counting the number of edges with value one in the `E.b` field. The number of branching edges in the graph should be in agreement with the variable `nb`, which is updated during the add

and delete move (Chapter 3 Section 3.1.1).

At the same time as a branching edge is encountered, the two edges, e_2 and e_3 above the top vertex of the branching edge, are checked to see if they can legally be deleted. If these edges are not branching, the labels of the edges must be in the array `ee` of deletable edges.

The data structures of the first n labels of the array `Nodes` are the data structures of the leaf vertices. This is checked by retrieving the label from the `Nodes` to verify if the field `V.type` has the value 1. All the other labels, except the label of the root and the label of the vertex above the root, in the `Nodes` array have a value 2 in the type field. The root has a value 4 and the vertex above the root has a value 5.

Other minor debugging routines are run which include a check for empty fields in the data structure, that the times of the leaves are zero and the time of the vertex above the root is equal to the maximum floating time of the computer software in use.

This collection of structural checks is run routinely after each acceptance of a new state and is switched off when doing the experiments to cut down on computer run time.

The second set of debugging routines consists of analytic and statistical tests.

5.2 Analytic tests

The following analytic results were used to debug the code. Let n be the number of individuals sampled from a population, i.e. the number of leaves in a graph. The expected time to the ultimate ancestor as derived by Neuhauser and Krone [17] is given as

$$E_n(t_{UA}) = 2\left(1 - \frac{1}{n}\right) + 2\sum_{r=1}^{n-1} \frac{1}{r(r-1)} \frac{e^{\sigma}}{\sigma^{r+1}} \int_0^{\sigma} t^{r+1} e^{-t} dt$$

When $\sigma = 0$ the expected time to the ultimate ancestor is equal to the expected time to the most recent common ancestor, MRCA,

$E_n(t_{UA}) = E_n(t_{MRCA}) = 2(1 - \frac{1}{n})$.

The times between events are exponentially distributed with rate $\binom{k}{2} + \frac{\sigma k}{2}$, where k is the number of edges between events. Therefore the expected time between events is $\frac{1}{\binom{k}{2} + \frac{\sigma k}{2}}$.

A graph g can be thought of a series of events, together with a label process $\{(\beta_m, \gamma_m), m \geq 1\}$ where m denotes the mth event, β_m denotes the edge which branches at a branching event and γ_m denotes the two edges merging at a coalescent event [17]. The label process, which is different from the labeling in Chapter 4, Section 4.3 describes the topology of the graph and is a powerful debugging tool.

Let $\mathcal{E}$ denote the set of events, ordered according to the time at which these events happen starting at the leaves. Let the symbols b and c represents a

branching and a coalescing event respectively. Suppose an ancestral selection graph has only two leaves and only one coalescent event i.e the graph is a tree, then the set $\mathcal{E} = \{c\}$. If the same graph has only one branching event, followed by two coalescing events, the set $\mathcal{E}$ representing the graph is $\mathcal{E} = \{b, c, c\}$. The example of a tree with two leaves and a graph with one branching event can be seen in Figure 5.1. Since the last event in a graph is always a coalescent event between two edges, the last element of the set $\mathcal{E}$ is always c.

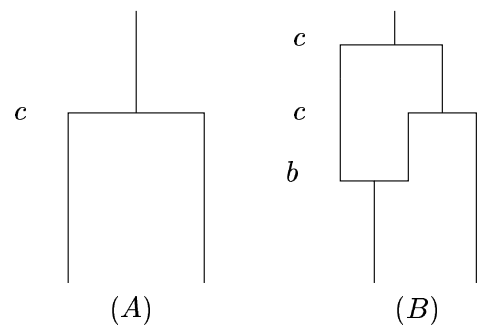


Figure 5.1: $\mathcal{E}_{g_1} = \{c\}$ and $\mathcal{E}_{g_2} = \{b, c, c\}$

The probability of a set $\mathcal{E}$ can be calculated by taking the product of the probabilities of the elements in set $\mathcal{E}$. For instance the probability of a branching event, b, given k edges, is $\mathbf{P}(b) = \frac{\frac{\sigma k}{2}}{\binom{k}{2}+\frac{\sigma k}{2}} = \frac{\sigma}{k-1+\sigma}$ and the probability of a coalescent event is $\mathbf{P}(c) = \frac{\binom{k}{2}}{\binom{k}{2}+\frac{\sigma k}{2}} = \frac{k-1}{k-1+\sigma}$.

Suppose a graph g with 2 leaves has the following structure, $\mathcal{E} = \{b, c, c\}$. The number of edges between events moving from the leaves to the root of the graph changes from $2 \rightarrow 3 \rightarrow 2 \rightarrow 1$ as can be seen in Figure 5.1. The probability of each element of $\mathcal{E}$ is $\mathbf{P}(b) = \frac{\sigma}{1+\sigma}$, $\mathbf{P}(c) = \frac{3}{3+\frac{3\sigma}{2}}$ and $\mathbf{P}(c) = \frac{1}{1+\sigma}$.

Therefore the probability to get a graph with the structure $\mathcal{E} = \{b\,, c\,, c\}$ is the product of the probabilities of the elements of $\mathcal{E}$, which is

$$\mathbf{P}(\{b, c, c\}) \;=\; \left(\frac{\sigma}{1+\sigma}\right)\left(\frac{3}{3+\frac{3\sigma}{2}}\right)\left(\frac{1}{1+\sigma}\right) \tag{5.1}$$

Label process

Each set $\mathcal{E}$ is built up by using combinations of the two elements, b and c. Associated with each graph, g, is a label process to describe the topology of the graph. By going from the leaves of the graph to the root of the graph, let m denote the mth event in the graph. For the element $b_m \in \mathcal{E}$, let β_m carry the label of the edge which branches. In the case of a coalescent event c_m, two edges coalesce. Therefore let γ_m contain the two labels of the two edges, which coalesce. Suppose the mth event is a branching event, b, where the edge j branches into two edges. Since this event cannot be a coalescent event at the same time, the variables $\beta_m = j$ and $\gamma_m = (0, 0)$. The corresponding element in the label process for this event is $(\beta_m = j, \gamma_m = (0, 0))$. The labels of the two resulting edges of the branching event are j and $m + |V_L|$, where $|V_L|$ is the number of leaves.

If the mth is a coalescent event between edges i and j, where $i < j$, the corresponding element in the label process is $(\beta_m = 0, \gamma_m = (i, j))$. The resulting edge has label i. For a more detailed discussion on the labeling process see the

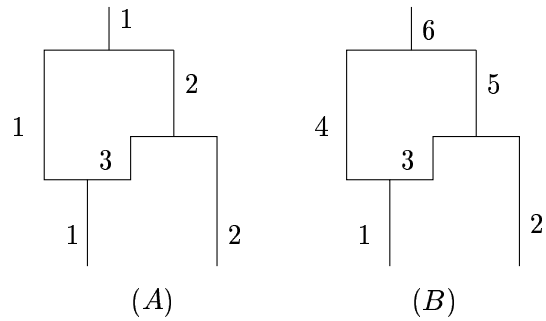


Figure 5.2: (A): Label process, (B): Labels for the code

paper by Neuhauser and Krone [17].

As can be seen in Figure 5.2 the labels of the label process are different from the labels assigned to the edges of the graph during the labeling of the graph. The label process describes the topology of the graph while the labels of the graphs are the names of the different edges. Two edges can not have the same name, while the same label can be assigned to two different edges in the label process. For instance the label process of the graph in Figure 5.2 (A) is $\{\beta_1 = 1\, \gamma_1 = (0,0);\ \beta_2 = 0\, \gamma_2 = (2,3);\ \beta_3 = 0\, \gamma_3 = (1,2)\}$ while the labels of the edges during the implementation of the code are $1, 2, 3, 4, 5, 6$ Figure 5.2 (B).

In Figure 5.3 are two graphs with 2 leaves which have the same event set $\mathcal{E} = \{b, c, c\}$, but each with a different label process. The graph in Figure 5.3 (A) has a label process $\{\beta_1 = 1, \gamma_1 = (0,0); \beta_2 = 0, \gamma_2 = (1,4); \beta_3 = 0, \gamma_3 = (1,2)\}$ and the label process of the graph in Figure 5.3 (B) is $\{\beta_1 = 1, \gamma_1 = (0,0); \beta_2 = 0, \gamma_2 = (2,4); \beta_3 = 0, \gamma_3 = (1,2)\}$. A tree with only two leaves has set $\mathcal{E} = \{c\}$ and label process $\{\beta_1 = 0, \gamma_1 = (1,2)\}$.

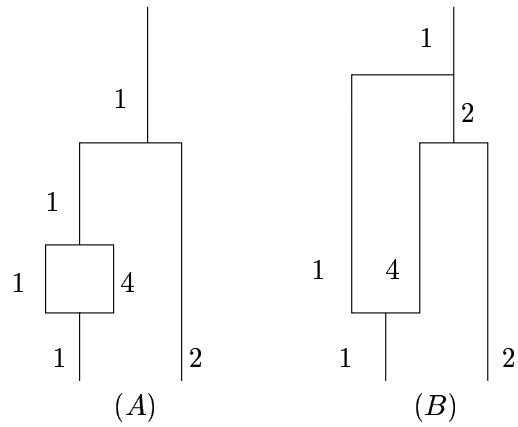


Figure 5.3: Two graphs with the same $\mathcal{E} = \{b, c, c\}$ and different label processes

The probabilities of the graphs in Figure 5.3 are

$$\mathbf{P}(\{\beta_1 = 1, \gamma_2 = (2,4), \gamma_3 = (1,2)\}) = \frac{4\sigma}{(1+\sigma)(6+3\sigma)} \frac{1}{1+\sigma} \tag{5.2}$$

and

$$\mathbf{P}(\{\beta_1 = 1, \gamma_2 = (1,4), \gamma_3 = (1,2)\}) = \frac{2\sigma}{(1+\sigma)(6+3\sigma)} \frac{1}{1+\sigma} \tag{5.3}$$

The debugging process is started by generating graphs with two leaves and a small value of σ. Small values of σ cause low levels of branching and short graphs i.e the time to the ultimate ancetor is not high. This enables the easy calculations of the ratios and the probabilities involved.

The debugging process of the MCMC is done in two stages. During the first stage the MCMC draws samples $\{\Phi_i\} \sim f_G(g|\sigma)$ (Chapter 2, Equation 2.1). After the first stage of debugging is completed satisfactorily, the second stage is started, where the samples $\{\Phi_i\}$ are drawn from $h(g, B_Y, B_I|\sigma, \theta = 1)$.

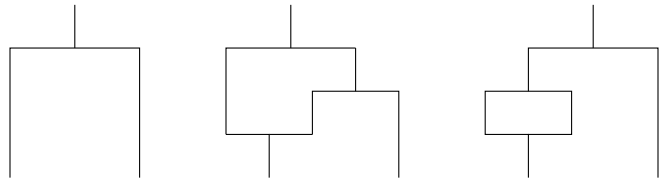


Figure 5.4: Diagram of the three different graphs used for debugging. The number of leaves is 2. The Markov chain generated for debugging is $\{\Phi_i\} \sim f_G(g|\sigma)$

.

Stage I

The three graphs in Figure 5.4 are used as debugging tools. The graphs have only two leaves to make the calculation of the probabilities easier. During the debugging phase the results of MCMC are compared to the analytic results of graphs with zero branching and graphs with one branching event for three different values of σ.

Following from the discussion in Section 5.2 of this chapter, the probability of $\mathcal{E} = \{c\}$ given two edges, is $\mathbf{P}(c) = \frac{1}{1+\sigma}$. The other analytic results of Table 5.1 are calculated using Equation 5.1 of Section 5.2 and Equations 5.2 and 5.3 of Section 5.2.

The analytic results and the debugging results are compared in Table 5.1 for $\sigma = 0, \sigma = 0.5$ and $\sigma = 1.5$.

	Analytic			MCMC		
	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1.5$	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1.5$
$\mathbf{P}(\{c\})$	1	0.6667	0.4000	1.0000	0.6799	0.4443
$\mathbf{P}(\{b, c, c\})$	0	0.1778	0.1371	0.0000	0.1725	0.1545
$\mathbf{P}(B)$	0	0.1185	0.0914	0.0000	0.1165	0.1068
$\mathbf{P}(C)$	0	0.0593	0.0457	0.0000	0.0560	0.0477
t_{UA}	1	1.1898	1.7615	1.0154	1.1713	1.8849
t_{FE}	1	0.6667	0.4000	1.0154	0.6544	0.2935

Table 5.1: Analytic and debugging results of the MCMC $\{\Phi_i\} \sim f_G(g|\sigma)$. $P(B)$ is the probability of one branching event, but not a bubble, Figure 5.4 (B). $P(C)$ is the probability of one bubble, Figure 5.4 (C). t_{UA}, time to ultimate ancestor and t_{FE} is time to first event.

Stage II

The difference between stage I and stage II of the debugging process is that-during the second stage, the mutation process is incorporated in the MCMC. Since the mutation process is propagated down from the root of the graph, independently on each edge, an even number of mutations along an edge causes the allelic type at the top of the edge to be the same as the allelic type at the bottom, while an odd number of mutation causes the allelic type at the bottom of the edge to be different than the allelic type at the top.

The probability that the allelic type at the bottom of an edge i of length τ is the same as the allelic type at the top of the edge, is the entry of the first column and the first row of the transition matrix associated with the edge i. The probability of a change in the type from the top to the bottom of the edge i, is the entry of the second column and the first row of the transition matrix as

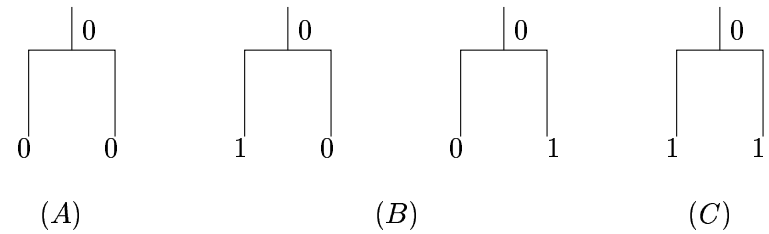


Figure 5.5: Diagram of the three different configurations of the selection graph with zero branching events used for debugging purposes. The number of leaves is 2. The Markov chain generated for debugging is $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$

.

discussed in Chapter 2.

The mutation process gives rise to the three allelic configurations for the graph $\mathcal{E} = \{c\}$ as can be seen in Figure 5.5. In Figure 5.5 (A), the allele type at the root, $B_{v_{UA}}$, is the same as both the allele types at the two leaves, $B_{v_{(1)}}$ and $B_{v_{(2)}}$. The difference in time between the root and the leaves is τ. Therefore the probability, $\mathbf{P}$, of the selection graph with two leaves, one coalescing event and with no change of allele type over either the edges from the top to the bottom of the edges, is $\mathbf{P}(A)$ where

$$\begin{aligned}\mathbf{P}(A) &= \mathbf{P}(B_{UA} = 0)\mathbf{P}(\mathcal{E} = \{c\})\mathbf{P}(B_{v_{(1)}} = B_{v_{UA}})\mathbf{P}(B_{v_{(2)}} = B_{v_{UA}}) \\ &= \mathbf{P}(B_{UA} = 0)(\frac{1}{1+\sigma})(0.5 + 0.5e^{(-\theta\tau)})(0.5 + 0.5e^{(-\theta\tau)}) \qquad (5.4)\end{aligned}$$

By following the same argument for the graph in Figure 5.5 (B), the allelic type at the root, $B_{v_{UA}}$ is different than the allele type at one of the leaf vertices, B_{v_L} and the same as the allele type at the other leaf vertex. The time difference

between the root and the leaf vertices is τ. The probability of the selection graph Figure 5.5 (B) is

$$\begin{aligned}
\mathbf{P}(B) &= 2 * \mathbf{P}(B_{UA} = 0)\mathbf{P}(\mathcal{E} = \{c\})[\mathbf{P}(B_{v_{(1)}} = B_{v_{UA}})\mathbf{P}(B_{v_{(2)}} \neq B_{v_{UA}}) \\
&\quad + \mathbf{P}(B_{v_{(1)}} \neq B_{v_{UA}})\mathbf{P}(B_{v_{(2)}} = B_{v_{UA}})] \\
&= \mathbf{P}(B_{UA} = 0)(\frac{1}{1+\sigma})[(0.5 + 0.5e^{(-\theta\tau)})(0.5 - 0.5e^{(-\theta\tau)}) \\
&\quad + (0.5 - 0.5e^{(-\theta\tau)})(0.5 + 0.5e^{(-\theta\tau)})] \qquad (5.5)
\end{aligned}$$

The selection graph in Figure 5.5 (C) shows a change in the allele type over both the edges. The probability $P(C)$ in Figure 5.5 (C) is

$$\begin{aligned}
\mathbf{P}(C) &= \mathbf{P}(B_{UA} = 0)\mathbf{P}(\mathcal{E} = \{c\})\mathbf{P}(B_{v_{(1)}} \neq B_{v_{UA}})\mathbf{P}(B_{v_{(2)}} \neq B_{v_{UA}}) \\
&= \mathbf{P}(B_{UA} = 0)(\frac{1}{1+\sigma})(0.5 - 0.5e^{(-\theta\tau)})(0.5 - 0.5e^{(-\theta\tau)}) \qquad (5.6)
\end{aligned}$$

For graphs with only two leaves the expected time to the first event is $\frac{1}{1+\sigma}$. Therefore the value of τ in the previous equations depends on σ. The probability that the type of the ultimate ancestor is 0 also depends on the value of σ. The values of the time to the ultimate ancestor and the probability that the type at the ultimate ancestor is 0 for the three values of σ can be seen in Table 5.2.

The analytic and the debugging results for $\mathcal{E} = \{c\}$ for the different values of σ can be seen in Table 5.3.

σ	τ	$\mathbf{P}(B_{UA} = 0)$
0	1.0000	0.5000
0.5	0.6667	0.4668
1.5	0.4000	0.3600

Table 5.2: Time to ultimate ancestor and probability that the ultimate ancestor is of type 0 for different values of σ when $\mathcal{E} = \{c\}$ and type configuration as in diagram 5.5

	Analytic			**MCMC**		
	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1.5$	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1.5$
$\mathbf{P}(A)$	0.2339	0.1782	0.1004	0.1900	0.1890	0.1180
$\mathbf{P}(B)$	0.2162	0.1146	0.0396	0.2284	0.0980	0.037
$\mathbf{P}(C)$	0.0499	0.0184	0.0039	0.0420	0.0100	0.003

Table 5.3: Analytic and debugging results of the MCMC $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$ where the graphs have event set $\mathcal{E} = \{c\}$. $P(A)$ is the probability of $B_{UA} = 0, B_{V_{L(1)}} = 0$ and $B_{V_{L(2)}} = 0$, Figure 5.5 (A). $P(B)$ is the probability of $B_{UA} = 0$ and $B_{V_{L(1)}} \neq B_{V_{L(2)}}$, Figure 5.5 (B). $P(C)$ is the probability $B_{UA} = 0$ and $B_{V_{L(1)}} = B_{V_{L(2)}} = 1$, Figure 5.5 (C).

During the second stage of the debugging process, a graph, $\mathcal{E} = \{b, c, c\}$ as can be seen in Figure 5.6 is used. The allele type of the ultimate ancestor is 0 and the allele types of both the two leaves are 0.

The probability if the graph in Figure 5.6 is calculated by using different combinations of allele types at a_1, a_2 and a_3. Let $\tau_1 = t_3 - t_2, \tau_2 = t_3 - t_1, \tau_3 = t_2 - t_1, \tau_4 = t_2 - t_0$ and $\tau_5 = t_1 - t_0$, where t_0, t_1, t_2 and t_3 are the times associated with each event in Figure 5.6.

The expected time between t_1 and t_0 is $\frac{1}{1+\sigma}$, the expected time between t_2 and t_1 is $\frac{1}{3+\frac{3\sigma}{2}}$ and the expected time between t_3 and t_2 is $\frac{1}{1+\sigma}$. Therefore the

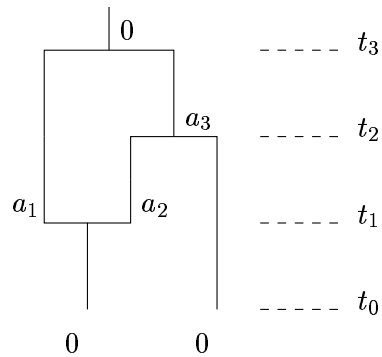


Figure 5.6: Diagram of the configurations of the selection graph with one branching event and zero bubbles used for debugging purposes. The number of leaves is 2. The Markov chain generated for debugging is $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$

.

σ	t_1	t_2	t_3
0.5	0.6667	0.9334	1.6001
1.5	0.4000	0.5905	0.9905

Table 5.4: The time t_0, t_1, t_2 and t_3 for $\sigma = 0.5$ and $\sigma = 1.5$ when the graph has event set $\mathcal{E} = \{b, c, c\}$ but no bubbles as can be seen in Figure 5.6

expected times t_1, t_2 and t_3 are dependent on the value of σ. In Table 5.4 the values of t_1, t_2 and t_3 are recorded for $\sigma = 0.5$ and $\sigma = 1.5$.

The probability of the graph in Figure 5.6, where the type of the ultimate ancestor is 0 and the types of both the leaf vertices are 0, is calculated by using different combinations of a_1, a_2 and a_3. The following equation was used to calculate the probability of the graph in Figure 5.6.

$$\begin{aligned}\mathbf{P}(\mathcal{E} = \{b, c, c\}) &= \mathbf{P}(B_{UA} = 0)\mathbf{P}(\mathcal{E} = \{b, c, c\}(0.5 \pm 0.5e^{-\theta\tau_1})(0.5 \pm 0.5e^{-\theta\tau_2}) \\ &\quad (0.5 \pm 0.5e^{-\theta\tau_3})(0.5 \pm 0.5e^{-\theta\tau_4})(0.5 \pm 0.5e^{-\theta\tau_5}) \qquad (5.7)\end{aligned}$$

As before $\theta = 1$ and the $\pm$ is $+$ if there is not a change over an edge and $-$ if there is a change over an edge. The value of $\mathbf{P}(B_{UA} = 0)$ is the same as in Table 5.2.

The analytic result as well as the debugging results of the graph in Figure 5.6 can be seen in Table 5.5 for the three different values of σ.

	Analytic			**MCMC**		
	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1.5$	$\sigma = 0$	$\sigma = 0.5$	$\sigma = 1.5$
$\mathbf{P}(A)$	0	0.0176	0.0144	0	0.0157	0.0110

Table 5.5: Analytic and debugging results of the MCMC $\{\Phi_i\} \sim h(g, B_Y, B_I|\sigma)$. $P(A)$ is the probability of the configuration as in Figure 5.6

The Likelihood of σ

Let n be the size of the data set sampled from a population and let n_0 be the number of type 0 in the data set. The probability of a type 0 individual in the population is assumed to be $\frac{n_0}{n}$. Therefore, the likelihood of σ, $\mathcal{L}(\sigma)$, is

$$\begin{aligned}\mathcal{L}(\sigma) &= \binom{n}{n_0}\left(\frac{n_0}{n}\right)^{n_0}\left(1-\left(\frac{n_0}{n}\right)\right)^{(n-n_0)} \\ &\quad \times \left(\frac{n_0}{n}\right)^{\theta-1}\left(1-\frac{n_0}{n}\right)^{\theta-1} e^{-(\sigma\frac{n_0}{n})} \end{aligned} \tag{5.8}$$

During the last debugging procedure, the marginal posterior density of σ, $P(\sigma)$, is compared to $\mathcal{L}(\sigma)$ for three different data sets.

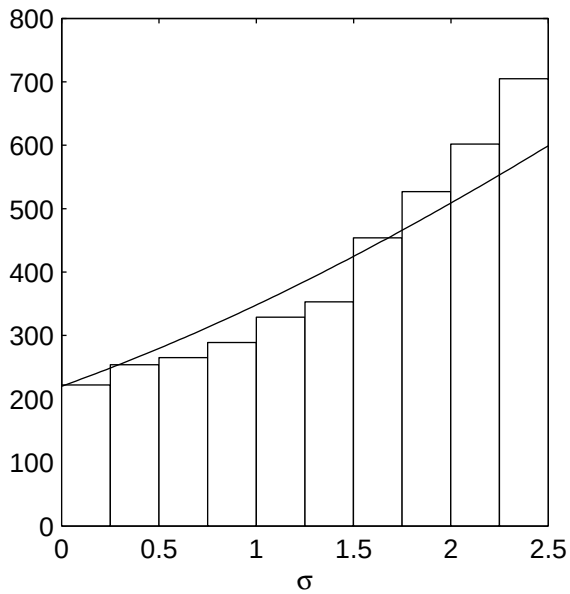


Figure 5.7: The histogram of $P(\sigma)$ where the data set consists of 10 individuals of type 1

In each of the Figures below the histogram of $P(\sigma)$ is compared to the solid line indicating $\mathcal{L}(\sigma)$.

In Figure 5.7, the data set consists of only type 1 individuals.

The same procedure is carried out, where all the leaves are of type 0, i.e. $n_0 = 10$. The corresponding results can be seen in Figure 5.8.

The last data set consists of 4 individuals of type 0. The results can be seen in Figure 5.9.

Since the aim of this project is to detect a signal for the selection parameter σ from a given data set, the results of these debugging procedures were satisfactory at this point and the experiments could then be carried out.

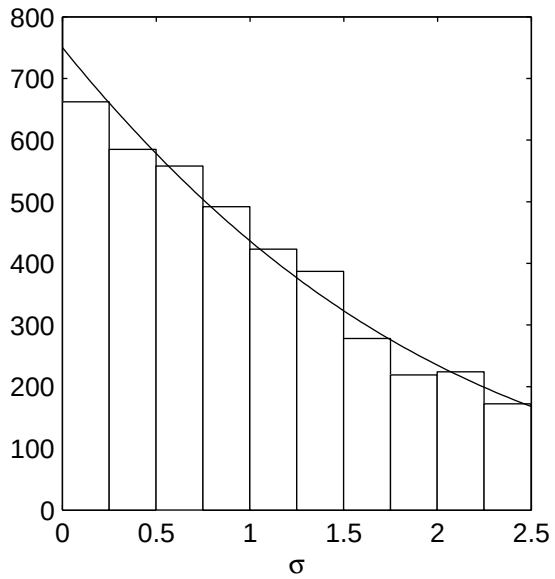


Figure 5.8: The histogram of $P(\sigma)$, where all of the 10 leaves are of type 0

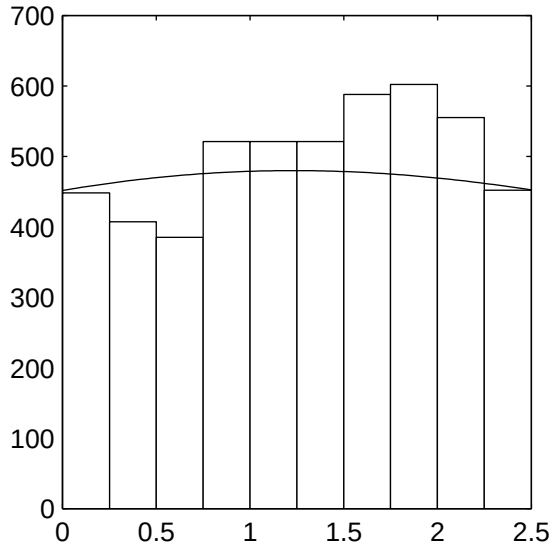


Figure 5.9: The histogram of $P(\sigma)$, where 4 of the 10 leaves are of type 0

Chapter 6

Experiments and Results

The principal aim of this project is to apply a Metropolis-Hastings algorithm using a Bayesian approach to determine how informative existing data is of a selection parameter where the data is taken from a population of organisms.

As with any other computational method a balance between the efficiency and the accuracy of the method should be maintained.

Therefore the experiments are divided into two main classes. One class consists of experiments to investigate the influence of the magnitude of σ on the integrated autocorrelation time, (IACT) and the acceptance rate of the MCMC. The second class of experiments is carried out to determine if existing data is informative of the selection parameter σ. This includes a look at what information is available and in what form. Although for these experiments simulated data is used, the cpu time and general performance of the MCMC was taken into account. The idea is to eventually apply this method to real

data. The latter experiments as well as the data used are discussed in more detail later in this chapter.

From Chapter 2, Section 2.4 the full posterior density is

$$h(g, B_Y, \theta, \sigma | B_I) \propto P(B_I | g, B_Y, \theta, \sigma) f_G(g|\sigma) P(\sigma) P(\theta) \mathbb{I}_{B_I = D}$$

where from Section 2.3 the density $f_G(g|\sigma)$ is

$$f_G(g|\sigma) = \left(\frac{\sigma}{2}\right)^{N_b} \prod_{m=1}^{2N_b+2n-2} e^{-\rho_m(t_m - t_{m-1})}$$

and the likelihood $P(B_I|g, B_Y, \theta, \sigma)$ as defined in Chapter 2, Equation 2.4 is $P(B_I|g, B_Y, \theta, \sigma) = p_{B_{UA}} \prod_{i \in E_g} P_{B_{v_{(i)}}, B_{v(i)}}$.

Since no information is available for the selection parameter σ, the uninformative prior density $P(\sigma) \sim U(0, 2.5)$ is used throughout all the experiments, unless otherwise specified. For the purpose of this thesis, $P(\theta = c) = 1$ where c is some constant.

Let $w(g, B_Y, \sigma, \theta | B_I, Tr) = h(g, B_Y, \sigma, \theta | B_I) \mathbb{I}_{(Tr \bigcap B_I = D)}$, for the indicator function $\mathbb{I}$ of the sub-tree of a graph g equal to Tr, the ancestry of the leaf individuals, B_I. The ancestry Tr is determined by following the rules in Chapter 2, Section 2.3.2.

Let $\Sigma = \{0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1.0,\ 1.2,\ 1.4,\ 1.6,\ 1.8,\ 2.0\}$ and let $k(g, B_Y | \sigma \in \Sigma, B_I) = h(g, B_Y, \sigma, \theta | B_I) \mathbb{I}_{\sigma \in \Sigma}$. The densities f_G and k are used for debugging

purposes and to investigate the influence of the parameter σ on the efficiency of the MCMC, while the densities h and w are of biological importance. For the densities $f_G(g|\sigma)$ and $k(g, B_Y|\sigma, B_I)$ the parameter σ takes values in the set Σ.

Let $\{\Phi_{if}\}$, $\{\Phi_{ih}\}$, $\{\Phi_{ik}\}$ and $\{\Phi_{iw}\}$ denote the Markov chains sampled from the densities $f_G(g|\sigma)$, $h(g, B_Y, \theta, \sigma|B_I)$, $k(g, B_Y|\sigma, B_I)$ and $w(g, B_Y, \sigma, \theta|B_I, Tr)$, respectively.

For the first set of experiments, two types of chains are created, namely separate and continuous chains.

Separate chains are started with exactly the same graph for each value of $\sigma \in \Sigma$. Each Markov chain is updated $8,000,000$ times and every 1000th state is sampled. This results in 11 separate Markov chains.

A continuous chain on the other hand is started with a random graph at $\sigma = 0$. The chain is updated $8,000,000$ times and every 1000th state is sampled. Without interrupting the run, σ is changed to 0.2. The last graph in the chain for $\sigma = 0$ is the first graph in the chain for $\sigma = 0.2$. This process is continued for each value of $\sigma \in \Sigma$. The continuous chain ends, when the $8,000,000^{\text{th}}$ update of $\sigma = 2.0$ is reached. In this case there is one run with $8,000,000,000$ updates and one Markov chain with $88,000$ observations.

During the comparisons of separate Markov chains the seed of the random generator is kept fixed and all aspects of the Markov chains stay the same between the runs.

Let $|V_L|$ denote the number of leaves of a graph g. Comparisons are made between separate Markov chains of variables with density functions $h(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ and $h(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ respectively.

Another set of comparisons is made between separate and continuous Markov chains. For instance the separate Markov chains of random variable $\{\Phi_{ih}^{(s)}\}$ where $\{\Phi_{ih}^{(s)}\} \sim h(B_Y, g|B_I, \sigma, |V_L| = 5)$ are compared to the single continuous Markov chain $\{\Phi_{ih}^{(c)}\}$ sampled from the same density h.

During the debugging process and the experiments interesting new features of graphs have been noticed. These features are as important as the inference of the parameter in question, since these features might influence the efficiency of the MCMC.

One such feature termed bubbles, was picked up during the debugging process and is discussed in full in Chapter 5. Bubbles are formed when the two edges above a branching event, coalesce with each other. Another feature, very tall graphs, called ladders, is discussed in the next section.

6.1 Ladders

6.1.1 $k(B_Y, g|\sigma, B_I)$ and $f_G(g|\sigma)$

During the debugging process, it was noticed that the trace of the time to the ultimate ancestor, t_{UA}, of the Markov chain $\{\Phi_{if}\}$ where $\{\Phi_{if}\} \sim f_G(g|\sigma)$ shows

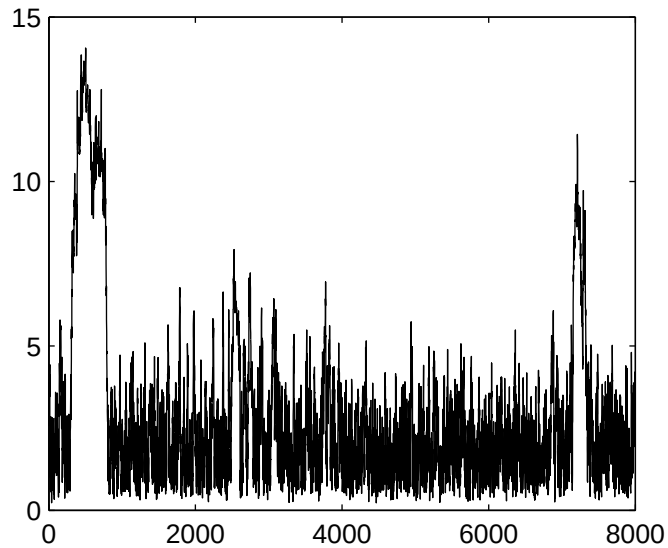


Figure 6.1: The trace of t_{UA} of the Markov chain $\{\Phi_{if}\} \sim f_G(g|\sigma = 1.8, B_I)$. The spikes show the presence of ladders.

spikes at some intervals. These spikes are caused by a steady increase followed by a rapid decrease in the time to the ultimate ancestor, with the period between spikes showing good mixing.

The trace of t_{UA} of the Markov chain $\{\Phi_{if}\} \sim f_G(g|\sigma = 1.8, |V_L| = 10)$ can be seen in Figure 6.1.

This phenomenon can be described as follows. When going from the bottom to the top of a graph, a branching event results into two edges. If these two edges coalesce at the top, both the edges are deletable. However, if an edge branches at the bottom and branches again at the top, the edge is not deletable. If an edge is added between an edge, which is the result of a branching event and some other edge, two branching events follow each other. See

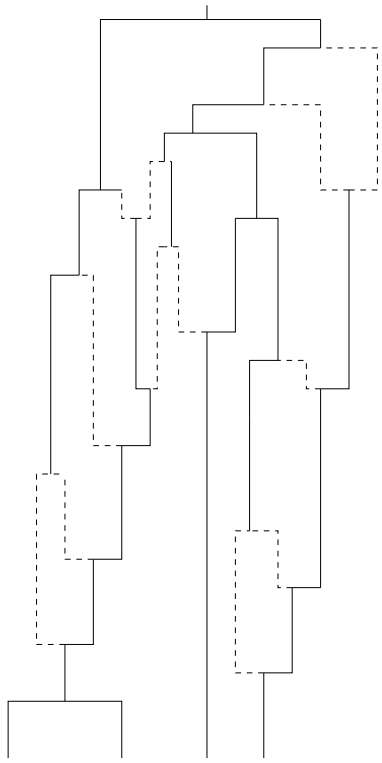


Figure 6.2: Part of a graph showing 10 branching events and 12 deletable edges. The dashed lines indicate the deletable edges.

Figure 6.2. This means that the first branching event results in one edge, which is deletable and one edge which is not deletable. If this happens often, each branching event results in one deletable and one non-deletable edge, instead of the two deletable edges per branching event.

A stair-like or ladder-like graph is created as can be seen on the left and right side of the graph in Figure 6.2. Therefore the apt name for this feature is a ladder. Deletable edges in Figure 6.2 and Figure 6.4 are presented by dashed lines. The graph in Figure 6.2 has 10 branching events and 12 deletable edges.

A ladder is a very tall graph with branching events followed by branching

events and without the presence of bubbles. If at a branching event the resulting edges coalesce with each other to form a bubble, both the resulting edges are deletable.

Since the maximum number of deletable edges in a graph is $2 \times N_b$ where N_b is the number of branching events, and the minimum number of deletable edges is $N_b + 1$ (proof in Appendix A) a statistic can be defined to indicate the presence of a ladder in an MCMC run.

The ladder index, LI is defined as the ratio of the number of branching events, N_b to the number of deletable edges nde, i.e. $LI = \frac{N_b}{nde}$. If a graph has two deletable edges for each branching event the ladder index is $LI = 0.5$. The graph in Figure 6.2 has a ladder index of $LI = \frac{N_b}{nde} = \frac{10}{12} = 0.8333$.

The trace of the ladder index LI, of the Markov chain $\{\Phi_{if}\} \sim f_G(g|\sigma = 1.8, |V_L| = 10)$ can be seen Figure 6.3.

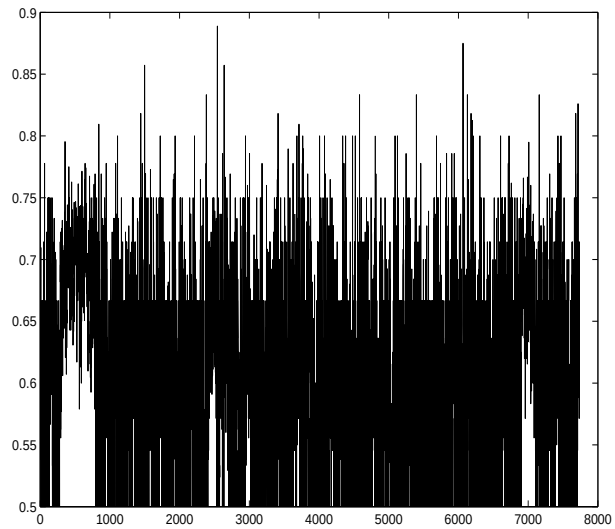


Figure 6.3: The trace of the ladder index,(LI), of the Markov chain $\{\Phi_{if}\} \sim f_G(g|\sigma = 1.8, |V_L| = 10)$

When a graph shows a lot of branching as well as a lot of bubbles the graph will be fatter and less tall as can be seen in Figure 6.4. This graph has a ladder index $LI = \frac{7}{8} = 0.875$. Although the ladder index is higher than the ladder index of the graph in Figure 6.2, the presence of a lot of bubbles makes the graph less ladder like.

Let $\hat{N}_{bk}$ denote the mean number of branching events of the Markov chain $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$. As can be seen in Figure 6.5, $\hat{N}_{bk}$, increases as σ increases from 0 to 2. Let $\hat{nde}_k$ denote the mean number of deletable edges of the Markov chain $\{\Phi_{ik}\}$. The graph in Figure 6.5 shows the $\hat{N}_{bk}$, and the mean number of deletable edges, $\hat{nde}_k$, of the Markov chain $\{\Phi_{ik}\}$. The mean value is on the y-axis and the σ value on the x-axis. The values $\hat{N}_{bk}$ are denoted by $(+)$ and $\hat{nde}_k$ are denoted by $(.)$. The dashed line, $(--)$

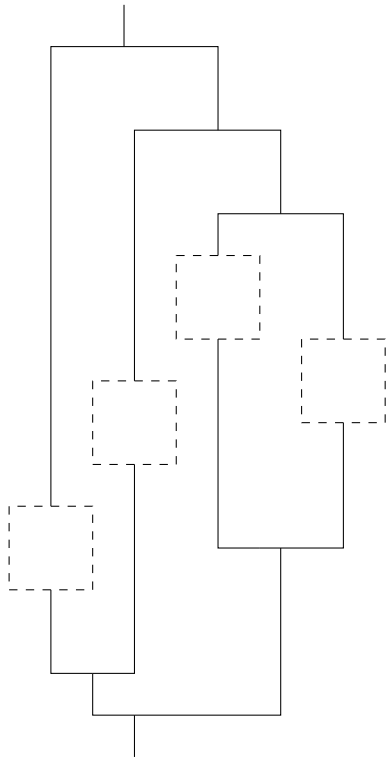


Figure 6.4: Part of a graph with 4 bubbles and $LI = 0.875$

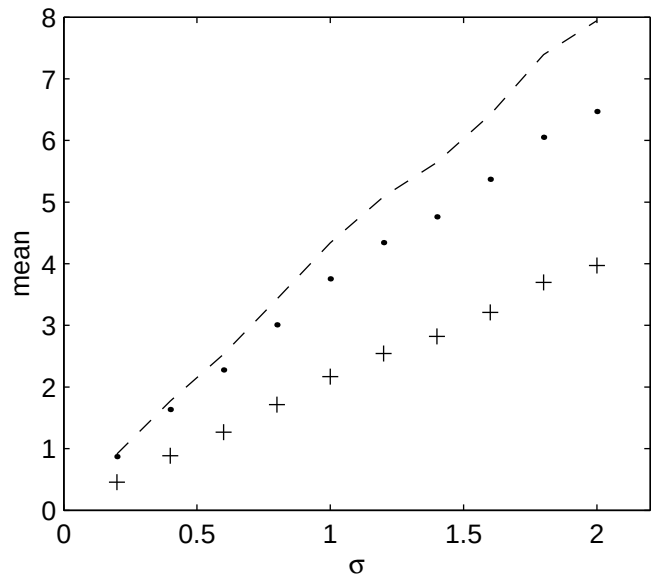


Figure 6.5: The mean number of branching events $\hat{N}_{bk}$ denoted by $(+)$, the mean number of deletable edges, $\hat{nde}_k$ denoted $(.)$ and the upper limit of the number of deletable edges denoted by a dashed line where $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$

shows the upper limit of the number of deletable edges for the given number of branching events. At $\sigma = 0.2$, the maximum possible number of deletable edges is $2 * N_b$. As σ gets bigger, the number of deletable edges becomes much smaller than the maximum number of $2 * N_b$.

The values in Table 6.1 were used to plot the graph in Figure 6.5.

Let $\hat{LI}_k$ and $\hat{LI}_f$ denote the mean ladder indices of the Markov chains, $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma, B_I, |V_L| = 5)$ and $\{\Phi_{if}\} \sim f_G(g|\sigma)$ respectively.

From Figure 6.6 one can see $\hat{LI}_k < \hat{LI}_f$ for $\sigma > 1$.

The preceding graphs were plotted from the values in Table 6.1, which contains $\hat{LI}_k, \hat{nde}_k$, and $\hat{N}_{bk}$ of the separate Markov chains $\{\Phi_k\} \sim k(B_Y, g|\sigma \in$

Density	σ	$\hat{LI}$	$\hat{N}_b$	$\hat{nde}$	$\hat{nbl}$
k	0.2	0.5164	0.4562	0.8705	0.1160
f		0.5152	0.4353	0.8319	0.1054
k	0.4	0.5285	0.8851	1.6349	0.2020
f		0.5295	0.8592	1.5837	0.1814
k	0.6	0.5355	1.2554	2.2780	0.2676
f		0.5429	1.4521	2.5810	0.2931
k	0.8	0.5495	1.7141	3.0085	0.3380
f		0.5568	2.0151	3.4810	0.3791
k	1.0	0.5565	2.1694	3.7544	0.4096
f		0.5644	2.4863	4.2276	0.4426
k	1.2	0.5618	2.5436	4.3440	0.4610
f		0.5778	3.5381	5.8184	0.5763
k	1.4	0.5690	2.8220	4.7599	0.4700
f		0.5865	4.2845	6.9173	0.6765
k	1.6	0.5742	3.2095	5.3705	0.5278
f		0.5998	5.5756	8.8262	0.8311
k	1.8	0.5829	3.6959	6.0525	0.5754
f		0.5996	5.3916	8.5441	0.7635
k	2.0	0.5864	3.9725	6.4689	0.5962
f		0.6070	6.0505	9.480	0.7906

Table 6.1: The mean ladder index $\hat{LI}$, the mean number of branching $\hat{N}_b$, the mean number of deletable edges $\hat{nde}$ and the mean number of bubbles $\hat{nbl}$ of separate Markov chains, $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, |V_L| = 5)$ and $\{\Phi_{ik}\} \sim k(g, B_Y|\sigma \in \Sigma, |V_L| = 5)$

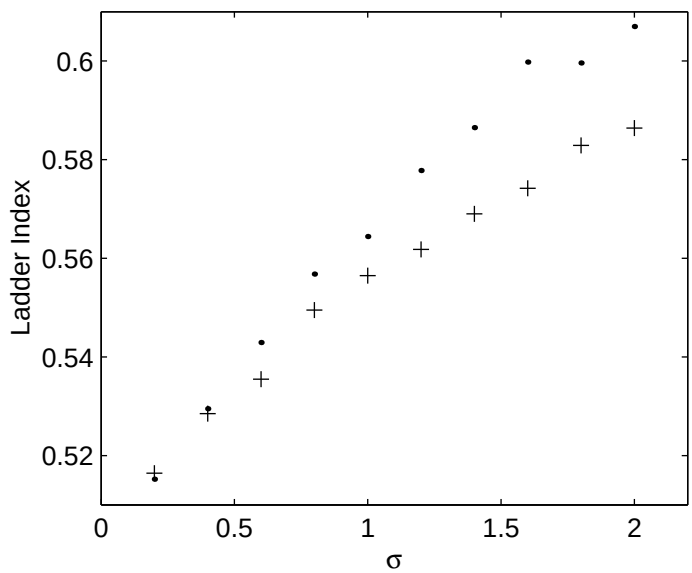


Figure 6.6: The mean ladder index, $\hat{LI}_k$ of $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, |V_L| = 5)$ denoted by $(+)$ and the mean ladder index $\hat{LI}_f$ of $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, |V_L| = 5)$ denoted by $(.)$

$\Sigma, B_I, |V_L| = 5)$.

The number of graphs having a ladder index $LI > 0.75$ for the Markov chain $\{\Phi_{ik}\}$ is between 15 at $\sigma = 0.2$ and 374 at $\sigma = 2$ out of 8000 graphs. These values and the number of graphs having a ladder index $LI > 0.85$ are recorded in Table 6.2. The frequency of graphs showing branching events of the Markov chain $\{\Phi_{ik}\}$ and the Markov chain $\{\Phi_{if}\}$ can be seen in the last column of Table 6.2. These frequencies are compared in Figure 6.7 where the frequency of graphs with branching events of the Markov chain $\{\Phi_{ik}\}$ is denoted by (.) and the frequency of the graphs showing branching events of the chain $\{\Phi_{if}\}$ is denoted by (+). As the value of σ increases the number of graphs with branching events increases but there is no visual difference between the chains $\{\Phi_{ik}\}$ and

Density	σ	$LI \geq 0.75$	$LI \geq 0.85$	$\max(LI)$	Freq.
k	0.2	15	0	0.8000	0.3298
f		17	0	0.8000	0.3206
k	0.4	61	0	0.8333	0.5076
f		73	0	0.8333	0.4956
k	0.6	105	1	0.8750	0.6110
f		135	2	0.8571	0.6306
k	0.8	175	1	0.8750	0.6893
f		192	2	0.8750	0.7138
k	1.0	193	5	0.8750	0.7540
f		209	3	0.8571	0.7664
k	1.2	232	10	0.8889	0.7973
f		282	5	0.8889	0.8190
k	1.4	228	7	0.8571	0.8111
f		327	9	0.8889	0.8521
k	1.6	270	11	0.8750	0.8429
f		385	8	0.8889	0.8885
k	1.8	341	12	0.9000	0.8594
f		386	10	0.88890	0.8903
k	2.0	374	16	0.9000	0.8698
f		435	11	0.8750	0.8968

Table 6.2: The number of graphs having a ladder index $LI \geq 0.75$ and a ladder index $LI \geq 0.85$. The maximum ladder index $\max(LI)$ and the frequency of graphs showing branching events of the Markov chains $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, |V_L| = 5)$ and $\{\Phi_{ik}\} \sim k(g, B_Y|\sigma \in \Sigma, |V_L| = 5)$

$\{\Phi_{if}\}$.

Another important aspect of the Markov chain is the rate at which the proposed new states are being accepted. Roberts [20] has shown that an acceptance rate of $10 - 40$ percent is good for a Gaussian distribution. As expected, the acceptance rate of the Markov chain $\{\Phi_{if}\}$ where $\{\Phi_{if}\} \sim f_G(g|\sigma, |V_L| = 5)$ is much higher than the acceptance rate of the Markov chain $\{\Phi_{ik}\}$ where

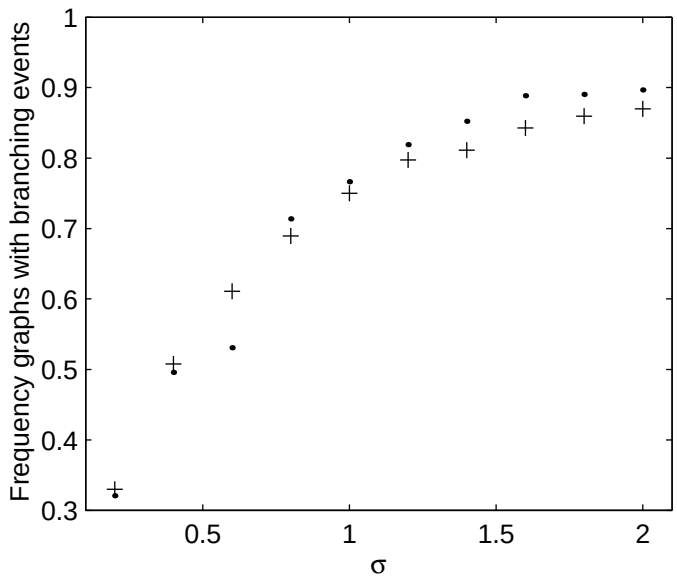


Figure 6.7: The frequency of graphs with branching events. $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, |V_L| = 5)$ is denoted by $(.)$ and of $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma)$ is denoted by $(+)$.

$\Phi_{ik} \sim k(B_Y, g|\sigma, B_I, |V_L| = 5)$. This can be seen in the graph in Figure 6.8 where the total acceptance rate of $\{\Phi_{ik}\}$ is denoted by $(+)$ and the acceptance rate of $\{\Phi_{if}\}$ is denoted by $(.)$. The total acceptance rate can be seen in Table 6.3.

In Table 6.3 the acceptance rate of the Markov chain $\{\Phi_{ik}\}$ where $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma, B_I, |V_L| = 5)$ is compared to the acceptance rate of the Markov chain $\{\Phi_{if}\}$ where $\Phi_{if} \sim f_G(g|\sigma, |V_L| = 5)$. The acceptance percentage is broken into the acceptance rate for the different moves. As can be seen from Table 6.3 the acceptance rate for the deleting and adding moves is considerably lower for the Markov chains $\{\Phi_{ik}\}$.

The ladders have an influence on the mean time to the ultimate ancestor

Density	σ	Delete	Add	Time	Type	Gate	Overall
k	0.2	0.41	0.41	11.64	15.50	5.45	33.39
f		1.29	1.29	37.16			39.74
k	0.4	0.65	0.65	11.76	15.25	8.37	36.67
f		2.10	2.10	37.83			42.03
k	0.6	0.82	0.82	11.80	15.04	10.21	38.69
f		2.69	2.69	38.42			43.80
k	0.8	0.94	0.94	11.83	14.91	11.57	40.19
f		3.10	3.10	38.97			45.17
k	1.0	1.04	1.04	11.83	14.70	12.54	41.15
f		3.43	3.43	39.45			46.31
k	1.2	1.1	1.1	11.82	14.54	13.10	41.66
f		3.66	3.66	39.88			47.20
k	1.4	1.16	1.16	14.54	13.53	11.88	42.27
f		3.85	3.85	40.30			48.00
k	1.6	1.21	1.21	11.90	14.44	14.05	42.81
f		3.98	3.98	40.61			48.57
k	1.8	1.23	1.23	11.90	14.37	14.32	43.05
f		4.13	4.13	40.97			49.23
k	2.0	1.27	1.27	11.90	14.30	14.53	43.27
f		4.22	4.22	41.30			49.74

Table 6.3: Acceptance rate,(%) of $\{\Phi_{if}\} \sim f_G(g, \sigma \in \Sigma, |V_L| = 5)$ and $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, |V_L| = 5)$

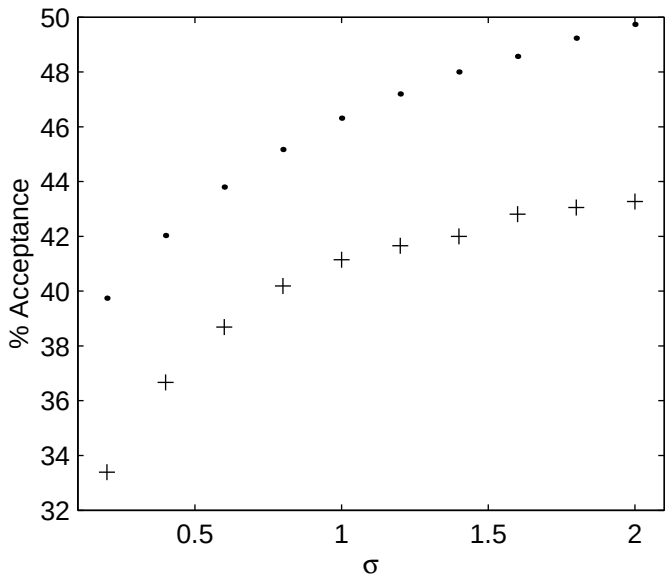


Figure 6.8: The acceptance rate $(\%)$ of $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(+)$ and the acceptance rate, $(\%)$ of $\{\Phi_{if}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(.)$

and the efficiency of the MCMC. The cause of the ladders is still unknown and the problem persists although two more operators were introduced to reduce the ladders. These two operators are the moves changing the selection type and the time at the internal vertices.

The addition of selection types at the internal vertices reduced the mean ladder index per run, but the number of graphs showing ladders is higher than the number of graphs showing ladders without the types at the vertices.

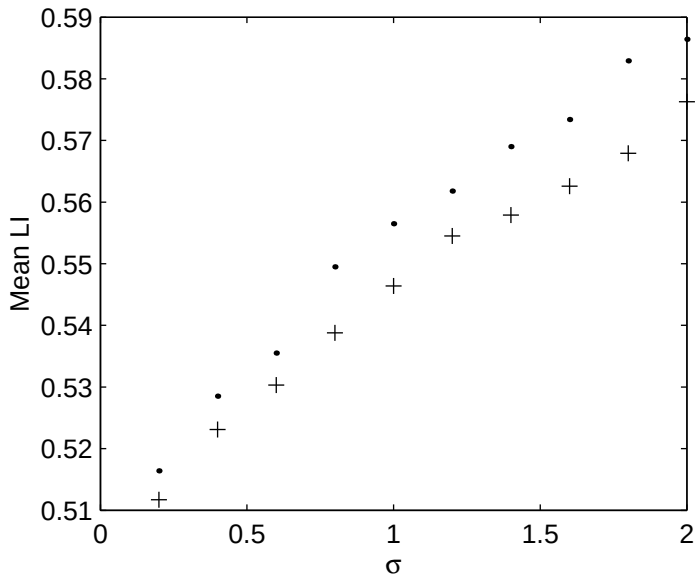


Figure 6.9: The mean ladder index, $\hat{LI}_k^{(5)}$ of $\{\Phi_{ik}^{(5)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (.) and the mean ladder index $\hat{LI}_k^{(10)}$ where $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ is denoted by $(+)$

6.1.2 $|V_L| = 5$ and $|V_L| = 10$

The influence of the number of leaves on the ladder index LI is investigated by creating and comparing the Markov chain $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ to the Markov chain $\{\Phi_{ik}^{(5)}\}$ with the Markov chain $\{\Phi_{ik}^{(5)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$. Separate chains are created for each value $\sigma \in \Sigma$.

The mean ladder index $\hat{LI}_k^{(10)}$ of the Markov chain $\{\Phi_{ik}^{(10)}\}$ is lower than the mean ladder index $\hat{LI}_k^{(5)}$ of the Markov chain $\{\Phi_{ik}^{(5)}\}$ as can be seen in Figure 6.9.

This interesting fact is further investigated by plotting the mean number of branching events $\hat{N}_{bk}^{(10)}$ and the mean number of deletable edges $\hat{nde}_k^{(10)}$ of the Markov chain $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ for the different values of

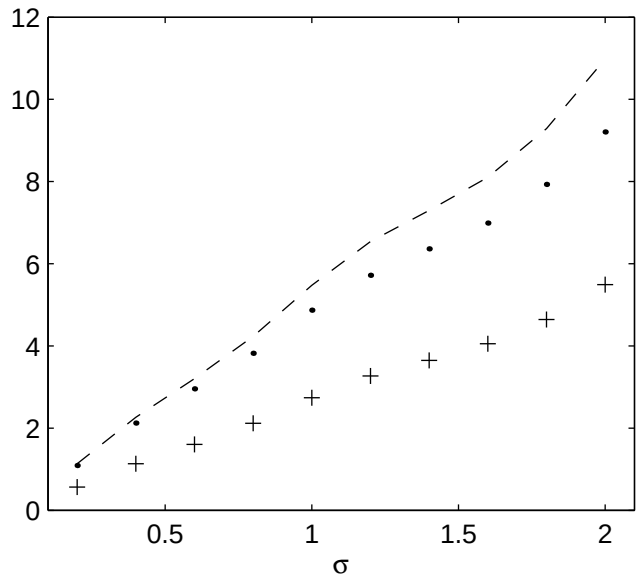


Figure 6.10: The mean number of branching events $\hat{N}_{b_k}^{(10)}$ denoted by $(+)$, the mean number of deletable edges, $\hat{nde}_k^{(10)}$ denoted by $(.)$ and the upper limit of the number of deletable edges by $(--)$ of the Markov chain $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$.

$\sigma \in \Sigma$. In Figure 6.10 $\hat{N}_{b_k}^{(10)}$ is denoted by (+) and $\hat{nde}_k^{(10)}$ is denoted by (.). The dashed line shows the upper limit of the number of deletable edges, $2 * N_b$.

The values used to plot the previous graphs can be seen in Table 6.4.

The acceptance rate for MCMC sampling of the Markov chain $\{\Phi_{ik}^{(5)}\}$ from the density $k(B_Y, g|\sigma, B_I, |V_L| = 5)$ is lower than the acceptance rate of the Markov chain $\{\Phi_{ik}^{(10)}\}$ sampled from the density $k(B_Y, g|\sigma, B_I, |V_L| = 10)$. The acceptance rates are compared in Figure 6.11 where the acceptance rate of the chain $\{\Phi_{ik}^{(10)}\}$ is denoted by $(+)$ and the acceptance rate of $\{\Phi_{ik}^{(5)}\}$ is denoted by $(.)$.

Density	σ	$\hat{LI}$	$\hat{N}_b$	$\hat{nde}$
k	0.2	0.5164	0.5655	1.0939
k	0.4	0.5285	1.1321	2.1227
k	0.6	0.5355	1.6025	2.9540
k	0.8	0.5495	2.1157	3.8199
k	1.0	0.5565	2.7389	4.8671
k	1.2	0.5618	3.2700	5.7176
k	1.4	0.5690	3.6496	6.3631
k	1.6	0.5734	4.0512	6.9917
k	1.8	0.5679	4.6445	7.9313
k	2.0	0.5763	5.4893	9.2069

Table 6.4: The mean ladder index, $\hat{LI}$, the mean number of branching, $\hat{N}_b$ and the mean number of deletable edges $\hat{nde}$ of Markov chain $\{\Phi_{ik}\}$ where $\{\Phi_{ik}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$

The acceptance rates for the MCMC of $\{\Phi_{ik}^{(10)}\}$ can be seen in Table 6.5.

The frequency of graphs with branching events in the Markov chain $\{\Phi_{ik}^{(10)}\}$ is higher than the frequency of graphs with branching events in the Markov chain $\{\Phi_{ik}^{(5)}\}$. This can be seen in Figure 6.12 and Table 6.2 and Table 6.6.

From the previous results the mean ladder index, $\hat{LI}_k^{(10)}$ and the number of graphs for which $LI \geq 0.75$ and $LI \geq 0.85$ are lower for the Markov chain $\{\Phi_{ik}^{(10)}\}$ than for the Markov chain $\{\Phi_{ik}^{(5)}\}$. The acceptance rate of the Markov chain $\{\Phi_{ik}^{(10)}\}$ is higher than the acceptance rate of the Markov chain $\{\Phi_{ik}^{(5)}\}$. This means that taking larger samples of individuals at the leaves of the graphs, D from a population to create an ancestral selection improves the efficiency of the MCMC.

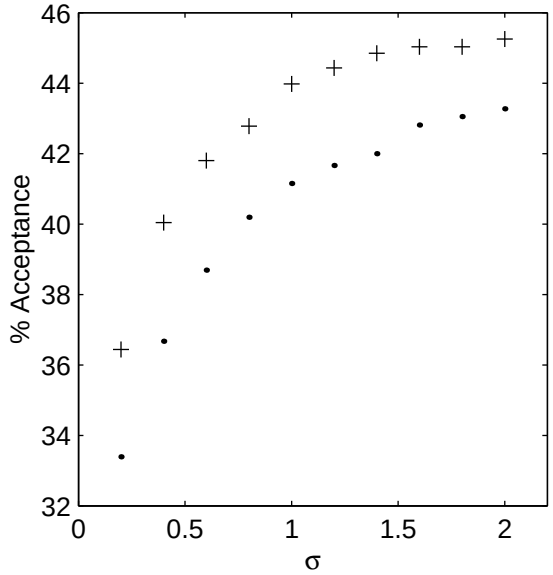


Figure 6.11: Acceptance rates $(\%)$ of $\{\Phi_{ik}^{(10)}\}$ denoted by $(+)$ and $\{\Phi_{ik}^{(5)}\}$ denoted by $(.)$

Density	σ	Delete	Add	Time	Type	Gate	Overall
k	0.2	0.29	0.29	13.30	15.87	6.69	36.44
k	0.4	0.48	0.48	13.23	15.65	10.20	40.04
k	0.6	0.62	0.62	13.06	15.33	12.17	41.80
k	0.8	0.73	0.73	12.89	15.05	13.38	42.78
k	1.0	0.84	0.84	12.91	15.00	14.39	43.98
k	1.2	0.91	0.91	12.87	14.85	14.89	44.43
k	1.4	0.97	0.97	12.85	14.78	15.28	44.85
k	1.6	1.02	1.02	12.77	14.63	15.58	45.02
k	1.8	1.07	1.07	12.65	14.45	15.79	45.03
k	2.0	1.10	1.10	12.63	14.41	16.01	45.25

Table 6.5: The acceptance rate,$(\%)$, of the Markov chain $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ broken in the different moves used for th MCMC

$\|V_L\|$	σ	$LI \geq 0.75$	$LI \geq 0.85$	$\max(LI)$	Freq
5	0.2	15	0	0.8	0.3298
10		18	0	0.8	0.4064
5	0.4	61	0	0.8333	0.5076
10		42	0	0.8333	0.6156
5	0.6	105	1	0.8750	0.6110
10		64	0	0.8333	0.7248
5	0.8	175	1	0.8750	0.6893
10		105	3	0.8571	0.7975
5	1.0	193	5	0.8750	0.7540
10		128	2	0.8511	0.8614
5	1.2	232	10	0.8889	0.7973
10		135	4	0.8511	0.8891
5	1.4	228	7	0.8571	0.8111
10		150	0	0.8333	0.9215
5	1.6	270	11	0.8750	0.8429
10		141	3	0.8750	0.9269
5	1.8	341	12	0.9000	0.8594
10		158	1	0.9000	0.9480
5	2.0	374	16	0.9000	0.8698
10		173	6	0.8750	0.9599

Table 6.6: The number of graphs, showing a ladder index $LI \geq 0.75$ and $LI \geq 0.85$. The maximum ladder index $\max(LI)$ and the frequency of graphs showing branching events where $\{\Phi_{ik}\}$ is sampled form $k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ and $k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ respectively.

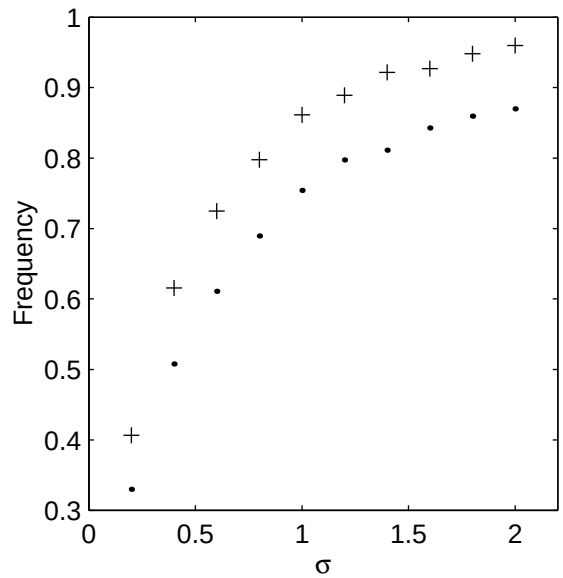


Figure 6.12: Proportion of graphs showing branching events, $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ denoted by $(+)$ and $\{\Phi_{ik}^{(5)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(.)$

One can conclude from these results that a larger sample from the population, i.e. more leaves at the bottom of the graph, creates less ladders and a higher acceptance rate.

6.1.3 Continuous versus separate Markov chains

When visually inspecting the graph of the mean ladder index, $\hat{LI}_f^{(s)}$, of the separate Markov chains $\{\Phi_{if}^{(s)}\}$ and the continuous Markov chain, $\{\Phi_{if}^{(c)}\}$ in Figure 6.13, there is no difference between the ladder indexes of the two Markov chains.

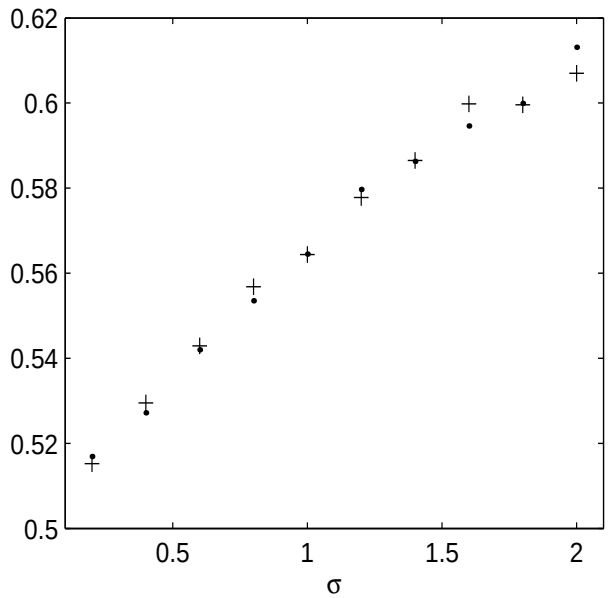


Figure 6.13: The mean ladder index $\hat{LI}_f$ of continuous Markov chain $\{\Phi_{if}^{(c)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (.) and the separate Markov chains $\{\Phi_{if}^{(s)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by $(+)$

The frequency of graphs showing branching events is the same for the separate Markov chains and the continuous Markov chains as can be seen in Figure 6.14.

The acceptance rate for the separate and the continuous Markov chains are the same as can be seen in Figure 6.15.

From these results it seems that by sampling the Markov chains $\{\Phi_i\}$ separately for each value of $\sigma \in \Sigma$ or by sampling one Markov chain continuously for all the values of $\sigma \in \Sigma$ will give the same results.

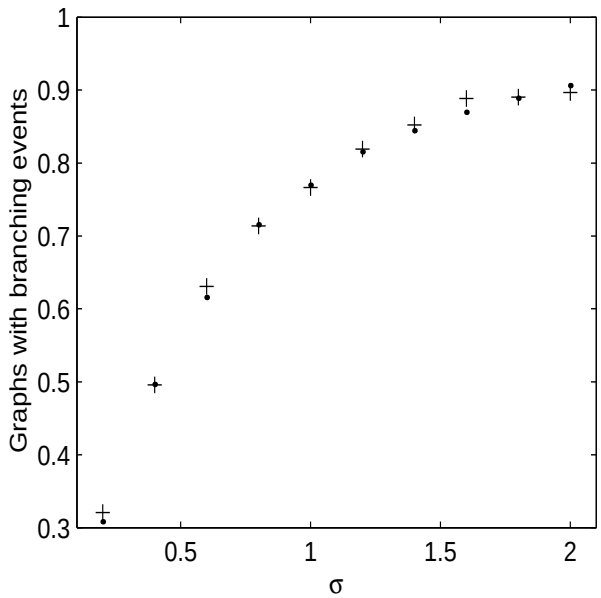


Figure 6.14: Frequency of graphs showing branching events where the continuous Markov chain $\{\Phi_{if}^{(c)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ is denoted by $(.)$ and the separate Markov chains $\{\Phi_{if}^{(s)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ is denoted by $(+)$

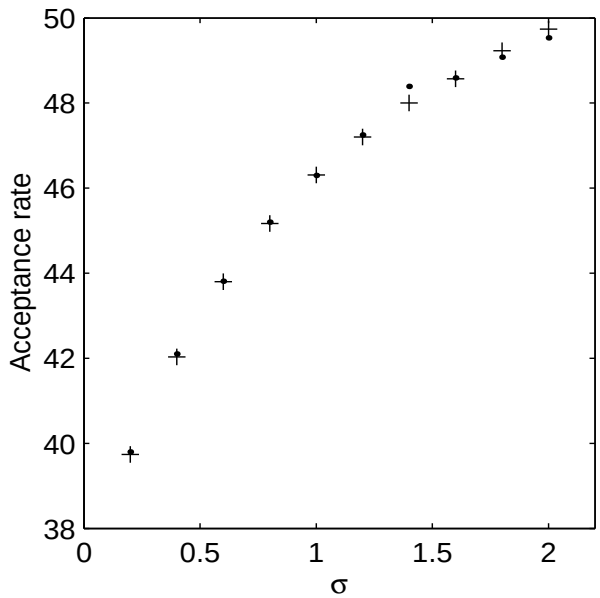


Figure 6.15: The acceptance rates, (%) of the continuous Markov chain $\{\Phi_{if}^{(c)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (.) and of the separate Markov chains $\{\Phi_{if}^{(s)}\} \sim f_G(g|\sigma \in \Sigma, B_I, |V_L| = 5)$ denoted by (+)

6.2 Integrated Autocorrelation Time

Let $\{\Phi_i\} = [g_i, B_{Yi}, \sigma_i] \in \Gamma$ be a sequence of random variables converging to the stationary posterior density $h(g, B_Y, \sigma|B_I)$ on the space $\tilde{\Gamma}$ and let ζ be a statistic of interest. The function ζ can be the time to the ultimate ancestor, t_{UA}, the number of branching events, N_b, in a graph or any other statistic of interest.

Since the variables are the output of an MCMC, the samples are highly correlated. The size of the error caused by estimating the parameters by using MCMC, is given by the central limit theorem adapted for correlated samples. In the correlated version of the central limit theorem, the difference between the estimated value of the mean of the function ζ and the mean itself converges in distribution to the normal distribution with mean 0 and variance $\tau_\zeta \text{var}_\zeta$ where τ_ζ is the integrated autocorrelation time (IACT) of the function ζ. The variance $\tau_\zeta \text{var}_\zeta$ can be estimated from the Markov chain itself.

The variance of the estimate $\hat{\zeta}$ from a correlated sample of size N is a factor τ_ζ larger than the variance of the estimate $\hat{\zeta}$ of independent samples. In order to reduce the variance of the estimator of correlated samples, for a fixed sample of size N, the aim is to get τ_ζ as small as possible.

Define the auto-covariance of $\{\Phi_i\}$ for the function ζ at lag s as $\gamma_\zeta(s) = \text{cov}(\zeta(\Phi_i), \zeta(\Phi_{i+s}))$. The integrated autocorrelation time $\tau_\zeta = \gamma_\zeta(2s) + \gamma_\zeta(2s + 1)$, the sums of adjacent pairs of auto-covariances as defined by Geyer [10]. According to Geyer [10] the function τ_ζ is strictly positive, strictly decreasing

and strictly convex. Let $\hat{\text{var}}_\zeta = -\gamma_0 + 2\sum_{i=0}^{M_\zeta} \tau_{i,M_\zeta}$ be an estimate of the variance var_ζ where M_ζ is the largest integer such that $\tau_{i,M_\zeta} > 0$.

The advantage of MCMC is that many different statistics can be investigated in one run. During all the experiments, the influence of different changes to the graphs and in particular the influence of the parameter σ on the IACT τ is investigated. The error bars on the graphs are calculated by using the formula by Sokal [22], $\text{var}(\hat{\tau}_\zeta) \approx \frac{2(2M+1)}{N}\tau_\zeta^2$.

The four statistics under investigation are the time to the ultimate ancestor, t_{UA}, the time to the first event, t_{FE}, the number of branching events, N_b and the ladder index, LI.

In the following experiment the Markov chain $\{\Phi_{ik}\}$ is sampled from the density $k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$. The Markov chain was updated $8,000,000$ times and every 1000th graph was sampled to give a sample of size 8000. For each value of σ, the chain was started at the exactly the same graph. The value $\sigma = 0$ is not included in the tables since no branching events happens in these graphs. Although t_{UA} can be recorded the number of branching events is equal to zero and the ladder index does not exist. For each statistic, ζ and each value of σ the value of τ_ζ, M_ζ and $\text{var}(\tau_\zeta)$ is recorded in a table followed by a plot of the values of τ_ζ on the y-axis against the σ values on the x-axis.

In each of the examples the IACT, the variance of the IACT and the cutting off value M are recorded in the respective tables for each statistic. The table is

σ	τ	var(τ)	M
0.2	1011.5	0.0102	$2,000$
0.4	1214.1	0.0221	$3,000$
0.6	1556.5	0.0485	$4,000$
0.8	2697.5	0.3638	$10,000$
1.0	2652.9	0.2463	$7,000$
1.2	2781.3	0.3481	$9,000$
1.4	3112.2	0.3390	$7,000$
1.6	3329.7	0.3881	$7,000$
1.8	5250.7	1.7921	$13,000$
2.0	5399.9	1.6038	$11,000$

Table 6.7: The integrated autocorrelation time τ, the variance of IACT var(τ) and the cutting off value M of the ladder index, LI. The variance is $\times 1.0e+008$

followed by a plot of the IACT on the y-axis and the values of σ on the x-axis. The IACT values on the $y-$axis need to be multiplied by $1,000$. The purpose of the figures is only to get a feel of the influence of the magnitude of σ on the IACT of each statistic.

The first table, Table 6.7 records the values of τ_{LI}, var(τ_{LI}) and M_{LI} of the ladder index, LI for the Markov chain $\{\Phi_i\} \sim h(g, B_Y|\sigma \in \Sigma, B_I, |V_L| = 5)$. A graph of the IACT, τ_{LI} with error bars for the ladder index can be seen in Figure 6.16.

The next table, Table 6.8, records the IACT, $\tau_{t_{UA}}$, the variance of IACT var($\tau_{t_{UA}}$) and the cutting off value, $M_{t_{UA}}$. The graph of the IACT $\tau_{t_{UA}}$ on the y-axis and the values of σ on the x-axis can be seen in Figure 6.17.

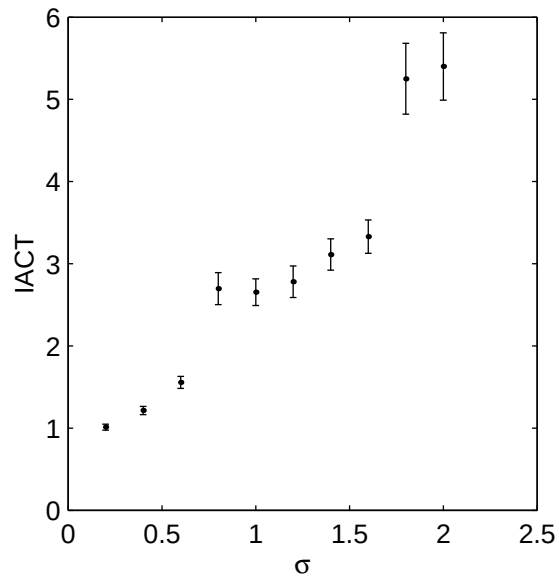


Figure 6.16: The IACT, τ of the ladder index on the y-axis and $\sigma \in \Sigma$ on the x-axis.

The Table 6.9 contains the IACT, $(\tau_{t_{FE}})$, the variance of IACT, var(τ) and the $M_{t_{FE}}$ value of the time to the first event,(t_{FE})).

In Figure 6.18 is a plot of the IACT, (τ) on y-axis and σ on the x-axis of t_{FE}.

Table 6.10 contains the IACT, $\tau_{t_{MRCA}}$, the variance of IACT, var$(\tau_{t_{MRCA}})$ and $M_{t_{MRCA}}$ of the time to the most recent common ancestor, $(t_{(MRCA)})$.

In Table 6.11 the IACT τ_{N_b}, the variance of IACT, var(τ_{N_b}) and M_{N_b} are recorded, followed by the graph of τ on the y -axis and σ on the x-axis of the number of branching events, N_b.

Let $\{\Phi_{ik}^{(10)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 10)$ and $\{\Phi_{ik}^{(5)}\} \sim k(B_Y, g|\sigma \in \Sigma, B_I, |V_L| = 5)$.

The statistic t_{UA} of the Markov chain $\{\Phi_{ik}^{(10)}\}$ is more correlated than the

σ	τ	var(τ)	M
0.2	2954.1	0.0022	$5,000$
0.4	5488.4	0.0181	$12,000$
0.6	6354.7	0.0242	$12,000$
0.8	11730.7	0.1032	$15,000$
1.0	14069.3	0.1584	$16,000$
1.2	18819.5	0.3188	$18,000$
1.4	13090.3	0.1371	$16,000$
1.6	17192.9	0.3252	$22,000$
1.8	33421.5	2.0665	$37,000$
2.0	32926.1	1.5178	$28,000$

Table 6.8: The IACT τ, the variance of IACT var(τ) and the cutting off value M of t_{UA}. The variance is $\times 1.0e + 0.010$

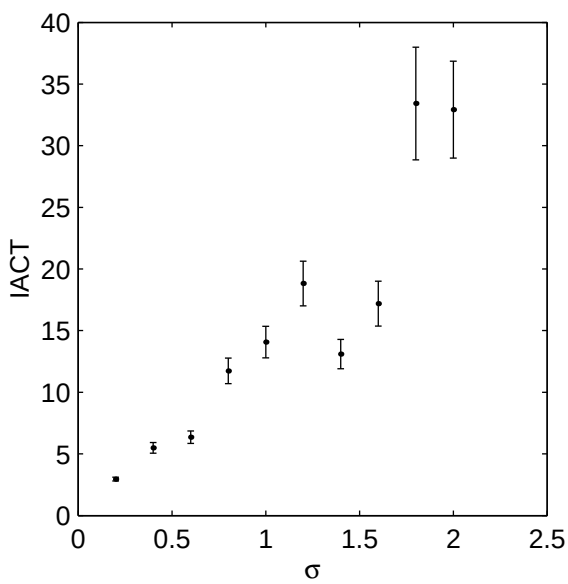


Figure 6.17: The IACT on y-axis and σ on x-axis of t_{UA}.

σ	τ	var(τ)	M
0.2	2111.7	1.3379	6,000
0.4	1879.6	1.4132	8,000
0.6	1998.0	1.1977	6,000
0.8	1674.2	0.7008	5,000
1.0	1734.5	0.6018	4,000
1.2	2022.1	0.8179	4,000
1.4	1934.1	0.9353	5,000
1.6	2271.5	1.2901	5,000
1.8	2894.3	3.3510	8,000
2.0	2371.9	2.2505	8,000

Table 6.9: The IACT, (τ), the variance of IACT, (var(t_{FE})) and the value M of t_{FE}. The variance is $\times 1.0e + 007$

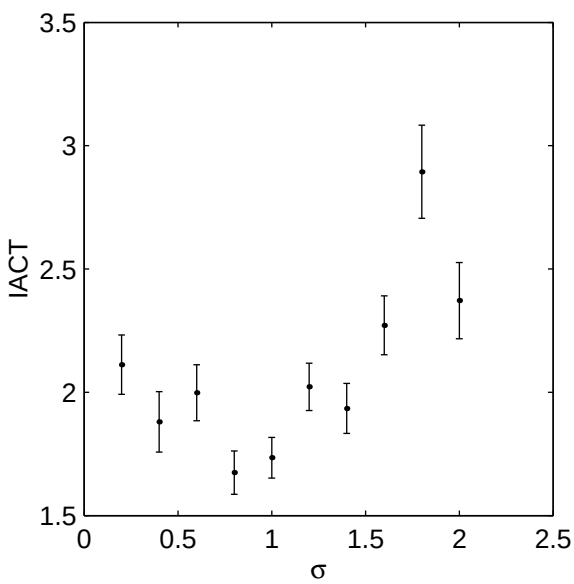


Figure 6.18: The IACT on the y-axis, with σ on x-axis of t_{FE}.

σ	τ	var(τ)	M
0.2	2942.9	0.0217	$5,000$
0.4	4092.9	0.0921	$11,000$
0.6	4324.5	0.1029	$11,000$
0.8	7358.5	0.4873	$18,000$
1.0	7267.3	0.2905	$11,000$
1.2	7752.4	0.3005	$10,000$
1.4	7704.6	0.5046	$17,000$
1.6	8274.1	0.4108	$12,000$
1.8	14492.2	2.3103	$22,000$
2.0	13469.0	1.5421	$17,000$

Table 6.10: The IACT, τ, the variance of IACT, var(τ) and M of t_{MRCA}. The variance is $\times 1.0e + 009$

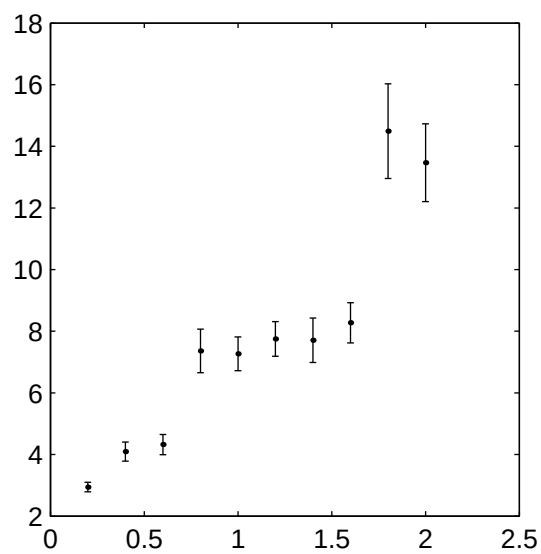


Figure 6.19: IACT on y-axis and σ on x-axis of t_{MRCA}.

σ	τ	var(τ)	M
0.2	1685.1	0.0114	8,000
0.4	2866.2	0.0329	8,000
0.6	3852.6	0.0520	7,000
0.8	7184.8	0.3614	14,000
1.0	9217.6	0.7223	17,000
1.2	13174.8	1.3019	15,000
1.4	9234.9	0.5544	13,000
1.6	11944.7	1.2841	18,000
1.8	20554.4	5.4924	26,000
2.0	25713.3	9.2566	28,000

Table 6.11: The IACT, τ, the variance of IACT, var(τ) and M of N_b. The variance is $\times 1.0e + 007$

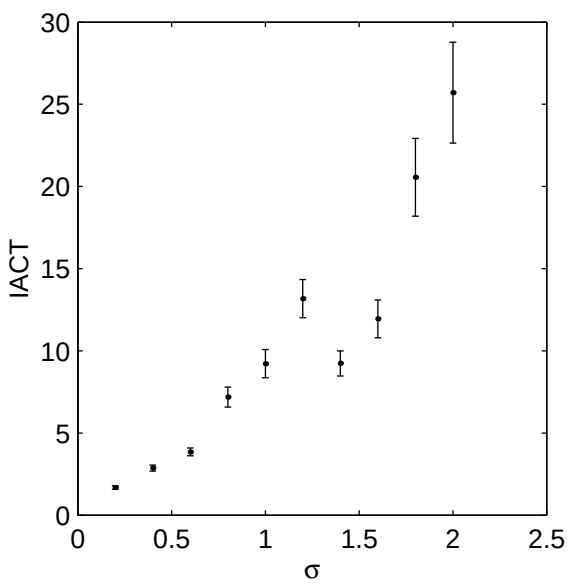


Figure 6.20: The IACT on the y-axis and σ on the x-axis of N_b

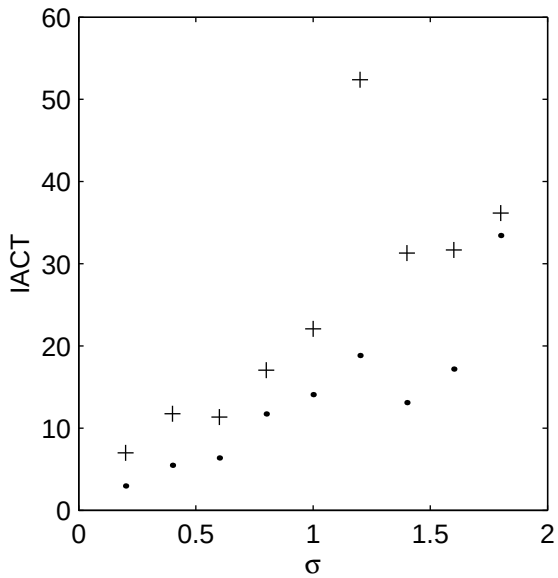


Figure 6.21: The IACT of t_{UA} of the Markov chain $\{\Phi_{ik}^{(10)}\}$ denoted by $(+)$ and of the Markov chain $\{\Phi_{ik}^{(5)}\}$ denoted by $(.)$

statistic t_{UA}, of the Markov chain $\{\Phi_{ik}^{(5)}\}$ as can be seen in Figure 6.21.

It can be seen from Figure 6.22 that the statistic LI of the Markov chain $\{\Phi_{ik}^{(10)}\}$ is also more correlated than the statistic LI of the Markov chain $\{\Phi_{ik}^{(5)}\}$.

As soon as $\sigma > 1$ the variance of IACT, var(τ) is increasing for all the different statistics. The IACT of the statistics for $\sigma > 1$ tends to be smaller than the IACT when $\sigma < 1$ before it starts to increase again. Therefore the accuracy of results when $\sigma > 1$ should be questioned.

Although the acceptance rate of the Markov chain $\{\Phi_{ik}^{(10)}\}$ is higher than the acceptance rate of the Markov chain $\{\Phi_{ik}^{(5)}\}$, the different statistics of the Markov chain $\{\Phi_{ik}^{(10)}\}$ are more correlated than the same statistics of the Markov chain $\{\Phi_{ik}^{(5)}\}$.

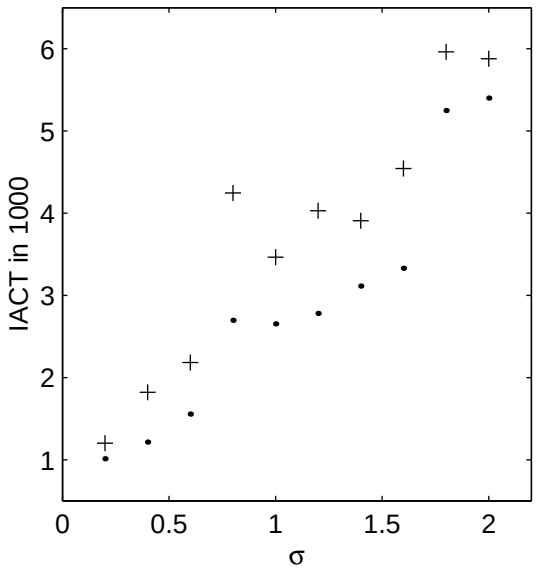


Figure 6.22: The IACT of the ladder index, τ_{LI}, of the Markov chain $\{\Phi_{ik}^{(10)}\}$ denoted by $(+)$ is more correlated than τ_{LI} of the Markov chain $\{\Phi_{ik}^{(5)}\}$ denoted by $(.)$

Therefore, in the quest to lower the variance, var(τ), the MCMC should draw more samples from the posterior density, i.e. a longer Markov chain and sample, the graphs further apart.

In conclusion, as can be seen in Section 6.1.2, the influence of ladders on the Markov chain is reduced when there are more leaves on a graph, i.e a bigger sample is taken form the population of organisms. The acceptance rate of the Markov chain $\{\Phi_i\}$, of graphs with more leaves is also higher than the acceptance rate of the Markov chain of graphs with less leaves. Unfortunately the samples within a Markov chain are more correlated when a bigger sample is taken from the population. This can be overcome by drawing more samples

from the density to create a longer Markov chain and to draw the samples further apart, i.e. instead of sampling each $1,000^{\text{th}}$ graph, sample each $10,000^{\text{th}}$ graph. This should be done taking computer time and computer memory into account.

6.3 The model parameter σ

Biologists are interested in determining a selection parameter σ where a group of organisms shows directional selection. In this section two different schemes are implemented in determining the selection parameter σ for simulated data, $B_I = D$.

6.3.1 Predetermined genealogy Tr

Computer software is available to determine the ancestry of a group of organisms. The first method incorporates the known ancestry in the determination of the selection parameter σ.

Each ancestral selection graph g, contains an ancestry of the leaf types which can be determined by following the rules in Chapter 2, Section 2.3.2. Let $\{\Phi_{iw}\}$ be the Markov chain sampled from the density $w(B_I, g, \sigma|B_I, Tr)$ where Tr is the ancestry of the organisms at the leaf tips. The Markov chain $\{\Phi_{iw}\}$ is created by using the operators as discussed in Chapter 3. The predetermined ancestral tree Tr is kept fixed throughout the updates. This is done by simply

rejecting the graphs which do not contain the tree Tr.

Let $|V_L|$ be the number of leaves of a graph g. The leaves represent the sample of individuals taken from a population of organisms. Since the population consists only of individuals of two types, the types of the vertices at the leaf tips are only zeros and ones. Let D be the vector denoting the types of the individuals in the sample. For instance $D(1)$ contains a zero or one according to the type of the vertex V with label 1, $D(2)$ contains the type of the vertex V with label 2, etc.

As before let

$$\chi_{V,E} = \{t_A; t_A \in [R_0, \infty)^{N_b+N_c} \text{ and for } \langle i, j\rangle \in E_g, t_i < t_j\},$$

(Chapter 2, Section 2.3) be the vector containing the times associated with the vertices of a graph, g, in ascending order. Therefore let $\chi_{V,E}^{(Tr)}$ be the vector containing the times of the ancestral tree.

Experiments were carried out using samples of 10 individuals from a population, i.e. $|V_L| = 10$. The simulated data consists of a known ancestral tree, Tr and the type of alleles of the 10 individuals in the sample. Since the data is created by backward simulation a graph g with the known embedded genealogy Tr is created. The graph is considered to be the true ancestral selection graph of the individuals in the sample at the leaf tips.

The data is simulated for two different values of the parameter $\sigma = 0.5$ and

$\sigma = 1.5$. Two different graphs are created with two different sets of allele types for the individuals in the sample D. The aim is to use MCMC to recover the values of σ.

For the first experiment $\sigma = 0.5$ and $\theta = 1$. Let $g_{0.5}, Tr_{0.5}$ and $D_{0.5}$ denote the ancestral graph, the genealogical tree and the data at the leaf tips when $\sigma = 0.5$ respectively.

The true graph $g_{0.5}$ can be seen in Figure 6.23 where the genealogy, $Tr_{0.5}$ is indicated by a thicker line. The types at the leaves are $D_{0.5} = [0\,0\,1\,1\,0\,1\,1\,0\,1\,1]$. As can be seen from Figure 6.23, the labels of the leaves are not in numerical order. The value $D_{0.5}(8) = 0$, which is the allele type of vertex with label 8. Since individual 1 and individual 8 have a common ancestor, vertices with labels 1 and 8 are grouped together and are next to each other.

The ancestral graph has 23 vertices and 25 edges. Two of the vertices represent branching events and 10 vertices represent the data.

Excluding the leaves, the tree $Tr_{0.5}$ has 9 events occurring at times $\chi_{V,E}^{(Tr)} = [\,0,\ 0.0024\ 0.0405\ 0.0882\ 0.1610\ 0.2108\ 0.2938\ 0.4554\ 0.4969\ 1.0184\,]$. The most recent common ancestor is the same as the ultimate ancestor.

The ancestry of the individuals at the leaf tips can be seen in Figure 6.28.

The MCMC was updated $10,000,000$ times and every $1,000^{\text{th}}$ was sampled to give a sample of size $10,000$ consisting of ancestral selection graphs. The parameter σ was sampled from an uniform density between 0 and 2.5. The trace of the parameter σ can be seen in Figure 6.24

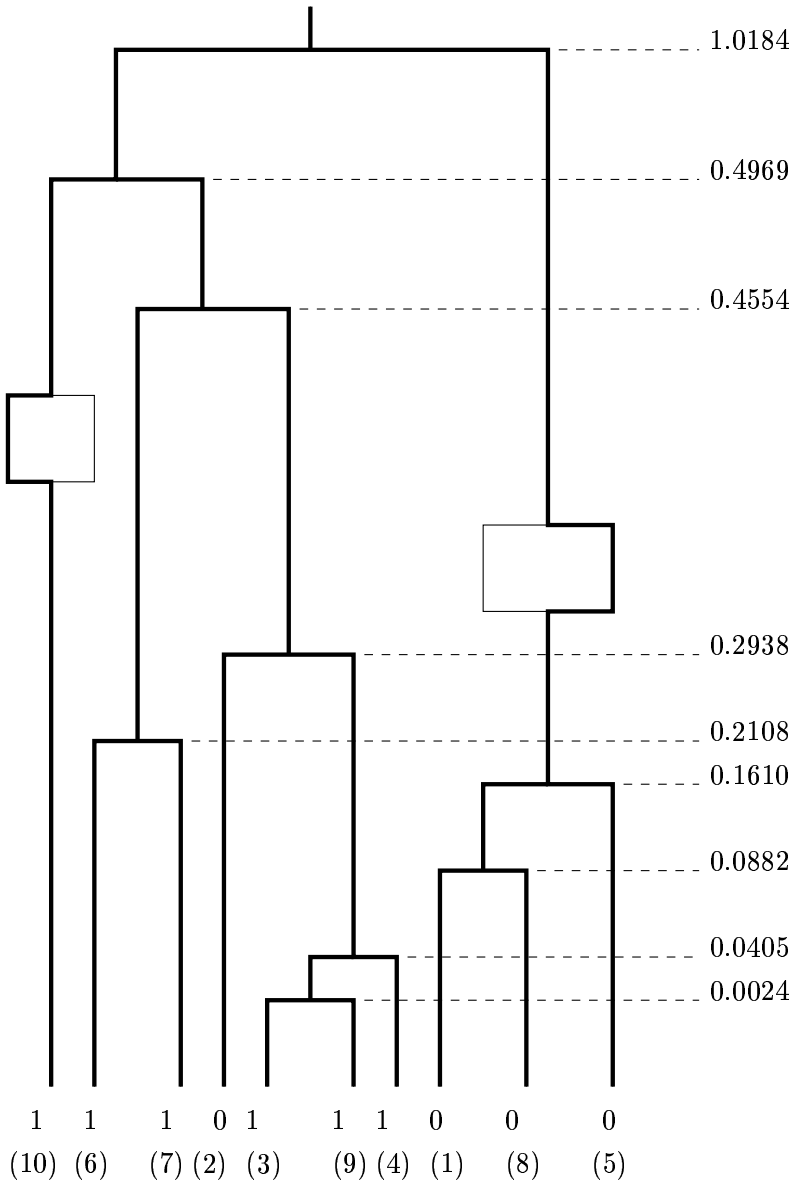


Figure 6.23: The ancestral selection graph, $g_{0.5}$. The ancestry, $Tr_{0.5}$ is denoted in a thicker line. The labels of the leaves are in brackets.

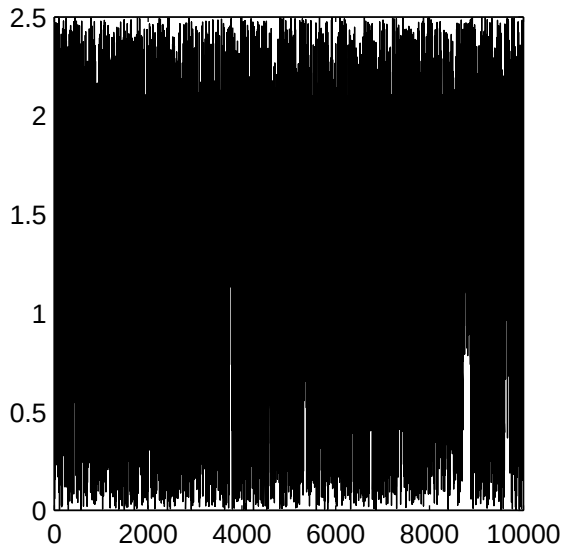


Figure 6.24: Trace of σ where the embedded ancestry is $Tr_{0.5}$

From the trace of t_{UA} in Figure 6.25, the flat side of the trace indicates that the time to the ultimate ancestor was never smaller than the time of the most recent common ancestor of the ancestral tree. This happened because all the graphs with a different ancestral tree than $Tr_{0.5}$ have been rejected. The spike at the end of the trace shows the presence of ladders.

Let $w(\sigma)$ denote the marginal posterior density of the parameter σ.The histogram of $w(\sigma)$ in Figure 6.26 indicates that the data is uninformative of the selection parameter σ.

The same experiment is carried out by creating a true ancestral graph $g_{1.5}$ for $\sigma = 1.5$. The allele type of the 10 individuals in the sample is $D_{1.5} = [1\,0\,1\,0\,1\,0\,1\,0\,1\,1]$. As before $D_{1.5}(2) = 0$ indicates that the allele type of the individual at the leaf vertex with label 2 is 0. The true ancestral graph can be

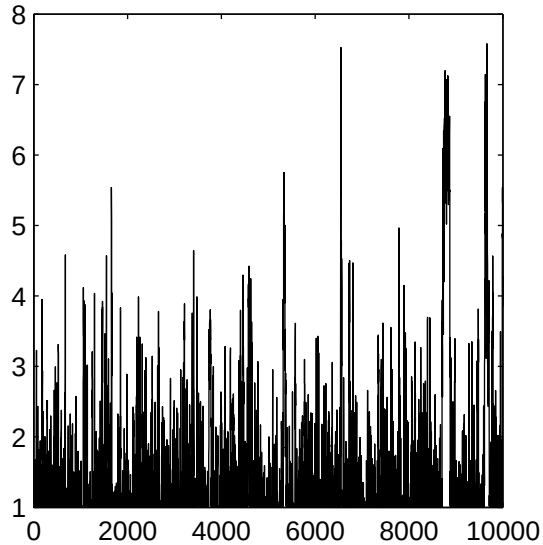


Figure 6.25: Trace of t_{UA} where the embedded ancestry is $Tr_{0.5}$.

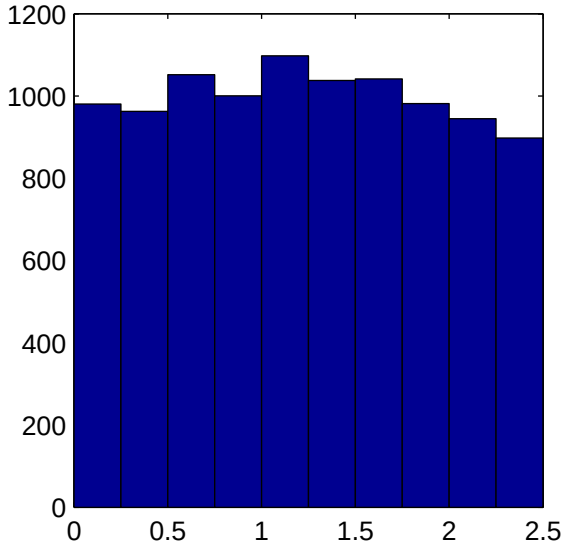


Figure 6.26: Histogram of the marginal posterior density, $w(\sigma)$, where the ancestry is $Tr_{0.5}$.

seen in Figure 6.27 where the leaf vertex with label 4 is the first leaf on the left of the graph. The thicker line indicates the ancestral path of the individuals at the leaf tips. The ancestral tree $Tr_{1.5}$, which is kept fixed throughout the MCMC, can be seen in Figure 6.28.

The prior density of σ, $P(\sigma)$, is the uniform density over the interval $(0, 2.5)$.

The trace of the parameter σ can be seen in Figure 6.29 and the trace of t_{UA} can be seen in Figure 6.30.

The histogram of the marginal posterior density $w(\sigma)$ can be seen in Figure 6.31. The data D at the leaf tips consists of 4 type zero individuals and 6 type one individuals. The histogram of the marginal posterior density $w(\sigma)$ is the same shape as the histogram for the likelihood of σ, which was created during the debugging stage. See Chapter 5, Figure 5.9. Therefore, the embedded genealogy $Tr_{1.5}$ does not reveal any more information on the selection parameter σ.

For the next experiment, the data D and the ancestral tree $Tr_{1.5}$ are the same as in the previous experiment, but the $P(\sigma) \sim U(0, 1.5)$ and not $U(0, 2.5)$. The trace of σ and the trace of t_{UA} can be seen in Figure 6.32 and Figure 6.33 respectively. The histogram of $w(\sigma)$ can be seen in Figure 6.34.

From these results it can be seen that if the selection pressure is high in a group of organisms and the data available consists of the allele types of a group of organisms as well as the ancestral tree of the group, the selection parameter can be inferred from the data.

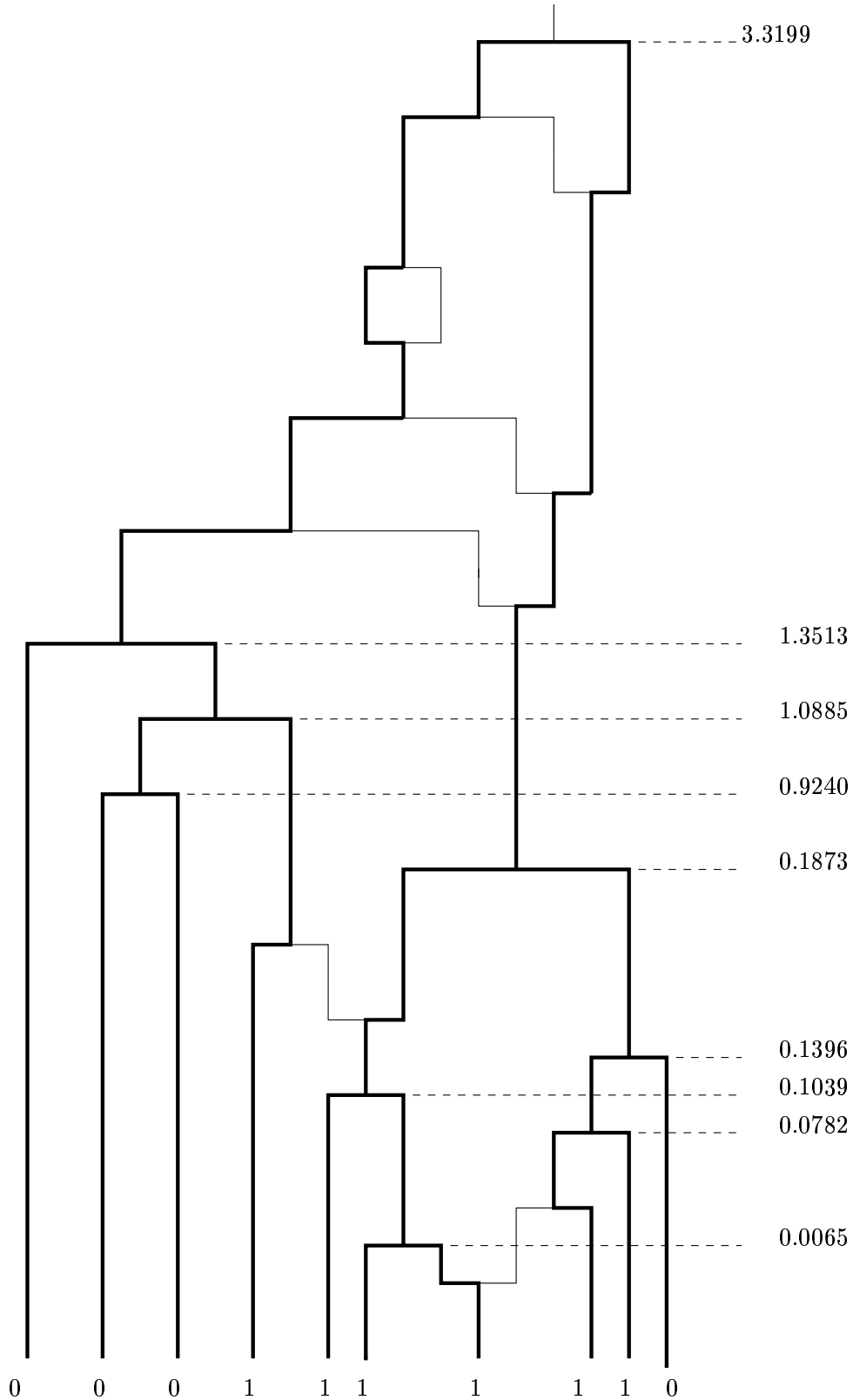


Figure 6.27: Ancestral selection graph, $g_{1.5}$ where the ancestry, $Tr_{1.5}$ is indicated by a thicker line

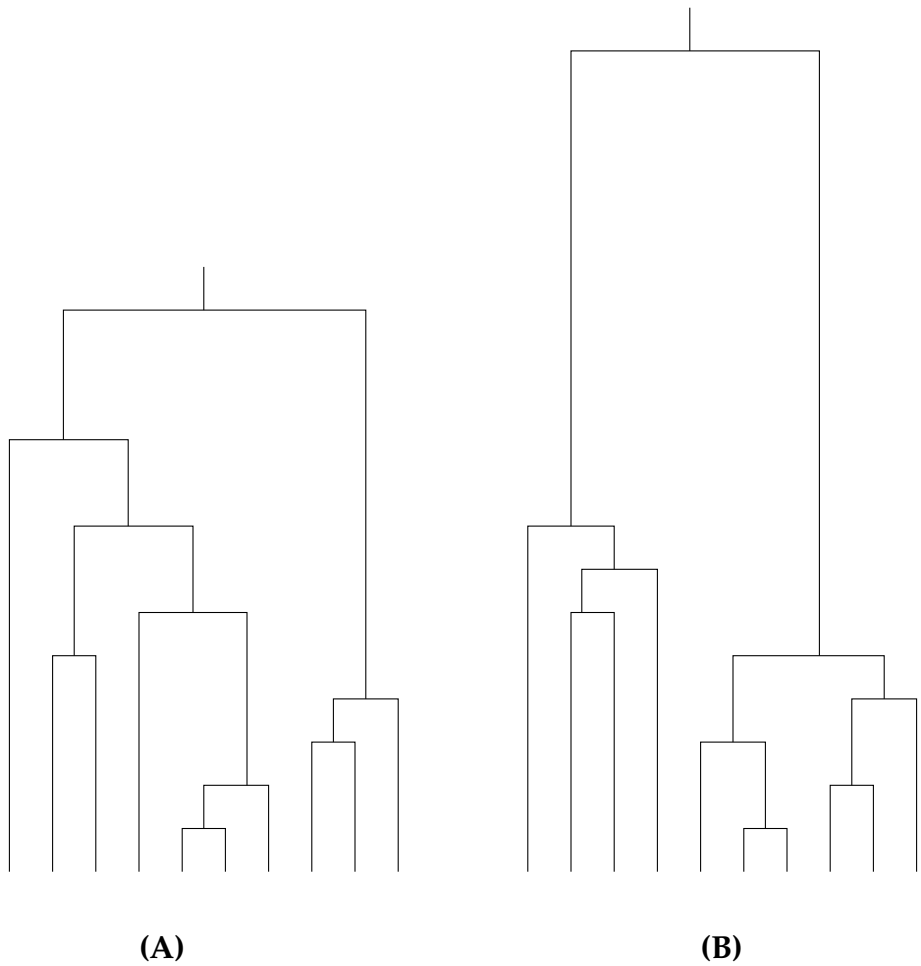


Figure 6.28: The ancestral trees $Tr_{0.5}$ (A) and $Tr_{1.5}$ (B) of the selection graphs in Figure 6.23 and Figure 6.27

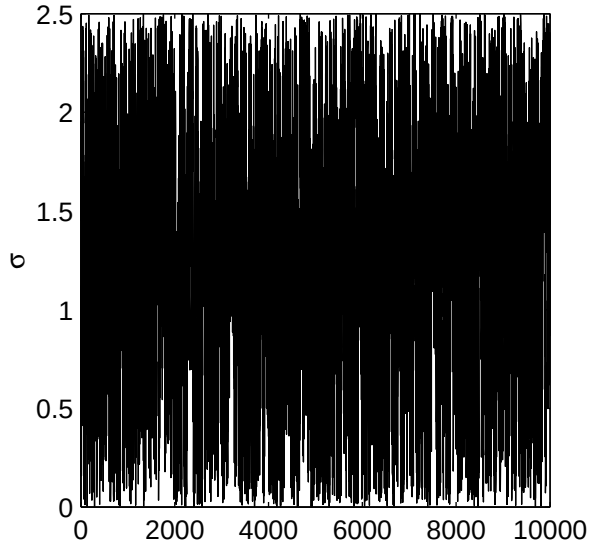


Figure 6.29: Trace of the parameter σ, $P(\sigma) \sim U(0, 2.5)$ and the ancestry is $Tr_{1.5}$

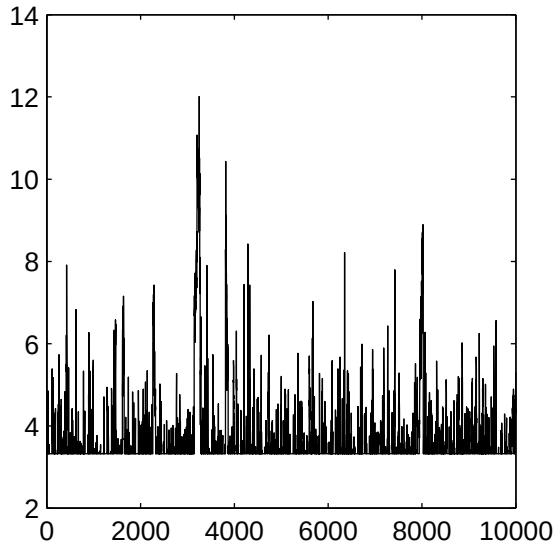


Figure 6.30: Trace of t_{UA},$P(\sigma) \sim U(0, 2.5)$ and the ancestry is $Tr_{1.5}$

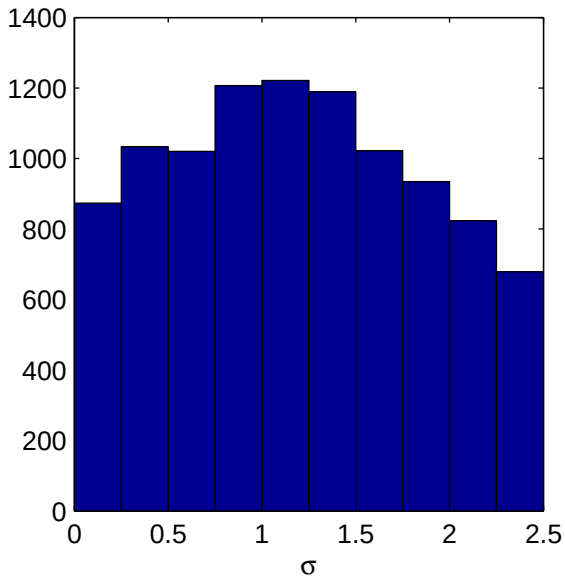


Figure 6.31: Histogram of $w(\sigma)$,$P(\sigma) \sim U(0, 2.5)$ and the ancestry is $Tr_{1.5}$

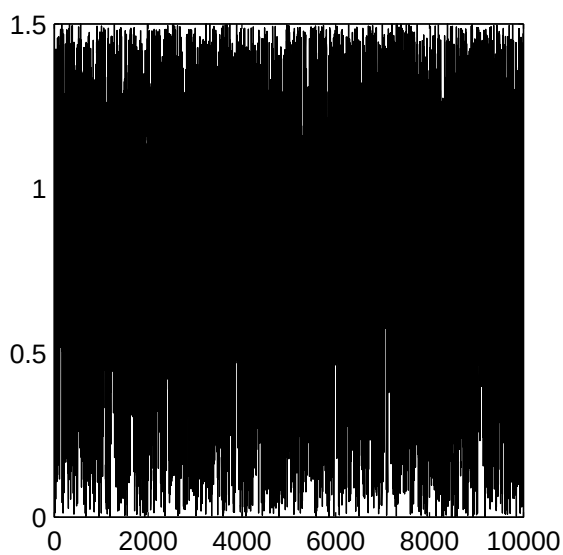


Figure 6.32: Trace of the parameter σ where $P(\sigma) \sim U(0, 1.5)$ and the ancestry is $Tr_{1.5}$

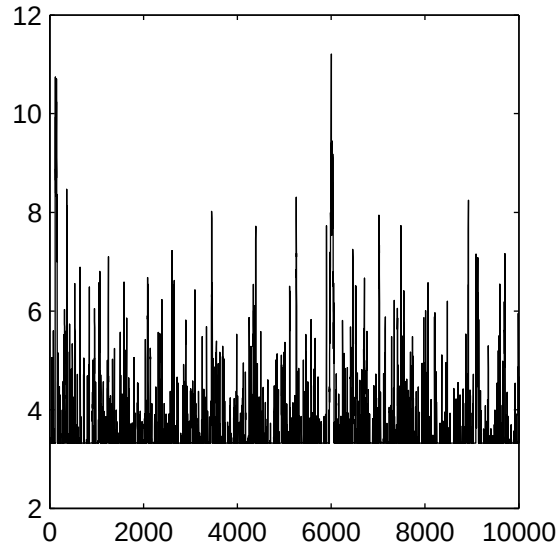


Figure 6.33: Trace of t_{UA} where $P(\sigma) \sim U(0, 1.5)$ and the ancestry is $Tr_{1.5}$

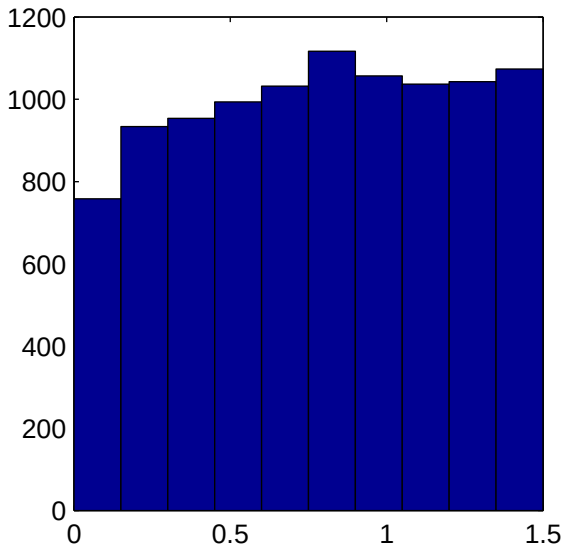


Figure 6.34: Histogram of $w(\sigma)$ where $P(\sigma) \sim U(0, 1.5)$ and the ancestry is $Tr_{1.5}$

For easier visualization the ancestral trees $Tr_{0.5}$ and $Tr_{1.5}$ can be seen in Figure 6.28. The tree in Figure 6.23 (A) is the embedded ancestry of the ancestral selection graph, $g_{0.5}$, in Figure 6.23 where $\sigma = 0.5$. Figure 6.28 (B) illustrates the embedded ancestral tree, $Tr_{1.5}$ for the ancestral selection graph, $g_{1.5}$ in Figure 6.27 where $\sigma = 1.5$.

The shape of genealogies can be altered by evolutionary processes. For instance when population growth is exponential, the genealogy shows shorter edges at the root of the tree. Golding [11] explored the effects of directional selection on the shape of the genealogy. He came to the conclusion that selection can not be detected by looking at the branch lengths of a genealogy.

For the next experiment, the question is reversed by asking what is the influence of a genealogy of a population under exponential growth on the selection parameter σ. Can directional selection be detected if the genealogy embedded in the ancestral selection graph is the genealogy of a population under exponential growth?

In this experiment the embedded ancestry Tr_{exp}, of the leaves shows exponential growth. The ancestral tree Tr_{exp}, has short edges below the MRCA. All the internal vertices, V_A of the tree Tr_{exp}, are close together at the top of the tree, as can be seen in Figure 6.35.

The vector $\chi^{Tr}_{V,E} = [0\ 2.4077\ 2.5450\ 2.7729\ 2.8000\ 2.9410\ 3.0\ 3.1\ 3.26\ 3.3199]$. The MRCA has the time 3.3199, while the first event happens at time 2.4077. The sample of 10 individuals is $D_{exp} = [1\,0\,1\,0\,1\,0\,1\,0\,1\,1]$.

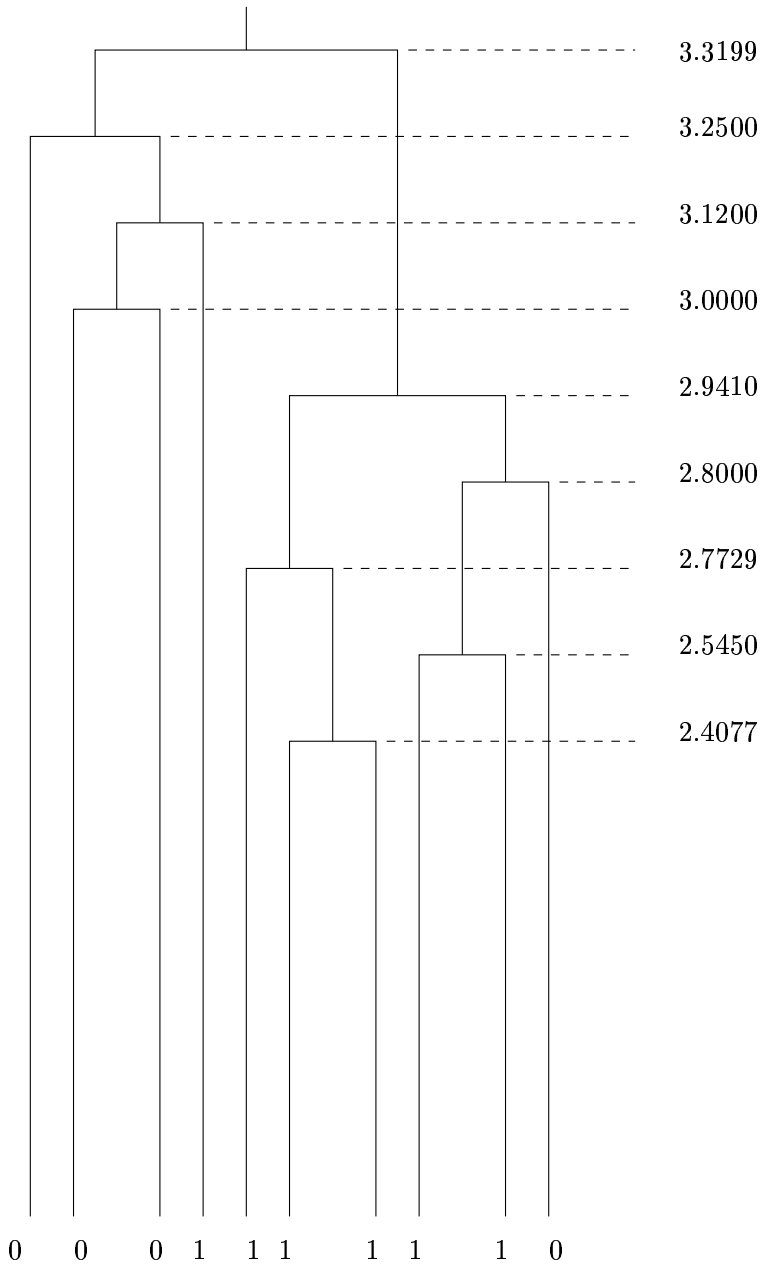


Figure 6.35: Ancestry, Tr_{exp}, of a sample taken from an exponential growing population. All the events happen close together at the top of the tree. The time of each internal event is followed by the allele type of the vertex

The MCMC was again updated $10,000,000$ times and every $1,000^{\text{th}}$ is sampled. During the MCMC, the embedded tree, Tr_{exp} is kept fixed and $P(\sigma) \sim U(0, 2.5)$.

The trace of the marginal posterior density $w(\sigma)$ can be seen in Figure 6.36. The histogram of the marginal posterior distribution $w(\sigma)$ is similar to an exponential distribution with mean 0.8687 and variance 0.0565 as can be seen in Figure 6.37.

An IACT of $97,820.1$ of the parameter σ, is much higher in the case where the embedded tree, Tr_{exp} shows exponential growth than an IACT of $19,078.2$ of σ, where the embedded tree, $Tr_{1.5}$ has a true selection parameter of 1.5. See Table 6.12 where the IACT, the acceptance percentage and the run time of the different MCMC are compared.

The MCMC was run on a high performance Sun Enterprise 4000 server with 7 processors running at 250 Mhz and 3.5 Gbytes of physical memory (RAM). The run time for each MCMC is recorded in Table 6.12 along with summary statistics of the marginal posterior density $w(\sigma)$.

The samples $\{\Phi_{ih}^{Tr_{exp}}\} \sim h(B_Y, \sigma, g|B_I, Tr_{exp})$ of the MCMC are highly correlated as can be seen from Table 6.12. The acceptance percentage of the Markov chain $\{\Phi_{ih}^{Tr_{exp}}\}$ is not much different than the acceptance rate of the other Markov chains, $\{\Phi_{ih}\}$. The run time of the MCMC where the samples are $\{\Phi_{ih}^{(Tr_{exp})}\}$ is a little higher than in the other cases but is still at an acceptable level.

The conclusion of these experiments is that all the information available

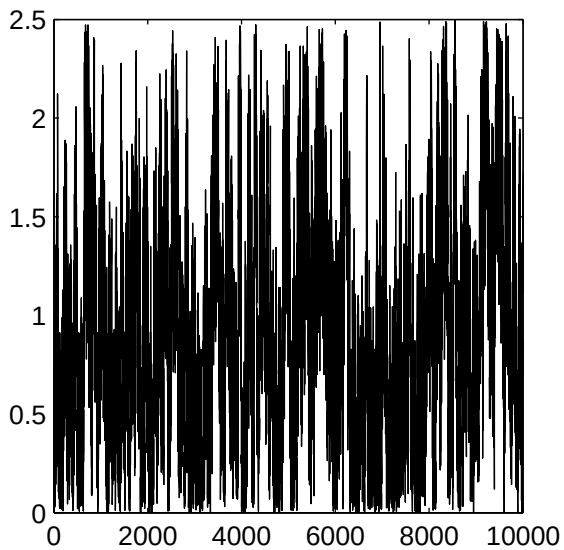


Figure 6.36: Trace of the parameter σ,where $P(\sigma) \sim U(0, 2.5)$. The ancestry is Tr_{exp}

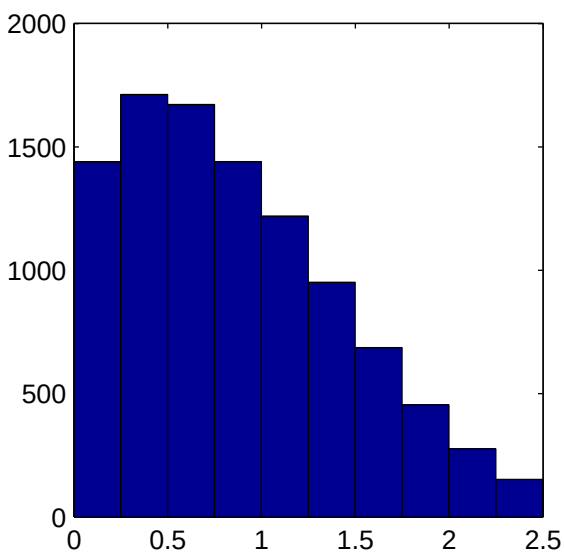


Figure 6.37: Histogram of $w(\sigma)$, where $P(\sigma) \sim U(0, 2.5)$. The ancestral tree Tr_{exp}

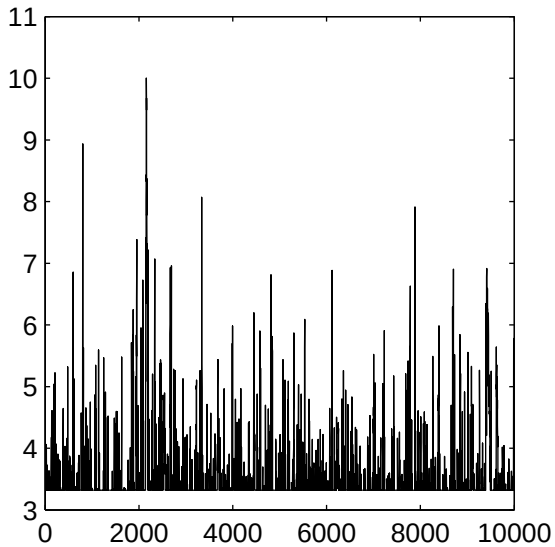


Figure 6.38: Trace of t_{UA} where $P(\sigma) \sim U(0, 2.5)$. The ancestral tree is Tr_{exp}.

	$Tr_{0.5}$	$Tr_{1.5}$	$Tr_{1.5}^{(15)}$	Tr_{expo}
mean	1.2350	1.1980	0.7828	0.8687
se	0.0115	0.0292	0.0114	0.0565
τ	$2,652.6$	$19,078.2$	$7,288.1$	$97,820.1$
acct	39.01	41.34	41.04	39.61
cpu time	$1.3035e+05$	$2.2574e+05$	$1.6954e+05$	$2.8924e+05$

Table 6.12: Summary of the statistics of $w(\sigma)$. τ is the IACT, acct is the overall acceptance percentage and cpu time is the run time of the MCMC in seconds. $Tr_{1.5}$ is the ancestral tree simulated for $\sigma = 1.5$ and $P(\sigma) \sim U(0, 2.5)$. $Tr_{1.5}^{(15)}$ indicate the ancestral tree simulated when $\sigma = 1.5$ and $P(\sigma) \sim U(0, 1.5)$.

should be incorporated in a model to be able to infer a selection parameter. As these experiments show, the data includes the ancestry, Tr of the sampled individuals from the population as well as the allele types of the individuals. The last experiment also indicates that a population under exponential growth can indicate the presence of directional selection.

6.3.2 Multiple loci

The next method involves the inference of σ by allowing for two alleles at multiple loci or positions on the chromosome. The assumption is that all the loci are under the same magnitude of selection.

Let L denote the number of loci. For the MCMC the Markov chain $\{\Phi_{ix}\}$ is created by sampling from the distribution $x(B_I, g, \sigma | B_I, L = 5)$. During this experiment the ancestry of the group of organisms is not kept fixed.

As an example a graph with 10 leaves is simulated at $\sigma = 0.8$ by backward simulation. The alleles at the leaves is denoted by $D^{(5)}$ and is a 5×10 matrix. The 10 columns represent the individuals and the rows the different loci. For instance in the example below leaf vertex with label 5 has the following types at the different loci, $(01100)^T$, where T indicates the transpose of the vector. The 10 individuals in this example have all 0 at the first locus or position.

In this experiment the matrix $D^{(5)}_{0.8}$ is as follows:

$$D_{0.8}^{(5)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Imagine 5 copies of the topology of the graph g on top of each other, but the copies differ from each other at the vertices by having different allele types.At an internal vertex the types is a 1×5 matrix. The time to the ultimate ancestor $t_{UA} = 2.1548$.

When determining the most recent common ancestor $MRCA$, the ancestry, Tr of the graphs g has to be determined first. At a branching event only one of the two edges out-going from the branching vertex at the top is included in the ancestry. Since all the loci are under the same selection pressure, only one position determines the edge included in the genealogy. The inclusion of the edge in the ancestry is determined by the same rules as discussed in Chapter 2. In this model the ancestry can not be determined if the loci are under different selection pressures, because according to the rule, one locus determines the inclusion of one edge at a branching vertex and another locus includes the other edge in the ancestry.

Unfortunately real data can not be applied using this method, since real data are not collected for 5 different loci on the same chromosome. In real data

the loci might also be under different selection pressure.

The main reason for not being able to use this method is it took a High performance Sun Enterprise 4000 $8.1223e+04$ seconds to generate a Markov Chain of only 98 samples. Maybe with a faster and stronger computer this method is better than the one where the ancestry of the leaves is kept fixed.

Chapter 7

Conclusion

The initial aim of inferring a selection parameter from data, sampled from a population of haploid organisms, extended to a question of what information is available to us. What exactly do we know about a sample of organisms and hence of the population and how can we incorporate all the information into a model?

To address these questions, a Bayesian approach was used. The advantage of a Bayesian framework is that all the information available can be incorporated into the posterior density. As new information becomes available, this can be easily added to the existing knowledge without too much work. This approach paid off in the end, since the data available is much more, than what we anticipated at the start of the project.

The graphical representation of the Moran model with selection, as introduced by Neuhauser and Krone is similar to the binary graphs representing

the neutral n-coalescent of Kingman. The ancestry or binary tree of the sampled organisms is embedded in the ancestral selection graph. Therefore, the question became, how can the known ancestry of a group of organism be used to infer the selection parameter σ.

We wanted to answer these questions by using a flexible and efficient MCMC.

During the initial experiments in Chapter 6 the influence of the magnitude of the selection parameter on the MCMC was investigated. These experiments led to the discovery of ladders, Chapter 6, Section 6.1 when $\sigma > 1$.

Although the ladders can be reduced, by taking larger samples from the population of organisms, i.e more leaves at the bottom of the graphs, the correlation among the sampled graphs of the MCMC increases. The correlation problem can be overcome if the MCMC draws more samples or if the samples are drawn further apart. In both cases it means longer Markov chains and more computer time.

The second set of experiments in Chapter 6 involved the incorporation of the known ancestry of the sampled organisms to infer a selection parameter. This is done by taking a sample of organisms from a population under low selection pressure. The ancestry of the sample is determined and the Markov chain is created by rejecting all the graphs which do not have the known ancestry as an embedded tree. The same experiment was carried out by taking a sample from a population under high selection pressure. The data consisting of the allele types and the ancestry of the sampled organisms under low

selection pressure is uninformative of the selection parameter σ as can be seen in Chapter 6, Section 6.3.1. Although the data consist of the allele types at the leaf tips and the known genealogy, Tr, the data from a population of organisms under high selection pressure, is still uninformative of the parameter σ as can be seen in Chapter 6, Section 6.3.1.

This led to the question: What can the ancestry tell us about the selection parameter if the population is under exponential growth? The ancestry of a sample of organisms from a population under exponential growth can be seen in Chapter 6, Section 6.3, Figure 6.35. The result of this experiment is that the data indicates that the selection parameter is approximately exponentially distributed. For a more detailed discussion see Chapter 6, Section 6.3.1.

The last experiment was carried out where the data consists of two allele types at 5 different loci on a chromosome. The assumption is that the loci are under the same selection pressure. At this point this method is not viable since the acceptance rate of the MCMC was very low and the MCMC runs for a very long time. Another disadvantage is the assumption that all the loci are under the same selection pressure.

7.1 Future work

The cause of the ladders for $\sigma > 1$ is still unknown. Future work involves the investigation into the cause of ladders and whether or not they can be

eliminated by incorporating other moves in the MCMC.

The ladder index and the incidence of ladders decrease as the size of the sample of organisms increases. This causes correlation among the samples of graphs of the MCMC to increase. Future work involves the determination of an optimal size for the sample of organisms, which takes into account the length of the Markov chain and the computer run time.

Throughout the project only simulated data were used. The challenge for future work is to apply the MCMC to real data. The idea is to take a sample of organisms from an exponentially growing population, determine the ancestry of the sample and apply the MCMC to the data.

In conclusion, the last experiment opened up more questions. Can data be collected differently to inform us more on the selection parameter? Are different loci under different selection pressure? And if the loci are under different selection pressure, can we invent a model to incorporate this fact?

The code for this MCMC is very delicate and long hours have been spent to write and debug the code. The MCMC involved the determination of nuisance parameters such as the time to the ultimate ancestor, the time to the most recent common ancestor, the number of branching events in a graph and the allele types of the ancestors. In the end, the only information we wanted was if the data available, is informative of the selection parameter.

I conclude with the statement: Future work involves development of a new efficient computational method, which uses all the information available to in-

fer only the parameters we are interested in. That is what I would call efficient, but this is a task for other brighter people.

Appendix A

Data structures

The data structure `V`(v) representing the vertices:

`V`(v)		
Fields	Structure	Description
`V.e`	(e_1, e_2, e_3)	The labels of the three edges associated with a vertex v
`V.time`	t	The time value associated with a vertex v
`V.type`	$1, 2, 4$ or 5	1: leaves 2: internal vertices 4: ultimate ancestor 5: the vertex above the root
`V.a`	0 or 1	Allele type of the vertex v
`V.next`	u	The label of the next event going from the root to the leaves
`V.prev`	w	The label of the previous event
`V.na`	i	The number of edges between the vertex v and the next event

The data structure `E`(e) representing the edges:

`E`(e)		
Fields	Structure	Description
`E.v`	(v_1, v_2)	v_1: Vertex label at bottom of edge e v_2: Vertex label at top of edge e
`E.cnt`	0 or1	0: Replacing edge 1: Continuing edge
`E.b`	0 or 1	0: Not a branching edge 1: Branching edge
`E.used`	0 or 1	0: Edge is not included in the genealogy 1: Edge is included in the genealogy
`E.allele`	$[a_1, a_2]$	a_1: Allele type at bottom of edge a_2: Allele type at top of edge

Appendix B

Theorems and proofs

Theorem 1 *Let g be an ancestral selection graph and let N_b be the number of branching events. The maximum number of deletable edges in g is $2N_b$ and the minimum number of deletable edges is $N_b + 1$.*

Proof:

If $N_b = 1$, both of the resulting edges outgoing from the branching vertex is deletable.

The proof is by mathematical induction for $N_b \geq 2$. Let $N_b = 2$. Suppose the branching events do not follow each other immediately. Then the two resulting edges of each branching events have coalescing events at the top. The number of deletable edges is 4 which is equal to $2N_b$. If one branching event follows immediately after another branching event, i.e. one of the resulting edges of one branching event branches again, then only one of the the edges of the one

branching event is deletable. The two resulting edges of the final branching events are both deletable and the number of deletable edges is $2 + 1 = 3$. The minimum number of deletable edges is $N_b + 1 = 2 + 1 = 3$.

Suppose the maximum number of deletable edges for $N_b = k$ branching events is $2N_b = 2k$ and the minimum number of deletable edges is $N_b + 1 = k + 1$.

Let $N_b = k+1$. Since only one branching event is added, one of three things can happen:

1. The new branching event is added immediately after an existing branching event. One of the resulting edges from the previous branching event is turned into a branching event. In this case one of the deletable edges is not deletable anymore and two more deletable edges are added as can be seen in Figure B.1. The number of deletable edges is $k+1-1+2 = k+2$, which is equal to the minimum number of deletable edges.

2. The branching event is between two existing branching events. This means the resulting edge from the previous branching event is a branching edge and one of the resulting edges from the new branching event is also branching. See Figure B.1. In this case only one deletable edge is added. The number of deletable edges is $k+1+1 = k+2$, which is equal to the number of branching events $N_b = k + 1$ plus one. This is equal to the minimum number of branching events.

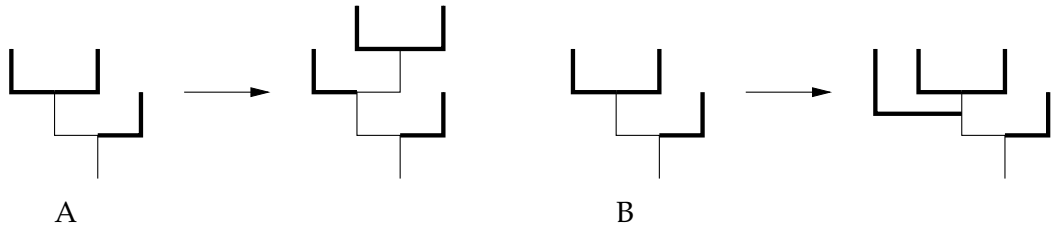


Figure B.1: The dark edges are deletable edges. Graph A has initially three deletable edges. A branching event is added at the top and one deletable edge is taken away while two deletable edges are added. The resulting graph has 4 deletable edges. Graph B has three deletable edges. After a branching event is added between two branching events, only one deletable edge is added and the graph has 4 deletable edges.

3. The third case is when the branching event is added at a point not connected with existing branching events. In this case two deletable edges are added. This is the maximum number of deletable edges that can be added. if the number of existing deletable edges is the minimum number of deletable edges $k+1$, then the number of deletable edges becomes $k+1+2 = k+3$. The number of deletable edges is less than the maximum number of deletable edges $2(k+1) = 2k+2$ and more than the minimum number of deletable edges $k+2$. If the number of existing deletable edges is the maximum number of deletable edges $2k$ then the number of deletable edges is $2k+2$ which is equal to the maximum number of deletable edges.

Theorem 2 *The Markov chain $\{\Phi_n\}$ is π-irreducible.*

Proof: The space $\tilde{\Gamma}$ can be partitioned into a collection of subsets $\{A_i, i =$

$0, 1, 2, ...\}$ with $A_i \cap A_j = \emptyset \quad , i \neq j$ where A_i is the subset of $\tilde{\Gamma}$ containing all ancestral selection graphs with the number of branching events $N_b = i$. The set A_0 contains all the binary trees or ancestral graphs with zero branching events.

Let $P((g, B_Y), A_i) = P(\Phi_n \in A_i | \Phi_{n-1} = (g, B_Y))$ be the probability the Markov chain Φ reached the subset A_i after one move by starting with type-labeled graph (g, B_Y). To show that the Markov chain $\{\Phi_n\}$ is irreducible on $\tilde{\Gamma}$ we only need to show that $P((g, B_Y), A_i) > 0$ where $(g, B_Y) \in A_{i+1}$ and that $P((g, B_Y), A_i) > 0$ when $(g, B_Y) \in A_{i-1}$.

Let $(g, B_Y) \in A_{i+1}$. Since the graph g has $i + 1$ branching events, the minimum number of deletable edges is $i+2$ and the maximum number of deletable edges is $2(i + 1)$ as shown in Theorem 1. This means that the probability to choose a deletable edge, $\frac{1}{|E_D|}$, is well-defined and not equal to zero. The transition probability to go from a state $(g, B_Y) \in A_{i+1}$ to a state (g', B'_Y) in A_i is $P((g, B_Y), A_i) = \psi(g', B'_Y | g, B_Y)\alpha(g', B'_Y | g, B_Y)$, where $\psi(g', B'_Y | g, B_Y) = p^\dagger \frac{1}{|E_D|}$. The function $\psi(g', B'_Y | g, B_Y)$ is a positive function since it is the product of the nonzero probability to delete an edge, $p^\dagger$, and the probability of choosing a deletable edge as explained in Section 3.1.1. The acceptance probability $\alpha(g' B'_Y | g, B_Y) > 0$ as can be seen in Section 3.1.1.

Let $(g, B_Y) \in A_{i-1}, i = 1, 2, 3....$ The set A_0 consists of all binary trees, i.e all ancestral selection graphs with zero branching events. The transition probability to go from a state $(g, B_Y) \in A_{i-1}$ to a state $(g', B'_Y) \in A_i$ is $P((g, B_Y), A_i) =$

$\psi'(g', B'_Y|g, B_Y)\alpha(g', B'_Y|g, B_Y)$, where $\psi'(g', B'_Y) = p^* \frac{1}{|E|^2} \frac{1}{\tau_1 \tau_2}$ as explained in Section 3.1.1. The ancestral selection graphs are all connected graphs. If a graph has zero branching events and n leaves then the number of edges is $2n - 1$. If a graph has N_b branching events and n leaves then the number of edges is $2n + 2N_b$. Since all the edges of a graph has non-zero lengths the function $\psi'(g', B'_Y|g, B_Y)$ is well-defined and non-zero.

By repeatedly applying the functions ψ and ψ' all the sets $A_i, i = 1, 2, 3...$ can be reached. Therefore the Markov chain Φ is irreducible. Q.E.D

Bibliography

[1] M. Bahlo and R.C. Griffiths. Inference from gene trees in a subdivided population. *Theoretical Population Biology*, 57:79–95, 2000.

[2] N. Cloete, G.K. Nicholls, and D.J. Scott. Simulation of ancestral selection graphs for Monte Carlo integration. *ANZIAM Journal*, 45:C391–C404, 2004.

[3] P. Donnelly and T. T. Kurtz. Genealogical processes for Fleming-Voit models with selection and recombination. *Ann. Appl. Probab.*, 9:1091–1148, 1999.

[4] A.J. Drummond, G.K. Nicholls, A.G. Rodrigo, and W. Solomon. Estimating mutation parameters,population history and genealogy simultaneously from temporally spaced sequence data. *Genetics*, 161:1307–1320, 2002.

[5] W.J. Ewens. *Mathematical Population Genetics*. Springer-Verlag New York,Inc., 2004.

[6] G. Ewing, G.K. Nicholls, and A.G. Rodrigo. Using temporally spaced sequences to simultaneously estimate migration rates, mutation rate and population sizes in measurably evolving populations (meps). Technical report, Bioinformatics Institute, University of Auckland, 2004.

[7] J. Felsenstein, M.K. Kuhner, J. Yamato, and P. Beerli. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 317:368–376, 1981.

[8] R.A. Fisher. *The General Theory of Natural Selection*. Clarendon Press, 1930.

[9] C. J. Geyer and J. Møller. Simulation procedures and likelihood inference for spatial point processes. *Scand. J. Statist.*, 21:359–373, 1994.

[10] C.J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):472–511, 1992.

[11] G.B. Golding. The effects of purifying selection on genealogies. In P. Donnelly and S. Tavare, editors, *Progress in Population Genetics and Human Evolution*, pages 271–285. Springer-Verlag New York, Inc., 1997.

[12] P.J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[13] R.C. Griffiths and S. Tavaré. Simulating probability distributions in the coalescent. *Theoretical Population Biology*, 46:131–159, 1994.

[14] J. Hein, M.H. Schierup, and C. Wiuf. *Gene Genealogies, Variation and Evolution. A primer in coalescent theory*. Oxford University Press, 2005.

[15] M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217:624–626, 1968.

[16] J.F.C. Kingman. The coalescent. *Stochastic Processes and their Applications*, 13:235–248, 1982.

[17] S. M. Krone and C. Neuhauser. Ancestral processes with selection. *Theoretical Population Biology*, 51:210–237, 1997.

[18] C. Neuhauser and S.M. Krone. The genealogy of samples in models with selection. *Genetics*, 145:519–534, February 1997.

[19] R. Nielsen. Detecting selection in protein coding genes using the rate of nonsynonymous and synonymous divergence. In A.G. Rodrigo and Jr. G.H Learn, editors, *Computational and Evolutionary Analysis of HIV Molecular Sequences*, pages 253–267. Kluwer Academic Publishers, 2001.

[20] G.O. Roberts. Markov chain concepts related to sampling algorithms. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, *Markov chain Monte Carlo in Practice*, pages 45–57. Chapman and Hall/CRC, 1996.

[21] P.F. Slade. Simulation of selected genealogies. *Theoretical Population Biology*, 57:35–49, 2000.

[22] A.D. Sokal. Monte Carlo methods in statistical mechanics: Foundations and new algorithms. Technical report, Department of Physics New York University, New York, 1996.

[23] M. Stephens and P. Donnelly. Inference in molecular population genetics. *Journal of the Royal Statistical Society, Series B*, 62:605–655, 2000.

[24] D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic inference. In D.M. Hillis, C. Moritz, and B.K. Marble, editors, *Molecular Systematics*. Sinauer Associates, Sunderland, Mass., second edition, 1996.

[25] L. Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22(4):1701–1728, 1994.

[26] D. Welch, G.K. Nicholls, A. Rodrigo, and W. Solomon. Integrating genealogy and epidemiology: The ancestral infection and selection graph as a model for reconstructing host virus histories. Technical report, Bioinformatics Institute, University of Auckland, 2004.

[27] I.J. Wilson, M.E. Weale, and D.J. Balding. Inferences from DNA data: population histories, evolutionary processes and forensic match probabilities. *J. R. Statist. Soc. A*, 166:1–33, 2003.

[28] S. Wright. Evolution in Mendelian populations. *Genetics*, 16:97 – 159, 1931.