Equilibrium Policy Gradients for Spatiotemporal Planning

by

Mark Crowley

BA Computer Science, York University, 1999

MSc Computer Science, University of British Columbia, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University Of British Columbia

(Vancouver)

November 2011

Abstract

In spatiotemporal planning, agents choose actions at multiple locations in space over some planning horizon to maximize their utility and satisfy various constraints. In forestry planning, for example, the problem is to choose actions for thousands of locations in the forest each year. The actions at each location could include harvesting trees, treating trees against disease and pests, or doing nothing. A utility model could place value on sale of forest products, ecosystem sustainability or employment levels, and could incorporate legal and logistical constraints such as avoiding large contiguous areas of clearcutting and managing road access. Planning requires a model of the dynamics. Existing simulators developed by forestry researchers can provide detailed models of the dynamics of a forest over time, but these simulators are often not designed for use in automated planning.

This thesis presents spatiotemoral planning in terms of factored Markov decision processes. A policy gradient planning algorithm optimizes a stochastic spatial policy using existing simulators for dynamics.

When a planning problem includes spatial interaction between locations, deciding on an action to carry out at one location requires considering the actions performed at other locations. This spatial interdependence is common in forestry and other environmental planning problems and makes policy representation and planning challenging. We define a spatial policy in terms of local policies defined as distributions over actions at one location conditioned upon actions at other locations.

A policy gradient planning algorithm using this spatial policy is presented which uses Markov Chain Monte Carlo simulation to sample the

landscape policy, estimate its gradient and use this gradient to guide policy improvement. Evaluation is carried out on a forestry planning problem with 1880 locations using a variety of value models and constraints.

The distribution over joint actions at all locations can be seen as the equilibrium of a cyclic causal model. This equilibrium semantics is compared to Structural Equation Models. We also define an algorithm for approximating the equilibrium distribution for cyclic causal networks which exploits graphical structure and analyse when the algorithm is exact.

Preface

Chapters 6 and 7 are based upon published work: M. Crowley and D. Poole. Policy gradient planning for environmental decision making with existing simulators. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, Special Track on Computational Sustainability and AI.* San Francisco, 2011. I carried out the primary design, analysis and writing in close consulation with Professor D. Poole.

Chapter 4 is based on published work: M. Crowley, J. Nelson, and D. Poole. Seeing the forest despite the trees: Large scale spatial-temporal decision making. In *Proceedings of the Twenty-Fifth Annual Conference on Uncertainty in Artificial Intelligence.* Montreal, 2009. I carried out the primary design, analysis and writing in close consulation with D. Poole. Professor J. Nelson provided valuable input on the nature of the decision problem in forestry and the kinds of policies and value models that are used in practice.

Chapter 5 is being prepared as a conference submission. D. Poole carried out some of the evaluations and wrote the original draft of this paper which forms the basis for Sections 5.2 to 5.6.

Table of Contents

List of Tables

List of Figures

List of Symbols

Chapter 1

C	p.2	A spatial landscape is partitioned into a set of cells $c \in C$.
$a_c \in A$	p.2	The set of actions which can be taken at some location c
$\pi_c(a_c)$	p.7	A local policy is defined as a distribution over actions $Lac \in A$.
$\Pi(a_c)$	p.7	A landscape policy is a distribution over joint actions at all locations in the landscape.

Chapter 3

$\mathcal{T}$	p.30	Transition probability for a Markov chain
w	p.30	Equilibrium distribution $\delta(X)$ for a Markov chain
W	p.30	Equilibrium matrix where each row is w
$\delta(X)$	p.31	Distribution over set X
$\mathtt{pa}(X)$	p.32	Set of parents of a node X in a Bayesian network or causal network
$p(X\|\mathtt{pa}(X))$	p.32	Probability of node X given its parents in a Bayesian network
$\mathbf{S}$	p.36	Finite set of landscape states of an MDP
$\mathbf{A}$	p.36	Finite set of landscape actions available in an MDP
$\mathbf{s} \in \mathbf{S}$	p.36	A particular landscape state in an MDP

$\mathbf{s}'$	p.36	The next landscape state after $\mathbf{s}$ in an MDP
$\mathbf{a} \in \mathbf{A}$	p.36	A particular landscape action in an MDP
$r(\mathbf{s}, \mathbf{a})$	p.36	*Reward function* which returns the expected immediate reward received by starting in landscape state $\mathbf{s}$, then taking landscape action $\mathbf{a}$
$\mathcal{T}(\mathbf{s}'\|\mathbf{s}, \mathbf{a})$	p.36	*Dynamics* of an MDP which specifies the probability of transitioning from landscape state $\mathbf{s}$ to landscape state $\mathbf{s}'$ given landscape action $\mathbf{a}$
$k = \langle \mathbf{s}^0, \mathbf{a}^0, \mathbf{s}^1, \mathbf{a}^1, \ldots \rangle$	p.36	A *trajectory* is a series of landscape states and actions.
K	p.36	Set of all trajectories k being considered during planning
$\mathbf{s}^{k,t} \in \mathbf{S}$	p.36	A particular state state at time step t in trajectory k
$\mathbf{a}^{k,t} \in \mathbf{A}$	p.36	A particular action at time step t in trajectory k
T	p.36	The number of time steps in a trajectory is called the *planning horizon*.
$\gamma \in [0, 1]$	p.37	Discount factor which models that current rewards are worth more than future rewards.
$\mathcal{V}^\pi(k)$	p.37	Expected discounted return of policy π
$\mathcal{R}(k)$	p.37	The *discounted return* of a trajectory is a function K $\rightarrow \Re$ given by $\mathcal{R}(k) = \sum_t \gamma^t r(\mathbf{s}^t, \mathbf{a}^t)$
$\mathcal{V}_i^\pi(k)$	p.38	Estimate of the expected discounted return of policy π after i backups

$\mathcal{V}^*(k)$	p.38	Expected discounted return of the optimal policy
$\mathcal{K}$	p.40	Set of all possible trajectories k
θ	p.40	Parameters of a policy
ρ	p.40	Number of components in the policy parameters θ
$\pi(\mathbf{a}\|\mathbf{s},\theta)$	p.40	A *stochastic policy* describes a probabilistic rule for how an agent acts in the world. A policy is a function $\pi : S \times \Theta \to \mathbf{A}$ from states to actions controlled by the parameters θ.
$p(k\|\theta)$	p.40	The probability of a trajectory is: $p(k\|\theta) = p(\mathbf{s}^0)\prod_{t=1}^{\mathrm{T}} \mathcal{T}(\mathbf{s}^{k,t}\|\mathbf{s}^{k,t-1},\mathbf{a}^{k,t-1})\,\pi(\mathbf{a}^{k,t}\|\mathbf{s}^{k,t},\theta)$
$\nabla_\theta \mathcal{V}^\pi$	p.41	Gradient of the value function with respect to the policy parameters θ
$b[\theta]$	p.43	Baseline for a policy paramtrized by θ which can be subtracted from the reward to reduce the variance of the gradient

Chapter 4

Notation	**Page**	**Description**
C	p.47	Finite set of cells representing locations in a spatial landscape
S	p.47	Finite set of states of a cell
A	p.47	Finite set of actions available at a cell
$\mathbf{S}$	p.47	Set of landscape states S^C
$\mathbf{A}$	p.47	Set of joint landscape actions A^C

Chapter 4

Notation	Page	Description
$\mathbf{s} \in \mathbf{S} \quad s_c \in S$	p.47	A particular landscape state and cell state
$\mathbf{a} \in \mathbf{A} \quad a_c \in A$	p.47	A particular landscape action and cell action
$\mathbf{a}_{-c}$	p.47	Particular cell actions for all cells other then cell c
$r(\mathbf{s}, \mathbf{a})$	p.48	*Reward function* which returns the expected immediate reward received by starting in landscape state $\mathbf{s}$, then taking landscape action $\mathbf{a}$
$\mathcal{T}(\mathbf{s}'\|\mathbf{s}, \mathbf{a})$	p.48	*Dynamics* of an MDP which specifies the probability of transitioning from landscape state $\mathbf{s}$ to landscape state $\mathbf{s}'$ given landscape action $\mathbf{a}$
F	p.48	Set of features describing the state of a cell
$f_c(\mathbf{s})$	p.48	Value of a feature $f \in F$ on cell c given landscape state $\mathbf{s}$
$\mathcal{R}(k)$	p.49	The *discounted return* of a trajectory is a function K $\rightarrow \Re$ given by $\mathcal{R}(k) = \sum_t \gamma^t r(\mathbf{s}^t, \mathbf{a}^t)$
$\theta[f, a]$	p.50	Weight of feature f for action a
$\psi(a, c, \mathbf{s}, \theta)$	p.50	Potential function combining the weights and features for a cell, given by: $\psi(a, c, \mathbf{s}, \theta) = \sum_f \theta[f, a] f_c(\mathbf{s})$

Chapter 4

Notation	Page	Description
$\pi_c(\mathbf{a}\|\mathbf{s},\theta)$	p.50	A cell policy is a log-linear distribution over cell actions using the potential function: $\pi_c(\mathbf{a}\|\mathbf{s},\theta) = \frac{\exp(\psi(a,c,\mathbf{s},\theta))}{\sum_{b_c \in A}\exp(\psi(b,\mathbf{s},\theta))}$
$\Theta : C \to \theta$	p.51	A set of policy parameters for each cell. Parameters for a single cell denoted Θ_c. If the policy is spatially stationary then $\Theta = \theta$
$\Pi^C(\mathbf{a}\|\mathbf{s},\Theta)$	p.51	Landscape policy combining independent cell policies using a seperate set of parameters for each cell $\Pi^C(\mathbf{a}\|\mathbf{s},\Theta) = \prod_{c\in C}\pi_c(a_c\|s_c,\Theta_c)$
$\Pi^0(\mathbf{a}\|\mathbf{s},\theta)$	p.52	Landscape policy combining independent cell policies using a single set of parameters for all cells $\Pi^0(\mathbf{a}\|\mathbf{s},\theta) = \prod_{c\in C}\pi_c(a_c\|s_c,\theta)$

Chapter 5

Notation	Page	Description
$do(\mathbf{Y})$	p.34,71	Indicates that the variables $\mathbf{X}$ have had their value set by a causal intervention rather than an observation in some conditional distribution such as $p(X\|do(\mathbf{Y})$

Chapter 5

Notation	Page	Description
$X_1 < X_2 < X_3$	p.71	A sample ordering specifying variables are sampled in the order X_1, X_2, X_3.
$\mathtt{pa}_X = \mathtt{pa}_X^- \cup \mathtt{pa}_X^+$	p.71	In a cyclic causal network, $\mathtt{pa}_X$ are the parents of variable X which are partitioned, given a sample ordering, into $\mathtt{pa}_X^-$, the parents before X in the ordering, and $\mathtt{pa}_X^-$, the parents after X in the ordering.

Chapter 6

Notation	Page	Description
$f_c(\mathbf{a}_{-c}, \mathbf{s})$	p.90	Value of a feature $f \in F$ on cell c given actions at other cells $\mathbf{a}_{-c}$ and a landscape state $\mathbf{s}$
$\psi(a, c, \mathbf{a}_{-c}, \mathbf{s}, \theta)$	p.90	Potential function which combines the policy weights with each feature and action for a single cell: $$\psi(a, c, \mathbf{a}_{-c}, \mathbf{s}, \theta) = \sum_f \theta[f, a] f_c(\mathbf{a}_{-c}, \mathbf{s})$$

Chapter 6

Notation	Page	Description
$\pi_c(\mathbf{a}\|\mathbf{a}_{-c}, \mathbf{s}, \theta)$	p.90	The cell policy where the actions at other cells are used is a function $\pi_c(\mathbf{a}\|\mathbf{a}_{-c}, \mathbf{s}, \theta) : A^{C-1} \times \mathbf{S} \times \Theta \to \delta(A)$ A distribution over the actions A at cell c is conditioned on the landscape state $\mathbf{s}$ and actions at all other cells $\mathbf{a}_{-c}$ defined as a log-linear function: $\pi_c(\mathbf{a}\|\mathbf{a}_{-c}, \mathbf{s}, \theta) = \frac{\exp(\psi(a, c, \mathbf{a}_{-c}, \mathbf{s}, \theta))}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{a}_{-c}, \mathbf{s}, \theta))}$
$\mathbf{a}^\tau$	p.92	$\mathbf{a}^\tau$ is the action sampled at step τ during Gibbs sampling while estimating the gradient. A chain of samples $\mathbf{a}^0, \ldots, \mathbf{a}^\tau$ will be generated to estimate the gradient for a single time period t during planning for a fixed $\mathbf{s}^t$
κ	p.92	Number of steps in a particular sampled Markov chain.
ω	p.97	Maximum length Markov chains used for sampling.
$\pi_c(\mathbf{a}\|\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta)$	p.92	Transistion model of a Markov chain using the local policy π_c where values for parents earlier than c in the sample ordering come from the current sample step and parents after c in the sample ordering come the previous sample step
$\Pi(\mathbf{a}\|\mathbf{s}, \theta)$ or $\Pi(\mathbf{a})$	p.92	Landscape policy defined by the equilibrium of the Markov chain

Chapter 6

Notation	Page	Description
$p_\kappa(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^\tau)$	p.93	Probability of transitioning from $\mathbf{a}^{\tau-\kappa}$ to $\mathbf{a}^\tau$ in κ steps
$M_i(a_c)$	p.94	Marginal probability over the actions at cell c exactly marginalizing out i^{th} generation ancestors of c
σ	p.101	Landscape action for which the gradient of the equilibrium policy needs to be computed
$\hat{\Pi}(\mathbf{a}\|\mathbf{s}, \theta)$	p.102	An estimate of the landscape policy

Glossary

AAC Annual Allowable Cut

AGG Action-Graph Game

AI Artificial Intelligence

BN Bayesian Network

BP Belief Propagation

DBN Dynamic Bayesian Network

DEC-MDP Decentralized MDP

DEC-POMDP Decentralized POMDP

EBN Equilibrium Bayesian Network

ENAC Episodic Natural Actor-Critic

EPG Equilibrium Policy Gradient planning

EP Expectation Propagation

Factored MDP Factored MDP

FSSAM Forest Service Spatial Analysis Model

GA Genetic Algorithms

GBP Generalized Belief Propagation

GMDP Graph-based Markov decision process

GPI Generalized Policy Iteration

IBP Iterative Belief Propagation

IIA Iterative Improvement Algorithm

MCMC Markov Chain Monte Carlo

MDP Markov Decision Process

MPB Mountain Pine Beetle

MRF Markov Random Field

NAC Natural Actor-Critic

OR Operations Research

PG Policy Gradient

POMDP Partially Observable Markov Decision Process

RL Reinforcement Learning

SEM Structural Equation Model

Acknowledgements

Completing this thesis was only possible with the generous help of many people. I would like to deeply thank my supervisor David Poole, who always forced me to simplify and helped me to see through to the fundamental issues in anything we discussed. Most of the ideas in this thesis started from conversations with David. His insight and imagination have deeply influenced the way I think about research and I know this will serve me well in the future.

To my supervisory committee; Jim Little, John Nelson and Nando de Freitas; thank you so much for all of your helpful feedback and guidance over the years. The interdisciplinary nature of my topic led to many fascinating discussions which were very helpful and enjoyable. Other faculty have also been very helpful along the way, especially Alan Mackworth, Anne Condon, Kevin Leyton-Brown and Kevin Murphy.

Thank you to the many other graduate students in LCI who've helped in many ways from technical help, logistical advice and even emotional support, especially Michael Chiang, Jacek Kisynski, Matt Hoffman, David Buchman, Mohammad Emtiyaz Khan, Benjamin Marlin, Mike Klaas and Frank Hutter. I'd also particularly like to thank Cornel Lencar who provided the inspiration for looking at the forestry planning problem and helped me a great deal initially learning about the field.

Thank you to the always helpful staff in the Computer Science department who have helped so much over the years to smooth the path towards this goal, it would have been much harder to find my destination without your guidance: special thanks to Joyce Poon, Kletnathee Imhiran, Colleen

Diamond and Valerie McRae.

Thank you to my parents Mary and Anthony who taught me that anything is possible and that anything worth doing is worth doing right. To my sister Clara and her husband Ken, you've helped in more ways than you can imagine, thank you and remember to never stop dreaming.

To Lily, the love of my life, without whose support I literally would not have been able to do this financially, logistically or emotionally. You taught me that the only way to reach your destination is to know where you are headed and to never, ever give up. Thank you.

To Lily,
your turn!

Chapter 1

Introduction

One of the primary research questions in Artificial Intelligence is how to design an agent which can act in the world to maximize its utility over time. But what if the agent needs to act in many locations in the world simultaneously? What if there are complex relationships tying actions at different locations together? We call these types of problems spatiotemporal planning problems and they arise in many important areas of decision making in industry, environmental planning and government.

A spatiotemporal planning problem is one where an agent must choose actions at multiple locations in space at each moment in time. The agent attempts to optimize actions over some time horizon based on a model of its values about different outcomes in the world. A brief discussion of some example domains will make this more concrete and help to highlight the computational challenges.

1.1 Example Domains

The example domain we will use as a running example will be *forestry planning* [Baskent and Keles, 2005]. The spatiotemporal planning problem in forestry is to decide whether to cut trees, perform maintenance activities or leave the trees to grow in each one of thousands of small areas of the forest each year. These plans may need to forecast decades or centuries into the future to satisfy values and constraints relating to economic, logistical, reg-

ulatory and ecological issues. Values can involve a wide range of properties such as the price of lumber, the overall health of the forest or the spatial pattern of harvested areas in a landscape.

Urban planning is another spatiotemporal planning domain which involves deciding when and where different types of urban development will be allowed or forbidden. Municipal governments need to regularly consider zoning regulations and plans as the population grows and spreads. Planners need to consider likely growth and migration patterns and decide which development plans for housing, business, education or other infrastructure to approve and which areas may need encouragement to develop. Part of the planning problem comes down to assigning zoning types to all the regions of an urban area. Properly forming a plan requires considering spatial relations between parts of the urban area as well as uncertainty about the prediction models of population growth and future business needs.

The problem of infectious disease control [Best et al., 2005; Clements et al., 2006] is another good example of spatiotemporal planning. Treatment plans for diseases need to specify where and how much medicine to distribute given a limited supply. Finding an optimal plan could require considering predictions about the variation of the disease year to year, how it spreads and likely participation rates in different population centres.

1.2 Problem Definition and Goals

An area of space called a *landscape* is divided into a set C of decision locations called *cells*. The decision problem is to choose, for every cell $c \in C$, an action a_c from a set of choices A. Actions need to be decided for each point in time over a finite planning horizon. Each action a_c can depend on the state of the world in the cell as well as on the states of other cells or actions taken in other locations, such as neighbouring cells.

A *landscape state* is an assignment of a state to every cell in the landscape. Similarly, a *landscape action* assigns an action to every cell in the landscape. A series of actions will be judged based on a value model which may take any combination of states and actions into account.

Some of the computational and modelling challenges that arise in spa-

tiotemporal planning problems can be understood by considering the properties of these problems, such as: the size of the domain, the complexity of the value and dynamics models, the presence of uncertainty, and need for solutions which are interpretable and usable by practitioners.

1.2.1 Problem Size

The space of possible landscape states and actions is often extremely large in spatiotemporal planning problems. The number of possible landscape states is the number of ways states can be combined from every cell. For example, for a spatial planning problem with 1000 cells with each cell having a binary state, the number of landscape states is $2^{1000} \approx 10^{300}$. One way to deal with this is to use a factored form for the state so that states do not need to be enumerated.

In spatiotemporal planning this same exponential blow-up also occurs for actions; this makes any planning method that relies on enumerating actions impractical to use. Yet, a very large number of actions does not necessarily make a problem intractable if all the cells are independent. For example, linear programming methods are regularly used to solve problems with thousands of variables but an assumption is usually made that the variables are independent or at least that only the counts of variable values are needed. If the identity of combinations of correlated cells is important then finding an optimal plan directly can be intractable. This *spatial interdependence* between locations is an important requirement to satisfy in the problem we are considering and leads us to the need for a new kind of spatial policy representation. Spatial interdependence arises from a number of different sources, including the value model and dynamics for the problem.

1.2.2 Value Models

Planning is the process of finding ways to act optimally, or as close as possible to optimal, relative to some value model. In spatiotemporal planning, that value model can have local and non-local components. A *local value component* can be computed by independently summing the values at each cell. A *non-local value component* requires some combination of cells to be

computed such as the count of all cells with some property. Non-local value components could also be spatial, attaching a value to a particular spatial arrangement of cells, such as all adjacent cells having the same action. Non-local value components make the planning problem more challenging by creating interdependence between cells.

1.2.3 Uncertainty

Planning in the presence of uncertainty about the current state is a notoriously hard problem. However, even if the state is treated as known, the dynamics in environmental planning problems will often be stochastic. When the dynamics of the system are noisy, we need to reason about the expected value of a policy; deterministic planning methods will be difficult to apply here.

1.2.4 Existing Dynamics Models

In complex planning domains, researchers develop sophisticated simulation models for the dynamics of the problem. We would like to reuse this extensive modelling work by utilizing existing simulators, when available.

Using existing simulators is challenging because they may be difficult to modify or access directly and often do not conform to the ideal of a state transition function that simply follows the actions provided to it. The explicit transition models underlying existing simulators are complex and are often not designed to be accessible in a convenient analytical form. We treat the dynamics as a black box that takes actions and states as input and provide a future state of the world as output.

The dynamics in environmental planning domains such as forestry often model processes which move across the landscape such as pests or fire. This further adds to the interdependence between locations in the landscape. Generating simulations can also be expensive, so we need to treat simulation data as a precious resource to be used as effectively as possible.

1.2.5 Interpretability and Usage

A major goal of any decision support system is to alleviate policy makers from activities that are repetitive, computationally difficult or outside their

area of expertise. In environmental planning problems where the policy makers are generally domain experts, it is important to have access to tools which take in expert knowledge with as few limitations as possible and use that knowledge to provide instructive feedback that can help with human decision making. Policy makers should be able to focus on modelling their domain and defining their values rather than spending time modifying optimization or planning algorithms. They should be able to focus on modelling features which describe the important aspects of the domain for decision making. They should be able to focus on defining their values concretely and accurately. A decision support system can use these features and values to search for optimal policies.

In spatiotemporal planning problems many different stake-holders are often involved and the final policy will generally be used as an input into a broader planning process. Therefore, it is also important to present a policy which can justify its advice and which can be interpreted directly. The policy should be usable by practitioners on the ground to inform their decisions given their surroundings. It would also be beneficial if the policy can be queried interactively with different "what-if" questions to aid decision making.

1.2.6 Notation

Throughout this thesis, upper case letters, such as X, will be used to define sets or random variables while specific instantiations will be in lower case so that $X = x$. Variables in bold will indicate a set of random variables or a vector containing some set, state or other entity for each cell in a landscape. For a Boolean variable, the truth value will be denoted as the lower-case of the variable, for example, $A = true$ is written as a and $A = false$ is written as $\neg a$.

1.3 Our Approach

This thesis motivates, defines and evaluates a general approach for spatiotemporal planning. A stochastic policy is defined that models a spatial landscape distribution using locally parametrized policies. The optimization

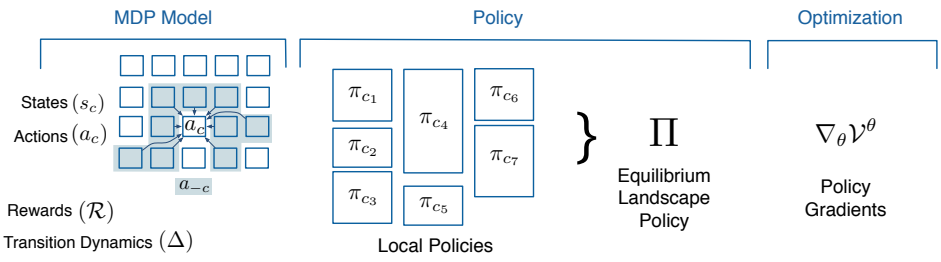


Figure 1.1: Overview of the Equilibrium Policy Gradient planning framework.

method used is policy gradient planning, a type of reinforcement learning, which allows a policy to be improved based on simulated experience without needing to enumerate all states and actions.

Our approach can be can be broken into three general components (summarized in Figure 1.1):

- **Model** - The model includes: all the information about the state of the world at each cell; the actions that can be taken at each cell; an external simulator to model the dynamics of how the world changes when actions are taken; and a value model which defines utilities for what is important to the user. The model is defined as a factored Markov Decision Process with states and actions divided into components for different locations.

- **Policy** - A rule for deciding what action an agent should take, given the current state of the world. The focus is on how to build a landscape policy for all cells that is composed of locally defined policies from each cell. A policy defined in terms of the actions of an agent at a particular location should be directly interpretable and usable by decision makers and practitioners on the ground in environmental planning problems.

- **Optimization** - A method for improving the policy as measured by the agent's value model. Given the scale of the problems in environmental planning it is often infeasible to compute a single optimal policy. Thus, the goal is to produce high value policies to be used as an aid to human

decision making.

1.4 Contributions

The contributions of this thesis relate to defining an interpretable landscape policy which can represent spatial interdependence between locations and using this policy for spatiotemporal planning. Our approach is developed in chapters 4 through 7 and can be briefly summarized in three parts.

1.4.1 Cell Policy and Landscape Policy

For each cell in the landscape, $c \in C$, there are a number of actions, A, that can be taken. For example, actions in forestry include cutting all trees at that location, treating the trees for disease or doing nothing. A local policy is a probability distribution over these actions for each cell:

$$\pi_c(a_c) = \text{distribution over actions } a_c \in A \text{ for cell } c \tag{1.1}$$

Local policies for multiple locations could share the same distributions or could be defined separately for each location.

A landscape action is the combined set of actions at all locations in the landscape. A landscape policy is a distribution over the joint actions taken at all locations in the landscape. Consider the simple case where the local policy for each cell is independent of all other cells. The landscape policy in this case would be

$$\Pi(\langle a_1, a_1, \ldots, a_C \rangle) = \pi_1(a_1)\pi_2(a_2)\ldots\pi_C(a_C) \tag{1.2}$$

This landscape policy will be used in Chapter 4 as the starting point for implementing policy gradient planning.

1.4.2 Spatial Landscape Policy as a Cyclic Causal Distribution

An important requirement for spatial planning is that we go beyond independent policies such as (1.2) and model interdependence between locations. Chapter 5 investigates how to model spatial distributions that could be used

for planning. Our formulation uses local policies to define a *cyclic causal model*. The desired distribution over landscape actions is the equilibrium of a Markov chain constructed from local policies. Computing this equilibrium directly can be computationally expensive. An iterative improvement algorithm is presented that exploits the structure in the cyclic causal model to build a dynamic model with structured latent variables called an Equilibrium Bayesian Network (EBN). An EBN can be used to compute marginal queries from the equilibrium distribution. This approach generalizes Structural Equation Models (SEMs) [Pearl, 2009] commonly used for causal modelling. Our approach can represent the same causal distributions as an SEM as well as handling cyclic causal distributions. Analysis shows cases when this approach produces exact answers and when answers are approximate.

1.4.3 Equilibrium Policy Gradient Planning

Existing research in planning does not extensively look at the problem of optimizing policies which are themselves the equilibrium distributions of complex systems. Chapter 6 presents equilibrium landscape policies as a natural way to represent complex spatial interactions compactly using interacting, conditional probabilities. To be practical for problems with many cells, the policy is modelled as a Markov chain and stochastic simulation is used to perform the operations of sampling actions from the policy, computing marginal probabilities of actions, and computing the gradient of the policy with respect to its parameters.

Chapter 7 presents an algorithm for spatial planning, Equilibrium Policy Gradient planning (EPG). EPG extends the standard Policy Gradient (PG) algorithm [Sutton et al., 2000] to work with equilibrium landscape policies and external dynamics in spatiotemporal domains. As far as we are aware, this is a novel approach to spatiotemporal planning. The algorithm is evaluated on a forestry planning problem with 1880 cells using an existing forest simulator in use by the British Columbia Forest Service.

The remaining chapters consist of background needed for understanding these contributions and discussion of related and future work. Chapter 2 provides a detailed overview of forestry planning as an example of a spa-

tiotemporal planning problem. Chapter 3 presents a summary of relevant planning and modelling techniques from Artificial Intelligence. Chapter 8 presents an overview of other related work in planning and AI. Finally, Chapter 9 outlines possible future research directions and summarizes the contributions of the thesis.

Chapter 2

Example Planning Domain: Forestry

The motivating spatiotemporal planning problem and running example throughout this thesis will be that of forestry planning. In this chapter we define forestry planning, discuss the computational problems that arise and give an overview of some methods currently being used. This chapter should provide an introduction to forestry planning for AI researchers as well as provide a case study of a typical spatiotemporal planning problem in an environmental planning domain.

Forestry planning is the problem of deciding how to sustainably manage forests while balancing complex economic, logistical, regulatory and ecological values. Decisions need to be made about management options for thousands or hundreds of thousands of forest locations each year over a planning horizon of centuries.

This chapter describes forestry planning as it is practised in British Columbia, Canada. Forestry is a very important industry in British Columbia generating about 13% of the province's GDP and employing around 200,000 people [Council of Forest Industries, 2007]. Large regions of forest of up to several hundred thousand hectares are licensed by the government to forestry companies that manage and harvest the forest. The government regulates management activities by setting a maximum annual allowable cut and spec-

ifying areas that are off limits for harvest. Regulations also specify spatial constraints to avoid a high density of cut areas and to protect wildlife and habitats. Violations of these constraints are enforced with fines and possible revocation of a company's license.

In recent years, the field of forestry planning has been undergoing a transition as practitioners focus more on sustainable harvesting in the presence of uncertainty, complex spatial constraints and deeper interaction with the public about the values placed on different uses for forests. This makes our research timely from the forestry planning point of view.

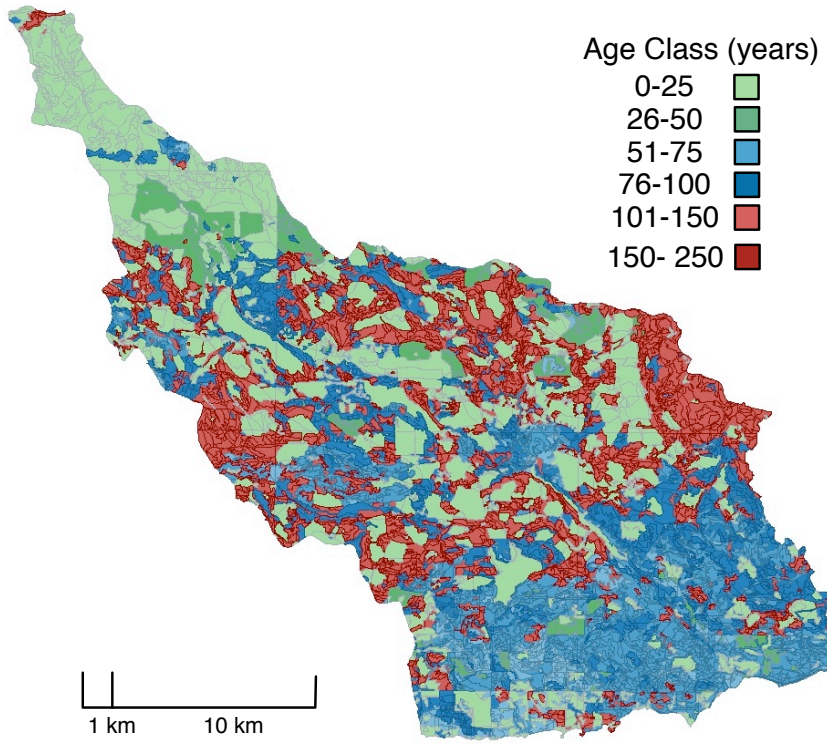


Figure 2.1: A region of forest in British Columbia with 1880 cells. Each colour indicates the dominant age class, in years, of trees in the cell.

2.1 Describing the Forest State

The total harvestable area of British Columbia is over 18 million hectares(ha) [Eng et al., 2004a]. Planning activities are regularly carried out by forestry companies or the government on hundreds of thousands or millions of hectares. For practical purposes these areas are divided up into regions for different levels of planning. The smallest division of forest for our purposes is a *stand* which can range in size from 1-50 ha. Stands correspond to the cells in a spatiotemporal planning problem, so we will use the more general term cells to refer to stands from now on.

A common problem instance in forestry planning would need to deal with thousands of cells. For provincial level planning, several hundred thousand cells would need to be considered. Each cell has a number of features describing the state of the forest within that cell. The boundaries of each cell are defined by a polygon chosen to produce a cell with fairly homogenous feature values throughout its area. This means the boundaries of a cell can be very irregular. The number of adjacent neighbours to a cell can range from 2 to 30 in the example shown. Some example features of a cell are shown in Table 2.1. Figure 2.1 shows a sample area of forest near Kelowna, BC, where the colours indicate the value of an age feature describing the dominant age class of trees in the cell. Figure 2.2 shows the same landscape where colours indicate the species of tree that is dominant in each cell.

Roads, streams and lakes will usually be defined by their own bounding polygons or delineate boundaries of the polygons for cells. Developing methods for defining the boundaries for cell polygons is a difficult and active area of research in its own right [Coburn and Roberts, 2004; Lowell, 1999]. For simplicity, we will assume that the polygons defining cells have already been created and are included with other landscape data.

2.1.1 Pest Infestations

One major impact on forest health is insect infestation such as the Mountain Pine Beetle (MPB) [Eng et al., 2004b]. MPB burrow under the bark of pine trees laying eggs, cutting off nutrients and leaving a deadly blue fungus that kills the tree. MPB are an endemic species in BC, however, in recent

decades, a lack of cold winters and the large number of older trees resulting from years of forest fire suppression have provided the MPB population with the conditions they need for an explosive epidemic. Cutting down trees before a brood spread can kill the beetles but the rice-sized beetles are hard to detect until a year or two after an attack. That is when the thousands of killed trees are easily spotted by their distinctive red colour. The infestation has devastated the forests of BC, wiping out most of the harvestable pine in the past 15 years and is beginning to spread into northern Alberta.

- Area of cell (1-50ha)
- Volume of trees in cell (0-100,000m^3)
- Dominant tree species (pine, spruce, balsam)
- Distribution of tree species (percentage coverage for each species)
- Number of trees (0-50,000)
- Distribution of age of trees (0-250 years)
- Presence of a road or stream (boolean)
- Presence or severity of pests such as the Mountain Pine Beetle (boolean or risk categories:low/med/high)
- Elevation of cell (0-2000m)
- Soil type
- Bio-climatic zone (the climate classification representing the expected weather conditions in an entire region of the province)

Table 2.1: Some examples of basic features defined for every cell in a forestry planning problem.

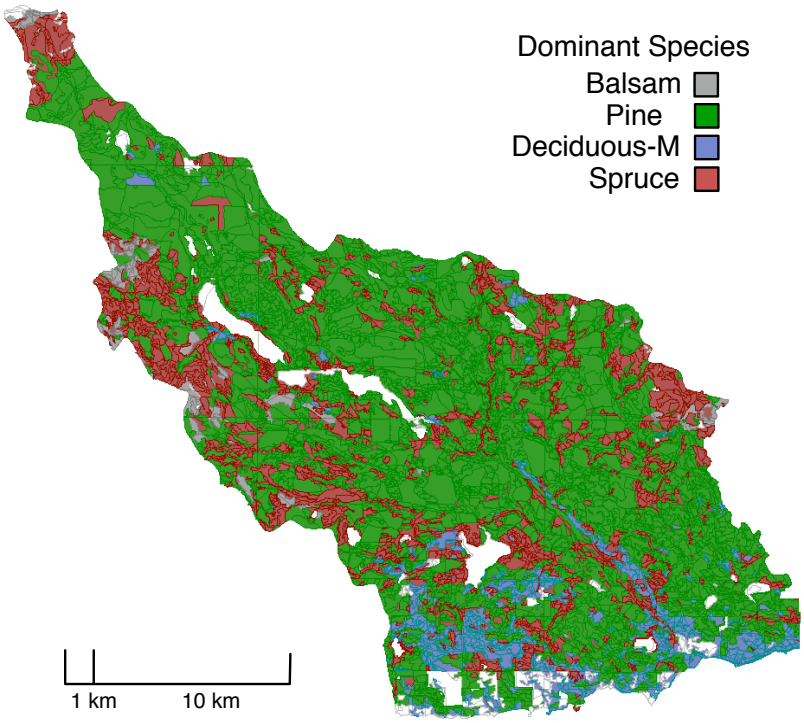


Figure 2.2: A region of forest in British Columbia used for testing. Each colour indicates the dominant species of trees in the cell.

2.2 Actions

The difference in scale between the entire landscape and the actions at a single cell necessitates a division of labour and expertise that leads to a hierarchical planning structure. The highest levels of general planning are referred to as *strategic planning*. Strategic plans are generally not concerned with individual cells and look at long time horizons. *Tactical planning* focuses on satisfying logistical constraints within smaller regions and assigns particular actions to individual cells over a shorter time horizon. This hierarchical planning structure is an important feature of forestry planning [Davis et al., 2001].

There are a small set of actions available at each cell each year. The entire cell could be *clear-cut*, which involves cutting all the trees and replanting. The cell could be partially cut either by clear-cutting parts of a cell or *thinning* the number of trees throughout the cell to reduce the distance between trees. Thinning can help reduce the spread of pests [Whitehead and Russo, 2005]. *Salvaging* available trees which are dead due to fire or pests is another action since this wood is handled differently from normal harvesting of healthy trees. *Single tree treatments* include poisoning trees or cutting and burning individual trees to reduce pest infestation, disease or fire risk. Finally, in the vast majority of cells, no action will be taken; the forest will be allowed to grow for another time period.

The material obtained from the forest is known collectively as *timber* and is sold directly, or after further processing, as different types of forest products such as *logs*, *lumber* (processed from logs sent to mills) and *wood chips* (scraps and chipped logs used for producing fabricated wood and pulp for paper mills).

Forestry planning methods often reason about entire treatment schedules, called *prescriptions*, rather than individual actions on a cell at one point in time. Prescriptions are pre-computed based on a likely scenario of future development of the cell and describe a fixed series of actions to take in a cell without taking new observations into account. An example of a prescription is a pre-determined harvest schedule that defines which year to perform an initial partial cut and which year to perform a followup cut of the remaining trees.

2.3 Value Model

In forestry planning, the value model contains local and non-local components which could be utility values to be optimized or constraints on actions to be satisfied.

Values that planners seek to optimize could include the total forest size, harvest yields, risk of fire, risk of pest outbreaks, costs of building roads, the market value of lumber and chips, employment levels in different communities and recreational uses of forests.

Sometimes, it is convenient to express values as constraints on actions. Constraints come from various sources, such as fundamental logistical requirements or government regulations. Constraints can be hard, banning certain types of actions entirely, or soft, penalizing policies that deviate from some desired goal. Some types of constraints commonly used in forestry planning are:

- *Even flow* constraint: The volume of timber harvested year to year may not vary more than some constant amount. This is an attempt to maintain a sustainable forest level both for stable revenue as well as ecological balance. Similarly, an even flow constraint could be placed on the forest population level itself rather than on the amount harvested.

- *Green-up* constraint: Cells that have been recently harvested and replanted are designated as being in a green-up phase which cannot be cut for some fixed number of years.

- *Spatial Cutting* constraints: These are restrictions on the spatial arrangement of cuts allowed in the forest used to limit impact on wildlife habitats and ecosystem health. A common example of this is a constraint against adjacent cutting requiring that adjacent cells not be cut in the same year. This is often tied to the green-up period so that if cell c is cut then all cells adjacent to c may not be cut until the green-up period for c is complete.

- *Viewsheds* : Restrictions on the visibility of cuts from populated areas [B. C. Ministry of Forests, 2005].

- *Annual Allowable Cut (AAC)* : The total volume cut each year may not exceed the AAC set for each region defined by the province but also must be more than the minimum required to maintain a license to operate.

- *Road Access* : The areas to be harvested must have the necessary road access to carry out the plan.

- *Biodiversity age bounds* : These are targets for what portion of the forest must fall within a particular age classification (such as 0-20 years old, 21-40 years old, etc). An example of a biodiversity age bound is a constraint that at least 30% of the forest is old growth trees (over 200 years old). Another could be that no more than 25% of the forest is ever less than 10 years old. The purpose of this type of constraint is to maintain a balanced ecosystem. It also makes economic sense by providing more regular yearly harvest levels and increasing the diversity of the tree population which discourages spread of pests and disease. Mathey and Nelson [2007] provide a good description of these constraints in more detail.

- *Connectivity* constraints : These could require that spatially distant habitats for migrating animals have connected corridors of natural landscape [Eng et al., 2004a]. Computing optimal corridors to aid multiple species is an active area of research in computational sustainability [Lai et al., 2011].

2.3.1 Accessibility of Value Models

Information about the utilities and constraints that make up the value model in forestry planning problems are usually embedded in various reports and studies. Some values are considered by strategic planners setting targets for a region while others are considered by those carrying out planning on the ground. Rarely are all the relevant components of the full value model written down in one location. This can make it challenging to discern the entire value model being used for the entire planning process. Boyland et al. [2005] provides a good overview of some objectives used in forestry planning.

2.4 Dynamics

Researchers in forestry planning have developed numerous detailed simulations of their domain through extensive study of tree growth patterns, forest ecosystem life-cycles and the epidemiology of pests. Insect infestations such as MPB create a source of spatial correlation between stands in the dynamics as they spread across the landscape each year [Whitehead and Russo,

2005].

One common use of simulators is to allow human planners to search through different parametrizations manually and view the resulting plan that would result so that they can compare outcomes [Davis et al., 2001].

2.4.1 Simulation-Planners

In practice, planning and simulation are often combined together into one software system we will call a *simulation-planner*. A simulation-planner takes some input from the user, such as desired harvest levels, and a starting state. The system then does further optimization of the actions while simulating an entire trajectory. One way to think about simulation-planners is that the full policy π is divided into two parts, π^{μ} and π^{θ}. The policy π^{θ} is an external policy provided by the user to guide actions. The user provides some guidance about how they want the actions to be carried out using π^{θ}. These actions are considered and the system then applies other expert knowledge hardcoded as π^{μ}, which is internalized within the existing simulator as a constraints to be satisfied. We have no control over this part of the policy. These constraints check the input actions and modify them to maintain adherence to well known rules and regulations or hard supply constraints.

2.4.2 The FSSAM Simulation-Planner

An example of a simulation-planner that we use is the Forest Service Spatial Analysis Model (FSSAM). FSSAM is used by the British Columbia Forest Service to analyse the impact of setting different provincial and regional harvest targets and the impact of these targets on sustainable ecological and economic goals. FSSAM uses detailed models of forest growth patterns and consistency and logistical constraints that need to be satisfied during harvest.

The input to FSSAM is an initial forest state provided as a GIS map and the desired harvest level in, cubic meters, for each year. FSSAM is a complex system with many different options and modules but for our purposes it can be broken into three major phases which are carried out for each time step:

Querying the Policy, Constraint Optimization and Simulation.

2.4.2.1 Policy Phase

After the data for the current state is initialized, a module is called which chooses an ordering of the cells. The program then loops through all cells using this ordering, considering whether each cell should be acted upon or left alone. The action for each cell is already determined in the model and could be set to clear-cut, thin, treat or other conditional cutting actions. However, the choice at each cell is binary; take the predefined action at the cell immediately, or do nothing. There are a number of deterministic sorting functions built in to FSSAM for choosing the ordering in which to consider cells. These functions can sort cells ascending or descending by age, by volume or sort by other predetermined priority values. The program also has a hook to call an external policy function to determine the cell ordering at each time step. We utilize this hook to integrate with our policy in Chapter 7.

Cells can also be flagged as blocked so that they are not considered for cutting until unblocked. This is used internally to allow a green-up period after cutting.

2.4.2.2 Constraint Optimization Phase

The program attempts to cut enough cells to achieve the desired harvest target for that year while not violating any enabled constraints. Constraints can include a green-up adjacent cutting constraint, biodiversity constraints as well as maximum and minimum volume cut.

The maximum harvest volume is a number provided for each harvest year by the user as input. During the policy phase, using the built-in cell ordering modules, the program attempts to cut as many cells as possible until reaching the specified maximum harvest volume. Thus, this input is both a constraint and a target. The difference between the actual cut and the target harvest volume is often used as a performance measure *shortfall*.

2.4.2.3 Simulation Phase

Once an action has been chosen for each cell, the effects of those treatments are run through the simulation to generate an updated state for the entire forest. This updated state includes simulating tree growth and death. The program then prepares for the next iteration and returns to the policy phase.

2.5 Dividing Up Space

There are three main approaches to dividing up space into actions in preparation for planning which are commonly used in forestry planning [Davis et al., 2001]:

Non-spatial Land Classification - Each cell is tracked with its own decision variable and its own history. Each cell is treated independently, so requirements can be expressed relative to neighbouring cells.

Strata-based - Tracking each cell individually is computationally intensive in large problems, so a popular method of reducing the state space are strata-based models. Cells that have *identical feature values* are assigned to a group called a *stratum* or *analysis area*. Each stratum is associated with a number of cells and one decision variable. The assumption is that the same decisions will be applied to all cells with identical features. In this model strata are non-spatial; cells are assigned to stratum regardless of their location in space. See figure 2.3 for an example of a strata-based model. Strata work best when the features associated with cells have small, discrete domains, otherwise small differences between cells may lead to there being a stratum for each cell which would defeat the point of using strata. Strata-based approaches [Baskent and Keles, 2005] are popular because they can be converted to a linear programming problem, which can be solved with widely available optimization tools. This approach is outlined in more detail in the next section.

Spatial Land Classification - The most general approach is to fully represent the spatial relationships in the data. This is necessary for plan-

ning that needs to consider spatial habitat requirements for wildlife, dealing with infestations, building roads and avoiding buffers around water. Some work has been done in forestry planning using this approach but it seems less widely pursued than the strata based method. A naive implementation of this with one decision variable per cell is generally infeasible given the size of the problem [Davis et al., 2001]. Section 2.6.2.2 discusses some optimization techniques that are used to partially deal with space in this way.

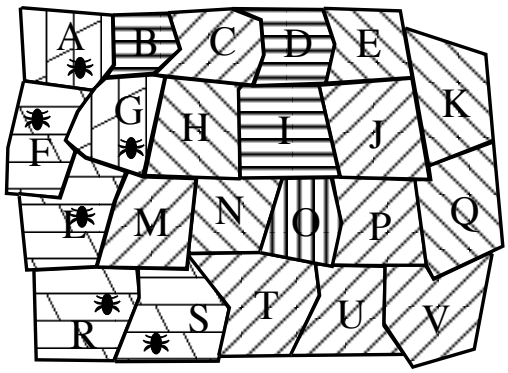


Figure 2.3: An example of a strata model applied to cells. Each labelled polygon encloses a cell. Each pattern represents a grouping of cells into a stratum. For example, the set of cells $\{C, J, M, P, T, U, V\}$, are all in one stratum. All cells in the same strata have identical features.

2.5.1 Type I vs. Type II Models

Strata based models are useful in forestry planning because they use the repetitive structure in the landscape to reduce the number of distinct regions that are considered during decision making. Two popular models for decision making are called Type I [Sessions et al., 1996] and Type II [Johnson and Sheurmann, 1977]. Both of these models begin by partitioning the landscape into strata. Generally, a number of prescriptions are then precomputed based on the likely future development of the cells. The utility of following

some prescription into the future on the cells within a particular stratum is defined by a local utility function. This utility could be based on features such as timber yield, cost or other components discussed in Section 2.3. The full objective function for the entire problem is then expressed as a linear function of all these local utilities summed over all strata and over all time steps into the future. The desired solution is an assignment of a prescription to every stratum such that this linear objective function is maximized.

The distinction between Type I and Type II models comes from the way in which harvested cells are dealt with. A Type I model keeps track of how many hectares were harvested of each stratum over in each year of planning. Suppose an area of forest is thinned, then treated for pests. After a few years pass the entire area is then harvested fully. In a Type I model this entire history is available for each cell during policy optimization and afterwards for analyzing the output. In a Type II model, cells are decoupled from their history once they are harvested. The cells that are harvested are added to a pool of regeneration variables that track how many acres have been harvested and how long ago the harvest happened. The regeneration variables act like a special stratum just for harvested cells. After the regeneration period has passed, the cell can be added back to an appropriate active stratum based on its feature values. In this approach when new trees are planted in a cell, there is no way to tell what happened to that area before the last harvest.

The value being tracked in a Type I model is the number of hectares of a stratum that are harvested each period by some prescription with the total cell history being represented. Type II models track the number of hectares of each stratum that are harvested each period and how long they will take to regenerate under some prescription with no further cell history.

Type II models have the advantage that fewer variables need to be tracked at any moment and less history is stored. This makes them more efficient for use with exact optimization methods but comes at the expense of losing access to some historical and spatial data.

2.6 Current Solution Methods

We present an overview of three approaches currently in use in forestry planning: manual simulation modelling, automated optimization of prescriptions and direct policy optimization.

2.6.1 Simulation Modelling

Simulation modelling is an interactive approach where the user provides as input an initial state in the form of a map and a specification of constraints and preferences for various cutting strategies. The software then carries out a simulation while choosing actions consistent with the user's constraints and preferences. Some example simulation tools are ATLAS (http://www.forestry.ubc.ca/atlas-simfor) and SELES [Fall et al., 2003]. The results from these simulation planners are often then fed through other tools for analysis after which the user can alter the parameters of the simulation and run it once again. The FSSAM tool described in Section 2.4.2 is used for simulation modelling by government planners to explore different scenarios that may result form various provincial harvest targets such as the AAC.

2.6.2 Optimization Techniques for Strata Based Models

A very common approach in forestry planning currently is defining planning as an *optimization problem* over prescriptions [Davis et al., 2001]. An optimization problem contains a set of variables, a set of constraints on the possible values of those variables and an objective function over the values [Nocedal and Wright, 1999]. The objective function tells us how good any assignment of the variables is. The goal is to find a variable assignment that satisfies all of the constraints and that maximizes the objective function. In forestry the variables represent cells (either individually or grouped into strata) and we must find an assignment of a prescription to each cell to determine the actions to take in that cell in the future. Since a prescription describes a series of actions over time, the optimization problem does not need to consider multiple points in time, only the assignment of an entire prescription to a cell or strata.

There are two main forms of optimization used in forestry planning which we will describe in detail: linear programming, which assumes all cells or strata are independent and exactly optimizes a prescription assignment; and stochastic local search, which searches for an optimal assignment which allows for more consideration of spatial interdependence.

2.6.2.1 Linear Programming

If certain conditions are met then *linear programming* [Dantzig, 1963] can be used that provides guaranteed globally optimal solutions quickly. The main conditions that need to be met are a linear objective function and linear constraints on the value of that function of the following form:

$$\begin{aligned} &\text{Maximize } c^T x \\ &\text{Subject to } Ax \leq b \\ &\text{Where } x \geq 0 \end{aligned}$$

The coefficients c^T define the objective function. The matrix of coefficients A define linear constraints over the variables bounded by coefficients b. Many software packages are available for solving linear programming problems such as CPLEX (http://www.ilog.com/products/cplex/).

The strata based Type I and Type II models meet the conditions for a linear program. The objective function is linear and the variables to optimize are the decision variables for each stratum which assign a particular prescription to that stratum. Given a prescription for every stratum the objective function returns the utility of the resulting plan carried out into the future. Constraints on actions are expressed as linear inequalities on variables. Examples of linear constraints would be bounding the total area cut to be less than the annual allowable cut or bounding the total volume of salvaged dead trees to be more than some constant amount.

The forestry planning tool Woodstock [Walter, 1993] uses a Type II model and outputs a linear program definition that is solved with an external solver such as CPLEX. Type I systems are still often preferred because more data is available for analysis later on. The US forestry service's tool,

Spectrum, uses a Type I model [Kent et al., 1991].

Linear programming systems have a limitation in that they cannot easily solve problems with an inherently spatial distribution. These distributions, such as MPB infestations or spatial constraints require groups of cells to be considered simultaneously while optimizing. But LP requires that all variables be linearly independent. Regional context can be partially obtained in LP systems by defining constraints specifically on spatial features such as proximity to streams or roads [Kurttila, 2001]. Another approach, which is used by an add-on tool for Woodstock, called Stanley, is to solve the problem first with LP and then satisfy spatial constraints by use of Lagrangian relaxation [Sessions et al., 1996]. This approach can produce solutions that satisfy spatial constraints but it searches only locally from the non-spatial LP solution. So it is possible that the solutions found are not near the optimal.

2.6.2.2 Stochastic Local Search

When faced with spatial constraints or non-linear objective functions, forestry planners increasingly turn to approximate optimization techniques to find solutions. Most of these techniques fall in the family of *stochastic local search (SLS)*, also referred to as *meta-heuristics*. SLS algorithms make random moves to iteratively approach better solutions. The algorithms often find good solutions quite quickly but generally provide no concrete guarantees of finding the globally optimal solution. SLS algorithms are more flexible than LP in the way the objective functions and constraints are defined. This flexibility makes SLS attractive to forestry planners who need to express spatial constraints and complex objective functions combining several goals or expressing nonlinear values. The last decade has seen an explosion of interest in using SLS in forestry planning.

We will describe the general concept of SLS algorithms and list a few of the popular variants as they are used currently in forestry planning [Pukkala and Kurttila, 2005; Baskent and Keles, 2005].

In the most basic type of local search, a variable is initially set to some random starting value and then moved to another local value in state space

that improves the objective function. The meaning of "local" depends on the state space, it could mean small changes in values along particular dimensions. A new state is evaluated using an objective function and if the new value is better than the current value of the variable then the new value is kept. This process is continued for all variables until a stopping criterion is met. The stopping criterion could be that the value changes by less than some minimum amount between iterations or that a maximum number of iterations has been reached. There are no guarantees in this basic form of local search that the solution found is near the optimal solution.

Variants on the basic local search algorithm provide ways to improve the quality or speed of search or provide guarantees of optimal solutions. A few of the major variants are:

Simulated Annealing - Simulated annealing [Kirkpatrick et al., 1983] tries to avoid the local maxima that pure local search can get stuck in. It does this by randomly choosing sub-optimal local search steps from time to time. The probability of a sub-optimal step is related to the "annealing temperature". This temperature is a randomization parameter that is slowly lowered following a provided cooling schedule. It has been shown that this algorithm converges to a global optimum given an infinitely slow cooling schedule. In practice [Bettinger et al., 2002] it produces very good results with faster schedules but is usually slower than standard local search methods.

Tabu Search - Tabu search [Glover, 1986] is local search carried out with the addition of a *tabu list* of recent values. The idea is to avoid moving to states that are on the tabu list in order to avoid becoming stuck in loops around local optima.

Genetic Algorithms - Genetic Algorithms (GA) [Holland, 1975; Goldberg, 1989] explore many potential solutions. New sample solutions are considered at each step by locally varying existing solutions and combining pairs of solutions to produce hybrid "offspring" that have some of the properties of both original solutions. The rules for creating these offspring solutions are inspired by biological genetic rules and

each solution is often described as a chromosome composed of genes, which encode the values assigned to particular variables.

To demonstrate how meta-heuristics are used in forestry planning, consider the case of simulated annealing [Lockwood and Moore, 1993; Pukkala and Kurttila, 2005]. The basic structure is usually as follows:

1. Generate a small set of plausible treatment schedules for each cell using a forestry simulator. The treatment schedule is a prescription that will define the actions that can be taken in the cell into the future.

2. Randomly assign a prescription to each cell and start with a high annealing temperature. Each improvement step will initially have a high chance of randomly choosing a lower value state.

3. Loop through the following until the stopping criterion is met:

   - Iterate through all cells and try another prescription for that cell.
   - If the objective function gives an equal or higher utility for the new prescription on this cell, then assign the prescription to the cell. Otherwise, randomly choose to assign it or not based on the annealing temperature.
   - Lower the annealing temperature.

An example stopping criterion could be that the assignment of prescriptions to cells has not changed over many iterations.

It is generally assumed when using these SLS techniques that the actions to be chosen are prescriptions (entire treatment schedules) rather than atomic actions at different points in time. Comparative studies [Pukkala and Kurttila, 2005; Liu et al., 2006] have indicated that some of these techniques that iterate through all cells are fast for simple problems but cannot find good solutions for problems with spatial constraints. Simulated annealing is generally slower than simpler local search methods but does better at solving spatial problems [Baskent and Keles, 2005; Bettinger et al., 2002]. There is discussion in forestry planning over whether genetic algorithms find

better solutions in spatial problems [Pukkala and Kurttila, 2005] but it is generally agreed that they are too slow to be useful, as they are orders of magnitude slower than simulated annealing.

2.6.3 Policy Search Methods

Researchers in forestry recognize the need to develop methods for planning that deal with some of the complex issues of spatial interdependency and uncertainty that arise in their decision problem [Kimmins et al., 2005; Baskent and Keles, 2005]. Two ways to progress on this goal would be moving away from the strata based simplified form of the problem and shifting the focus to optimizing policies rather than prescriptions. This could come in the form of modifying harvest schedules during optimization or by optimizing a general policy which defines how to act at different times based on the observed conditions.

2.6.3.1 Cellular Automata Approach

Mathey and Nelson [2007] describe a method which addresses the problem of forestry planning under spatial constraints such as biodiversity constraints. They use *cellular automata* to model each cell as an agent which attempts to optimize its local management schedule. The state of each agent is a management schedule over the entire planning horizon. Each agent also has a transition function which updates the management schedule by performing a linear optimization on when to cut that cell. This optimization is performed relative to an objective function with weights updated based on local conditions as well as conditions and actions in other cells nearby.

This method is described as being similar in approach to simulated annealing with the major difference being that local as well as global objective values are integrated into the local cell optimizations. In simulated annealing, each change in management schedule for a cell is judged by its impact on the global objective functions. Since there is noise in the dynamics of the system this algorithm does produce slightly different outcomes each run, so the authors propose performing multiple runs to produce a range of outputs and produce a confidence level for a set of plans.

2.6.3.2 Genetic Algorithms for Viewshed Design

Chamberlain and Meitner [2009] implemented an automated planner for improving the visual aesthetics of a harvest pattern using genetic algorithms. Their algorithm divides the landscape into a grid of cells and iteratively improves the cutting pattern within a single time period. This is done by using spatial masks representing local cut patterns that are used to generate a landscape harvest pattern. These masks are stochastically varied and merged to improve the value of the resulting harvest pattern. The value model takes into account several spatial patterns that research has shown people find visually appealing and weights occurrences of these patterns more highly when searching for a harvest solution.

2.6.3.3 Optimization of a General Policy

Forsell et al. [2009b] describe a forestry planning problem of minimizing wind damage to trees. They apply their Graph-based Markov decision process (GMDP) model to this domain by using extensive domain knowledge to simplify the problem and solve portions of it with linear programming. These exact solutions are then used as solutions to subproblems in an iterative process of improving a general policy. In [Forsell et al., 2009a] they compare their methods to a number of linear programming and reinforcement learning approaches.

The authors also point out there is a need for scalable, model-free planning methods that can take advantage of existing simulators for natural resource planning problems.

Chapter 3

Background

This chapter gives an overview of four important techniques from Artificial Intelligence research that will be used throughout the thesis:

- Markov Chains
- Probabilistic Graphical Models
- Markov Decision Processes
- Policy Gradient planning

3.1 Markov Chains

Markov chains are a general representation tool for modelling dynamics, stochastic processes and form the basis for research in a wide variety of fields. A *Markov chain* of length T is a sequence of random variables $\langle X_0, \ldots, X_{\mathrm{T}-1} \rangle$ over a set of states X. A transition function $\mathcal{T}(X_t|X_{t-1})$ gives the probability of transitioning between states in a single step. A chain has the Markov property if the probability of future states are independent of past states given the present. $\mathcal{T}$ can also be expressed as a $|X| \times |X|$ *transition matrix* such that all the rows sum to one.

A Markov chain is *aperiodic* if there is no state that the chain returns to after a regular interval. A Markov chain is *irreducible* if every state can eventually be reached from every other state. If a Markov chain is aperiodic

and irreducible then it is called *ergodic* which means that it eventually will visit all states. Let $\delta(X)$ be a probability distribution over set X. Ergodic Markov chains are guaranteed to converge to a unique stationary distribution $w = \delta(X)$ described by the following balance equation also known as the Chapman-Kolomgorov equation [Bertsekas, 2001]:

$$w = w\mathcal{T} \tag{3.1}$$

One way to find w is to run the chain forward many steps. For some arbitrary initial distribution w_0 the probability after one step is:

$$w_1 = w_0\mathcal{T}$$

After t steps the distribution is

$$w_t = w_0\mathcal{T}^{t-1}$$

In the limit the initial state w_0 will be forgotten and the chain will converge to

$$W = \lim_{\kappa\to\infty} \mathcal{T}^t \tag{3.2}$$

where W is a matrix with the equilibrium distribution w in each row.

Another way to express this convergence which will be useful in chapter 5 is by expanding the Chapman-Kolmogorov equation showing all the calculations happening as the transition matrix is iterated:

$$w_1(X_1) = \sum_{X_0} w_0(X_0)\mathcal{T}(X_1|X_0) \tag{3.3}$$

$$w_t(X_t) = \sum_{X_t} w_{t-1}(X_{t-1})\mathcal{T}(X_t|X_{t-1}) \tag{3.4}$$

$$= \sum_{X_t}\sum_{X_{t-1}}\cdots\sum_{X_0} w_0(X_0)\mathcal{T}(X_t|X_{t-1})\ldots\mathcal{T}(X_1|X_0) \tag{3.5}$$

Essentially, we are computing the probability of the t-step transition from

X_0 to X_t by summing over all the possible paths in intervening steps.

3.2 Probabilistic Graphical Models

In a flat representation of a probability distribution, a single variable X represents the entire state space. In a factored representation, a state is represented by multiple variables. Bayesian Networks can be used to represent this factored form and their dynamic generalization Dynamic Bayesian Networks can be used to represent stochastic dynamic processes with factored states.

3.2.1 Bayesian Networks

A *Bayesian Network (BN)* [Pearl, 1988], or Bayes net, is a directed acyclic graph that represents the conditional independence relationships amongst a finite set of random variables, $\{\mathbf{Y} = Y_1, \ldots, Y_n\}$. Figure 3.1 shows a Bayesian network with five random variables each represented by a node in the graph. The *parents* of a node Y_i are those nodes connected to it through incoming arcs and are denoted $\mathtt{pa}(Y_i)$. Variables can be *observed* by fixing them to one of the values in the their domain; all other nodes are refereed to as *unobserved*.

Bayesian networks have a number of useful properties for modelling probabilistic variables. A node in a Bayesian network is conditionally independent of its non-descendants given its parents. So Y_3 is independent of Y_0 if Y_1 has been observed. If a descendant of an unobserved node is observed then it informs us about the value of that node (e.g. observing Y_2 will influence the distribution of Y_0 but not of Y_4). Without any observations, the variables Y_1 and Y_4 are independent of each other. However, if Y_3 is observed then its parents become interdependent since they both could have influenced the observed value. This property is one of the causes of complexity in the dynamic Bayesian network model described in the next section.

Bayesian networks allows a very compact representation when there is conditional independence between variables and there are no cycles. Each node has a corresponding conditional distribution, $p(Y_i|\mathtt{pa}(Y_i))$, which could be expressed as a table or a function, modelling its dependence on its parents.

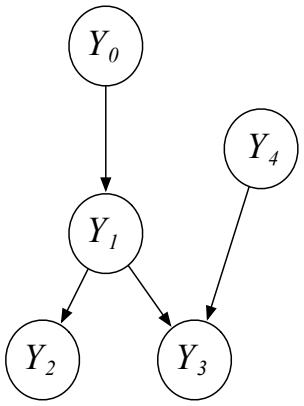


Figure 3.1: An example of a Bayesian network

The joint probability of all the variables in a BN can be factorized in the following way:

$$p(\mathbf{Y}) = \prod_{i=1}^{n} p(Y_i|\mathtt{pa}(Y_i)) \tag{3.6}$$

where nodes with no parents simply have a prior probability of $p(Y_i)$.

The process of computing the marginal probability of a set of variables in the Bayesian network is called *inference*. A variety of methods can be used to perform inference in Bayesian networks [Zhang and Poole, 1994; Dechter, 1996; Jensen et al., 1994]. Regardless of the method used, the process involves propagating information from the conditional distributions of each variable, integrating any evidence that has been observed and marginalizing out all the nodes which are not part of the query set of nodes. Bayesian network inference takes time exponential in the *tree-width* of the network. The tree-width is a measure of graph sparseness and is related to the size of the family of the most connected node in the network [Mateescu et al., 2002]. Thus, inference on large, highly connected networks is often infeasible.

It is natural think of the arcs in a BN as causal since the distributions of parent nodes influence the distributions of their children. However, causality is not necessary for a consistent interpretation of BNs; in fact it leads to

different answers for the marginal distributions [Pearl, 1993, 2009]. There is a difference between *observing* the value of a variable and *intervening* to set the value of a variable. When the value of a variable, such as Y_3, is set in a causal model, this cannot be used to infer anything about the values of Y_1. Pearl [2009] distinguishes intervention by writing the probability as $p(Y_1|do(Y_3))$ rather than the probability under observation $p(Y_1|Y_3)$. A Bayesian network can be treated as a purely *causal network* by modifying the behaviour of observations so that when an intervention occurs on Y_i all of the arcs from variables in $\texttt{pa}(Y_i)$ to Y_i are cut. The usual Bayesian network inference methods can then be used to compute marginal probabilities.

At each time step t each variable $Y_{i,t}$ has parents $\texttt{pa}(Y_{i,t})$ which could be in the current step $\mathbf{Y}_t$ or the previous step $\mathbf{Y}_{t-1}$. In the example in Figure 3.2, Y_3 depends on both Y_1 and Y_4 in each step and Y_0 depends on Y_3 from the previous step. The conditional distributions for each variable $p(Y_{i,t}|\texttt{pa}(Y_{i,t})$ define the transition model of the Dynamic Bayesian Network (DBN) and normal BN inference on the entire DBN can yield the distribution over the variables $\mathbf{Y}$. Arbitrary Markov chain transition models can be represented this way very compactly if there is an independency structure to exploit.

3.2.2 Dynamic Bayesian Networks

A *Dynamic Bayesian Network (DBN)* [Dean and Kanazawa, 1989] models a Markov chain with the state $\mathbf{Y}_t$ represented by a Bayesian network over variables $\mathbf{Y}_t = \{Y_{i,t}|i \in [0, n)\}$.

There is a limit on the size of problem that can be computed exactly using a DBN due to the complexity of performing inference. During inference the variables in the DBN become entangled with each other so that their probabilities are all correlated [Koller and Friedman, 2009b]. The problem is that even if two variables within a single time step of the DBN are conditionally independent, their marginal distributions will become correlated over time either due to common ancestors at some point in the past or the effects of observations. Inference takes time exponential in the size of the largest connected clique of variables in a BN. In a DBN all of the variables become connected, so exact inference can quickly become infeasible directly

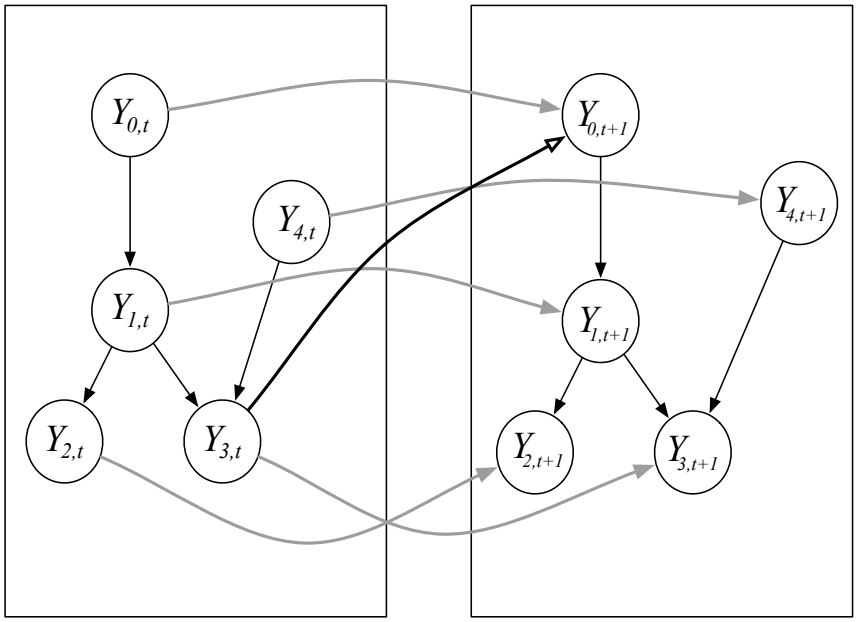


Figure 3.2: An example of a two-step slice of a Dynamic Bayesian Network

on the DBN structure itself.

Chapter 5 introduces a new approach for performing inference in a DBN when the DBN is seen as the equilibrium distribution of a causal system with cycles. Significant work has been done on finding approximations of the belief state of a DBN to improve inference, some of this work is reviewed in Chapter 8.

3.2.3 Approximating DBNs

When exact computation of the equilibrium of a Markov chain or DBN is infeasible, Markov Chain Monte Carlo (MCMC) methods can be used to approximate the equilibrium distribution. One of the simplest approaches is *Gibbs sampling* [Geman and Geman., 1984]. The Gibbs sampling algorithm begins by setting some initial value for all the variables in $\mathbf{Y}_0$. For each time step t, each variable is sampled in turn using the current values of the other

variables

$$y_{i,t} \leftarrow p(Y_{i,t}|\mathtt{pa}(Y_{i,t})) \qquad \forall i \in [1, n)$$

The samples acquired from this algorithm can be used to compute any marginal distribution over the variables.

A more general approach is the Metropolis-Hastings MCMC algorithm [Metropolis et al., 1953; Hastings, 1970] which uses a likelihood function to choose which samples to reject and which to keep. Gibbs sampling can be seen as an instance of this approach where all samples are kept.

3.3 Markov Decision Processes

A Markov Decision Process (MDP) [Bellman, 1957; Puterman, 1994] is a tuple $\langle \mathbf{S}, \mathbf{A}, r, \mathcal{T} \rangle$, where:

$\mathbf{S}$ is a finite set of states.

$\mathbf{A}$ is a finite set of possible actions that can be taken in any time step.

$r : \mathbf{S} \times \mathbf{A} \to \Re$ is a reward function which returns the expected immediate reward received by starting in state $\mathbf{s} \in \mathbf{S}$, then taking action $\mathbf{a} \in \mathbf{A}$.

$\mathcal{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to [0, 1]$ is a dynamics model which specifies the probability of transitioning from state $\mathbf{s}$ to state $\mathbf{s}'$ given action $\mathbf{a}$.

A *trajectory* is a series of states and actions $k = \langle \mathbf{s}^0, \mathbf{a}^0, \mathbf{s}^1, \mathbf{a}^1, \ldots \rangle \in \mathrm{K}$. When it is not clear from context we will slightly abuse notation by using k to index the states, $\mathbf{s}^{k,t}$, and actions, $\mathbf{a}^{k,t}$. The number of elements in the trajectory is called the *planning horizon*. We consider an infinite planning horizon MDP where only the first T states and actions are modelled.

The *discounted return* of a trajectory is a function, $\mathcal{R} : \mathrm{K} \to \Re$ defined by

$$\mathcal{R}(k) = \sum_t \gamma^t r(\mathbf{s}^t, \mathbf{a}^t) \tag{3.7}$$

for discount factor $\gamma \in [0, 1]$ which models the fact that current rewards are worth more than future rewards.

A *stochastic policy* describes a rule for how an agent acts in the world. A policy is a function, $\pi : \mathbf{S} \rightarrow \delta(\mathbf{A})$, from states to a distribution over actions.

The *value* of following the policy for a given state $\mathbf{s}$ is the expected discounted return for starting in that state and continuing on with the policy from then on:

$$\mathcal{V}^{\pi}(\mathbf{s}) = \mathbb{E}_{\pi}\left[\mathcal{R}(k)\right] \tag{3.8}$$

$$= \mathbb{E}_{\pi}\left[\sum_{t} \gamma^{t} r(\mathbf{s}^{t}, \mathbf{a}^{t}) \middle| \mathbf{s}^{0} = \mathbf{s}\right] \tag{3.9}$$

where $\mathbb{E}_{\pi}$ is the expectation over all actions given that the agent follows the policy π. Solving an MDP involves finding a policy that maximizes this value. A recursive relationship holds for the value function which is known as the *Bellman equation*[Bellman, 1957; Sutton and Barto, 1998]:

$$\mathcal{V}^{\pi}(\mathbf{s}) = \sum_{\mathbf{a} \in \mathbf{A}} \pi(\mathbf{a}|\mathbf{s}) \left(r(\mathbf{s}, \mathbf{a}) + \gamma \sum_{\mathbf{s}' \in S} \mathcal{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) \mathcal{V}^{\pi}(\mathbf{s}') \right) \tag{3.10}$$

The Bellman equation defines the value of following a policy starting from state $\mathbf{s}$, collecting an immediate reward and following the policy from then on. This result has proved very useful in many fields such as economics, physics, statistical analysis in addition to planning.

Another form of the Bellman equation that is useful is the *action-value* function, $\mathcal{Q}^{\pi}(\mathbf{s}, \mathbf{a})$. The action-value function gives the expected reward of taking action $\mathbf{a}$ in state $\mathbf{s}$ and following policy π thereafter, its Bellman equation is:

$$\mathcal{Q}^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \sum_{\mathbf{s}' \in S} \mathcal{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) \sum_{\mathbf{a}' \in \mathbf{A}} \pi(\mathbf{a}'|\mathbf{s}') \mathcal{Q}^{\pi}(\mathbf{s}', \mathbf{a}') \tag{3.11}$$

The Bellman equation for the optimal value functions are a special case:

$$\mathcal{V}^*(\mathbf{s}) = \max_{\mathbf{a}\in\mathbf{A_s}} \left(r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'\in S} \mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{a})\mathcal{V}^*(\mathbf{s}') \right) \tag{3.12}$$

$$\mathcal{Q}^*(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'} \mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{s}) \max_{\mathbf{a}'} \mathcal{Q}^*(\mathbf{s}',\mathbf{a}') \tag{3.13}$$

The optimal $\mathcal{Q}$ and $\mathcal{V}$ functions are related as follows:

$$\mathcal{V}^*(\mathbf{s}) = \max_{a\in\mathbf{A}} \mathcal{Q}^*(\mathbf{s},\mathbf{a}) \tag{3.14}$$

$$\mathcal{Q}^*(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'} \mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{a})\mathcal{V}^*(\mathbf{s}') \tag{3.15}$$

Given an optimal value function a deterministic *greedy policy* that selects the best action for every state is the optimal policy:

$$\pi^*(\mathbf{s}) = \arg\max_{\mathbf{a}\in\mathbf{A}} \left(r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'\in S} \mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{a})\mathcal{V}^*(\mathbf{s}') \right) \tag{3.16}$$

$$= \arg\max_{\mathbf{a}\in\mathbf{A}} \mathcal{Q}^*(\mathbf{s},\mathbf{a}) \tag{3.17}$$

Note that the greedy policy can be simply read off the optimal action-value function without further work.

3.3.1 Solving MDPs

Two methods for finding the optimal policy are *value iteration* [Bellman, 1957] and *policy iteration* [Howard, 1960] which use dynamic programming to iteratively improve the estimate $\mathcal{V}^\pi$ until it converges to the true value of the policy[1].

In value iteration, equation (3.12) is turned into an iterative update $\mathcal{V}_i^\pi$ called a *backup*:

$$\mathcal{V}_{i+1}^\pi(\mathbf{s}) = \max_{\mathbf{a}\in\mathbf{A}} \left(r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'\in S} \mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{a})\mathcal{V}_i^\pi(\mathbf{s}') \right) \tag{3.18}$$

[1] If a stochastic policy $\pi(a|s)$ is being used then the maximum action is given probability 1 and all others 0. If there is a tie then all tied actions can split the probability.

This update is carried out for all states $\mathbf{s} \in S$, then repeated again for $\mathcal{V}^{\pi}_{i+1}$ using the new values $\mathcal{V}^{\pi}_{i}$. This update converges geometrically to $\mathcal{V}^{*}$ as i increases [Bertsekas and Tsitsiklis, 1996].

Policy iteration methods involve two main steps; beginning with some arbitrary policy π^0:

I Policy Evaluation : Compute $\mathcal{V}^{\pi^i}$ by solving the system of linear equations defined by equation (3.10). Alternatively equation (3.10) can be applied multiple times until convergence to compute $\mathcal{V}^{\pi^i}$.

II Policy Improvement : Use the current $\mathcal{V}^{\pi^i}$ to improve the policy. For example, greedy updating can be defined as:

$$\pi^i(\mathbf{s}) = \arg\max_{\mathbf{a}\in\mathbf{A}} \left(r(\mathbf{s},\mathbf{a}) + \gamma \sum_{\mathbf{s}'\in S} \mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{a}) \mathcal{V}^{\pi^i}(\mathbf{s}') \right) \tag{3.19}$$

The original policy iteration algorithm by Bellman [1957] applies step I repeatedly until $\mathcal{V}^{\pi^i}$ stops changing between each iteration. Then step II is applied to compute a new policy based on the current value estimate and the process is repeated again.

Puterman and Shin [1978] demonstrated that convergence could still be guaranteed even if the policy is not evaluated fully after each policy update, this algorithm is called *Modified Policy Iteration*. Sutton and Barto [1998] pointed out that any combination of step I and II can be carried out as long as no states are completely inaccessible during either step, an algorithm they call *Generalized Policy Iteration (GPI)*. GPI improves the policy without fully evaluating it and evaluates the policy without having necessarily maximized it. This algorithm is also referred to as the *Actor-Critic* method. The "actor" continuously attempts to improve its performance based on the latest evaluation from the "critic". Meanwhile the critic continues evaluating each attempt by the actor even when the value of the current policy has not fully converged.

The field of *Reinforcement Learning (RL)* [Sutton and Barto, 1998] studies how an agent can act in the world using only its own experiences and

rewards to guide it. This problem can be represented by an MDP where the agent knows $\mathbf{s}$ and $\mathbf{a}$ but only has access to $r(\mathbf{s}, \mathbf{a})$ and $\mathcal{T}(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ through interacting with the world. There is a wide literature on methods for solving RL problems, see Szepesvári [2010] for a recent overview. Many methods attempt to estimate the value functions $\mathcal{V}^*(\mathbf{s})$ and $\mathcal{Q}^*(\mathbf{s}, \mathbf{a})$ and then use a greedy policy to act optimally.

3.4 Parametrized Policies and Trajectory Formulation

If there are a large number of states and actions, the policy cannot be represented as a table of probabilities. The policy can instead be represented using a parametrized function. Assume Θ is a tuple of ρ real-valued parameters. A parametrized policy is a function, $\pi : S \times \Theta \to \delta(\mathbf{A})$, from states to a distribution over actions given $\theta \in \Theta$. Policies will be written $\pi(\mathbf{a}|\mathbf{s}, \theta)$ to make the parameters of the policy explicit.

In some cases the reward signal is only available after an entire trajectory is complete, $\mathcal{R}(k)$ is available but $r(\mathbf{s}^t, \mathbf{a}^t)$ is not. The probability of a trajectory is:

$$p(k|\theta) = p(\mathbf{s}^0) \prod_{t=1}^{\mathrm{T}} \mathcal{T}(\mathbf{s}^t|\mathbf{s}^{t-1}, \mathbf{a}^{t-1}) \pi(\mathbf{a}^{t-1}|\mathbf{s}^{t-1}, \theta) \tag{3.20}$$

The value function from (3.8) can be restated using trajectories directly:

$$\mathcal{V}^\pi = \mathbb{E}_\pi \Big[\mathcal{R}(k) \Big] \tag{3.21}$$

$$= \sum_{k \in \mathcal{K}} p(k|\theta) \mathcal{R}(k) \tag{3.22}$$

where $\mathcal{K}$ is the set of all possible trajectories given a fixed start state $\mathbf{s}$. This form can always be converted to (3.8) by dividing the total return amongst all of the time steps, but this encounters the *temporal credit problem* of how to assign credit to each step for the total return.

3.5 Policy Gradient Planning

Another way to approach solving very large MDPs would be for an agent to determine *how it should act* rather than the *value of acting*. This is the approach taken by direct policy search algorithms such as Policy Gradient (PG) planning.

The policy gradient algorithm contains two main steps: *generating* samples from the policy and *updating* the policy. These steps correspond to the evaluation and improvement steps in Generalized Policy Iteration. Instead of evaluating the full value of the policy, policy gradient algorithms just compute the *direction* to improve the policy and update it directly.

Policy gradient algorithms are an active area of research with early work by Williams [1992] with his REINFORCE method and more recently by many others [Sutton et al., 2000; Riedmiller et al., 2007b; Kersting and Driessens, 2008; Baxter and Bartlett, 2000]. The core insight used here is that GPI can be carried out without needing to actually evaluate the policy at all, by following the policy's derivative instead. This insight manifests differently if rewards are assigned to individual state-action pairs or to entire trajectories.

Using the state-action reward formulation of the value function in (3.10) it can be shown [Sutton et al., 2000; Cao, 2005] that

$$\nabla_\theta \mathcal{V}^\pi = \sum_{\mathbf{s}} p(\mathbf{s}) \sum_{\mathbf{a}} \nabla_\theta \pi(\mathbf{a}|\mathbf{s}, \theta) \mathcal{Q}^\pi(\mathbf{s}, \mathbf{a}) \tag{3.23}$$

where $p(\mathbf{s})$ is the probability of state $\mathbf{s}$ arising in the equilibrium of the Markov chain induced by the policy π. In practice both $p(\mathbf{s})$ and $\mathcal{Q}^\pi(\mathbf{s}, \mathbf{a})$ are often not available. We can use stochastic sampling of the MDP to estimate $p(\mathbf{s})$ and use the actual return $\mathcal{R}(\mathbf{s}, \mathbf{a})$ from the sampling to approximate the value function. This is how the REINFORCE method of Williams [1992] works.

We will be primarily interested in the trajectory formulation of the value

function from equation (3.22). Using this formulation it can be shown that:

$$\begin{aligned}\nabla_\theta \mathcal{V}^\pi &= \nabla_\theta \sum_{k\in\mathcal{K}} p(k|\theta)\mathcal{R}(k) \\ &= \sum_k \nabla_\theta p(k|\theta)\mathcal{R}(k) \\ &= \sum_k p(k|\theta)\nabla_\theta \log p(k|\theta)\mathcal{R}(k) \end{aligned} \tag{3.24}$$

using the fact that $\nabla f(x) = f(x)\nabla \log f(x)$.

To compute the gradient $\nabla_\theta \log p(k|\theta)$ only requires knowing the gradient of the log policy because the probability of the initial state and the transition dynamics do not depend on the policy parameters, so the gradient of these terms is zero.

$$\begin{aligned}\nabla_\theta \log p(k|\theta) = \nabla_\theta \log p(\mathbf{s}_0) &+ \nabla_\theta \sum_{t=1}^{\mathrm{T}} \log \mathcal{T}(\mathbf{s}^{k,t}, \mathbf{a}^{k,t-1}, \mathbf{s}^{k,t-1}) \\ &+ \nabla_\theta \sum_{t=1}^{\mathrm{T}} \log \pi(\mathbf{a}^{k,t-1}|\mathbf{s}^{k,t-1}, \theta) \\ &= \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \end{aligned} \tag{3.25}$$

The final gradient of the value function is thus:

$$\nabla_\theta \mathcal{V}^\pi = \sum_{k\in\mathcal{K}} p(k|\theta)\mathcal{R}(k) \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \tag{3.26}$$

$$\approx \frac{1}{|\mathrm{K}|} \sum_{k\in\mathrm{K}} \mathcal{R}(k) \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \tag{3.27}$$

where a set $\mathrm{K} \subseteq \mathcal{K}$ of sampled trajectories is used to estimate the expected value.

3.5.1 General Policy Gradient Algorithm

The policy gradient planning algorithm [Williams, 1992; Sutton et al., 2000], shown in Figure 3.3, uses the approximation of the gradient of the value function in (3.22) to implement a generalized policy iteration algorithm which

improves the policy iteratively based on simulated experience.

Given are an initial state $\mathbf{s}^0$ and a transition model $\mathcal{T}$ which can be used for sampling. The policy gradient planning algorithm begins with an initial set of parameters θ (these could be randomly selected) and an empty set of trajectories K. The algorithm involves two interacting processes which are iterated until convergence of the gradient or some maximum time is reached.

First, a new sample trajectory, k, is generated using the current policy parameters, Θ, and this trajectory is used to compute $\nabla_\Theta \mathcal{V}^\theta$. Then, the policy parameters are updated by following the gradient of the policy value:

$$\theta' = \theta + \lambda \nabla_\theta \mathcal{V}^\theta \tag{3.28}$$

Where λ is a learning rate that controls the size of the policy update steps.

3.5.2 Reducing the Variance of the Gradient

The varying magnitude of $\mathcal{R}(k)$ can lead to high variance in the estimate of $\nabla_\theta \mathcal{V}^\pi$, which will impede learning. Part of the variance can be removed by subtracting a constant *baseline* b from each occurrence of $\mathcal{R}(k)$ in equation (3.27). This is valid since $\nabla_\theta \sum_k p(k|\theta) = \nabla_\theta 1 = 0$ [Riedmiller et al., 2007b]. The optimal baseline [Weaver and Tao, 2001] for our problem is computed for each policy parameter θ as follows:

$$b[\theta] = \frac{\sum_k \mathcal{R}(k) \left[\nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta)\right]^2}{\sum_k \left[\nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta)\right]^2} \tag{3.31}$$

Another technique used to improve policy gradient performance is called Rprop [Riedmiller et al., 2007b] which replaces scaled updating using the full gradient used in equation (3.28) with an update-value, Δ_θ, which has the *same direction* as $\nabla_\theta \mathcal{V}^\pi$ but a magnitude that is unrelated to the gradient. This is achieved by setting Δ_θ to some small initial value (0.1 is common) and after the gradient is computed, θ is incremented by a small constant amount in the direction of the gradient without regard to its size.

Using this method the magnitude of Δ_θ will remain similar to the magnitude of the parameters. To update the policy we compute $\theta' = \theta + \Delta_\theta$

I Generate sample trajectories :

i Generate $k = \langle \mathbf{s}^{0:\mathrm{T}}, \mathbf{a}^{0:\mathrm{T}-1} \rangle$ using the provided dynamics $\mathcal{T}(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1})$, sampling an action for each time period t from the landscape distribution $\pi(\mathbf{a}^t|\mathbf{s}^t, \theta)$.

ii Add k to the set of simulated trajectories K.

II Update the policy:

i Compute the gradient of the policy: For each trajectory $k \in \mathrm{K}$ and each time step $t \in [0, \mathrm{T}]$, compute the combined gradient $\nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta)$ of the current policy for each state and action encountered at each time period t, in each trajectory k.

ii Combine gradients: The expected policy value, $\mathcal{V}^\pi$, weights the total return, $\mathcal{R}(k)$, received for each trajectory by the probability of that trajectory under the current policy. The gradient of $\mathcal{V}^\pi$ is:

$$\nabla_\theta \mathcal{V}^\pi \approx \frac{1}{|\mathrm{K}|} \sum_k \mathcal{R}(k) \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \tag{3.29}$$

iii Update policy parameters:

$$\theta' = \theta + \lambda \nabla_\theta \mathcal{V}^\pi \tag{3.30}$$

where λ is a learning rate.

Figure 3.3: The general policy gradient algorithm.

and then increment the value of Δ_θ in the appropriate direction based on the gradient.

3.5.3 Natural Policy Gradients

An alternative method that combines the benefits of both baselining and gradient scaling is to use the *natural gradient* of the value function [Amari, 1998; Kakade, 2002]. Peters et al. [2005] described the Natural Actor-Critic (NAC) framework which generalizes the Actor-Critic algorithm. NAC algorithms use sampled trajectories to gather sufficient statistics to compute the *Fisher information* of the gradient. The Fisher information represents the covariance of the gradient of the *score* (i.e. the log likelihood) of the policy [Riedmiller et al., 2007a]. This information is then used to produce an unbiased estimate of the gradient vector using linear regression on the gradients of the policy parameters and the rewards.

The natural gradient is a more efficient direction to follow than the gradient as it represents the steepest descent direction with respect to the variance of the policy. Thus, by following the natural gradient the value of the policy is being maximized in a way that minimizes the variance of the gradients estimated from multiple sampling runs.

The Natural Actor-Critic approach has been shown to provide significant improvements in performance over other RL methods, converging in fewer steps than other gradient methods and avoiding local minima more effectively. It also automatically balances the scale of the derivatives, thus removing the need to compensate in the algorithm and reducing the sensitivity of the algorithm to the value of the learning rate [Peters et al., 2005; Riedmiller et al., 2007b].

One straightforward instance of NAC which fits our purposes is the Episodic Natural Actor-Critic (ENAC) algorithm. This algorithm operates on an entire trajectory, or episode, as its data for sufficient statistics and assumes a fixed start state. This is very appropriate for spatiotemporal planning since value models could include non-local components that extend over time and we usually operate from a single fixed starting state representing the present.

In matrix form the natural gradient can be solved as follows :

$$\mathbf{X} = \begin{bmatrix} \nabla_{\theta_1} \sum_t \log \pi(\mathbf{a}^{1,t}|\theta) & \dots & \nabla_{\theta_\rho} \sum_t \log \pi(\mathbf{a}^{1,t}|\theta) & 1 \\ \dots & \dots & \dots & 1 \\ \nabla_{\theta_1} \sum_t \log \pi(\mathbf{a}^{|\mathrm{K}|,t}|\theta) & \dots & \nabla_{\theta_\rho} \sum_t \log \pi(\mathbf{a}^{|\mathrm{K}|,t}|\theta) & 1 \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} \mathcal{R}(k=0) \dots \mathcal{R}(k=|\mathrm{K}|-1) \end{bmatrix}$$

$$\begin{bmatrix} \delta\theta_1 \\ \delta\theta_2 \\ \dots \\ \delta\theta_\rho \\ \mathcal{V}^\pi \end{bmatrix} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

In later algorithms the above computation will be referred to as $[\delta\theta \quad \mathcal{V}^\pi] = \texttt{naturalGradient}(\mathbf{X}, \mathbf{Y})$. The complexity of this computation depends on the method used; using the normal equations method with Cholesky factorization the complexity is $O(\rho^2|\mathrm{K}| + \rho^3/3)$ [Higham, 1996].

Chapter 4

Spatiotemporal Planning using Policy Gradients

This chapter defines the spatiotemporal planning problem as an MDP with actions and states factored across locations in space. A policy will be defined as a distribution over landscape actions built from local, conditional policies for each cell. For now, these local policies are considered independently. Later chapters will look at policies that can handle interrelated locations. Optimization of the spatial policy will use a policy gradient planning algorithm. A forestry planning problem is used for evaluation of the algorithm. The focus of the evaluation is on comparing two policies defined at different levels of abstraction.

4.1 Spatial Planning as a Factored MDP

Let C be the set of all cells. Each cell defines an area in the landscape. The number of cells is assumed to be finite. Let S be the finite set of states of a single cell and A the finite set of actions that can be taken in any cell.

A spatial planning problem is a factored Markov decision process(MDP) $\langle \mathbf{S}, \mathbf{A}, r, \mathcal{T} \rangle$ where:

- $\mathbf{S}$, the set of *landscape states*, is S^C, the set of functions from C into S. A landscape state is denoted as $\mathbf{s} \in \mathbf{S}$. The state at a particular

cell c is denoted s_c. A cell state is defined by a set, F, of features. A feature is a function $f_c(\mathbf{s}) : C \times \mathbf{S} \rightarrow [0, 1]$.

- $\mathbf{A}$, the set of *landscape actions*, is A^C, the set of functions from C into A. A landscape action is denoted as $\mathbf{a} \in \mathbf{A}$. The action at a particular cell c is denoted a_c.

- $r : \mathbf{S} \times \mathbf{A} \rightarrow \Re$ is a reward function which return the expected reward received by starting in landscape state $\mathbf{s}^t$, then taking landscape action $\mathbf{a}^t$ and ending up in landscape state $\mathbf{s}^{t+1}$ at the next time step.

- $\mathcal{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$ is a dynamics model which specifies the probability of transitioning from landscape state $\mathbf{s}^t$ to landscape state $\mathbf{s}^{t+1}$ given landscape action $\mathbf{a}^t$.

The factored form of this MDP is similar to other factored models such as the Factored MDPs described by Boutilier et al. [1999] as well as multi-agent factored MDPs [Guestrin et al., 1996] and Decentralized MDPs (DEC-MDPs) [Bernstein et al., 2002]. Section 8.2 contains more discussion of results on these models and exact solution methods that can be used for small problems.

4.2 Forestry Planning as a Factored MDP

To make things more concrete we will map the forestry problem defined in Chapter 2 to this MDP formulation.

4.2.1 Actions

The actions that can be taken at any cell are those described in Section 2.2 such as clear-cut harvesting, thinning or doing nothing. There are a small set of available actions per cell but the space of possible landscape actions is huge. Consider a spatial planning problem with 1000 cells and binary actions. The number of possible landscape actions is $2^{1000} \approx 10^{300}$ Clearly any method that relies on enumerating actions would be impractical here.

4.2.2 States

The features, F, defining the state of each cell describe the conditions of the forest at that location and could include any of the properties described in Section 2.1. In this chapter, F contains four features:

`Age` **(0-250 years old)** - The dominant age class of trees in the cell

`PercentPine` **(0.0-1.0)** - Fraction of the trees in the cell that are pine

`MPBLevel` **(low/medium/high)** - A severity rating for the MPB infestation

`Area` **(1-50ha)** - The area of the cell

As with the action-space, the state-space can quickly become impractically large. Consider a problem with 1000 cells and 10 binary features; the number of landscape states would be $(2^{10})^{1000} \approx 10^{3000}$. The complexity of the state and action spaces is one of the reasons we looked to Policy Gradients for planning since this can avoid the need to enumerate states and actions.

4.2.3 Rewards

The reward function assigns value for cutting individual trees and penalizes various properties of the landscape state, such as: a quadratic penalty on the deviation from a desired total forest population size, linear penalties for overcutting, linear penalties for the number of trees killed by MPB, and base costs for maintaining the forest that will be incurred even with no cutting.

In forestry planning problems, the reward is often best described in the form of the *total expected return* for an entire trajectory described in Section 3.3. Recall that a trajectory is a series of states and actions generated over time, $k = \langle \mathbf{s}^0, \mathbf{a}^0, \mathbf{s}^1, \mathbf{a}^1, \ldots \rangle$. The return is useful because the evaluation of a trajectory may need to account for changes over time as with the even flow constraints discussed in Section 2.3. A simple form for the expected return is the expected discounted reward $\mathcal{R}(k) = \sum_t \gamma^t r(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1})$.

4.2.4 Dynamics

We developed a simple forest simulator to evaluate the performance of policy gradient planning on this problem. The simulator includes state features for the distribution of tree species and age classes, the level of MPB in a cell and its neighbouring cells. The dynamics include tree birth, growth and death, replanting of young trees after clearcutting, killing of trees by MPB and the spread of MPB to nearby cells year to year. This simulator takes in landscape state and action and return the next landscape state after following the given action directly.

4.3 Stochastic Spatial Policies

This section presents a stochastic spatial policy for landscape actions that combines a set of independent local policies for actions each cell. Later chapters expand this approach to cases where local policies can be interdependent.

4.3.1 Cell Policy Definition

A cell policy, $\pi_c(a_c|\mathbf{s}, \theta)$, is a distribution over actions to take at cell c, given the state of the landscape $\mathbf{s}$ and policy parameters θ. The policy parameters $\theta[f, a]$ provide a weight for each combination of feature $f \in F$ and action $a \in A$. An example instance of the policy parameters are shown in Figure 4.1. Features are used to define a potential function ψ combining parameters and features:

$$\psi(a, c, \mathbf{s}, \theta) = \sum_f \theta[f, a] f_c(\mathbf{s})$$

The cell policy is defined by a log-linear distribution (also called a Boltzman or Gibbs distribution) over cell actions using the potential function:

$$\pi_c(a|\mathbf{s}, \theta) = \frac{\exp(\psi(a, c, \mathbf{s}, \theta))}{\sum_{b \in A} \exp(\psi(b, \mathbf{s}, \theta))} \tag{4.1}$$

In the case of binary actions, it is not strictly necessary to model a policy parameter for each combination of feature and action; the same expressive-

Action	Age	PercentPine	MPBLevel	Area
Cut	-3.42	0.77	3.45	-2.63
NoCut	7.75	5.69	-1.60	6.13

Figure 4.1: Example of a set of policy parameters for four features.

ness could be achieved with one parameter per feature. For any set of cell actions A, the minimum number of parameters required is $|F| \times |A-1|$. So, while $|F|$ parameters could be eliminated by dropping parameters for one action, we have chosen to add the additional action to provide a uniform structure for the policy model. This should make the parameters easier for practitioners to interpret without needing work out the value of the remaining dependent action. Positive and negative correlations between each feature and each action can be read directly off the table.

4.3.2 Landscape Policy

The landscape policy $\Pi(\mathbf{a}|\mathbf{s}, \Theta)$ represents the probability of an agent choosing a landscape action $\mathbf{a}$ given the landscape state $\mathbf{s}$. The landscape-parameters Θ define parameters for each local cell-policy. Two parametrizations of the landscape-policy will be used in this chapter, Π^C and Π^0. If the actions at each cell are independent of each other then the landscape policy can be defined as the product of (4.1) for each cell.

The first landscape policy, Π^C, explicitly maintains separate parameters, $\Theta : C \to \theta$, for each cell

$$\Pi^C(\mathbf{a}|\mathbf{s}, \Theta) = \prod_{c \in C} \pi_c(a_c|\mathbf{s}, \Theta_c) \tag{4.2}$$

The second landscape policy, Π^0, uses a single set of parameters, $\theta \in \Theta$, for all cells in the landscape

$$\Pi^0(\mathbf{a}|\mathbf{s}, \theta) = \prod_{c \in C} \pi_c(a_c|\mathbf{s}, \theta) \tag{4.3}$$

These two formulations will be used within a general policy gradient algorithm and compared.

Both Π^C and Π^0 define a policy for each cell but Π^0 does not distinguish between cells based on their identity. All cells are treated equally based on their state features.

One way to think about the difference between Π^C and Π^0 is by an analogy to time. A stationary policy defines one set of parameters for all time steps. A policy can be stationary or non-stationary with respect to time. Similarly, Π^0 is stationary with respect to space. This *spatially stationary policy* defines a distribution over actions based on cell features that apply to any cell in the landscape. A spatially stationary policy is well-defined for any number of cells, so the number of policy parameters is independent of the problem size. A spatially stationary policy could have many advantages during planning, allowing us to easily change scale or apply a learned policy to different sub-regions of the landscape without modification. However, it is not clear if a spatially stationary policy would perform better during planning than one that is able to fit the results of individual cells. This is one of the questions dealt with in the experiments in this chapter.

4.4 Policy Gradients

The Policy Gradient planning algorithm described in Figure 3.3 is an attractive approach for this type of problem because it avoids the need to enumerate the huge number of states and actions; it also allows learning from simulated experiences.

Within a trajectory k, the specific landscape states and landscape actions at time step t are denoted as $\mathbf{s}^{k,t}$ and $\mathbf{a}^{k,t}$ respectively. The gradient of a value function using Π^C is expanded from the general definition of the

gradient in equation (3.27):

$$\begin{aligned}\nabla_{\Theta}\mathcal{V}^{\Pi^C} &\approx \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k)\sum_t \nabla_{\Theta}\log\Pi^C(\mathbf{a}^{k,t}|\mathbf{s}^{k,t},\Theta)\\ &= \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k)\sum_t \nabla_{\Theta}\log\prod_c \pi_c(\mathbf{a}^{k,t}|\mathbf{s}^{k,t},\Theta_c)\\ &= \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k)\sum_t\sum_c \nabla_{\Theta_c}\log\pi_c(\mathbf{a}^{k,t}|\mathbf{s}^{k,t},\Theta_c) \qquad (4.4)\end{aligned}$$

The gradient of a value function using Π^0 is:

$$\begin{aligned}\nabla_{\theta}\mathcal{V}^{\Pi^0} &\approx \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k)\sum_t \nabla_{\theta}\log\Pi^0(\mathbf{a}^{k,t}|\mathbf{s}^{k,t},\theta)\\ &= \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k)\sum_t \nabla_{\theta}\log\prod_c \pi_c(a_c^{k,t}|\mathbf{s}^{k,t},\theta)\\ &= \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k)\sum_t\sum_c \nabla_{\theta}\log\pi_c(a_c^{k,t}|\mathbf{s}^{k,t},\theta) \qquad (4.5)\end{aligned}$$

These gradients are applied to build two instances of the PG algorithm, $\texttt{SPG}^C$ and $\texttt{SPG}^1$, by modifying the policy update formula (3.28).

Algorithm $\texttt{SPG}^C$ uses $\nabla_{\Theta}\mathcal{V}^{\Pi^C}$ to define the policy update

$$\Theta' = \Theta + \lambda\nabla_{\Theta}\mathcal{V}^{\Pi^C} \qquad (4.6)$$

Algorithm $\texttt{SPG}^1$ uses $\nabla_{\theta}\mathcal{V}^{\Pi^0}$ to define the policy update

$$\theta' = \theta + \lambda\nabla_{\theta}\mathcal{V}^{\Pi^0} \qquad (4.7)$$

where λ is a learning rate that controls the size of the policy update steps. This learning rate is notoriously difficult to choose as it needs to scale with the magnitude of the derivative. Riedmiller et al. [2007b] describe some techniques called *optimal base-lining* and *Rprop* to get around the difficulty of setting λ and to also reduce the variance of the gradient. These methods were described in Section 3.5.2.

4.4.1 Gradient of the Spatial Policies

The formulations for $\nabla \mathcal{V}^{\pi}$ in (4.4) and (4.5) require computing the gradient of the cell policy, $\nabla_{\theta} \log \pi_c(a|\mathbf{s}, \theta)$ for any action $a \in A$. The log-linear policy parametrization allows us to express this analytically. The partial derivative $\nabla_{\alpha f} \log \pi_c(a|\mathbf{s}, \theta)$ with respect to parameter $\theta[\alpha, f]$ for every $\alpha \in A$ and $f \in F$ is defined as:

$$\begin{aligned}
\nabla_{\alpha f} \log\, \pi_c(a|\mathbf{s}, \theta) &= \nabla_{\alpha f} \log \left(\frac{\exp(\psi(a, c, \mathbf{s}, \theta))}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{s}, \theta))} \right) \\
&= \nabla_{\alpha f} \psi(a, c, \mathbf{s}, \theta) - \nabla_{\alpha f} \log \sum_{b \in A} \exp(\psi(b, c, \mathbf{s}, \theta)) \\
&= \nabla_{\alpha f} \psi(a, c, \mathbf{s}, \theta) - \frac{\nabla_{\alpha f} \sum_{d \in A} \exp(\psi(d, c, \mathbf{s}, \theta))}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{s}, \theta))} \\
&= \nabla_{\alpha f} \psi(a, c, \mathbf{s}, \theta) - \frac{\sum_{d \in A} \nabla_{\alpha f} \exp(\psi(d, c, \mathbf{s}, \theta))}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{s}, \theta))} \\
&= \nabla_{\alpha f} \psi(a, c, \mathbf{s}, \theta) - \frac{\sum_{d \in A} \exp(\psi(d, c, \mathbf{s}, \theta)) \nabla_{\alpha f} \psi(d, c, \mathbf{s}, \theta)}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{s}, \theta))} \qquad (4.8)
\end{aligned}$$

Recall that ψ is defined as the sum of features $f_c(\mathbf{s})$ weighted by parameters θ. Thus, the partial derivative of ψ with respect to a parameter $\theta[f, \alpha]$ is set to zero for every term except the single one matching f and α:

$$\nabla_{\alpha f} \psi(a, c, \mathbf{s}, \theta) = \begin{cases} f_c(\mathbf{s}) & \text{if } \alpha = a \\ 0 & \text{otherwise} \end{cases}$$

Now (4.8) can be simplified in each case. If $\alpha = a$ then

$$\begin{aligned}
\nabla_{\alpha f} \log\, \pi_c(a|\mathbf{s}, \theta) &= f_c(\mathbf{s}) - \frac{\exp(\psi(a, c, \mathbf{s}, \theta)) f_c(\mathbf{s})}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{s}, \theta))} \\
&= f_c(\mathbf{s}) - \pi_c(\mathbf{s}, \alpha, \theta) f_c(\mathbf{s}) \\
&= (1 - \pi_c(\mathbf{s}, \alpha, \theta)) f_c(\mathbf{s})
\end{aligned}$$

If $\alpha \neq a$ then

$$\begin{aligned}\nabla_{\alpha f} \log\ \pi_c(a|\mathbf{s},\theta) &= -\frac{\exp(\psi(d,c,\mathbf{s},\theta))}{\sum_{b\in A}\exp(\psi(b,c,\mathbf{s},\theta))} \\ &= -\pi_c(\mathbf{s},\alpha,\theta) f_c(\mathbf{s})\end{aligned}$$

Both cases can be combined under one notation

$$g_c(a,\mathbf{s},\theta) = \begin{cases} (1-\pi_c(\mathbf{s},\alpha,\theta)) & :\ \text{if } \alpha = a \\ -\pi_c(\mathbf{s},\alpha,\theta) & :\ \text{if } \alpha \neq a \end{cases} \tag{4.9}$$

The final gradient of the value under policy Π^0 is:

$$\nabla_\theta \mathcal{V}^{\Pi^0} \approx \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k) \sum_t \sum_c g_c(a_c^{k,t}, \mathbf{s}^{k,t}, \theta) f_c(\mathbf{s}) \tag{4.10}$$

For the final gradient of the value under policy Π^C we note that the gradient of $\nabla_{\Theta_d}\psi(a,c,\Theta_c) = 0$ since the cell policies for any two cells c and d are independent. Thus, for each cell policy the gradient is:

$$\nabla_{\Theta_c} \mathcal{V}^{\Pi^C} \approx \frac{1}{|\mathrm{K}|}\sum_k \mathcal{R}(k) \sum_t g_c(a_c^{k,t}, \mathbf{s}^{k,t}, \Theta_c) f_c(\mathbf{s}) \tag{4.11}$$

4.5 Spatial Policy Gradient Algorithm

Two spatial policy gradient algorithms were implemented: `SPG`^{C} shown on page 57 and `SPG`^{1} shown on the next page followed by supporting algorithms to generate trajectories and simulate dynamics. The algorithms are identical except for the fact that `SPG`^{C} uses the policy from equation (4.2) and `SPG`^{1} uses the policy from (4.3). Both algorithms use optimal baselining and Rprop gradient scaling to reduce the variance of the gradient search as described in Section 3.5.2. When `runSim` is called by `SPG`^{1} each instance of Θ_c then $\Theta_c = \theta$ for all c.

Algorithm 1: $\texttt{SPG}^1$ (s_0)

initialize θ randomly
$\Delta_\theta = 0.1$; $\mathrm{K} = \emptyset$
repeat *maxSamples* **times**
  `// Sample new trajectory`
  $\langle \mathbf{s}, \mathbf{a}, R \rangle$ = generateTrajectory($\mathbf{s}_0$, θ)
  $K = K \cup \langle \mathbf{s}, \mathbf{a}, R \rangle$
  `// Compute gradient of policy and baseline`
  foreach (k, θ) **do**
    $G_\theta[k] = \sum_t \sum_c g_c(a_c^{k,t}, \mathbf{s}^{k,t}, \theta) f_c(\mathbf{s})$
  $b[\theta] = \frac{\sum_k \mathcal{R}(k)[G_\theta[k]]^2}{\sum_k [G_\theta[k]]^2}$
  `// Update policy`
  $\nabla_\theta \mathcal{V}^{\Pi^0} = \frac{1}{|\mathrm{K}|} \sum_k (R(k) - b[\theta]) G_\theta[k]$
  update Δ_θ using Rprop with $\nabla_\theta \mathcal{V}^{\Pi^0}$
  $\theta = \theta + \Delta_\theta$
return θ

4.6 Experiments

$\texttt{SPG}^1$ and $\texttt{SPG}^C$ were implemented in Matlab and run on a dual processor Pentium 4 3.2GHz PC with 2GB RAM running Windows XP.

The initial landscape states were varied randomly around representative values for state features based on common distributions present in data for BC forests for tree species, tree age, MPB presence and other features.

4.7 Results

Figure 4.2 shows a typical result for the total reward received by the two algorithms. The reward is shown for each trajectory sample and is averaged over 20 trials for a small problem with 5 cells and 5 time steps. Each trial sampled 200 trajectories and updated the policy after every 5 samples using all trajectories sampled up to that point. The following distribution for the action components of the parameters was used as an initial policy for each trial as uniform weights: $\langle DoNothing = 1.0, ClearCut = 0.0, Thin = 0.0 \rangle$.

Algorithm 2: `SPG`^{C} (s_0)

initialize Θ randomly

$\Delta_\Theta = 0.1$; $\mathrm{K} = \emptyset$

repeat *maxSamples* **times**

`// Sample new trajectory`

$\langle \mathbf{s}, \mathbf{a}, R \rangle$ = generateTrajectory($\mathbf{s}_0$, Θ)

$K = K \cup \langle \mathbf{s}, \mathbf{a}, R \rangle$

`// Compute gradient of policy and baseline`

foreach (k, Θ_c, c) **do**

$G_\theta[k] = \sum_t g_c(a_c^{k,t}, \mathbf{s}^{k,t}, \Theta_c) f_c(\mathbf{s})$

$b[\Theta, c] = \frac{\sum_k \mathcal{R}(k)[G_\theta[k]]^2}{\sum_k [G_\theta[k]]^2}$

`// Update policy`

foreach $c \in C$ **do**

$\nabla_{\Theta_c} \mathcal{V}^{\Pi^C} = \frac{1}{|\mathrm{K}|} \sum_k (R(k) - b[\Theta, c]) G_\theta[k]$

update Δ_Θ using Rprop with $\nabla_\Theta \mathcal{V}^{\Pi^C}$

$\Theta = \Theta + \Delta_\Theta$

return Θ

Algorithm 3: `generateTrajectory` ($\mathbf{s}_0$, Θ)

$R = 0$

for $t = 0$ **to** T **do**

$\langle \mathbf{a}^t, r^t, \mathbf{s}^{t+1} \rangle$ = runSim($\mathbf{s}^t$, Θ)

$R = R + \gamma^t r^t$

return $\langle \mathbf{s}, \mathbf{a}, R \rangle$

Algorithm 4: `runSim` ($\mathbf{s}^t$, Θ)

foreach c *in* C **do**

`// sample action distribution`

$a_c^t \sim \pi(A | \mathbf{s}_c^t, \Theta_c)$

$\mathbf{s}_{t+1}$ = externalSimulator($\mathbf{s}^t, \mathbf{a}^t$)

r^t = reward($\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1}$)

return $\langle \mathbf{a}^t, r^t, \mathbf{s}^{t+1} \rangle$

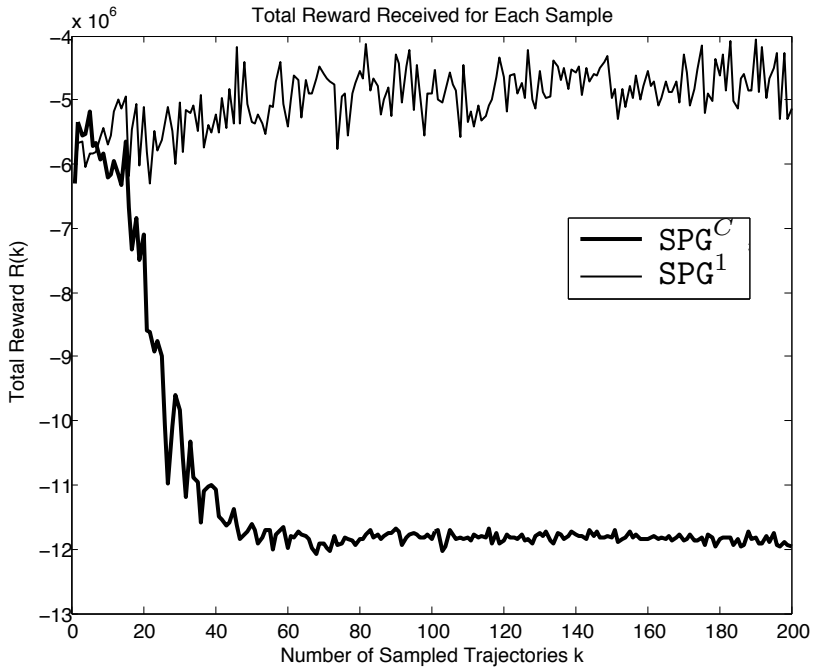


Figure 4.2: Total reward received averaged over 20 trials for $\texttt{SPG}^C$ and $\texttt{SPG}^1$ on 5 cells with 5 time steps after 200 samples with policy updates every five samples. Initial policy was $\langle DoNothing = 1.0, ClearCut = 0.0, Thin = 0.0 \rangle$.

The initial Rprop value of the gradient update-value, Δ_Θ was set to a value of 0.1; this has been found to be a reasonable value for many problems [Riedmiller et al., 2007b]. All time steps and cells were initialized to the same action distribution.

$\texttt{SPG}^1$ consistently finds higher value policies than $\texttt{SPG}^C$. The abstract policy of $\texttt{SPG}^1$ is also more robust across multiple trials, whereas $\texttt{SPG}^C$ fixes onto a deterministic set of action assignments to particular cells. Unlike policy iteration, policy gradient algorithms are not guaranteed to increase their value at each step. $\texttt{SPG}^C$ randomly finds lower value policies and then cannot get out that area. This is possibly because $\texttt{SPG}^C$ lacks the flexibility to explore the full action space as $\texttt{SPG}^1$ can.

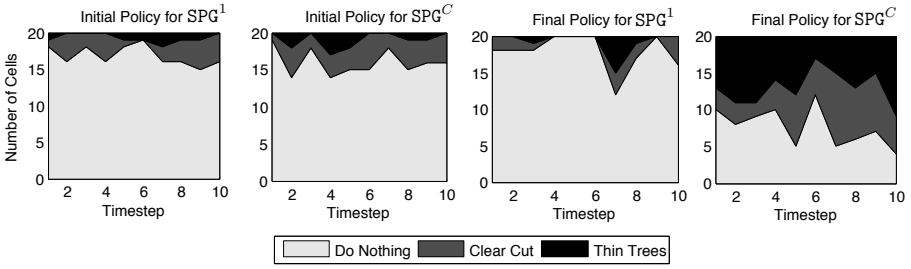


Figure 4.3: In each cell, the available actions are ⟨"Do Nothing", "Clear Cut", "Thin Trees"⟩. The number of cells assigned each of these actions are shown as shaded areas. The initial and final policies after optimization for both SPG^1 and SPG^C are shown. This trial used 200 sampled trajectories using, 10 time steps and 20 cells. The initial policy was set to $\langle DoNothing = .8, ClearCut = .15, Thin = .05\rangle$ for both algorithms.

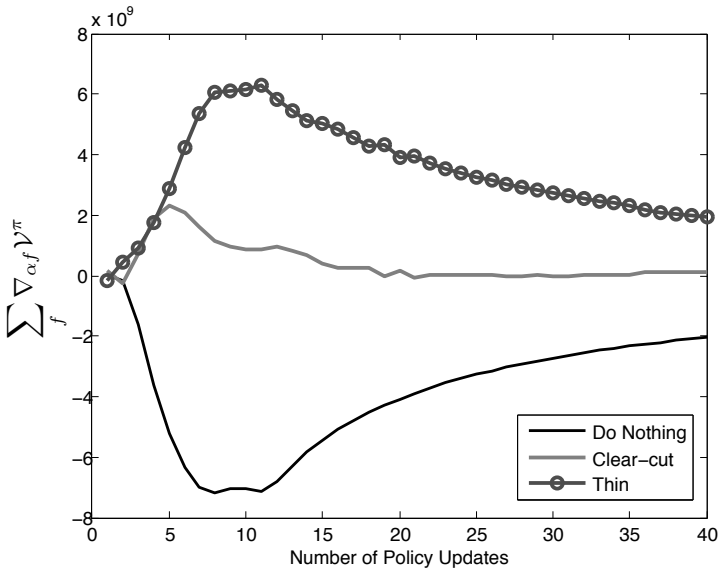
Figure 4.3 shows the initial and final policies for the two algorithms on a single trial of a 20 cell planning problem. The initial policy in this trial was set to $\langle DoNothing = .8, ClearCut = .15, Thin = .05\rangle$ for both algorithms. The final policies are very different. SPG^1 has found a policy that does less cutting than the initial policy; clear-cutting a few cells for the first few years, then cutting almost nothing and focussing on thinning in later years to achieve a higher value. The SPG^C algorithm has found a policy with a much higher proportion of cutting. The policy found by SPG^C is deterministic. Each cell, at each time step, always has the same action taken over many different trajectory samples. SPG^C cannot break out of this policy, even though it is incurring major penalties for overcutting, because the value of a policy is based on weighting rewards by the likelihood of past trajectories under the current policy. This makes a deterministic policy that doesn't change between samples very attractive. Once a deterministic policy is found, diverging on some cell will only lower the expected value of the policy. Algorithm SPG^1 does not have this problem since there are

fewer deterministic policies in which to get stuck and they all have very low reward, such as `Cut` in all cells or `NoCut` in all cells.

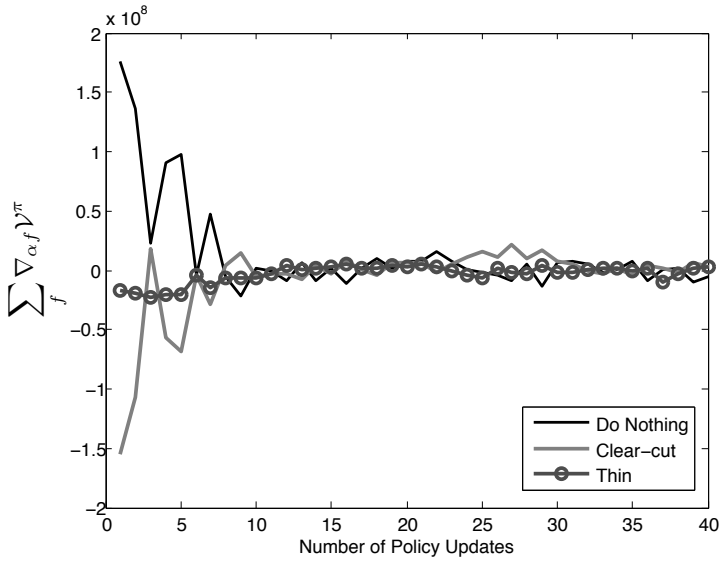
Figure 4.4 shows the gradients of the combined parameters for each action in a policy for the same trial as in Figure 4.3. After the $\texttt{SPG}^C$ policy converges to a narrow range of rewards, the gradient begins converging. For $\texttt{SPG}^1$, the variation in the gradient drops significantly once a good policy is found.

The $\texttt{SPG}^1$ algorithm runs about three times faster than $\texttt{SPG}^C$ on the same number of trials and trajectories. This is not surprising since both algorithms sample actions for every cell and time step while $\texttt{SPG}^C$ uses more memory and time managing the larger number of parameters. The runtimes for $\texttt{SPG}^1$ range from 2 minutes for a 5 cell, 5 time step problem, to 230 minutes for a 10 time step, 30 cell problem. With this implementation, we estimate that solving a problem with a few thousand cells would take approximately two days.

The policies Π^C and Π^0 do not model any interaction between actions in different cells. Thus, correlations between the actions taken at different locations cannot be exploited to improve the value of the policy. To improve upon this we need to step back and consider the meaning of the local conditional policies and how to combine them in a consistent manner, which is the subject of the next chapter.



(a) Algorithm $\texttt{SPG}^C$



(b) Algorithm $\texttt{SPG}^1$

Figure 4.4: Policy gradients for parameters relating to each action, summed across all cells and time steps for one trial with 200 sampled trajectories, 20 cells and 10 time steps.

Chapter 5

Cyclic Causal Models

A limitation of the landscape policies Π^C and Π^0 in the previous chapter are that they assume actions at each cell can be chosen independently of actions at any other cell. This makes it straightforward to represent a landscape policy but makes it difficult to model realistic spatial constraints and values. The rest of this thesis looks at how to represent, manipulate and perform planning with a landscape policy that can model actions at spatially interrelated locations. This chapter focusses on the theoretical basis for representing a joint distribution over many interrelated variables using a cyclic causal model built from local distributions over the variables. The goal is find a compact representation that can flexibly model different types of spatial interaction and also have some way of being interpreted and used by practitioners on the ground.

We explored a number of other approaches to computing or approximating a landscape distribution. One of the approaches we explored but abandoned was an ensemble of directed probabilistic models. In this model, the distribution at each location is represented as a weighted combination of the estimates of multiple Bayesian networks (BN). Each BN would use a different directed acyclic graph over the nodes. By combining multiple BNs with different graphs covering all the relevant influences on a node, a reasonable approximation of the spatial interactions should be representable. The hope was that a compact representation could be found which used

only a small number of BNs to achieve a reasonable representation of the joint distribution. For example, a square lattice could be covered with four BNs rooted at each of the four corners and the marginal probabilities from each BN could be averaged together to get an estimate for that node. However, we were unable to find an arrangement of small numbers of BNs which worked well and eventually abandoned this approach.

We then turned to the idea of using a causal model for the landscape policy. A causal model is appropriate for a local policy since actions are fundamentally causal, in the following sense. Each local cell policy answers the question: *"What action would an agent take at this location if the actions at all other locations were already decided."* This makes a local interpretation of the policy very clear. This is important for spatiotemporal planning because it matches the fact that actions are often carried out by many people on the ground given their local conditions. However, a model representing the interaction of causal cell policies contains cycles as the actions at each cell influence the likely actions at other cells. The challenge is how to construct a consistent distribution over landscape actions, a *spatial landscape policy*, that is consistent with all of these interacting local cell policies.

Strotz and Wold [1960] gave an interpretation of what the joint distribution of a cyclic causal model means; the distribution is the *equilibrium of a dynamic system* where the local causal distributions define the transition dynamics. Cyclic causal distributions have not been a major part of research into causal modelling. Existing representations of causal distributions, such as SEMs, generally assume acyclic structure and do not try to take advantage of the structure in cyclic causal models. So, the question remains how can a landscape distribution over multiple variables in a cyclic model be computed efficiently and in what cases, if any, can it be computed exactly? Addressing this question is the focus of this chapter.

We begin by reviewing the theoretical grounding for the equilibrium as an interpretation of a cyclic causal network, then an iterative algorithm is presented for computing the marginal distributions from the equilibrium of arbitrary subsets of variables. This algorithm can often compute the exact marginals but at the expense of a more complex formulation that increases

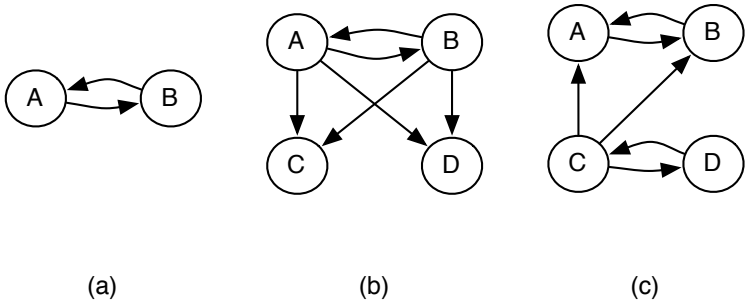


Figure 5.1: Three simple cyclic causal networks.

the tree-width of the graph.

5.1 A Multi-agent Example

Spatiotemporal planning problems can be seen as co-operative multi-agent planning problems where at each location there is an agent deciding how to act based on its local context. All agents share the goal of maximizing a global reward function.

To make cyclic, causal models clearer, the running example in this chapter will use a simple multi-agent decision problem with a naturally sparse causal structure. Consider a simple economic problem composed of four agents: Alfred, Betty, Cindy and Doug. Every day, each agent decides whether they will buy or sell a fixed amount of gold. The only information agents will use to make their decisions is their own internal desire to own gold conditioned on their reaction to how other agents behave. This simple domain can lead to quite complex behaviour when there are cycles, even without other complexities such as price and external information about gold, so we ignore these other complexities here. To avoid runaway purchasing, we assume that each agent has a fixed number of units of gold they can hold at any time. When deciding whether to buy a unit of gold today or not, each agent can take into account the most recent action of any other agent.

Example 1. Consider first a simple model containing only Alfred and Betty who always take each other's activities into account when making their own decision. This can be represented by the cyclic causal network shown in 5.1(a). Alfred and Betty are represented as causally interdependent. Under an equilibrium interpretation this model says that at each moment in time, Alfred takes Betty's most recent actions into account and Betty takes Alfred's most recent actions into account. We say that A and B are symmetrically correlated.

Example 2. In 5.1(b), Alfred and Betty trust each other and pay close attention to each other's activities and have a high probability of following each other's lead. Meanwhile, Doug and Cindy's activities are ignored by everyone but they both pay close attention to the activities of Betty and Alfred. We say that in this network the variables are all correlated but are not symmetric.

Example 3. In 5.1(c), Alfred and Betty closely observe each other's activities; similarly, Cindy and Doug observe each other's behaviour. Cindy also influences Alfred and Betty, perhaps by stopping by and talking to them every day, but Cindy completely ignores the activities of Alfred and Betty.

The goal is to compute the distribution over the amount of gold being bought or sold over time by each individual agent. The general solution is to use the equilibrium distribution that arises after these networks are rolled out in time. The examples above will be used to demonstrate how inconsistencies can arise in the equilibrium and how to deal with them to perform approximate inference. As we will see, cyclic causal models are related to existing work on Structural Equation Models and filtering in dynamic Bayesian networks.

5.2 Modelling Cyclic Causality with SEMs

Pearl [2009] proposed Structural Equation Models (SEMs) as a representation for causality. A structural equation model consists of a deterministic function for each variable in terms of other variables and (independent) noise inputs. A modal logic for SEMs was presented by Halpern [2000].

Example 4. Consider the simple cyclic causal model with two Boolean variables A and B from Example 1. As a policy, this could model two agents that influenced by each other's previous actions before acting. The causal model can be defined in terms of 4 parameters:

$$p_1 = P(a|do(b))$$
$$p_2 = P(a|do(\neg b))$$
$$p_3 = P(b|do(a))$$
$$p_4 = P(b|do(\neg a))$$

This can be represented as a structural equation model:

$$a \leftrightarrow (b \wedge u_1) \vee (\neg b \wedge u_2) \tag{5.1}$$
$$b \leftrightarrow (a \wedge u_3) \vee (\neg a \wedge u_4) \tag{5.2}$$

where the U_i are independent exogenous Boolean variables and $P(u_i) = p_i$.

This model gives the anticipated result for all interventions, except for the case of no interventions.

An exogenous variable is **extreme** if its probability distribution contains zeros, and is **non-extreme** if its probabilities are all strictly between 0 and 1.

Proposition 1. *The noise variables $U_1, \ldots, U_4$ in the SEM cannot be both non-extreme and independent.*

Proof. The assignment $U_1 = true$, $U_2 = false$, $U_3 = false$, $U_4 = true$ is logically inconsistent, as it implies $(a \leftrightarrow b) \wedge (b \leftrightarrow \neg a)$, and so must have probability zero. For the u_i to be independent, $P(u_1 \wedge \neg u_2 \wedge \neg u_3 \wedge u_4) = P(u_1) \times (1 - P(u_2)) \times (1 - P(u_3)) \times P(u_4) = 0$. One of the variables must be extreme, having probability zero. Similarly, $u_1 = false$, $u_2 = true$, $u_3 = true$, $u_4 = false$ implies $(a \leftrightarrow b) \wedge (a \leftrightarrow \neg b)$ which is inconsistent and must have probability zero as well. □

Another way of saying this is that if the cyclic model contains a possibil-

ity of an inconsistent assignment then requiring the noise on the variables to be independent will require that at least one of them is extreme. This proposition does not rely on there being two binary variables, but has to do with cyclic causality. The following result shows that it happens quite generally. The operator $\models$ represents logical entailment and $\rightarrow$ represents logical implication. $A \models B$ means A logically implies B.

Proposition 2. *If there exists an assignment $u_1 \ldots u_k$ to exogenous variables $U_1 \ldots U_k$ such that for all values v for X,*

$$u_1 \ldots u_k \models (X{=}v) \rightarrow (X{=}v')$$

where $v' \neq v$, the variables $U_1 \ldots U_k$ cannot be non-extreme and probabilistically independent.

Proof. If $u_1 \ldots u_k \models (X{=}v) \rightarrow (X{=}v')$ for $v' \neq v$ then $u_1 \ldots u_k \models (X{\neq}v)$. If this occurs for all v, then $u_1 \ldots u_k$ are logically inconsistent. Thus, $P(u_1 \ldots u_k) = 0$. If the variables $U_1 \ldots U_k$ are independent, then $P(u_1 \ldots u_k) = \prod_i P(u_i)$. This product could only equal zero if at least one of the $P(u_i)$ is zero. □

Note that this just requires a weak but sound reasoning procedure (i.e. one that implements $\models$). If there is a stronger logic behind it, such as $\mathcal{L}^+(S)$ of Halpern [2000], the results still hold, but the logic may be able to prove the inconsistency directly. Note that Halpern's logic does not include exogenous variables and so is not directly applicable.

Example 5. One way to avoid inconsistency is to make the noise variables dependent. For example, creating the dependency $u_2 \rightarrow u_1$ makes $u_2 \wedge \neg u_1$ inconsistent. This can be modelled by making $u_2 = u_1 \wedge u_5$ for some noise u_5. Equation (5.1) then becomes:

$$a \leftrightarrow (b \wedge u_1) \vee (\neg b \wedge u_1 \wedge u_5) \tag{5.3}$$

This can be reduced to:

$$a \leftrightarrow (b \vee u_5) \wedge u_1 \tag{5.4}$$

This is the style of many SEMs (see page 29 of Pearl [2009]). An SEM using formula (5.4) (with the corresponding equation for B) incorporates prior knowledge that A and B are positively correlated, as making one true can only increase the probability of the other being true. This SEM does not result in a unique probability for A or for B as there are many consistent solutions to the variables.

5.3 Equilibria in SEMs

An alternative to SEMs is an equilibrium model [Strotz and Wold, 1960], where the causes of each variable form a transition model of a Markov chain, and we are interested in the equilibrium distribution of this Markov chain. Strotz [1960] describes the contribution of that paper as:

> *If a causal interpretation of an interdependent system is possible it is to be provided in terms of a recursive system. The interdependent system is then either an approximation to the recursive system or a description of its equilibrium state.*

Equilibria in SEMs were explored by Iwasaki and Simon [1994] but they focussed on an equilibrium where the *values* of the variables are invariant. In the Markov chain semantics, the equilibrium is on a distribution over the variables rather than a particular value assignment.

One of the properties of the causal theories of Pearl [2009] is that local causal models are sufficient to predict all combinations of interventions (including the case of no interventions). For each variable X, with parents $\mathtt{pa}_X$, and for each combination of values, $\overline{v}$, to $\mathtt{pa}_X$, the probabilities $P(X|do(\mathtt{pa}_X = \overline{v}))$ fully specify the model. For this to still be true in a causal model with cycles we must work with the equilibrium of the system.

For Example 4, this semantics is defined in terms of a Markov chain with variables $A^0, A^1, \ldots$ and $B^0, B^1, \ldots$, where the superscript represents

a time point, with transition probabilities such as:

$$p_1 = P(a^t|b^t)$$
$$p_2 = P(a^t|\neg b^t)$$
$$p_3 = P(b^t|a^{t-1})$$
$$p_4 = P(b^t|\neg a^{t-1})$$

where a^t is the proposition that A is true at time t. The above model can be specified as an SEM where variables on the right hand sides can refer to a previous time (in such a way that there are no cycles in the temporally extended graph). E.g.:

$$a^t \leftrightarrow (b^t \wedge u_1^t) \vee (\neg b^t \wedge u_2^t) \tag{5.5}$$
$$b^t \leftrightarrow (a^{t-1} \wedge u_3^t) \vee (\neg a^{t-1} \wedge u_4^t) \tag{5.6}$$

where for all t, U_i^t are independently and identically distributed variables with probability p_i. The use of the previous time is used to avoid cycles in the temporally extended models. The aim is to determine the equilibrium distribution — the distribution over the variables that does not change in time.

This Markov chain has an equilibrium that satisfies:

$$P(a) = p_1 P(b) + p_2(1 - P(b)) \tag{5.7}$$
$$P(b) = p_3 P(a) + p_4(1 - P(a)) \tag{5.8}$$

Solving the simultaneous equations gives:

$$P(a) = \frac{p_1 p_4 + p_2(1 - p_4)}{1 - (p_1 - p_2)(p_3 - p_4)} \tag{5.9}$$
$$P(b) = \frac{p_3 p_2 + p_4(1 - p_2)}{1 - (p_1 - p_2)(p_3 - p_4)} \tag{5.10}$$

which are well defined for all $p_i \in [0, 1]$, except for the two cases: $p_1 = 1$, $p_2 = 0$, $p_3 = 1$, $p_4 = 0$ (which corresponds to $a \leftrightarrow b$) and $p_1 = 0$, $p_2 = 1$,

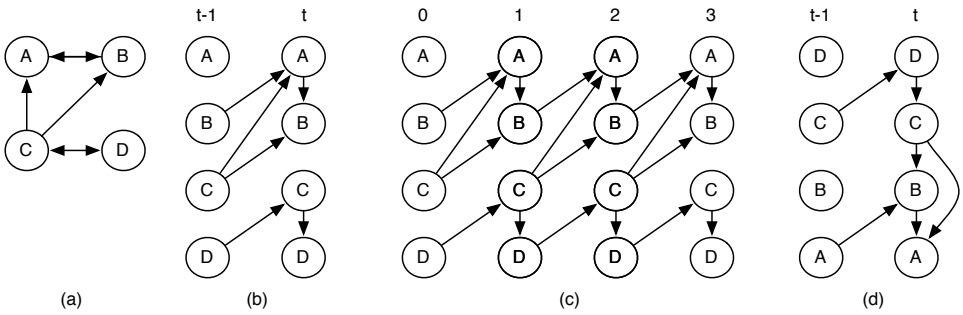


Figure 5.2: A causal network, its two-stage DBN for sample ordering $A < B < C < D$, its unrolled DBN, and the two-stage DBN for sample ordering $D < B < C < D$

$p_3 = 0$, $p_4 = 1$ (which corresponds to $a \leftrightarrow \neg b$). In these cases, there is an equilibrium for every value in $[0, 1]$. For the rest of this discussion, we ignore extreme probabilities that give these two extreme cases.

In the Markov chain, the A's at different times are different variables. There is no logical inconsistency that leads to the problem in the proof of Proposition 1.

To specify Equations (5.5) and (5.6), we need not only specify that A and B are dependent, but also that B depends on the previous value of A, and A depends on the current value of B. Intuitively, for each time, we sample B then A.

5.4 Equilibrium Models

In this section we define equilibrium models using Markov chains as an alternative to SEMs for representing causal knowledge. These models are slightly more complex than SEMs as the equilibrium distribution depends on the order in which the variables are sampled as well as when the distribution is sampled.

We assume finitely many discrete-valued variables and that all conditional probabilities are non-extreme. The non-extreme assumption is reasonable for learned models, where we may not want to assume a priori that any transition is impossible, but may not be appropriate for all domains. It

simplifies the discussion as all of the Markov chains are then ergodic, with a unique equilibrium distribution, independent of the starting state [Bremaud, 1999].

The **parents** of variable X are defined to be a minimal set of variables $\mathbf{Y}$ such that for all sets of variables $\mathbf{Z}$, where $\{X\}$, $\mathbf{Y}$ and $\mathbf{Z}$ are disjoint sets, the following statement holds:

$$P(X|do(\mathbf{Y})) = P(X|do(\mathbf{Y}, \mathbf{Z})).$$

That is, for all interventions where the variables in $\mathbf{Y}$ are set to particular values, changing the value of any other variables $\mathbf{Z}$ does not affect X. This is like the standard definition of conditional independence, but involves interventions rather than observations [Pearl, 2009]. To carry out an intervention on a variable X, the usual causal distribution for X is replaced with $P(X{=}v) = 1$. This parent relation induces a directed graph that can contain cycles, but is irreflexive (there is no arc from a variable to itself).

Define a **causal network** to be an irreflexive directed graph where the nodes are random variables. A causal mechanism for each variable X consists of a conditional probability $P(X|do(\mathtt{pa}_X))$ where $\mathtt{pa}_X$ is the set of parents of X in the causal network.

The post-intervention semantics can be defined by constructing a two stage dynamic Bayesian network (DBN) [Dean and Kanazawa, 1989]. A two-stage DBN specifies how each variable at the current stage depends on variables at the current stage and variables from the previous stage. Defining a DBN depends on both the structure of the causal network and a **sample ordering** which is a total ordering of the variables, such as $X_1 < X_2 < \cdots < X_n$. For each variable X, define $\mathtt{pa}_X^-$ to be the set of those parents of X that come before X in the sample ordering, and $\mathtt{pa}_X^+$ to be the set of those parents of X that come after X in the sample ordering. Thus, the parents of X are $\mathtt{pa}_X = \mathtt{pa}_X^+ \cup \mathtt{pa}_X^-$. Each X_i, depends on its parents in $\mathtt{pa}_X^-$ at the current stage, and on its parents in $\mathtt{pa}_X^+$ at the previous stage.

A causal network with variables $\{X_1, \ldots, X_n\}$ and sample ordering $X_1 < X_2 < \cdots < X_n$ defines a decomposition of a discrete-time Markov chain

where the state S^t at time t can be described by the variables $X_1^t, \ldots, X_n^t$ for each time t, and for each causal variable X for each time t, the Markov chain variable X^t has parents $\{Y^t : Y \in \mathtt{pa}_X^-\} \cup \{Y^{t-1} : Y \in \mathtt{pa}_X^+\}$. X^t is independent of all variables $Z^{t'}$ for $t' < t$ given these parents and is independent of all variables Z^t where $Z < X$ given these parents in the Markov chain. Thus the causal network with the sample ordering defines the decomposition of the state transition function:

$$\begin{aligned} P(S^t|S^{t-1}) &= P(X_1^t, \ldots, X_n^t|S^{t-1}) \\ &= \prod_{i=1}^{n} P(X_i^t|X_1^t \ldots X_{i-1}^t, S^{t-1}) \\ &= \prod_{i=1}^{n} P(X_i|\mathtt{pa}_{X_i^t}^- \cup \mathtt{pa}_{X_i^{t-1}}^+) \end{aligned} \tag{5.11}$$

where the conditional probabilities for the Markov chain, $P(X_i^t|\mathtt{pa}_{X_i^t}^- \cup \mathtt{pa}_{X_i^{t-1}}^+)$, come from the $P(X|\mathtt{pa}_X)$ in the causal network. The distribution of the causal model (after interventions) is the equilibrium distribution of the induced Markov chain.

Example 6. Consider the causal network in Figure 5.2 (a), where the double-ended line segment represents two arcs. In this example, the parents of A are B and C, the parents of B are A and C, the parent of C is D, and the parent of D is C.

Figure 5.2 (b) shows the two-stage DBN with sample ordering $A < B < C < D$. The left nodes represent the variables at time $t-1$ and the right nodes represent the variables at time t. This represents the Markov chain given in Figure 5.2 (c), where the structure is repeated indefinitely to the right. Each of the conditional probabilities is defined as part of the causal network.Figure 5.2 (d) shows the two-stage DBN for the same causal network with sample ordering $D < C < B < A$.

5.5 The Sample Ordering

The following example shows that the equilibrium distribution can depend on the sample ordering:

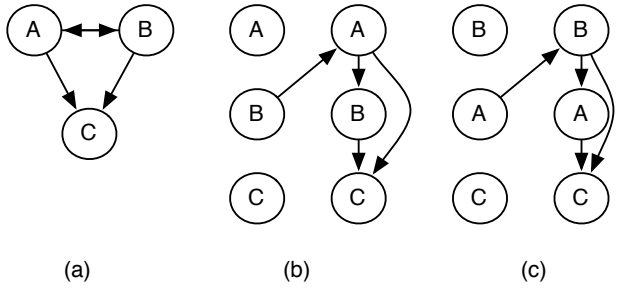


Figure 5.3: A causal network and two induced two-stage Bayesian networks

Example 7. Consider the causal network shown in Figure 5.3(a), with the causal probabilities:

$$P(a|do(b)) = 0.1 \quad P(a|do(\neg b)) = 0.9$$
$$P(b|do(a)) = 0.9 \quad P(b|do(\neg a)) = 0.1$$
$$P(c|do(a \wedge b)) = P(c|do(\neg a \wedge \neg b)) = 0.9$$
$$P(c|do(\neg a \wedge b)) = P(c|do(a \wedge \neg b)) = 0.1$$

One way to understand this network is that doing B tends to change A to be different to B, and doing A tends to change B to be the same as A. C has high probability if A and B have the same value.

Figure 5.3 (b) shows the two-stage DBN with the sample ordering $A < B < C$. Figure 5.3 (c) shows the two-stage DBN with the sample ordering $B < A < C$.

In the equilibrium distribution of Figure 5.3 (b), $P(c) = 0.82$, whereas in the equilibrium distribution of Figure 5.3 (c), $P(c) = 0.18$. Intuitively, in (b), A is sampled, then B is sampled, based on that value of A, and so they tend to have the same value and so C tends to be true. Whereas in (c), B is sampled, then A is sampled, based on that value of B, and so they tend to have different values and so C tends to be false.

5.5.1 The Sample Ordering is Part of the Model

Dependency on the sample ordering is not a side-effect of a particular method or an error to be minimized; the dependency arises as an unavoidable part of the model which needs to be specified.

Example 8. Consider the causal model shown in Figure 5.4(a). All the arcs represent positive correlations except for Betty's influence on Alfred which is negatively correlated; Alfred tends to do the opposite of whatever Betty, did but Betty often imitates Alfred's behaviour.

Suppose each agent always buys or sells gold at their own fixed time every day. This will induce a consistent causal ordering on the information each other agent has over time as they make their trading decisions. One possible schedule is shown in Figure 5.4(b) with the appropriate causal arcs from the model added. Under this ordering, Betty tends to follow Alfred's actions from that morning while Alfred often acts opposite to what Betty did yesterday. Cindy, who trades after Alfred in the morning, always observes that Betty and Alfred's trades are different, so Cindy is always aware of this when making her purchasing decisions. Meanwhile, Doug trades at the end of the day and observes that Betty's behaviour is largely in sync with Alfred's. Doug acts on the belief that Alfred and Betty are in general agreement. Cindy and Doug could have identical dependencies (ie. $p(C|do(A), do(B) \equiv p(D|do(A), do(B))$ on the actions of Alfred and Betty, yet Cindy and Doug's beliefs about the joint behaviour of Alfred and Betty would be very different. This means the behaviour of Cindy and Doug (ie. the marginal probabilities $p(C)$ and $p(D)$) could be very different, solely due to the sample ordering of the variables.

Example 8 shows that when a node, such as C, has two parents in a cycle, the marginal distribution of C in the equilibrium can depend on the sample ordering. In this example the dependency is caused by the fact that A's values are positively correlated with B's value while B's values are *negatively correlated* with A's value. In general, this dependency on order can occur whenever there is a node connected to at least two other nodes in a cycle that have a non-zero probability of an inconsistent assignment.

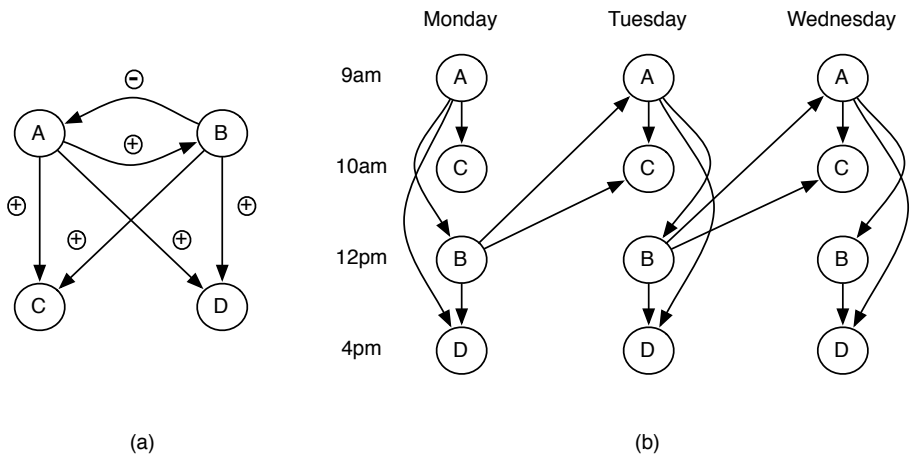


Figure 5.4: Example schedule (b) for gold trades of four people with a cyclic causal interaction based defined by causal model (a). Positive and negative correlations indicated on arcs.

5.6 Inference in Equilibrium Models

The inference problem we consider here is, given a causal network and a sample ordering, determine $P(X|do(\mathbf{Y}), \mathbf{Z})$ for disjoint sets of variables $\{X\}$, $\mathbf{Y}$ and $\mathbf{Z}$, which means the posterior distribution of X after doing $\mathbf{Y}$ and then observing $\mathbf{Z}$ in the equilibrium distribution[1]. This can be computed by replacing the causal mechanisms of the variables in $\mathbf{Y}$ with the intervention values, computing the equilibrium distribution, conditioning on $\mathbf{Z}$ and marginalizing over the remaining variables.

One way to compute the equilibrium distribution is to sample from it, sampling each variable in turn, according to the sample ordering. This is an instance of Markov Chain Monte Carlo (MCMC) sampling [Bremaud, 1999]. A state is an assignment of a value to each variable. For each sampling step, a state S^t is generated using the transition model conditioned on previous state

[1]Note that this is *not* counterfactual reasoning [Pearl, 2009], which would be observing then doing. In general, there could be arbitrary sequences of observing and doing. It is what Dash [2005] calls the manipulated-equilibrated model, but our equilibrium is over distributions.

S^{t-1}. The samples generated, after some burn-in period, can be considered as random samples from the equilibrium distribution.

When variables are selected according to a fixed sample ordering, the probabilities from the causal model can be used directly by using Gibbs sampling. Suppose the sample ordering being used is $X_1 < X_2 < \cdots < X_n$. At each sample step there is already a value assigned to each node from previous steps or from the random initialization. Using equation (5.11), each node X_i is sampled independently where the parent values $\mathtt{pa}^{-}_{X_i^t} \cup \mathtt{pa}^{+}_{X_i^{t-1}}$ are provided by the current values of all the other variables. A new value for X_i is chosen and set, then sampling proceeds to sample variable X_{i+1}. Thus Gibbs sampling has a natural correspondence to the form of the equilibrium distribution. This MCMC approach is what will be used for representing large spatial policies in the next chapter. In this chapter we will show that, at least for smaller models, we can do better than sampling.

The equilibrium distribution can also be computed directly using Gaussian elimination, which takes time polynomial in the number of states. An alternative iterative method is to start with a probability distribution over states and repeatedly use the model to compute a distribution over the next state. This converges to the stationary distribution (unlike MCMC which gives samples that are distributed according to the stationary distribution) at a geometric rate [Bremaud, 1999]. This algorithm is polynomial in state space as it entails computing the probability of each state.

AI research over the last few decades has demonstrated that we can typically do much better than polynomial in the state space by exploiting the sparseness of the representation. Exploiting conditional independence is the basis for efficient inference in Bayesian networks [Pearl, 1988], using techniques such as variable elimination [Zhang and Poole, 1994]. These techniques are exponential in the tree-width [Mateescu et al., 2002], and are linear for a fixed tree-width.

5.7 Equilibrium Bayesian Networks

We now consider a compact model of the equilibrium distribution of the Markov chain induced by the cyclic causal network. The model takes ad-

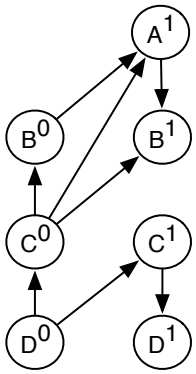


Figure 5.5: An equilibrium belief network

vantage of the sparse structure of the causal network to represent the equilibrium compactly. The goal is to be able to use exact inference on this model to compute marginal distributions from the equilibrium of a cyclic causal network. Depending on the amount of structure used to represent the equilibrium, the answers to marginal queries may be exact or may be approximations.

First, consider that a two stage DBN is like a Bayesian network with two parts, the "previous variables" and the "current variables", where the distribution over the previous variables is not specified. Suppose we now create a Bayesian network over just these previous variables. For each previous variable, there is a conditional probability given some other previous variables as parents such that this previous network is acyclic. This previous network can be constructed to answer all marginal queries not answered by the current network. If all these queries are answered as they would be in the equilibrium distribution then this previous network represents the equilibrium distribution. If a previous network represents the equilibrium distribution then the current variables which depend on them will also represent the equilibrium distribution. If the previous network is an approximation to the equilibrium distribution, the current variables will represent a closer approximation to the equilibrium distribution.

An *Equilibrium Bayesian Network (EBN)* for causal network G, sample

ordering $\mathtt{SO}$ and previous ordering $\mathtt{PO}$, is a Bayesian network, where for each variable X in G, there is a *current* node, X^1, and there is a *previous* node X^0 if X has a child in G earlier in the sample ordering. The set of all nodes from both stages is $\mathbf{V}$. The parents of current nodes are derived from the causal network and the sample ordering: X^1 has parents $\{\mathbf{Y}^1 : Y \in \mathtt{pa}_X^-\} \cup \{\mathbf{Y}^0 : Y \in \mathtt{pa}_X^+\}$. The parents of previous nodes must be previous nodes that are earlier in the previous ordering. The parents of a node X anywhere in an EBN will be given by $\mathtt{pa}(X)$.

Example 9. Consider the causal network from Example 6 with sample ordering $A < B < C < D$. Figure 5.5 shows the EBN for this network given the previous ordering $B < C < D$. Unlike the two-stage DBN, shown in Figure 5.2(b), the EBN is complete in itself and is not a template for a temporally extended network.

An EBN contains two sets of parameters representing the conditional probability of a node given its parents. The probabilities from the causal network are modelled by γ which defines the conditional probability of each current node given its parents, $G^\gamma(X^1|\mathbf{Y}^1, \mathbf{Y}^0)$. The conditional probabilities for nodes in the previous step of an EBN are modelled by θ as $G^\theta(X^0|\mathbf{Y}^0)$ and define a distribution over all previous nodes $G^\theta(\mathbf{X}^0) = \prod_{X^0 \in \mathbf{X}^0} G^\theta(X^0|\mathbf{Y}^0)$. The θ parameters are the ones that will be modified to model the equilibrium distribution. The parameters γ are given and remain fixed.

The factorization of the marginal probability of a set of nodes $\mathbf{X}$ represented by an EBN is:

$$G^{\theta,\gamma}(\mathbf{X}) = \sum_{\mathbf{Y}^1} \sum_{\mathbf{Y}^0} G^\gamma(\mathbf{X}|\mathbf{Y}^1, \mathbf{Y}^0) G^\gamma(\mathbf{Y}^1|\mathbf{Y}^0) G^\theta(\mathbf{Y}^0) \tag{5.12}$$

5.7.1 The Iterative Improvement Algorithm

The purpose of an EBN is to allow efficient querying of marginal probabilities of variables in the causal network without needing to instantiate a DBN. These queries could be the marginal probability of a single variable, the joint

probability of multiple variables or conditional distributions of a subset of the variables. Answering queries about an equilibrium distribution using an EBN involves three steps:

- Determine the structure of the previous stage of the EBN.
- Iteratively compute the conditional probabilities for the previous stage, setting the parameters for the previous stage using the converged conditional probabilities.
- Condition on and query the current variables.

Given the structure of an EBN, the *Iterative Improvement Algorithm (IIA)*, shown on the following page, computes the θ parameters by repeatedly updating the conditional probability tables for the previous variables. For each previous variable X^0 in the EBN with parents $\mathbf{Y}^0$, IIA computes $G^{\theta,\gamma}(X^1|\mathbf{Y}^1)$ from the full model, where X^1 and $\mathbf{Y}^1$ are the current counterparts of X^0 and $\mathbf{Y}^0$. The value for $G^{\theta}(X^0|\mathbf{Y}^0)$ is then replaced by this computed value. Since X^1 may have some different parents than X^0, computing $G^{\theta,\gamma}(X^1|\mathbf{Y}^1)$ requires marginalizing out any other nodes, $\mathbf{W}$. For example, for the EBN from Figure 5.5, the value for $G^{\theta}(B^0|C^0)$ would be updated by the result of the following computation:

$$G^{\theta'}(B^0|C^0) = \sum_{A^1,B^0,C^0,D^0} G^{\gamma}(B^1|C^0,A^1)G^{\gamma}(A^1|B^0,C^0)G^{\gamma}(C^1|D^0)\times \\ G^{\theta}(B^0|C^0)G^{\theta}(C^0|D^0)G^{\theta}(D^0)$$

5.8 Analysis

The IIA algorithm can be shown to converge geometrically to the marginal probabilities from the equilibrium distribution if the belief network on the previous variables can represent the equilibrium distribution. When the previous variables are fully connected, IIA converges since the algorithm is then equivalent to iteratively applying the full Markov chain transition matrix to

Algorithm 5: `IIA`- Iterative Improvement Algorithm

input: G, γ, ϵ

Set θ to random initial values

$\Delta = \infty$

while $\Delta \geq \epsilon$ **do**

  for $(x^0, \mathbf{y}^0) \in (X^0, \mathtt{pa}(X^0))$ **do**

    $\mathbf{W} = \mathbf{V} - (X^1 \cup \mathbf{Y^1})$

    $G^{\theta'}(x^0|\mathbf{y}^0) = \sum_{\mathbf{W}} G^{\gamma}(X^1 = x^0|\mathbf{Y}^1 = \mathbf{y}^0, \mathbf{W}) G^{\theta}(\mathbf{W})$

  $\Delta = |\theta - \theta'|$

  $\theta = \theta'$

return θ

a representation of the state, which is known to converge geometrically [Bremaud, 1999]. This section shows that if the sub-graph is not fully connected, IIA can still converge exactly to the marginals from the equilibrium if the belief network over the previous variables is "expressive enough", otherwise IIA provides an approximation to the equilibrium distribution. Whether a belief network over the previous variables is expressive enough will be defined in terms of the representation of cliques of previous variables.

A set $\mathbf{S}$ of nodes forms a **previous clique** in an equilibrium belief network if, for each node $N \in \mathbf{S}$, $\{M^0 : M \in \mathbf{S}$ and $N^0 < M^0$ in the previous ordering$\} \subseteq \mathtt{pa}_{N^0}$. A set of nodes $\mathbf{S}$ is **represented** in an EBN if the set of previous nodes that are ancestors of elements of $\mathbf{S}$, i.e., $\{M : \exists A \in \mathbf{S}$ such that M^0 is an ancestor of $A^1\}$, forms a previous clique and that previous clique is also represented. This recursive definition ensures that all cliques of ancestors relevant for computing marginals of represented variables are available in the previous stage.

Example 10. Figure 5.6 (a) shows a 5 node causal network. (b) is the induced two-stage DBN under the sample ordering $A < B < C < D$, which expands to the temporally extended network shown in (c). (d) shows an equilibrium belief network where the previous nodes are disconnected. The iterative improvement algorithm converges exactly to the correct marginals distributions from the equilibrium for all of the variables. However, it

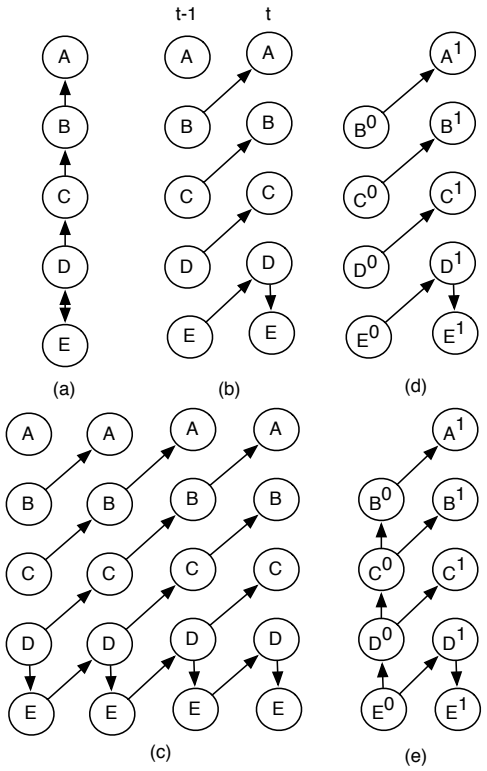


Figure 5.6: (a) A causal network, (b) its induced two-stage DBN (c) the unfolded DBN and (d) and (e) two equilibrium belief networks

is not exact for marginals on non-singleton sets of variables (except for DE). (e) shows an equilibrium belief network under the previous ordering E, D, C, B, A where neighbouring nodes are joined. For this EBN, the algorithm converges exactly to the correct marginal for each singleton variable, as well as for AB, BC, CD and DE. However, the resulting network does not fully represent the equilibrium distribution as it gets the wrong answer, for example, for $P(A|D)$. To understand this, consider the unrolled DBN in (c). A and D are dependent, but this dependence cannot be fully represented by the equilibrium belief network (e).

5.8.1 Proof of Convergence

We now prove that if an EBN is defined so that the query set $\mathbf{S}$ is represented in the previous step, then the parameters θ will be sufficient to compute marginals of the variables in $\mathbf{S}$ from the equilibrium distribution. The Iterative Improvement Algorithm is guaranteed to converge geometrically to a unique set of parameters, θ^*, that defined the equilibrium distribution.

The following two well known theorems will be useful in showing the convergence of our algorithm.

Proposition 3 (Banach Fixed Point Theorem - [Agarwal et al., 2001])**.** *Let (X, d) be a complete metric space of points X and distance measure $\delta(x, y)$ for all $x, y \in X$. Define a transformation mapping $T : X \to X$ which is a* contraction*; meaning that there exists $k \in [0, 1)$ such that*

$$\delta(T(x), T(y)) \leq k\ \delta(x, y) \tag{5.13}$$

for all x, y in X. Applying T iteratively n times is denoted as $T^n(x)$.

Then starting from any point $x \in X$ there is a unique fixed point, $x^* \in X$, which can be found by

$$\lim_{n\to\infty} T^n(x) = x^*. \tag{5.14}$$

The system converges by $k^n/1 - k$ over n steps.

Proposition 4 (Cauchy Inequality - [Cauchy, 1821])**.** *For any two vectors x and y the following inequality holds:*

$$\left(\sum_i x_i y_i\right)^2 \leq \sum_i x_i^2 \sum_i y_i^2 \tag{5.15}$$

We begin by defining a complete metric space (Θ, δ) where Θ is the set of all latent model parametrizations of an EBN and δ is a distance measure. The L2-norm can be used to define the distance between distributions represented by two EBNs G^θ and G^ψ with two latent model parametrizations

$\theta, \psi \in \Theta$. The distance measure is:

$$\begin{aligned}\delta(\theta, \psi) &= \|G^{\theta}(\mathbf{X}^0) - G^{\psi}(\mathbf{X}^0)\|_2 \\ &= \Bigg[\sum_{\mathbf{x}^0 \in \mathbf{X}^0} \Big(G^{\theta}(\mathbf{x}^0) - G^{\psi}(\mathbf{x}^0) \Big)^2 \Bigg]^{1/2} \end{aligned} \tag{5.16}$$

where $\mathbf{x}^0 \in \mathbf{X}^0$ are all the complete assignments of values to all the variables in the latent model.

The transformation mapping, T from Banach's theorem, is the one step update for the latent model distribution, which is:

$$G^{\theta_t}(\mathbf{X}_t^0) = \sum_{\mathbf{X}_{t-1}^0} G^{\gamma}(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0 | \mathbf{X}_{t-1}^0) G^{\theta_{t-1}}(\mathbf{X}_{t-1}^0) \tag{5.17}$$

To show convergence of the algorithm, it suffices to show that $G^{\theta_t}(\mathbf{X}_t^0)$ is a contraction with respect to δ.

Lemma 1. *The iterative improvement algorithm converges to a distribution which is a unique fixed point.*

Proof. To show this we expand the distance metric using (5.17):

$$\delta(\theta,\psi) = \|G_t^\theta(\mathbf{X}_t^0) - G_t^\psi(\mathbf{X}_t^0)\|_2 = \left[\sum_{\mathbf{X}_t^0}(G_t^\theta(\mathbf{X}_t^0) - G_t^\psi(\mathbf{X}_t^0))^2\right]^{1/2}$$

$$= \left[\sum_{\mathbf{X}_t^0}\Bigg(\sum_{\mathbf{X}_{t-1}^0} G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)G^{\theta_{t-1}}(\mathbf{X}_{t-1}^0) \; - \right.$$

$$\left.\sum_{\mathbf{X}_{t-1}^0} G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)G^{\psi_{t-1}}(\mathbf{X}_{t-1}^0)\Bigg)^2\right]^{1/2}$$

$$= \left[\sum_{\mathbf{X}_t^0}\left(\sum_{\mathbf{X}_{t-1}^0} G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)\left[G^{\theta_{t-1}}(\mathbf{X}_{t-1}^0) - G^{\psi_{t-1}}(\mathbf{X}_{t-1}^0)\right]\right)^2\right]^{1/2}$$

$$\leq \left[\sum_{\mathbf{X}_t^0}\sum_{\mathbf{X}_{t-1}^0}\left(G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)\right)^2 \times \right.$$

$$\left.\sum_{\mathbf{X}_{t-1}^0}\left[G^{\theta_{t-1}}(\mathbf{X}_{t-1}^0) - G^{\psi_{t-1}}(\mathbf{X}_{t-1}^0)\right]^2\right]^{1/2}$$

$$= \left[\sum_{\mathbf{X}_{t-1}^0}\left[G^{\theta_{t-1}}(\mathbf{X}_{t-1}^0) - G^{\psi_{t-1}}(\mathbf{X}_{t-1}^0)\right]^2 \times \right.$$

$$\left.\sum_{\mathbf{X}_t^0}\sum_{\mathbf{X}_{t-1}^0}\left(G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)\right)^2\right]^{1/2}$$

$$= \left[\sum_{\mathbf{X}_{t-1}^0}\left[G^{\theta_{t-1}}(\mathbf{X}_{t-1}^0) - G^{\psi_{t-1}}(\mathbf{X}_{t-1}^0)\right]^2\right]^{1/2} \times$$

$$\left[\sum_{\mathbf{X}_t^0}\sum_{\mathbf{X}_{t-1}^0}\left(G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)\right)^2\right]^{1/2}$$

$$= \delta(\theta_{t-1},\psi_{t-1})\left[\sum_{\mathbf{X}_t^0}\sum_{\mathbf{X}_{t-1}^0}\left(G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0|\mathbf{X}_{t-1}^0)\right)^2\right]^{1/2} \tag{5.18}$$

Note that, as required by Banach's theorem, (5.18) contains only the distance metric on the untransformed points and a constant term since the second term involves only G^γ which uses the constant parameters γ. The maximum value of the second term of equation (5.18) is $\sqrt{|\mathbf{X}^1|}$ where $|\mathbf{X}^1|$ is the number of variables in the model. Thus, the constant k needed to bound the contraction for Theorem 3 is:

$$
\begin{aligned}
k &= \frac{1}{\sqrt{|\mathbf{X}^1|}} \left[\sum_{\mathbf{X}_t^0} \sum_{\mathbf{X}_{t-1}^0} \left(G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0 | \mathbf{X}_{t-1}^0) \right)^2 \right]^{1/2} \\
&= \left[\frac{1}{|\mathbf{X}^1|} \sum_{\mathbf{X}_{t-1}^0} \sum_{\mathbf{X}_t^0} \left(G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0 | \mathbf{X}_{t-1}^0) \right)^2 \right]^{1/2} \\
&< \left[\frac{1}{|\mathbf{X}^1|} \sum_{\mathbf{X}_{t-1}^0} \sum_{\mathbf{X}_t^0} G^\gamma(\mathbf{X}_{t-1}^1 = \mathbf{X}_t^0 | \mathbf{X}_{t-1}^0) \right]^{1/2} \\
&= \left[\frac{1}{|\mathbf{X}^1|} \sum_{\mathbf{X}_{t-1}^0} 1 \right]^{1/2} \\
&= 1
\end{aligned}
$$

The inequality holds if one of the probabiltiies is non-extreme since $p^2 < p$ if $0 < p < 1$.

By Banach's theorem the transition will have a unique fixed point θ^* and the algorithm will converge at a rate of $k/(1-k)$ per iteration. □

Proposition 5. *For each represented set of variables in an EBN, the iterative improvement algorithm will converge to the marginal of the equilibrium distribution of the Markov chain on those variables.*

Proof. Imagine having an oracle which could exactly answer any marginal queries from the equilibrium distribution. Suppose we use this oracle to answer queries about the previous stage. In computing the marginal distribution for a represented set of variables in the current stage of the EBN,

the nodes that are not ancestors can be pruned and all others at the current stage can be summed out. What remains will be a marginal probability distribution over previous variables. The oracle can provide answers to queries about this distribution. A sufficient oracle is one that can answer these queries about the equilibrium distribution (but not necessarily any others).

If the equilibrium distribution of the Markov chain were known and projected onto the previous stage network G^θ of an EBN, then the previous stage would be a sufficient oracle for computing any represented marginals from the equilibrium. This is sufficient because by definition a represented set of variables has its previous clique represented in the previous stage, so a sufficient oracle will need to model all of this information. Thus, the equilibrium distribution of the Markov chain, is also an equilibrium of the EBN. Lemma 1 shows that using the Iterative Improvement Algorithm, the EBN is guaranteed to converge to a unique equilibrium, so it must be converging to the equilibrium of the Markov chain on each represented set of variables.

□

5.9 Evaluation

In this section we empirically evaluate the accuracy of the algorithm when it is not exact. If the structure on the previous variables does not match the condition of Proposition 5 then we do not expect the equilibrium of the EBN to exactly match the Markov chain equilibrium, but expect it to be an approximation. The following example gives an empirical evaluation of the approximation.

Example 11. One case where the previous variables need to be fully connected to satisfy Proposition 5 is when the causal network is a double linked chain:

$$A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow \ldots$$

The iterative improvement algorithm converges to the equilibrium distribution on the represented variables when the previous variables are fully connected. If the previous variables are not fully connected, it converges

bound	#iters	total error	max error
1	82	1.7	0.36
2	381	0.072	0.023
3	374	0.0031	0.00066
4	369	0.00016	3.8e-05
5	385	5.0e-05	1.6e-05
6	387	4.7e-07	1.7e-07
7	384	5.0e-08	1.8e-08
8	382	5.5e-09	1.2e-09
9	383	4.4e-10	2.1e-10
10	379	8.2e-11	3.6e-11

Figure 5.7: Empirical results from Example 11

to an approximation. To test how good various approximations are, we created such a chain with 16 binary variables. We then constructed 1000 random parametrizations (with various levels of skewed distributions, where the probabilities were of the form r^k or $1 - r^k$ for r a random number and k an integer in $[1, 4]$). We then chose the 10 parametrizations where the approximations were worst. Here we report on the worst parametrization (as this was typical of the others).

Instead of considering all of the ancestors in the definition of *represented*, we only considered the ancestors to a depth bound. For this example, when the depth bound is 1, the previous variables are disconnected. When the depth bound in 2, each variable, except the last one, has one parent. When the depth bound is b, each variable (other than the boundary cases) has $b - 1$ parents. Figure 5.7 gives the number of iterations until convergence (the probabilities change by less than 10^{-15}), the sum of the errors on the marginals of the individual variables, and the maximum error for individual variables (to two significant digits) as a function of the bound.

5.10 Commentary

The idea that a cyclic causal model represents an equilibrium distribution of a Markov chain is not new; Strotz [1960] argued that when there are variables that are interdependent in a cyclic ordering, the fixed point in values was a specification error. One consistent interpretation of a cyclic model was an equilibrium of the system of causal equations. Fisher [1970] followed this by giving conditions for the equilibrium to be well defined. There is also a vast literature on learning cyclic causal models that focuses on the interpretation that causal cycles are caused by unmeasured latent variables [Glymour and Spirtes, 1988; Schmidt and Murphy, 2009].

If SEMs are the right model for causality, they should work for simple cases. We have argued that SEMs impose undesirable dependencies, and proposed an Equilibrium Bayesian Network models as an alternative. We gave an algorithm for constructing a network that can answer queries about the equilibrium.

It should also be noted that the counterexample of Neal [2000] to Pearl and Dechter [1996] does not work for the equilibrium semantics. D-separation holds with the Markov chain equilibrium semantics as there are no cycles in the Markov chain.

Cyclic causal models have a natural connection to spatial landscape policies composed of local, conditional policies. The Equilibrium Bayesian Network model and Iterative Improvement Algorithm presented here can provide exact solutions and structured approximations for cyclic causal models. However, the `IIA` algorithm is exponential in the tree-width of the EBN[2]. So, it is difficult to use this method directly for representing spatial landscape policies for large planning problems due to the computational cost of exact inference on models with many variables. The next chapter shows how to use stochastic simulation to define an equilibrium spatial policy.

[2]Recall that tree-width is a measure of sparseness that is related to the size of the largest family of nodes in the network (see Section 3.2.1).

Chapter 6

Equilibrium Landscape Policies

The previous chapter showed how a cyclic causal model could be used to define a joint distribution over a number of interrelated variables using local, conditional distributions. A landscape policy is a distribution over joint actions taken at multiple locations. This distribution can be defined as the equilibrium distribution of a cyclic causal model where each variable represents a local distribution over actions to be taken at a cell given actions at other relevant cells. For the forestry planning problem, this means defining a cyclic causal model containing thousands of action variables. It is not feasible to use the EBN method from the previous chapter directly for estimating the landscape policy on networks with many, interconnected variables. However, stochastic simulation can be used to estimate the equilibrium distribution for large numbers of variables. This chapter defines an equilibrium landscape policy and presents algorithms for sampling actions from it, estimating the marginal probability for individual cells and estimating the gradient of the landscape policy with respect to the policy parameters of the local cell policies.

6.1 Cyclic Local Cell Policy

A local cell policy will be defined as a log-linear distribution similar to (4.1) except now the features can also model the cyclic dependency of an action at one location on the actions at other locations.

A *cell policy*, $\pi_c(a|\mathbf{a}_{-c}, \mathbf{s}, \theta)$, is a causal distribution over the actions at cell c conditioned on the landscape state $\mathbf{s}$, actions at all other cells $\mathbf{a}_{-c} \in A$ and parameters θ. Intuitively, π_c models the probability the agent would choose action a in cell c if c were the last cell to be decided, after the actions for all other cells, $\mathbf{a}_{-c}$, had been decided. Using Pearl's causal notation [Pearl, 2009] this would be written as $\pi_c(a|do(\mathbf{a}_{-c}), \mathbf{s}, \theta)$. The actions for other cells are set as interventions rather than observations. We omit the $do(\cdot)$ notation for conciseness.

The features, $f_c(\mathbf{s})$, used in section 4.3.1 can now be extended to be $f_c(\mathbf{a}_{-c}, \mathbf{s})$ which can include information about actions at other cells $\mathbf{a}_{-c}$. This makes the potential function:

$$\psi(a, \mathbf{a}_{-c}, \mathbf{s}, \theta) = \sum_f \theta[f, a] f_c(\mathbf{a}_{-c}, \mathbf{s}) \tag{6.1}$$

The new spatial cell-policy is:

$$\pi_c(a|\mathbf{a}_{-c}, \mathbf{s}, \theta) = \frac{\exp(\psi(a, c, \mathbf{a}_{-c}, \mathbf{s}, \theta))}{\sum_{b \in A} \exp(\psi(b, c, \mathbf{a}_{-c}, \mathbf{s}, \theta))} \tag{6.2}$$

As in Section 4.3.1, the policy is spatially stationary since the same parameters are used across all cells and each cell depends on the states and actions of its surroundings using the same function. The identity of the cell is not used.

6.1.1 Aggregate Features

One way to represent relations between cells is by defining features which aggregate information from a number of neighbouring cells. In the original landscape policy definition from Chapter 4, aggregate features could be defined relating to the *states* of other cells, such as:

- total MPB infestation level in neighbouring cells
- neighbouring cell with the largest volume of trees
- percentage of nearby cells (see biodiversity constraints in section 2.3)

The spatial cell policy in (6.2) can now also contain features aggregating the *actions* being taken at other cells, for example:

- number of neighbouring cells being cut

6.2 The Landscape Policy

The cell policy π_c is a cyclic definition with each cell depending on the actions at other cells. Thus, (6.2) does not *directly* define a conditional distribution of an action at a cell given the distribution of actions at other cells. As discussed in Chapter 5, one natural interpretation of a cyclic system is as the equilibrium of a Markov chain.

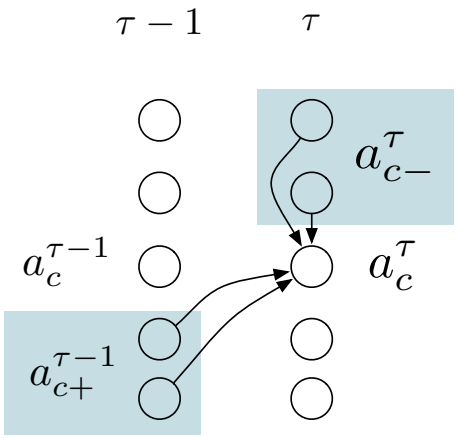


Figure 6.1: Visualization of a single sample update in the Markov chain. For an action a_c, the actions at other cells are split into two partitions, $\mathbf{a}_{-c} = \mathtt{pa}_{a_c^\tau}^{+} \cup \mathtt{pa}_{a_c^\tau}^{-}$, based on the sample ordering that will be used by the local cell policy.

Recall from Section 5.4 that a sample ordering is a total ordering of the cells. Given a sample ordering, a Markov chain can be induced from the

cell policy. Let sampled landscape actions at step τ of the Markov chain be indexed[1] as $\mathbf{a}^\tau$. During each step, new actions are sampled for each cell according to the sample ordering using equation (6.2). When the action for cell c is being sampled, the actions for all other cells are fixed to their most recently sampled values. The actions at all other cells $\mathbf{a}_{-c}$ are split into two partitions using the same notation introduced on page 71, as shown in Figure 6.1. The partitions are the cells before c in the sample ordering, $\mathtt{pa}^-_{a^\tau_c}$, and the cells after c in the sample ordering, $\mathtt{pa}^+_{a^\tau_c}$. The probability $\pi_c(a|\mathbf{a}_{-c}, \mathbf{s}, \theta)$ in equation (6.2) can now be restated as $\pi_c(a|\mathtt{pa}^+_{a^\tau_c} \cup \mathtt{pa}^-_{a^\tau_c}, \mathbf{s}, \theta)$. For conciseness, the given state and parameters will often be omitted, referring to the spatial cell policy as $\pi_c(a|\mathtt{pa}^+_{a^\tau_c} \cup \mathtt{pa}^-_{a^\tau_c})$. This conditional probability defines the transition model of the Markov chain. The sampling algorithm `sampleStepGibbs`, on page 94, samples from the *landscape policy* $\Pi(\mathbf{a}|\mathbf{s}, \theta)$ using Gibbs sampling to sample from the equilibrium distribution on the Markov chain.

The equilibrium of the Markov chain can be expressed in terms of the landscape transition probability $p_\kappa(\mathbf{b}, \mathbf{a})$, which is the probability of moving from landscape action $\mathbf{b}$ to landscape action $\mathbf{a}$ in κ steps.

In terms of the Markov chain being sampled, the definition of the single-step transition between any two particular landscape actions $\mathbf{a}^{\tau-1}$ and $\mathbf{a}^\tau$ is:

$$p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) = \prod_c \pi_c(a^\tau_c|\mathtt{pa}^+_{a^\tau_c} \cup \mathtt{pa}^-_{a^\tau_c}) \tag{6.3}$$

The probability, in the equilibrium distribution, of reaching $\mathbf{a}^\tau$ can be expressed as the expected probability of transitioning to $\mathbf{a}^\tau$ in one step from any other landscape action:

$$\Pi(\mathbf{a}^\tau) = \mathbb{E}_{\mathbf{a}^{\tau-1} \in \mathbf{A}} \left[p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \right] \tag{6.4}$$

$$= \sum_{\mathbf{a}^{\tau-1} \in \mathbf{A}} \Pi(\mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \tag{6.5}$$

[1] Throughout this chapter we use τ to denote a sampling step in a Markov chain; this is not to be confused with t which refers to a time period during a planning process.

Since we are interested in the action at end of the chain rather than the beginning, we define the chain extending backwards from the final sampled action $\mathbf{a}^\tau$ rather than forward from some start state. The κ-step transition probability, ending in step τ, is defined recursively as:

$$p_\kappa(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^\tau) = \sum_{\mathbf{a}^{\tau-\kappa+1} \in \mathbf{A}} p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau)$$

where $p_0(\mathbf{a}^\tau, \mathbf{a}^\tau) = 1.0$. Using this κ-step transition, the equilibrium probability in (6.5) can be extended back κ steps from $\mathbf{a}^\tau$:

$$\begin{aligned}
\Pi(\mathbf{a}^\tau) &= \sum_{\mathbf{a}^{\tau-1}} \Pi(\mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) && (6.6)\\
&= \sum_{\mathbf{a}^{\tau-2}} \Pi(\mathbf{a}^{\tau-2}) \sum_{\mathbf{a}^{\tau-1}} p_1(\mathbf{a}^{\tau-2}, \mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \\
&= \sum_{\mathbf{a}^{\tau-2}} \Pi(\mathbf{a}^{\tau-2}) p_2(\mathbf{a}^{\tau-2}, \mathbf{a}^\tau) \\
&= \sum_{\mathbf{a}^{\tau-1}} \dots \sum_{\mathbf{a}^{\tau-k}} \Pi(\mathbf{a}^{\tau-k}) p(\mathbf{a}^\tau | a^{\tau-1}) \dots p(\mathbf{a}^{\tau-k+1} | \mathbf{a}^{\tau-k}) \\
&= \sum_{\mathbf{a}^{\tau-k}} \Pi(\mathbf{a}^{\tau-k}) p_\kappa(\mathbf{a}^{\tau-k}, \mathbf{a}^\tau) \\
&= \mathbb{E}_{\mathbf{a}^{\tau-k}} \left[p_\kappa(\mathbf{a}^{\tau-k}, \mathbf{a}^\tau) \right] && (6.7)
\end{aligned}$$

The probabilities in (6.7) are all non-zero and all landscape actions have a non-zero probability of being sampled at each step, thus this Markov chain is *ergodic*. Ergodic Markov chains are guaranteed to converge to a unique equilibrium distribution as the length of the chain goes to infinity [Bremaud, 1999]. This is because as the length of this chain becomes longer, the probability of the starting point $\mathbf{a}^\kappa$ becomes irrelevant. The equilibrium can be defined solely by the limit of transition probabilities as the length of the chain ending in $\mathbf{a}^\tau$ grows to infinity:

$$\Pi(\mathbf{a}^\tau) = \lim_{\kappa \to \infty} p_\kappa(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^\tau) \tag{6.8}$$

Algorithm 6: `sampleStepGibbs` $(\mathbf{s}, \theta)$ - Perform a single step of Gibbs sampling and return the new action. Also optionally return the sampling distribution used for each local policy.

foreach $c \in C$ **do**
  $\delta_c = \pi_c(A|\mathbf{a}_{-c}, \mathbf{s}, \theta)$
  $\mathbf{M}[c] = \mathbf{M}[c] + \delta_c$
  $a_c \sim \delta_c$ `// sample from distribution`
return $\mathbf{a}, \langle \mathbf{M} \rangle$

6.2.1 Estimating the Equilibrium Distribution

The conditional probabilities, $\pi_c(a_c^\tau | \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-)$, of the most recently sampled actions can be used to produce an estimator for the marginal distribution over the actions at a single cell a_c:

$$M_0(a_c) = \frac{1}{\mathcal{M}} \sum_{\tau=0}^{\mathcal{M}} \mathbb{I}(A_c^\tau = a_c)$$

where $\mathcal{M}$ is the total number of samples and $\mathbb{I}(x)$ is an indicator that equals one when the term x is true and equals zero otherwise. This estimator, sometimes called the *Gibbs count*, simply counts the number of occurrences of each action for each cell over the course of the chain [Koller and Friedman, 2009a].

The following estimators weight the Gibbs count by the conditional probability of each sample given some number of previous steps of ancestors of the cell c:

$$M_1(a_c) = \frac{1}{\mathcal{M}} \sum_{\tau=0}^{\mathcal{M}} \pi_c(a_c^\tau | \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-)$$

$$M_2(a_c) = \frac{1}{\mathcal{M}} \sum_{\tau=0}^{\mathcal{M}} \pi_c(a_c^\tau | \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-) \pi_c(a_c^\tau | \mathtt{pa}_{a_c^{\tau-1}}^+ \cup \mathtt{pa}_{a_c^{\tau-1}}^-)$$

These estimators essentially use a simple form of Rao-Blackwellization [Rao, 1965]. The Rao-Blackwell theorem specifies that for any convex loss

function $\mathcal{L}$, such as mean squared error, the following inequality holds:

$$\mathcal{L}(M_2)(A_c) \leq \mathcal{L}(M_1)(A_c) \leq \mathcal{L}(M_0)(A_c) \tag{6.9}$$

This means that the estimator of a conditional probability is never made worse by including information from more ancestors of a node in the Markov chain.

A key requirement of the theorem is that the update depends only on the local parameters θ and not on the true distribution of A_c or the current estimate $M(A_c)$; this is true in the cases above.

6.2.2 Equilibrium Landscape Policy Sampling Algorithm

For use in later algorithms, the sampling and marginal estimation methods are combined into one algorithm, `sampleSpatialPolicy`, shown on the current page. When only the sampled action is required, the returned marginal estimates are simply discarded. The complexity of this algorithm is $O(C\mathcal{M})$ since `sampleStepGibbs` makes an update to each cell and this is performed for $\mathcal{M}$ samples.

Algorithm 7: `sampleSpatialPolicy` $(\mathbf{s}, \theta)$ - Sample a landscape action from the equilibrium distribution using Gibbs sampling. Optionally also return an estimate of the marginal distribution of each cell individually.

$\mathbf{M} = [0]^{C \times A}$
for τ **in** 0 **to** $\mathcal{M}$ **do**
  $\mathbf{a}^\tau, \mathbf{M}' =$`sampleStepGibbs` $(\mathbf{s}, \theta)$
  $\mathbf{M} = \mathbf{M} + \mathbf{M}'$
return $\mathbf{a}^\tau, \mathbf{M}/\mathcal{M}$

6.2.3 Alternative MRF Representation

An alternative way to think of a distribution over landscape actions should be mentioned; instead of using a directed, cyclic model, an undirected Markov Random Field (MRF) [Koller and Friedman, 2009a] could be used (also called a Markov Network). In an MRF, a joint distribution over vari-

ables is defined by positive potential functions over fully connected cliques of nodes. Each family in our directed representation defines a clique of nodes. There is already a potential function, $\psi(a_c, \mathtt{pa}_c, \mathbf{s}, \theta)$, over the nodes in a clique where the parents from the direct model, $\mathtt{pa}_c$, are neighbours in the undirected model. In our formulation, the cell policies are defined as distributions, so ψ is normalized over all the local actions $b \in A$. In an MRF, the distribution is defined by normalizing over all landscape actions, which is intractably large to compute. However, during Gibbs sampling each clique can be normalized locally and everything from outside that clique is cancelled out. Thus, Gibbs sampling on an MRF would be very similar to sampling of our landscape policy. Rather than an arbitrary MRF structure with difficult to interpret potential functions, the directed model is one built on the idea of each cell having its own causal, conditional distribution. Causal distributions make sense for policies where actions at one location depend on actions elsewhere. Each clique of nodes makes sense as having a node at the centre, surrounded by its influencing neighbours. Gibbs sampling is a natural solution for sampling given the causal cycles discussed in the previous chapter.

6.3 Gradient of the Equilibrium Policy

Using an equilibrium landscape policy for planning will require the gradient of the equilibrium landscape policy with respect to the policy parameters. The gradient of the equilibrium landscape policy can be derived from equa-

tion (6.5) as follows:

$$
\begin{aligned}
\nabla_\theta \Pi(\mathbf{a}^\tau) &= \sum_{\mathbf{a}^{\tau-1}} \nabla_\theta [\Pi(\mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau)] \\
&= \sum_{\mathbf{a}^{\tau-1}} \nabla_\theta \Pi(\mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) + \Pi(\mathbf{a}^{\tau-1}) \nabla_\theta p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \\
&= \sum_{\mathbf{a}^{\tau-1}} \sum_{\mathbf{a}^{\tau-2}} \Big[\nabla_\theta \Pi(\mathbf{a}^{\tau-2}) p_1(\mathbf{a}^{\tau-2}, \mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \\
&\qquad + \Pi(\mathbf{a}^{\tau-2}) \nabla_\theta p_1(\mathbf{a}^{\tau-2}, \mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \Big] \\
&\qquad + \Pi(\mathbf{a}^{\tau-1}) \nabla_\theta p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \\
&= \sum_{\mathbf{a}^{\tau-2}} \nabla_\theta \Pi(\mathbf{a}^{\tau-2}) p_2(\mathbf{a}^{\tau-2}, \mathbf{a}^\tau) + \\
&\quad \sum_{\mathbf{a}^{\tau-2}} \Pi(\mathbf{a}^{\tau-2}) \sum_{\mathbf{a}^{\tau-1}} \nabla_\theta p_1(\mathbf{a}^{\tau-2}, \mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) + \\
&\quad \sum_{\mathbf{a}^{\tau-1}} \Pi(\mathbf{a}^{\tau-1}) \nabla_\theta p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau)
\end{aligned}
$$

Since, $p_2(\mathbf{a}^{\tau-2}, \mathbf{a}^\tau) = \sum_{\mathbf{a}^{\tau-1}} p_1(\mathbf{a}^{\tau-2}, \mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau)$.

After substituting $\Pi(\mathbf{a}^{\tau-i})$ with $\Pi(\mathbf{a}^{\tau-i-1})$ for i up to some maximum length ω and grouping terms, the general form of the gradient becomes:

$$
\begin{aligned}
\nabla_\theta \Pi(\mathbf{a}^\tau) = &\sum_{a^{\tau-\omega}} \nabla_\theta \Pi(\mathbf{a}^{\tau-\omega}) p_\omega(\mathbf{a}^{\tau-\omega}, \mathbf{a}^\tau) + \\
&\sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \Pi(\mathbf{a}^{\tau-\kappa}) \sum_{\mathbf{a}^{\tau-\kappa+1}} \nabla_\theta p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) \qquad (6.10)
\end{aligned}
$$

The gradient of a policy describes how a small change in the policy parameters would impact the probability of a given action under the policy, in this case the action $\mathbf{a}^\tau$, sampled after τ steps of a Markov chain. The first component of (6.10) describes how a change in the parameters would affect the probability of each landscape action ω steps back in the chain, weighted by the probability of reaching $\mathbf{a}^\tau$ from that action after ω steps. The second component describes how a parameter change would affect each transition in each chain, for chains of lengths 1 to ω that end up at the landscape

action $\mathbf{a}^\tau$.

Using (6.8), the first component vanishes as the length, ω, of the chain increases to ∞:

$$\begin{aligned}\lim_{\omega\to\infty}\sum_{\mathbf{a}^{\tau-\omega}}\nabla_\theta\Pi(\mathbf{a}^{\tau-\omega})p_\omega(\mathbf{a}^{\tau-\omega},\mathbf{a}^\tau) &= \sum_{\mathbf{a}^{\tau-\omega}}\nabla_\theta\Pi(\mathbf{a}^{\tau-\omega})\Pi(\mathbf{a}^\tau)\\ &= \Pi(\mathbf{a}^\tau)\sum_{\mathbf{a}^{\tau-\omega}}\nabla_\theta\Pi(\mathbf{a}^{\tau-\omega})\\ &= 0\end{aligned}$$

Thus, the first term in (6.10) can be removed, leaving the following approximation for finite ω:

$$\begin{aligned}&\nabla_\theta\Pi(\mathbf{a}^\tau) \qquad (6.11)\\ &\approx\sum_{\kappa=1}^{\omega}\sum_{\mathbf{a}^{\tau-\kappa}}\Pi(\mathbf{a}^{\tau-\kappa})\sum_{\mathbf{a}^{\tau-\kappa+1}}\nabla_\theta p_1(\mathbf{a}^{\tau-\kappa},\mathbf{a}^{\tau-\kappa+1})p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1},\mathbf{a}^\tau)\end{aligned}$$

6.3.1 Gradient of the Cell Policy

The remaining term needed to estimate the gradient in (6.11) is the gradient of the one-step transition probability, $\nabla_\theta p_1(\mathbf{a}^\kappa,\mathbf{a}^{\kappa+1})$, for some pair of consecutive samples in the Markov chain. Since the one-step transition model of this Markov chain is a product of the policy for each cell, the derivation is similar to the one shown in Section 4.4.1 for the independent landscape policy.

The gradient of the one-step transition model can be simplified by taking the logarithm and expressing the parameters $\theta[\alpha,f]$ in terms of its action $\alpha\in A$ and feature $f\in F$ components:

$$\begin{aligned}\nabla_{\alpha f}\log\ p_1(\mathbf{a}^{\tau-1},\mathbf{a}^\tau) &= \nabla_{\alpha f}\log\prod_c\pi_c(a_c^\tau|\mathtt{pa}_{a_c^\tau}^+\cup\mathtt{pa}_{a_c^\tau}^-)\\ &= \sum_c\nabla_{\alpha f}\log\pi_c(a_c^\tau|\mathtt{pa}_{a_c^\tau}^+\cup\mathtt{pa}_{a_c^\tau}^-) \qquad (6.12)\end{aligned}$$

$$
\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) \\
&= \nabla_{\alpha f} \log \left(\frac{\exp(\psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))} \right) \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) - \nabla_{\alpha f} \log \sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta)) \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) - \frac{\nabla_{\alpha f} \sum_{d_c \in A_c} \exp(\psi(d_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))} \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) - \frac{\sum_{d_c \in A_c} \nabla_{\alpha f} \exp(\psi(d_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))} \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) - \\
&\qquad \frac{\sum_{d_c \in A_c} \exp(\psi(d_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta)) \nabla_{\alpha f} \psi(d_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta)}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))}
\end{aligned}
\tag{6.13}
$$

The definition of ψ in (6.1) is a sum over features weighted by the parameters θ, so the derivative with respect to each parameter $\theta[\alpha, f]$ sets all but one term to zero:

$$
\nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) = \begin{cases} f_c(\mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}) & \text{if } \alpha = a_c \\ 0 & \text{otherwise} \end{cases} \tag{6.14}
$$

Using (6.14) and the definition of π_c in (6.2) then (6.13) reduces to:

$$
\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) - \\
&\qquad \frac{\exp(\psi(\alpha_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta)) f_c(\mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s})}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta))} \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) - \pi_c(\alpha_c^\tau | \mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s}, \theta) f_c(\mathtt{pa}_{a_c^\tau}^+ \cup \mathtt{pa}_{a_c^\tau}^-, \mathbf{s})
\end{aligned}
\tag{6.15}
$$

The gradient in (6.15) has two cases due to (6.14).

if $a_c^\tau = \alpha$ then:

$$\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) \\
&\quad = f_c(\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}) - \pi_c(\alpha_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) f_c(\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}) \\
&\quad = (1 - \pi_c(\alpha_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta)) f_c(\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s})
\end{aligned}$$

otherwise $a_c^\tau \neq \alpha$, in which case:

$$\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) \\
&\qquad\qquad = -\pi_c(\alpha_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) f_c(\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s})
\end{aligned}$$

We can combine both cases under one notation:

$$\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) \\
&\qquad = g(a_c^\tau, \alpha, \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) f_c(\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s})
\end{aligned} \tag{6.16}$$

where

$$g(a_c^\tau, \alpha, \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) = \begin{cases} 1 - \pi_c(\alpha_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) \text{ if} & \alpha = \alpha_c^\tau \\ -\pi_c(\alpha_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) \text{ if} & \alpha \neq \alpha_c^\tau \end{cases}$$

The gradient of the log transition model in (6.12) can now be expressed as:

$$\begin{aligned}
&\nabla_{\alpha f} \log p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \\
&\qquad = \sum_c \nabla_{\alpha f} \log \pi_c(a_c^\tau | \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}) \\
&\qquad = \sum_c g(a_c^\tau, \alpha, \mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s}, \theta) f_c(\mathtt{pa}^+_{a_c^\tau} \cup \mathtt{pa}^-_{a_c^\tau}, \mathbf{s})
\end{aligned} \tag{6.17}$$

6.3.2 The Combined Gradient

The gradient of the landscape policy $\nabla_\theta \Pi(\mathbf{a}^\tau)$ from equation (6.11) can now be derived using $\nabla\ f(x) = f(x)\nabla \log\ f(x)$ on the gradient from (6.17).

$$
\begin{aligned}
&\nabla_\theta \Pi(\mathbf{a}^\tau) \\
&\approx \sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \Pi(\mathbf{a}^{\tau-\kappa}) \sum_{\mathbf{a}^{\tau-\kappa+1}} p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) \nabla_\theta p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) \\
&= \sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \Pi(\mathbf{a}^{\tau-\kappa}) \sum_{\mathbf{a}^{\tau-\kappa+1}} p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) \times \\
&\qquad\qquad p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) \nabla_\theta \log p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) \\
&= \sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \Pi(\mathbf{a}^{\tau-\kappa}) \sum_{\mathbf{a}^{\tau-\kappa+1}} p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) \times \\
&\quad \sum_{c \in C} g(a_c^{\tau-\kappa+1}, \alpha, \mathtt{pa}^+_{a_c^{\tau-\kappa+1}} \cup \mathtt{pa}^-_{a_c^{\tau-\kappa+1}}, \mathbf{s}, \theta) f_c(\mathtt{pa}^+_{a_c^{\tau-\kappa+1}} \cup \mathtt{pa}^-_{a_c^{\tau-\kappa+1}}, \mathbf{s})
\end{aligned}
\tag{6.18}
$$

6.4 Estimating the Gradient

Since the distribution $\Pi(\mathbf{a})$ is unknown, (6.18) cannot be computed directly but this gradient can also be expressed as an expectation over paths of all lengths of Markov chain up to ω:

$$
\begin{aligned}
&\nabla_\theta \Pi(\mathbf{a}^\tau) \\
&\approx \sum_{\kappa=1}^{\omega} \mathbb{E}_{\mathbf{a}^{\tau-\kappa} \to \mathbf{a}^\tau} \Bigg[\sum_{c \in C} g(a_c^{\tau-\kappa+1}, \alpha, \mathtt{pa}^+_{a_c^{\tau-\kappa+1}} \cup \mathtt{pa}^-_{a_c^{\tau-\kappa+1}}, \mathbf{s}, \theta) \\
&\qquad\qquad f_c(\mathtt{pa}^+_{a_c^{\tau-\kappa+1}} \cup \mathtt{pa}^-_{a_c^{\tau-\kappa+1}}, \mathbf{s}) \Bigg]
\end{aligned}
\tag{6.19}
$$

where $\mathbb{E}_{\mathbf{a}^{\tau-\kappa} \to \mathbf{a}^\tau}[.]$ indicates an expectation over all paths of length κ ending in $\mathbf{a}^\tau$. An approximation algorithm using stochastic simulation can be used to estimate the expected gradient of (6.19).

When computing the gradient, a fixed landscape $\mathbf{s}$ and a fixed landscape action $\sigma \in \mathbf{A}$ are given. This state and action could correspond to a sample

from a single time step of some previous planning run possibly generated by a different policy. The gradient $\nabla_\theta \Pi(\sigma)$ quantifies how a change in each policy parameter would alter the probability of σ being sampled from the equilibrium of the Markov chain.

The `gradEstimate` algorithm uses samples from a single Markov chain of length ω to estimate the policy gradient for a given action and state. At each sample step $\tau \in [0, \omega)$ a new landscape action $\mathbf{a}^\tau$ is generated and the transition probabilities $p_1(\mathbf{a}^\tau, \mathbf{a}^{\tau+1})$ and $p_1(\mathbf{a}^\tau, \sigma)$ are stored. The algorithm then computes paths of different lengths that all end with the query action σ. This provides ω chains of length two, $\omega - 1$ chains of length three, $\omega - 2$ chains of length four, etc.; finally, a single chain of length ω is computed. This process is visualized in Figure 6.2; the chains ending in $\mathbf{a}^5$ are highlighted.

The sum of the gradients computed on all the different length chains is:

$$\nabla \hat{\Pi}(\sigma) = \sum_{\tau=0}^{\omega-1} p_1(\mathbf{a}^\tau, \sigma) \nabla_\theta \log p_1(\mathbf{a}^\tau, \sigma) + \sum_{\kappa=1}^{\omega-1} \left[\sum_{\tau=0}^{\omega-\kappa-1} p_1(\mathbf{a}^{\tau+\kappa}, \sigma) \nabla_\theta \log p_1(\mathbf{a}^\tau, \mathbf{a}^{\tau+1}) \prod_{i=0}^{\kappa-1} p_1(\mathbf{a}^{\tau+i}, \mathbf{a}^{\tau+i+1}) \right] \tag{6.20}$$

This formula can be used as an estimator of the true gradient $\nabla_\theta \Pi(\sigma)$. In the limit as $\omega \to \infty$, `gradEstimate` would visit all possible states and all possible chains ending in σ would be sampled infinitely often. Eventually the estimate (6.20) would match the expectation (6.19).

To estimate the gradient of the log policy, $\nabla \log \hat{\Pi}(\sigma)$, the above gradient estimate can be combined with an estimate of the equilibrium, $\hat{\Pi}(\sigma)$, generated from the same chain and using the equivalence $\nabla \log \hat{\Pi}(\sigma) \approx \frac{\nabla \hat{\Pi}(\sigma)}{\hat{\Pi}(\sigma)}$.

The gradient estimation algorithm `gradEstimate`, shown on page 105, computes two transitions, $p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau)$ and $p_1(\mathbf{a}^\tau \to \sigma)$, for each action sampled at step τ. These transitions are then used to create τ gradient

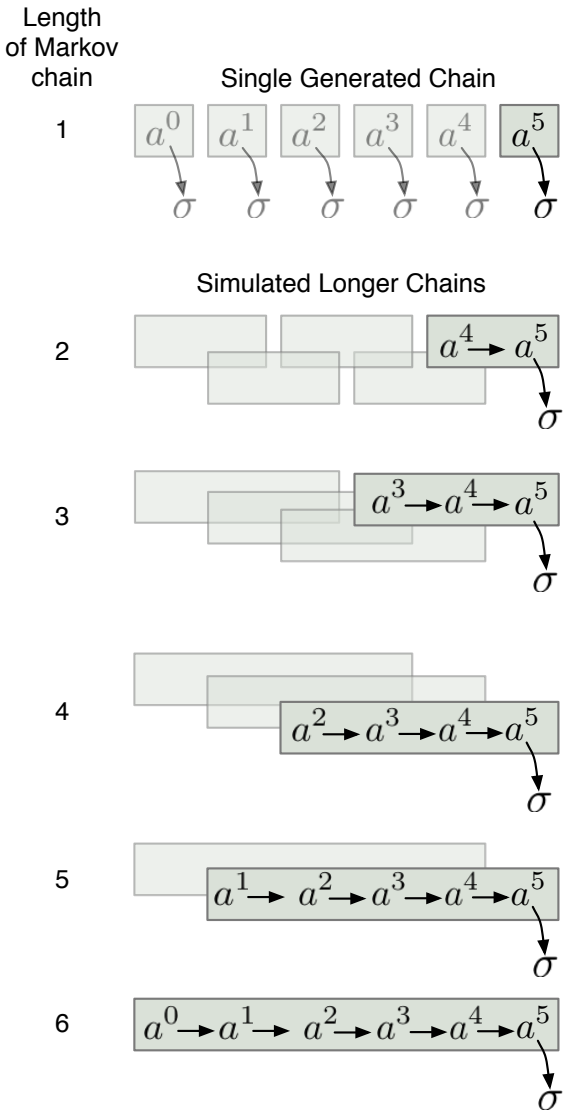


Figure 6.2: Visualization of gradient estimation algorithm on an example using a Markov chain of length 6. A single Markov chain is run and all transition probabilities are stored. Each action is also transitioned to the target state σ, then all lengths of chains are computed from each starting point. The chains ending in $\mathbf{a}^5$ are shown and all the other chains are indicated with empty boxes.

components and add them to the existing estimates for all starting points of the chain. Thus, all the terms in (6.20) do not need to be computed at each iteration; only τ new terms need to be computed at step τ. The complexity of `gradEstimate` for a maximum number of sample steps ω is:

$$\begin{aligned} O(\texttt{gradEstimate}) &= 2\omega C + \sum_{\tau=0}^{\omega} \tau \\ &= 2\omega + \frac{\omega(\omega+1)}{2} \\ &= O(\omega C + \omega^2) \end{aligned} \tag{6.21}$$

where C accounts for the sampling calculations in `sampleStepGibbs` which is computed over all cells for each sample step.

The `gradEstimate` algorithm is an online algorithm; any step of the algorithm has an estimate of the gradient encompassing all of the previous sampled steps. Thus, `gradEstimate` can be stopped at any time without further computation after a fixed time or after some other halting criteria is achieved.

6.4.1 Experimental Evaluation of Gradient Estimation Algorithm

While we cannot compare our gradient estimate to the exact gradient for the full size planning problem, other evaluations were performed to test the validity of the gradient estimation algorithm. Figure 6.3 shows the typical progress of the gradient estimate from four different state-action pair during a planning run with 1880 cells, binary actions and four independent features. Each plot shows the gradient for a different state and action so the value of the true gradient will be different in each situation. Each line shows the contribution to the estimated gradient of a particular policy parameter. The gradient estimation algorithm usually settles near the final gradient values within 500 sample steps for this domain; then minor adjustments and changes in relative position of the gradients can continue for several

Algorithm 8: `gradEstimate` $(\sigma, \mathbf{s})$ - Estimate gradient of policy for a given state and action. Compute estimate inline after each sample is acquired.

$G_\theta = [0]^{\omega \times |\theta|}$
$logpdelta = [0]^{\omega}$
$sumg_\theta = [0]^{\omega \times |\theta|}$
$a^0 = \sigma$
for τ **in** 0 **to** $\omega - 1$ **do**
  `// Transition from` a^τ `to` σ
  $dls_\theta = \nabla_\theta \log p(\sigma | a^\tau)$
  $logpsig = \log p(\sigma | a^\tau)$
  `// Transition from` a^τ `to next sample`
  $a^{\tau+1} =$ `sampleStepGibbs` (a^τ, s, θ)
  $dlp_\theta[\tau] = \nabla_\theta \log p(a^{\tau+1} | a^\tau)$
  $logptrans[\tau] = \log p(a^{\tau+1} | a^\tau)$
  `// Components for new lengths given sample` a^τ
  for κ **in** 0 **to** τ **do**
    if $\kappa = 0$ **then**
      $sumg_\theta[\kappa] += dls_\theta \exp(\frac{1}{C} logpsig)$
    else
      `// Add new` $a^{\tau-1}$ `to` a^τ `step to each chain`
      $logpdelta[\tau - \kappa] += logptrans[\tau - 1]$
      $sumg_\theta[\kappa] += dlp_\theta[\tau - \kappa] \exp\left(\frac{logpdelta[\tau-\kappa] + logpsig}{C(\kappa+1)}\right)$
    $G_\theta[\tau] += \frac{sumg_\theta[\kappa]}{\tau - \kappa + 1}$
return $\frac{1}{C} G_\theta$

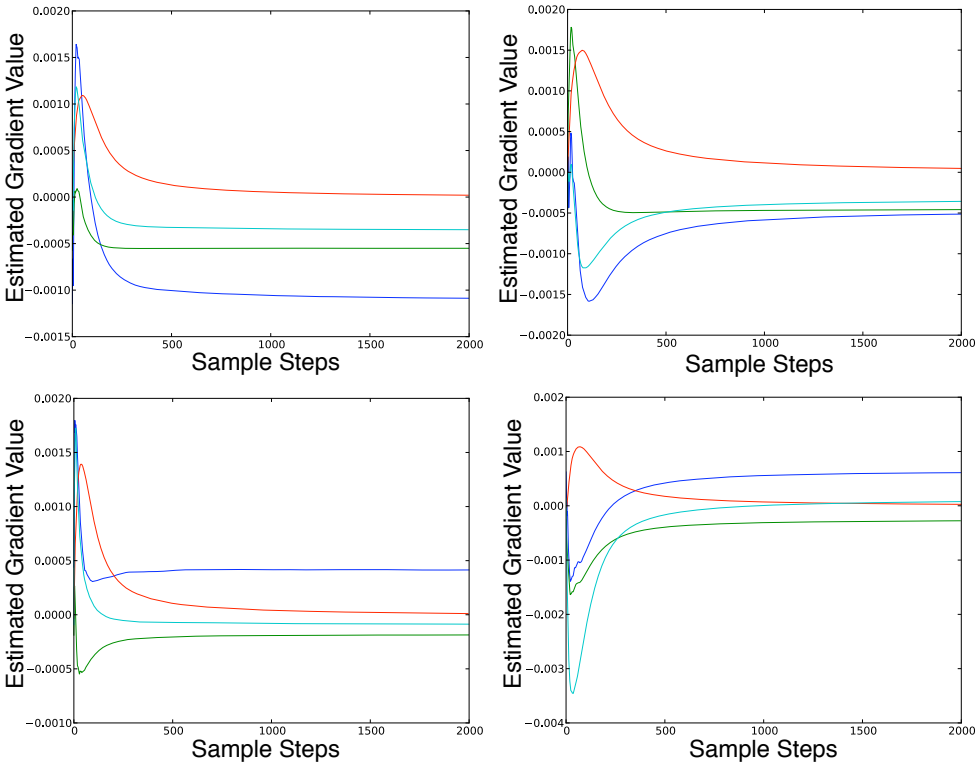


Figure 6.3: Estimates of the gradient for four different states and actions using the same policy parameters. Each line is the gradient estimate for a single feature parameter for the cut action. The same colour is used for each parameter across the multiple runs.

thousand more steps. Knowing this will help to select the number of samples to use when planning. The exact gradient is not needed to perform policy gradient planning, but it is desirable for the estimate of the gradient to be near to its correct value and hopefully to have the relative ordering between derivatives of different parameters correct.

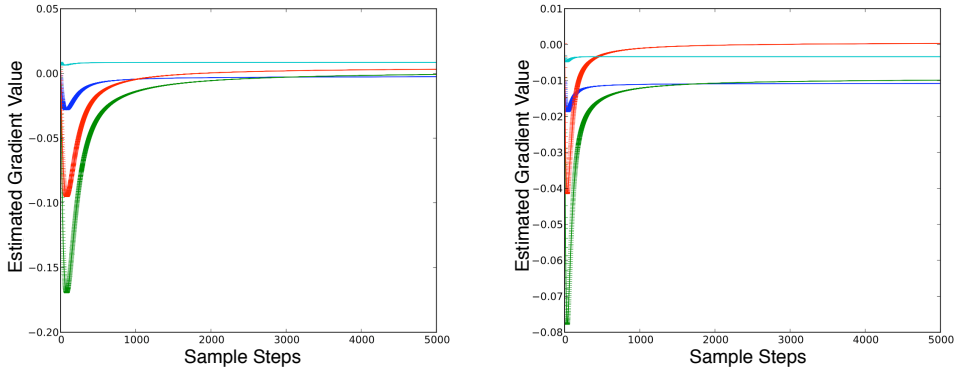


Figure 6.4: Two examples of 10 runs of the gradient estimation algorithm on the same state and action. The width of the lines indicate the variance at each sample step amongst the samples.

Figure 6.4 shows two examples of estimating the gradient for a single policy parametrization on a single state and action over 10 runs. This shows the same pattern of convergence as above and demonstrates that the estimates consistently converge to the same answer over multiple runs.

The gradient of the policy will be different for each set of parameters or for a different landscape action on the same set of parameters. Figure 6.5 shows the convergence of multiple runs with different final gradients. For our problem size and setup, the variance of the gradient estimates drop to their final values within a few hundred steps and stay at those values. This plot extends out to 5000 sample steps without further change.

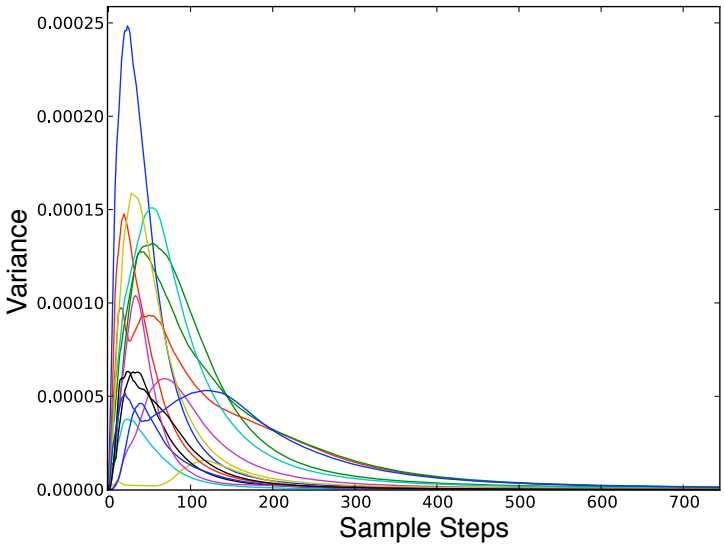


Figure 6.5: Each line indicates the combined variance of all the parameters around their final value showing that most runs reach near their final estimate within a few hundred steps.

Chapter 7

Equilibrium Policy Gradient Planning

This chapter describes how to use the equilibrium landscape policy described in Chapter 6 for planning using a policy gradient planning algorithm called Equilibrium Policy Gradient planning (EPG). EPG is a Natural Actor-Critic algorithm. The algorithm uses an equilibrium landscape policy, reuses its trajectory history while planning and can use a simulation-planner for its dynamics model, which requires more integration than a simple transition model. An analysis of the computational complexity of the algorithm is provided and experimental results are presented from implementing EPG on a forestry planning problem using the FSSAM simulation-planner discussed in Chapter 2.

7.1 Equilibrium Policy Gradient Algorithm

The `EPG` algorithm, shown on the next page, is an instance of the general policy gradient algorithm from Section 3.5.1 that brings together the components for policy sampling, gradient estimation and simulation.

The algorithm receives as input: $\mathbf{s}^0$, an initial landscape state; $\mathbf{H}$, the number of top trajectories to use for computing the gradient; and two functions:

- `sampleSimPlanner` : $\mathbf{S} \times \Theta \rightarrow \mathcal{K}$ - an episodic simulator which takes

an initial landscape state $\mathbf{s} \in \mathbf{S}$ and a set of policy parameters $\theta \in \Theta$. The function returns a trajectory $k \in \mathcal{K}$, generated from the the set of all possible trajectories $\mathcal{K}$, using the provided transition model $\mathcal{T}(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1})$. At each time period, a sample landscape action is acquired from the landscape policy $\Pi(\mathbf{a}|\mathbf{s}, \theta)$ using the `sampleSpatialPolicy` algorithm on page 95.

- $\mathcal{R} : \mathcal{K} \rightarrow \Re$: a reward function returning the expected, discounted reward for a trajectory $k \in \mathcal{K}$. Specific reward models will be described in the experiments section.

Algorithm 9: `EPG` $(s_0, \mathbf{H}, \texttt{sampleSimPlanner}, \mathcal{R})$

K $= \emptyset$; $\theta = [\texttt{random}]^{F \times A}$ $G_\theta = [0]^{\mathbf{H} \times |\theta|}$
repeat until termination condition
  //Stage I - Sample New Trajectory
  $k = \texttt{sampleSimPlanner}(\mathbf{s}_0, \theta)$
  $K = K \cup k$
  //Stage II - Update Policy Along Gradient
  `// Estimate policy gradients over` $\mathbf{H}$ `top trajectories`
  $H = \texttt{topTrajectories}(\text{K}, \mathcal{R}, \mathbf{H})$
  foreach $k \in H$ **do**
    $G_\theta[k] = \sum_t$ `gradEstimate` $(\mathbf{s}^{k,t}, \mathbf{a}^{k,t}, \theta, \omega)$
  `// Compute Natural Gradient`
  $[\delta\theta \quad \mathcal{V}^\pi]$ =`naturalGradient` $(G_\theta, \mathcal{R}(H))$
  `// Update policy`
  $\theta = \theta + \lambda\delta\theta$
return θ

Initially, the set of trajectories K is empty and the policy parameters θ are set to some random values or they can be biased towards a reasonable starting policy if one is known.

The algorithm involves two stages that are iterated until some termination condition is satisfied, such as convergence of the policy, maximum desired runtime or solution quality. In stage I of `EPG`, a new trajectory is

simulated using `sampleSimPlanner` and the resulting trajectory k is stored to the trajectory history K. When `sampleSimPlanner` is not based on a simple transition model $\mathcal{T}$ but rather on a simulation-planner (see Section 2.4.1), which does its own optimization and planning, the details of implementation becomes somewhat more complicated. Appendix A describes the implementation for `sampleSimPlanner` defined by the FSSAM simulation-planner introduced in Section 2.4.2.

Stage II first selects a subset of the current trajectories to use for computing the gradient. `topTrajectories` chooses the **H** trajectories in K with the highest reward; the set H stores this history. The gradient estimation algorithm `gradEstimate`, described on page 105, is then used to compute the gradient of the policy over all trajectories in H with respect to the current policy parameters θ.

Recall that `gradEstimate` simulates a Markov chain for each action $\mathbf{a}^{k,t}$ at time period t and trajectory k. The gradient for each trajectory is essentially estimating to what degree a change to each parameter in θ would influence the likelihood of generating the trajectory k again. The `naturalGradient` algorithm, described on page 45, is then used to compute the natural gradient, $\delta\theta$, of the value function using the raw policy gradients G_θ for each trajectory weighted by the reward received for each of those trajectories. Finally, the policy is updated by $\delta\theta$, weighted by a learning rate λ. The overall effect of the planning algorithm is to alter the policy to make the good trajectories more likely and the bad trajectories less likely; this produces a policy more likely to generate high value trajectories.

7.2 Complexity of the EPG Algorithm

Figure 7.1 shows a visual depiction of the components of the entire algorithm and their complexity using plate notation. Each plate depicts a module or loop which is named at the top of the plate. The bottom left corner shows the complexity of the component or the number of iterations which are executed.

The following variables define the complexity of the algorithm. Some of these variables are part of the given problem while others are user specified

and can be adjusted to trade-off runtime vs. performance.

C - The number of cells in the domain. In practice, this has a large impact on the speed of the algorithm as it can be quite large.
`Range:` In our experiments we use a forestry problem with 1880 cells but it is common for problems to contain hundreds of thousands of cells.
`Source:` Given by problem input.

F - The number of features used to described the state of the world in each cell.
`Range:` In our experiments **F** ranges from 4-8
`Source:` Given by problem input.

A - The number of actions that can be taken at each cell. The number of actions are not generally very large but do impact every level as most calculations need to combine all the actions for a cell to define distributions.
`Range:` In our experiments **A** is binary but **A** could range from 2-10 possible actions at a location.
`Source:` Given by problem input.

T - The number of time periods in the planning horizon. Most parts of the algorithm repeat at every time step within a simulated trajectory.
`Range:` In our forestry experiments, decisions are taken every year with a time horizon in the range 50-100 years. In forestry problems in general **T** could be as high as 300.
`Source:` Given by problem input.

R - The complexity of computing the reward on an entire trajectory. Since the reward model can be non-local it could require going through multiple passes over **C** or **T** to be computed.
`Range:` For our experiments the reward models are linear in $\mathbf{C} \times \mathbf{T}$
`Source:` Determined by provided reward model.

$\mathcal{M}$ - The number of MCMC iterations used to sample an action from the equilibrium landscape policy.

`Range:` 500-1000 in our experiments. Longer runs will hurt performance but provide more stable policy samples and improve the chance that the sampled utility for the trajectory is representative of the policy. The appropriate value could vary greatly depending on the number of cells, number of parameters and interdependency of action at different locations.

`Source:` User specified.

ω - The number of steps in the Markov chain used to estimate the gradient.

`Range:` In section 6.4.1, we showed experimentally that, for the domain being considered here, the gradient stabilized quickly. Values in the range 100-500 are used for experiments.

`Source:` User specified

K - The number of trajectories generated via simulation and stored. This is the size of the model of experiences being built up. Trajectories are expensive to generate so we want to make the best use of them by reusing them for later policy updates.

`Range:` Varies. In our experiments we generate up to 20 trajectories.

`Source:` User specified.

H - The number of trajectories used to estimate the gradient. As the number of trajectories increases, computing the gradient becomes more expensive as it requires samples over all previous trajectories.

`Range:` Less than or equal to **K**. Our experiments use 3-5 of the trajectories with the highest rewards.

`Source:` Given by problem input.

Z - The complexity of computing the transition to the next state given the current state and action.

`Range:` Will likely be proportional to **C** but could be more complex depending on spatial complexity of the model. In the case of the experiments we use the FSSAM simulation planner. **Z** is linear in the

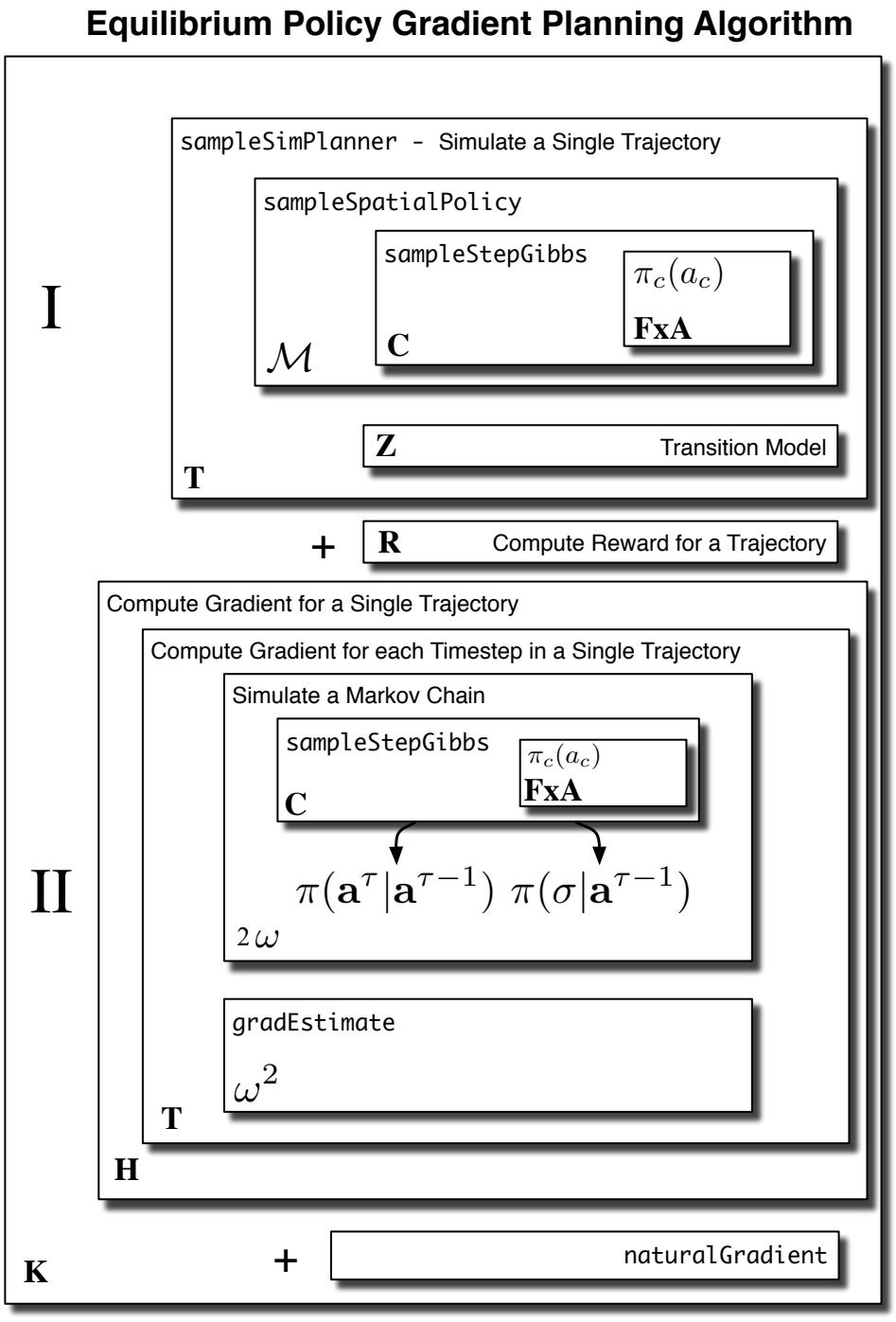


Figure 7.1: Overview of the Spatial Policy Gradient Planning Algorithm

number of cells plus a cost for integrating with the simulation planner
`Source:` Given by problem input.

The complexity of Stage I of the `EPG` algorithm is constant for each trajectory :

$$O(I) = O(\texttt{sampleSimPlanner}) \propto \mathbf{T}(\mathbf{Z} + \mathcal{M}\mathbf{CAF}) + \mathbf{R}$$

Stage II computes a new gradient for each time period in each of the top $\mathbf{H}$ trajectories:

$$O(\texttt{gradEstimate}) \propto \mathbf{T}(\omega\mathbf{C} + \omega^{\mathbf{2}})$$

since the complexity of a single gradient estimate is $O(\omega\mathbf{C}+\omega^2)$, as described on page 104. The complexity of the natural gradient computation was given on page 46, for our policy it is:

$$O(\texttt{naturalGradient}) \propto (\mathbf{F} \times \mathbf{A})^2\mathbf{H} + \frac{(\mathbf{F} \times \mathbf{A})^3}{3}$$

The complexity of each iteration of Stage II is:

$$O(II) = (O(\texttt{gradEstimate}) + O(\texttt{naturalGradient}))$$

For a planning run with $\mathbf{K}$ iterations, using a history of size $\mathbf{H}$ to estimate the gradient, the total cost of running the entire algorithm is:

$$\begin{aligned}
O(\texttt{EPG}) =& \mathbf{KH}(O(II)) + \mathbf{K}(O(I)) \\
=& \mathbf{KH}(O(\texttt{gradEstimate}) + O(\texttt{naturalGradient}))+ \\
& \mathbf{K}(O(\texttt{sampleSimPlanner})) \\
=& \mathbf{KHT}\omega^{\mathbf{2}} + \mathbf{KHT}\omega\mathbf{C} + \mathbf{KH}^2(\mathbf{F} \times \mathbf{A})^2 + \mathbf{KH}\left(\frac{(\mathbf{F} \times \mathbf{A})^3}{3}\right) + \\
& \mathbf{K}(\mathbf{T}(\mathbf{Z} + \mathcal{M}\mathbf{CAF}) + \mathbf{R})
\end{aligned} \tag{7.1}$$

Note that for larger $\mathbf{H}$ or when $\mathbf{K} = \mathbf{H}$, this is a loose upper bound since the number of trajectories grows on each iteration. Only the very final run

of Stage II will iterate over $\mathbf{K}$ trajectories; so $\mathbf{KH}$ would become $\frac{1}{2}(\mathbf{K}^2+\mathbf{K})$ rather than $\mathbf{K}^2$, which is strictly larger.

Running different parts of the algorithm in parallel would reduce this complexity since Stage I can be run independently of Stage II and Stage II can just use the most recent trajectory history generated by Stage I. Many parts of computing the gradient within Stage II can be performed in parallel as well. Section 9.1.7 has further discussion on parallelization of the EPG algorithm.

Note also that this complexity depends on `sampleSimPlanner` which contains an unknown constant $\mathbf{Z}$ which could be high if the external simulator being used is very slow.

7.3 Evaluation

The EPG algorithm is evaluated on a forestry planning problem, considering the goal of finding regular forest and harvest levels that can be sustained over decades without compromising overall forest health. A sustainable harvest level maintains a healthy forest ecosystem while providing a relatively steady harvest volume over time. While harvest targets do not need to be completely uniform year to year, it is undesirable to have large downward or upward trends in either the volume harvested or the overall size of the forest. These targets correspond to the *even flow constraint* discussed in Section 2.3. Using the `EPG` algorithm these targets are defined by three different reward models.

7.3.1 Reward Models

This evaluation looks at applying an even flow constraint on two different volume measures: the *harvested forest volume*, denoted $\texttt{HarvestVolume}_t$, which is the amount cut each year in m^3; and the *available forest volume*, denoted $\texttt{AvailVolume}_t$, which is the total volume of forest that is available for cutting each year. The available forest excludes parts of the forest that are under ecological protection and areas that are still too young to be cut.

Each of the reward models provides a positive reward in proportion to the mean volume cut per year. The mean values computed for available and

harvest levels are:

$$\mu_{AV} = \frac{1}{T}\sum_{t=0}^{T} \texttt{AvailVolume}_t$$
$$\mu_{H} = \frac{1}{T}\sum_{t=0}^{T} \texttt{HarvestVolume}_t$$

The reward models penalize deviation from two possible even-flow targets: one target is the total available volume of the forest and the other target is the amount of timber harvested. Deviation from an even flow target is defined as the sample standard deviation from the mean of the target value:

$$\texttt{AvailableDev} = \sqrt{\frac{1}{T}\sum_{t}(\texttt{AvailVolume}_t - \mu_{AV})^2}$$
$$\texttt{HarvestDev} = \sqrt{\frac{1}{T}\sum_{t}(\texttt{HarvestVolume}_t - \mu_{H})^2}$$

Each reward model also provides positive reward for the total harvested volume and has a spatial constraint penalizing adjacent cutting. For each cell, an adjacent cut occurs whenever the cell c is cut and the `AnyAdj` feature is true for that cell.

$$\texttt{adjCount}(a_c, \mathbf{a}_{-c}, \mathbf{s}) = \begin{cases} 1 & a_c = \texttt{Cut} \wedge \texttt{AnyAdj}_c(\mathbf{a}_{-c}, \mathbf{s}) = \mathit{true} \\ 0 & \text{otherwise} \end{cases}$$

The adjacency penalty counts, over all timesteps and cells, all the occurrences adjacent cuts across all timesteps:

$$\texttt{AdjDev}(a_c, \mathbf{a}_{-c}, \mathbf{s}) = \frac{1}{T}\sum_{t}\sum_{c} \texttt{adjCount}(a_c^t, \mathbf{a}_{-c}^t, \mathbf{s}^t)$$

The following constants weight each of the penalty components:

$$w_{AV} = .3 \qquad w_H = 1 \qquad w_{ADJ} = 50,000$$

The w_{AV} weights attempt to make one standard deviation from the harvest volume or the average available volume have similar magnitude penalties in the reward model. The adjacency weight, w_{ADJ}, was set to make one adjacent cut violation roughly equivalent to one standard deviation from the mean harvested volume.

The three reward functions are defined an an entire trajectory k:

$$\begin{aligned}
\texttt{HVR}(k) &= \mu_H - (\texttt{HarvestDev} * w_H) - \texttt{AdjDev}(\mathbf{a}^k, Ls^k) * w_{ADJ} \\
\texttt{AVR}(k) &= \mu_H - (\texttt{AvailableDev} * w_{AV}) - \texttt{AdjDev}(\mathbf{a}^k, Ls^k) * w_{ADJ} \\
\texttt{HAVR}(k) &= \mu_H - (\texttt{AvailableDev} * w_{AV} + \texttt{HarvestDev} * w_H) \\
&\qquad - \texttt{AdjDev}(\mathbf{a}^k, Ls^k) * w_{ADJ}
\end{aligned}$$

Harvest Volume Reward (HVR) - penalizes irregular harvest volumes over time

Available Volume Reward (AVR) - penalizes irregular available volume of the forest over time

Harvest and Available Volume Reward (HAVR) - penalizes both irregular harvest and available volumes over time

7.3.2 Fixed Harvest Levels

Figure 7.2 shows what happens if an unsustainable fixed harvest level, in this case 200,000m^3, is entered into the FSSAM simulator. The simulator will assign the maximum level of cut until it collapses the forest population. The available harvest line includes the volume of all trees that are old enough to be cut or are not in a reserved area. Setting a lower fixed target can avoid the collapse seen here but may under utilize the forest.

Figure 7.3 shows the result of a range of fixed maximum harvest levels on the available volume of the forest. We can see that for a fixed harvest level, the yearly volume harvested needs to be somewhere between the low but sustainable 25,000m^3 but less than 60,000m^3 which is trending towards collapsing the forest population after 100 years. While in this simple case it

may not be too difficult to manually search for a sustainable fixed harvest target, the complexity of this search increases with each new constraint and dimension added to the model.

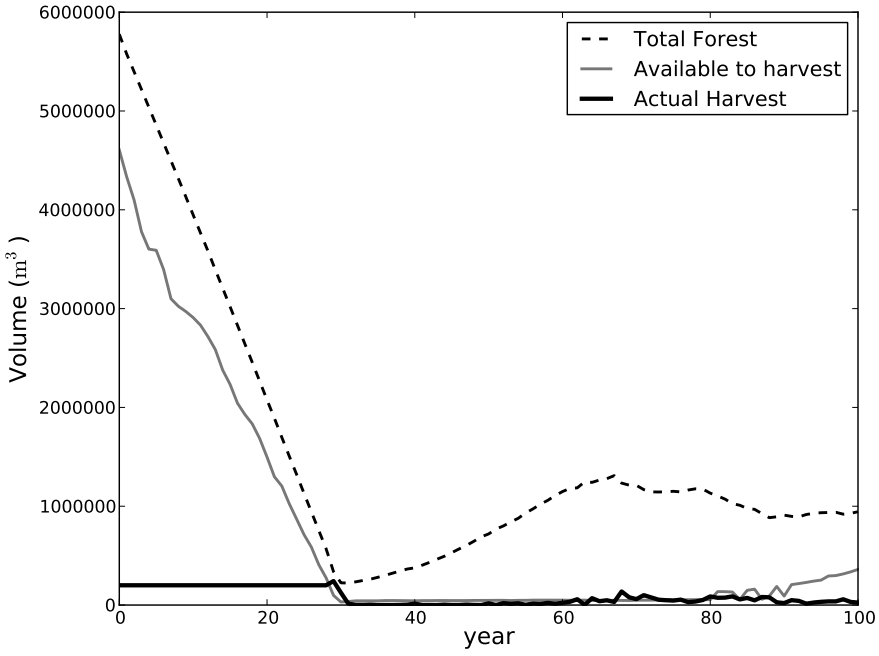


Figure 7.2: Volume of Harvested, Available and Total Forest for a fixed harvest level of 200,000m^3 using the FSSAM simulator-planner. Available cells include any cells which are not too young to be cut and are not in reserve areas.

7.3.3 Experimental Setup

There are 1880 cells in the forest model we are using and the planning horizon will be 100 years, which allows enough time for regrowth of trees cut early on. Actions will be binary, $A = \{\texttt{Cut}, \texttt{NoCut}\}$, where `Cut` removes all trees and replants new trees and `NoCut` puts off any activity for this year.

The following set of features F are defined for each cell:

1:Volume - the total available volume of trees in the current cell

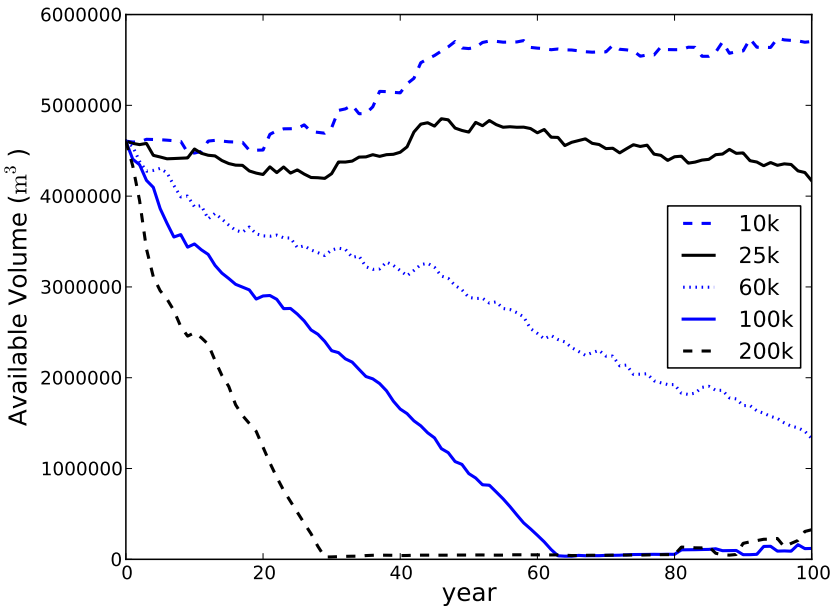


Figure 7.3: The available volume of forest each year under five different fixed maximum harvest levels. Each line shows the fixed volume in m^3 that is harvest each year using the default cutting strategy in FSSAM.

2:Age - the average age of the dominant group of trees in the cell

3:Maximum adjacent volume (MaxAV) - the volume of trees in the largest adjacent cell (by volume) which is available for cutting

4:Adjacent cut flag (AnyAdj) - true if any adjacent cell is being cut under the current landscape action

Features can model local state information (features 1 and 2), state information from spatial neighbours (feature 3) and action information from spatial neighbours (feature 4). Figure 7.3 indicates that a sustainable policy will need to cut relatively little each year. So, the initial policy is set to a uniform weighting of all features biased to a low cut level starting off cutting a small

number of cells out of 1880 in the first sample. This is done by setting all the weights uniformly to $\theta[f, \texttt{Cut}] = 0.0$ and the weights $\theta[f, \texttt{Cut}] = 5.0$ for all features $f \in F$. This is important as policy gradient methods work best when they start with reasonable policies. These settings were determined through trial and error by looking at simulations of different initial policy settings.

All experiments were carried out on a Quad-Core Intel i5 2.66GHz machine with 3GB of RAM. All planning and sampling code was written in Python 2.6 while the forestry simulator and connecting code was written in Java 5.

Each value model was run for ten policy update iterations and the policy that achieved the highest value was used. The algorithm consults the top 5 stored trajectories in addition to the most recently generated trajectory, to in the EPG algorithm $h = 6$.

7.3.4 Analysis

We analyse the behaviour of the algorithm by looking at a set of typical policies resulting from using each of the value models described on page 118. The policies will be referred to as `AVR`, `HVR`, `HAVR` and `HAVR-High`. The first three policies were generated after about 30 hours of runtime using 500 MCMC steps for each action sample and 10 gradient update steps running the gradient estimation algorithm for 70 samples. The policy `HAVR-High` shows a typical result from a longer run where more time is given for estimating the gradient at each step, using 500 MCMC steps; this run took 180 hrs to run.

Figure 7.4 shows the volumes actually harvested under the four policies. The difference between the three reward models is the penalty applied for deviation from an even flow for the harvest level or the available forest level. The `HVR` policy has a more regular harvest flow than the other policies, as expected. This can be seen by looking at Table 7.1 which shows both even flow deviations for the four policies optimized on each value model. Some variation in yearly harvest is unavoidable since the policies are not specifying partial cuts of cells, they are choosing whether to cut entire cells or not. The

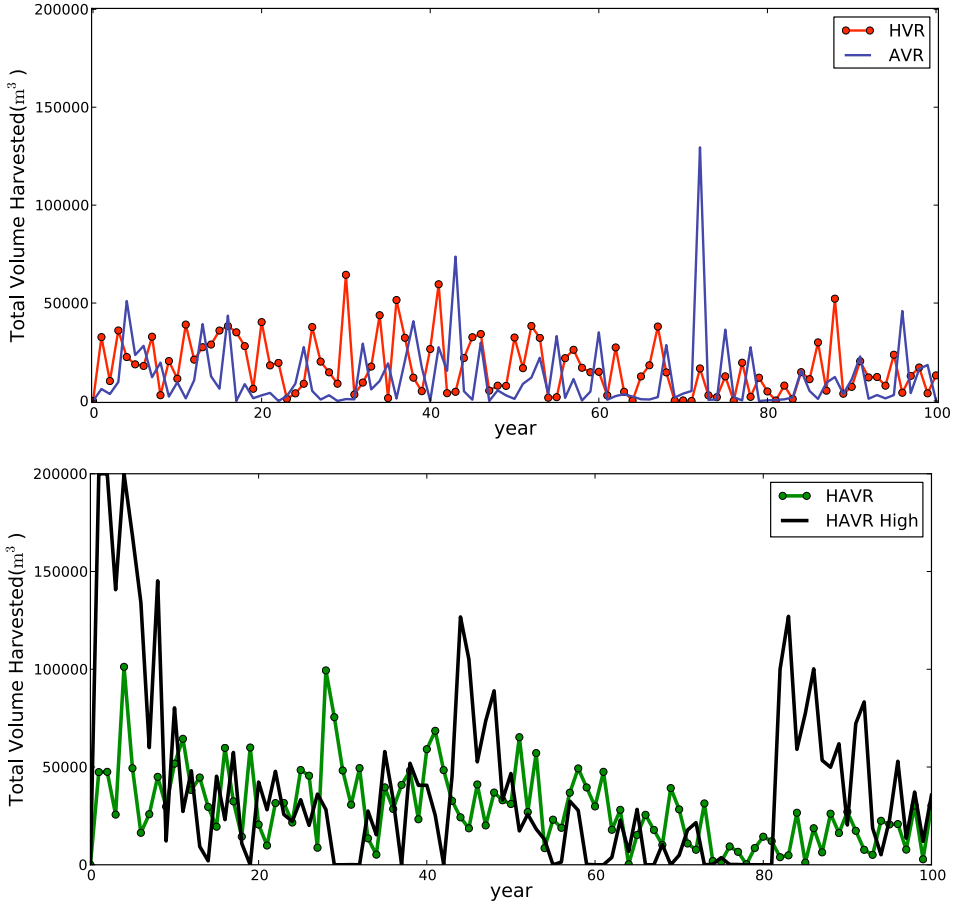


Figure 7.4: Harvest flows of policies computed using the `AVR`, `HVR`, `HAVR` and `HAVR-High` value models.

number of cells being cut ranges from 5-30 cells at each time step, so there are only so many discrete harvest levels that are available. Both the `AVR` and `HVR` value models tend to find very low cutting policies while searching for an even flow. The two `HAVR` policies, shown in Figure 7.4(b), are optimizing both harvest flow and available forest flow, so it is not surprising they have less regular harvest levels than `HVR` alone.

Figure 7.5 shows the available volume in each year in a 100 year period

Policy Under Value Model	Standard Deviation from Mean Harvest Level	Standard Deviation from Mean Available Forest Level
`AVR`	18,065	411,085
`HVR`	14,422	248,920
`HAVR`	20,309	224,212
`HAVR-High`	46,699	212,070

Table 7.1: The standard deviation from the mean for the harvest and available forest flows for each policy from Figure 7.4.

Value Model	Total Harvest Volume (m^3)
`AVR`	1,208,827
`HVR`	1,751,652
`HAVR`	2,923,499
`HAVR-High`	3,973,170
FSSAM-10k	1,000,000
FSSAM-25k	2,500,000
FSSAM-60k	6,000,000
FSSAM-100k	7,847,107
FSSAM-200k	8,311,268

Table 7.2: Comparison of the Total Harvested volumes summed over 100 years for each policy.

for the same four policies. We can see that all four value models are able to produce sustainable harvest policies that do not lead to a collapse of the forest population. `HVR` has a more regular available volume profile than `AVR` even though `AVR` penalizes deviation from irregular available flow (see Table 7.1). In fact, the `AVR` policy does worse than all the other policies at achieving a regular available forest flow. We find in general that `AVR` performs very erratically. This may be because it has no incentive to produce a regular harvest level and needs to search through a larger space of policies,

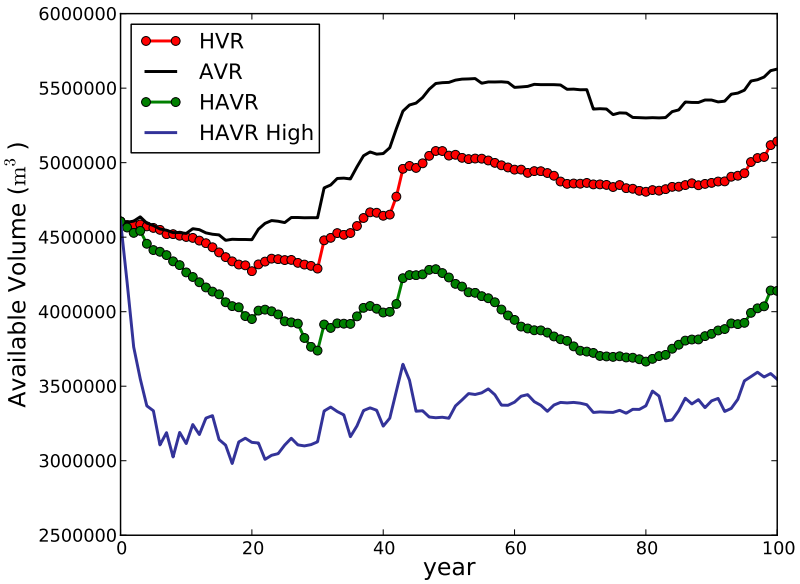


Figure 7.5: Available forest volume under policies computed using the `AVR`, `HVR` and `HAVR` value models.

leading to the more irregular behaviour.

The `HAVR` and `HAVR-High` policies achieve similar reward values under the `HAVR` value model. `HAVR-High` has a slightly higher reward than `HAVR` despite having a much more irregular harvest level. The penalty for deviation from the even flow harvest constraint is compensated for by having a more regular available forest level and by cutting significantly more trees over the 100 years than the `HAVR` policy. The total volumes harvested over the planning horizon under each policy are shown in Table 7.2.

The pattern of cuts used by `HAVR-High` matches a standard result from LP optimization of forest harvest levels [Davis et al., 2001]. When the goal is to find the maximum annual allowable cut level which can be maintained indefinitely, the optimal plan is often one where an older forest is first harvested down to a lower level and a sustainable flow at a lower harvest level is then maintained into the future. The value gained from an increase in

the total volume harvested can compensate for the penalties incurred for deviating from an even harvest flow.

These results emphasize the importance of having a good value model in order to get meaningful results from an automated planning system. The policies shown here are trying to balance three signals from their value models: the value gained from harvesting trees, a penalty on irregular cutting and a penalty in irregular available forest level. How these three components are weighted will influence which kinds of policies achieve higher rewards. For example, if regular harvest flow was weighted more strongly then `HAVR` may well dominate policies like `HAVR-High` while improving the policy. Thus, the implications of each component of a value model should be considered carefully. The EPG algorithm could be a useful tool for planners to use to improve their value model by seeing the ramifications of carrying through with policies that optimize to those values.

7.4 Interpretation and Usage of Policy

One of the goals for spatiotemporal planning tools discussed in the introduction is to enable policy makers to focus on modelling their problem and values rather than on algorithm design and to be able to interpret a policy to understand why it is suggesting particular actions. Tools which automatically explore the ramifications of a value model and free the policy maker from spending their time manually running simulations can help with this goal. The EPG algorithm only requires the user to specify a reward model describing how they value different outcomes and what features are important for making decisions locally. A particular policy can be analysed by looking at the parameters directly, by looking at the trajectories that were generated and by interacting with the predictions of the policy.

Chapters 5 and 6 explained how to interpret local conditional policies as the equilibria of cyclic causal models. The parameters of these spatial policies can be interpreted directly given this causal interpretation. Table 7.3 shows the final policy parameters for the `AVR`, `HVR`, `HAVR` and `HAVR-High` policies. Recall that each parameter is a weight in the cell-policy from Equation (6.2). Each weight describes the correlation between the value

$\theta_f(a)$ Parameters - Initial Values

Action	Age	Max AV	AnyAdj	Volume
`Cut`	1.0	1.0	1.0	1.0
`NoCut`	5.0	5.0	5.0	5.0

$\theta_f(a)$ Parameters - `HVR`

Action	Age	Max AV	AnyAdj	Volume
`Cut`	-3.17	-0.41	-2.08	0.45
`NoCut`	8.22	5.42	5.51	4.53

$\theta_f(a)$ Parameters - `AVR`

Action	Age	Max AV	AnyAdj	Volume
`Cut`	-4.17	0.04	3.68	1.65
`NoCut`	9.68	4.98	0.75	3.27

$\theta_f(a)$ Parameters - `HAVR`

Action	Age	Max AV	AnyAdj	Volume
`Cut`	-1.55	-1.98	0.71	0.29
`NoCut`	7.82	6.97	3.85	4.79

$\theta_f(a)$ Parameters - `HAVR-High`

Action	Age	Max AV	AnyAdj	Volume
`Cut`	-11.88	-3.78	0.63	5.49
`NoCut`	15.27	9.05	4.16	-0.40

Table 7.3: Initial and four final policy parameter settings.

of the feature at a cell and the probability of taking the associated action at that cell. The features for a cell can include spatial features which use information from other cells in the landscape. Thus, the parameters describe a local cell policy given that the actions for all other relevant locations have already been decided. Negative parameters indicate that the feature

is negatively correlated with the action. All else being equal, as the value of a feature f goes up, the probability of an action a goes up in proportion to $\theta[f, a]$ when $\theta[f, a] > 0$ and the probability of an action a goes down in proportion to $\theta[f, a]$ when $\theta[f, a] < 0$. For example, the `AVR` and `HAVR` policies strongly favor cutting less for cells with larger volume. This would favor cutting smaller cells and cutting less overall. The more aggressive policy optimized for `HAVR-High`, on the other hand, has a high correlation between cutting and volume so it targets large cells. The age parameters are all low which mean the policy tends to cut less as the age of the trees increases. This could be influenced by the particular feature values of the landscape we are using or the tradeoffs needed over time to maintain the forest. The `HAVR-High` policy most dramatically targets younger which could lead to a more uniform forest age makeup which helps to produce the very uniform forest population. Further work needs to be done on how to properly analyze parameters given the underlying data.

A spatial policy can also be visualized in several ways. Figures 7.6, 7.7 and 7.8 show an example of a way to visualize a policy. The maps show the cells cut under three different policies. Cells cut within each ten-year period have the same colour. A policy maker could look at these maps to get an idea of what each policy does over time. Since the policies are stochastic, multiple maps could also be generated from the same policy and the variation plotted on a map; regions that are cut under multiple runs could be starting points to consider cutting in the real world.

Another useful visualization could be to show the marginal probability of cutting at each cell under the policy. Interventions could be added to force cuts, or non-cuts, at particular locations. The marginal probabilities could then be recomputed, showing the updated policy for cutting given the existing cuts. The resulting probability map would show where the policy advises to cut next, given the initial cutting decision. By generating a full decision making policy optimized rather than a specific optimal prescription a policy can be given a partial instantiation of the plan and queried about the remaining actions. While we have made some progress on producing interpretable, spatial policies, there is still much work to do in the area.

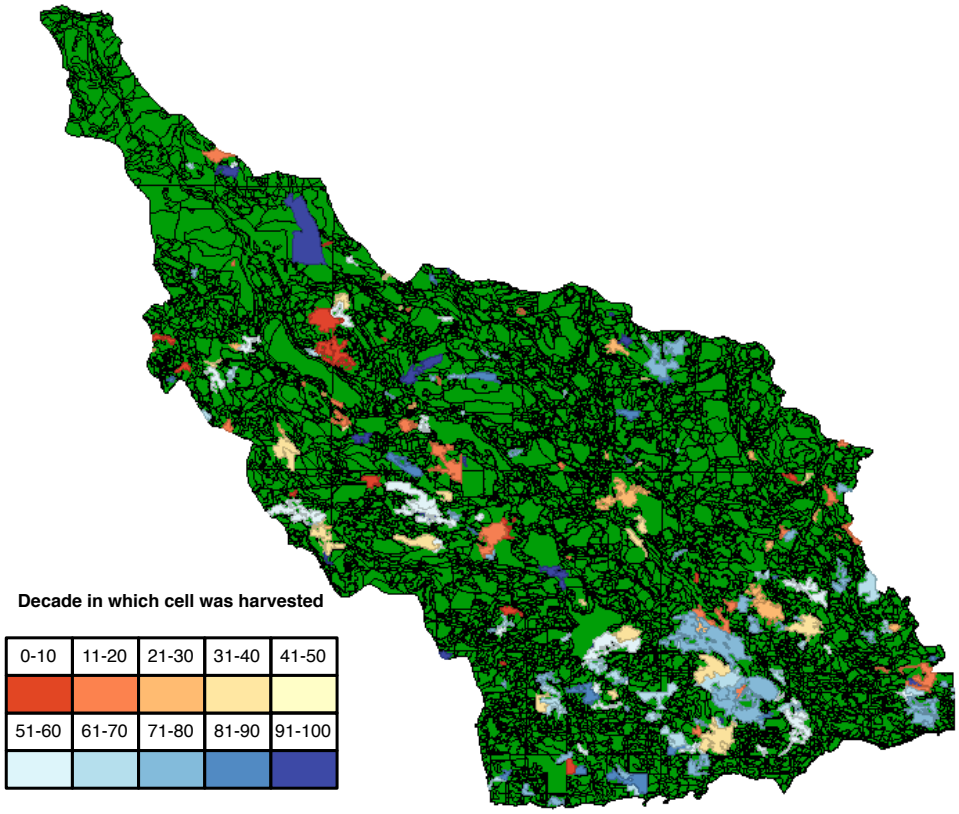


Figure 7.6: Map of Harvest using `AVR` policy. Each colour indicates cells cut at some point in each 10 year period for a run of the simulator.

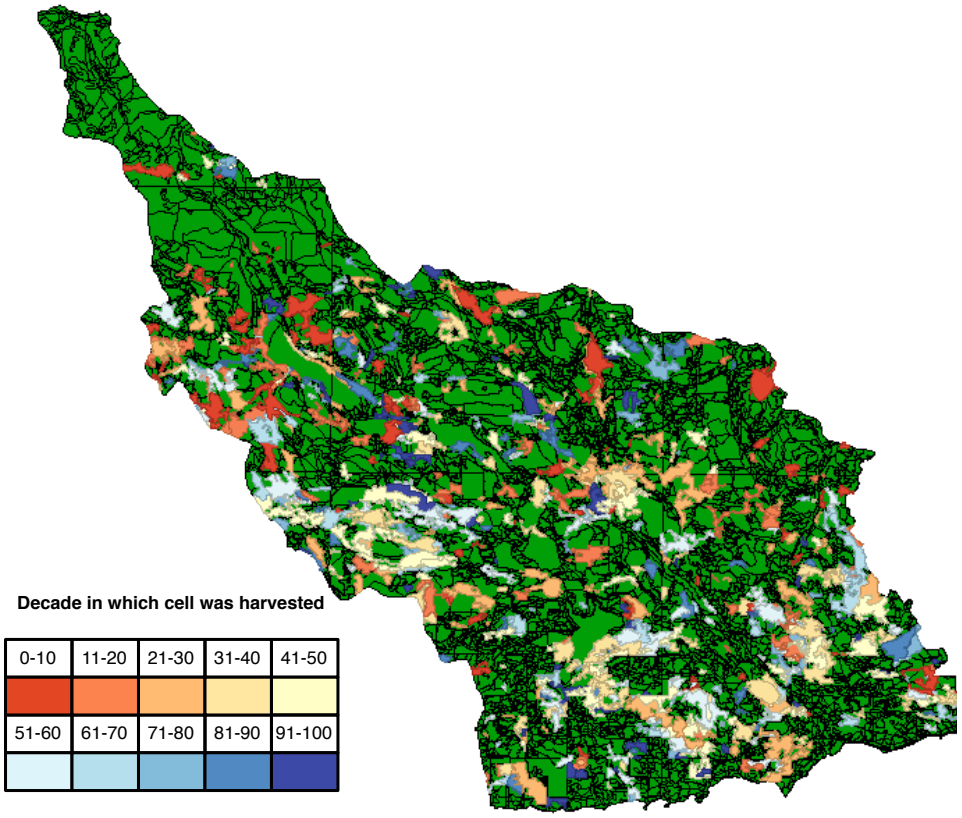


Figure 7.7: Map of Harvest using `HAVR` policy. Each colour indicates cells cut at some point in each 10 year period for a run of the simulator.

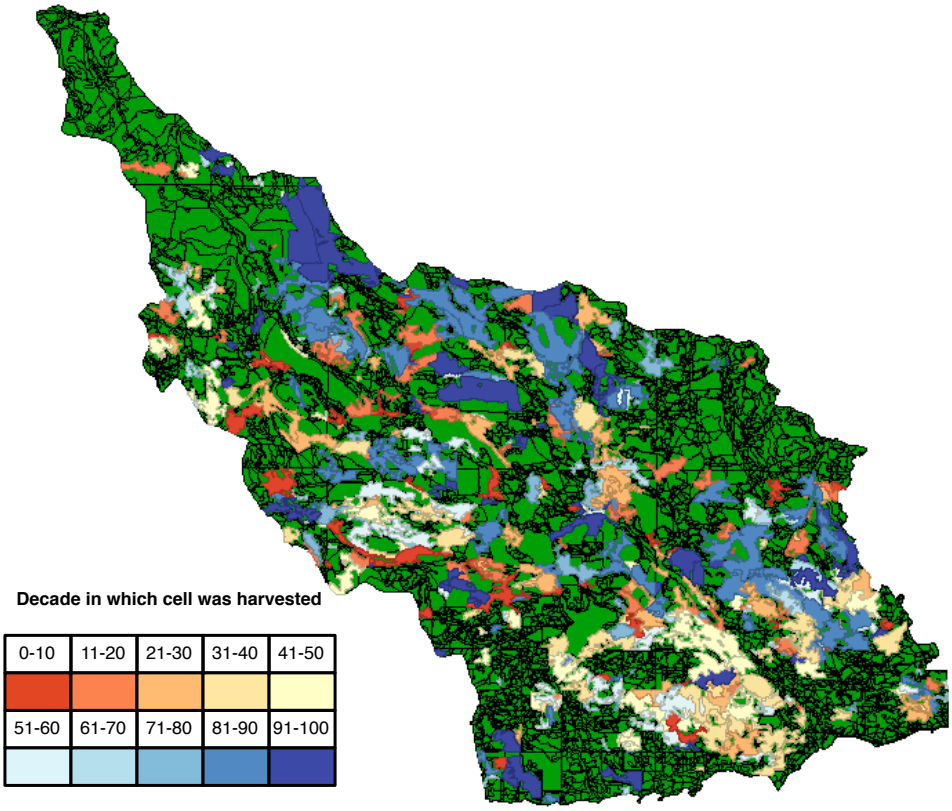


Figure 7.8: Map of Harvest using `HAVR-High` policy. Each colour indicates cells cut at some point in each 10 year period for a run of the simulator.

Chapter 8

Related Work

Spatiotemporal planning for environmental domains is an inherently interdisciplinary problem and relates to many fields of research beyond Computer Science such as Operations Research (OR), Economics, Statistics, Ecology and many others. This chapter discusses some other specific work which relates to the theoretical basis for techniques in this thesis or which may provide useful directions for future work.

There is a vast literature from OR on planning for resource management problems with large number of variables. Powell [2010] provides a good overview of the relationship between planning in the fields of Operations Research and Artificial Intelligence. OR tends to focuss on optimization via mathematical programming techniques rather than the logical inference, simulation and learning approaches more common in AI. Some of the traditional methods of OR include the meta-heuristic discussed in Chapter 2; these methods are especially popular in forestry planning and other environmental planning fields.

8.1 Markov Chain Analysis

When Sutton et al. [2000] introduced policy gradient planning to the Reinforcement Learning (RL) community he noted an alternate justification for direct policy search coming from the Markov chain analysis literature. Cao and Chen [1997] describe the field of Perturbation Analysis of Markov chains

as looking at how *sensitive* the stationary distribution of a Markov chain is to changes to the parameters of the transition distribution. Baxter and Bartlett [2001] describe how an Markov Decision Process (MDP) planning problem can be converted into a Markov chain filtering problem by:

1. Defining a transition model for the Markov chain, rolling the action distribution used by the policy into the MDP transition function.

2. Defining a performance measure using the value function for the current state.

In this approach, the dynamics of an MDP are viewed as a Markov chain where the transition model $\mathcal{T}^{\theta}(\mathbf{s}^t|\mathbf{s}^{t-1})$ contains the actions that would follow given the current policy. The value of the current state, $V(\mathbf{s})$, is used as a performance measure to maximize a Markov chain by modifying the parameters θ of the transition model. This uses the same mathematical formulation as RL which focusses on choosing actions except the transition model is $\Delta(\mathbf{s}^t|\pi^{\theta}(\mathbf{s}^{t-1}), \mathbf{s}^{t-1})$ where the actions are chosen to maximize $Q(\mathbf{s}, \mathbf{a})$.

Cao [2005] provides a good overview which ties together perturbation analysis methods with MDPs and RL. Most of these methods assume a very small state space (in some cases just seven states) and an explicit distribution over all states. Thus, although some insight is gained into the relation between MDPs and Markov chains, the methods used do not seem directly applicable to the large-scale spatiotemporal planning problems addressed in this thesis.

8.2 Factored Planning Approaches

A Factored MDP [Boutilier et al., 1999] is an MDP where the state or actions are factorized into a number of interacting variables. A Dynamic Bayesian Network (DBN) [Dean and Kanazawa, 1989] is an example of a factored state space that can be used for modelling an MDP. A DBN represents conditional distributions in a compact and natural way. Only local, direct relationships are specified in the model; belief propagation takes care of applying the effects of variables not closely related to each other. The

conditional structure of a DBN models the state and transition function in a compact way. The distribution represented by the network can be used to represent the probability of being in a state of the MDP.

Guestrin et al. [2001] define a *multi-agent factored MDP* as an MDP which has multiple action choices at each point in time. This is similar to the model we are using since each location in space with an action could be seen as an agent. The model is multi-agent and co-operative in that all the 'agents' are acting to maximize the same global value function. Their approach uses local value functions $Q_j(a_1, a_2)$ that model the contribution of joint actions of two agents, a_1 and a_2, to the overall utility model. A graphical formulation called a co-ordination graph is used to represent the interaction between agents using these Q_j functions to define undirected relationships. The most obvious difference from our model is that we use directed, conditional relations between agents with cycles rather than undirected potentials. This has a different semantics than an undirected approach although the same set of functions can be modelled. We also are not modelling the value function but rather the distribution over actions.

Guestrin et al. [2002] demonstrate a model-based reinforcement learning approach for solving factored MDPs. Their technique performs very well against other algorithms but requires exact inference on the value function so is limited to small problem sizes.

Melo and Veloso [2011] present an approach for generalizing actions in MDPs to handle larger, multidimensional action spaces. Their approach partitions the action space into sets of which the larger actions are taken after which they use traditional LP methods for solving the MDPs. This requires a domain where the agents interact only rarely with each other; this is not generally the case in spatiotemporal planning. They provide a nice overview/review of different RL communities dealing with large action spaces.

Pazis and Parr [2011] describe a method for compactly representing the value function (Q or V) in RL problems that very efficiently represents states and using this value function during learning. Their method requires a small MDP that can be solved exactly, so this is not feasible for large scale

spatiotemporal problems.

Forsell et al. [2009b] describe a forestry planning problem of minimizing wind damage to trees. They apply their Graph-based Markov decision process (GMDP) model, another form of factored MDP, to this domain by using extensive domain knowledge to simplify the problem and solve portions of it with linear programming. These exact solutions are then used as solutions to subproblems in policy iteration. [Forsell et al., 2009a] compare a number of linear programming and reinforcement learning approaches on forestry planning problems using GMDPs. They also point out the need for scalable, model-free planning methods that can take advantage of existing simulators for natural resource planning problems. Our approach attempts to address this need without requiring extensive domain knowledge to engineer an efficient problem representation.

8.2.1 Complexity of Factored Models

Bernstein et al. [2002] provides a complexity analysis of two models of decentralized control that apply to general factored MDP planning problems, which includes our Equilibrium Policy Gradient planning (EPG) approach. A Decentralized POMDP (DEC-POMDP) has multiple agents acting under possibly different information about the state; a Decentralized MDP (DEC-MDP) has the further restriction that the joint information observed by all the agents uniquely determines the state. Thus, every agent may not know about all of the state but no information that is needed to fully specify the state is hidden from *all* agents. They show that solving DEC-MDPs and DEC-POMDPs is NEXP-hard which is known to be greater than P; thus solving these problems exactly is provably intractable. Using the result by Madani et al. [1999] that the infinite horizon Partially Observable Markov Decision Process (POMDP) is undecidable this implies that infinite horizon DEC-MDPs and DEC-POMDPs are undecidable.

8.2.2 Cellular Automata

Mathey and Nelson [2007] present a factored approach to forestry planning which uses a local optimization algorithm for finding a single approximately

optimal plan in the presence of spatial constraints such as biodiversity constraints. The cyclic causal policy used in EPG is superficially similar to cellular automata in that local actions for each stand are conditioned on the actions and states of other stands and stands are optimized iteratively over multiple passes. However, there are significant differences between cellular automata and our approach. In the cellular automata approach:

- The actions for each stand are entire management schedules over the planning horizon rather than an action at a time step. This simplifies action choices to *when* a cut should occur rather than *how often* it should occur.
- Each stand is optimized using a linear method based on current local conditions. Our method does not attempt to choose or define an optimal action at each point, rather using a distribution over desired actions based on experiences so far.
- Local rewards are distributed evenly to all cells by dividing up a single global reward, in our model they can be modelled as completely separate from global rewards.
- The output of their algorithm, as with most other meta-heuristics used in forestry planning, is a single 'optimal' prescription to follow rather than a distribution over actions which could be interactively queried and interpreted.

8.3 DBNs and Belief Propagation

One way to view the Iterative Improvement Algorithm on EBNs from Chapter 5 is in terms of its relation to work on monitoring of dynamic stochastic processes such as the *BK algorithm* of Boyen and Koller [1998]. The BK algorithm reduces the entanglement of variables in a DBN by breaking the conditional link to the past at each step and approximating the current state.

In the BK algorithm the current state is projected to an *approximate belief state* using a factored representation. This factored, approximate belief

state removes the links between clusters of interrelated nodes *within a given time slice*. This is a reasonable approximation as long as interaction between variables in different clusters is sufficiently weak [Pfeffer, 2006]. The approximate belief state is structured so that the variables that make up the state space are divided up into small, weakly interacting components conditioned on some other aggregate variable. Under this approximation it can be shown that the approximation error does not increase at each step.

IIA does something similar to the BK algorithm, essentially projecting the belief state onto a carefully constructed belief network representation of the belief state, an EBN. This belief state uses hidden variables (the previous variables are latent variables for the representation of the equilibrium). The hidden model in our EBN can bee seen as a kind of projection at each step of a DBN where each time slice contains the entire, two-stage EBN. The convergence proof in Boyen and Koller [1999] is similar to the proof of convergence in Section 5.8.1 except that our proof deals only with interventions and not observations.

The BK algorithm is a special case of Iterative Belief Propagation (IBP) [Darwiche, 2009] which is the exact Belief Propagation (BP) algorithm [Pearl, 1988] applied to a graphical model with cycles. Research into IBP increased significantly after it was discovered by Richardson [2000] that it was equivalent to the very successful turbo-coding algorithm. This has led to several generalizations of IBP such as Generalized Belief Propagation (GBP) [Yedidia et al., 2001] and Expectation Propagation (EP) [Minka, 2001]. Our algorithm can also be directly described as an instance of EP.

In general, IBP algorithms and their related generalizations all have fixed points which are distributions in the exponential family. However, the algorithms are not guaranteed to converge to these fixed points except in special cases. In practice, IBP algorithms perform very well, especially when the approximate belief state used has a rich structure that matches the structure of the true distribution in some way.

Fixed points of IBP algorithms have also been identified by Yedidia et al. [2001] as identical to the stationary points of the Bethe free energy formulation from statistical physics. Thus, a fixed point can be found by

optimizing this energy function directly. However, the quality of such an answer is often a bad approximation of the exact distribution if IBP doesn't converge on its own.

8.3.1 Relation to Our Approach

Existing DBN inference research with which we are aware assumes that the goal is always to produce a full joint distribution and that projections which are used to reduce interrelation of variables can only involve existing variables. We have demonstrated that if a new set of variables with a different structure is allowed, then exact monitoring of marginal distributions is possible. We also found that in some cases the full joint can be computed with a compact representation of these latent variables. The Iterative Improvement Algorithm and EBN model presented here have the ability to compute exact distributions over a range of query sets from the marginals on individual variables.

8.4 Qualitative Spatial Reasoning

Qualitative spatial reasoning is the study of how to express spatial relations between different features or objects without using numerical values for those relations. The focus is on relationships such as 'in front', 'near', 'between' and 'before' rather than locations on a grid. A rich vocabulary for qualitative descriptions has been devised by Davis [1990] for many different types of spatial relations. This field of research, also referred to as *common-sense reasoning*, relates both to how to describe information in a spatial context and how to reason about it.

Work on spatial logics [Bailey-Kellog and Zhao, 2004; Gabelaia et al., 2005] provides insight into how to constrain the combinatorial explosion of spatial descriptions. There is also a community of research on *qualitative temporal reasoning* and the notion of temporal logics [Allen, 1983] that model the relation between events in time. Freksa [1992] use semi-intervals which denote the beginning and ending time of events rather than a list of instants. Qualitative reasoning is concerned with the semantics of relations between events; whether they overlap, precede or follow each other without

necessarily tying them to specific numerical times.

Qualitative models can lead to very compact representations of knowledge if the domain is taken into account or types of possible queries are known. For example, answering the question "Did Newton live before Einstein" does not require knowledge about their actual dates of birth or death if the intervals of their lives are represented relative to each other qualitatively. In essence, reasoning is being performed on qualitatively distinct equivalence classes of states. If the relative orientation of intervals or objects is what is important then these descriptions can greatly reduce the space of possibilities.

In the forestry planning problem considered here we usually have access to data expressed with numerical positions and attributes. We did not attempt to utilize qualitative spatial methods in this thesis. However, when the specific problem domain is known, the insights from this research could be helpful in guiding the design of features and devising compact representations of the state.

8.5 Action Graph Games

Jiang et al. [2011] describe a multi-agent game theoretic architecture call Action-Graph Games (AGGs) which generalize a number of different methods for expressing compact, structured utility models and computing equilibrium strategies. AGGs are well suited to representing spatial planning as each decision location can be seen as one of many co-operative agents acting across the landscape. AGGs contain a compact representation of the expected utility model built only from the local utilities of individual agents and their relation to the number of neighbouring agents taking some actions. There are some similarities here to the modular planning approach we are using. However, we leave the utility model to be provided externally and do not assume any structure in it. Thus, having a compact representation for estimating the expected utility within a single time step is not directly useful in our current approach. This could be an interesting avenue to explore in the future work.

8.6 Computational Sustainability

Over the past 20 years, the diversity of research and techniques available from the fields of AI, probabilistic modelling, optimization and machine learning has exploded. For practitioners in application domains, the people who actually need to solve a particular spatiotemporal planning problem in the world, the abundance of choices can be overwhelming and the learning curve to apply many advanced methods to a new problem can be steep. One effect of this is that when some method is described well for use in a particular domain it gets use over and over by practitioners in that domain. This is the case, for example, in forestry planning where a demonstration of Simulated Annealing by Lockwood and Moore [1993] has greatly influenced the field. This approach has some significant benefits over other methods for dealing with spatial correlations but there are even more flexible approaches available that are not widely used because no similar mapping has been made.

The growing interdisciplinary field of Computational Sustainability [Gomes, 2009] takes on the task of making these kinds of mappings from problems in environmental domains to the best methods for modelling, prediction and planning from the fields of AI and optimization. This improves the immediate quality of the solutions in the specific field but should also have the longer term benefit of better explaining these methods to non-experts. This reduces the learning curve for practitioners and gives them more choices for solving their problems. It also has the benefit of increasing the real world grounding for research problems in computer science through increased communication with application domains.

A wide variety of computational problems could fall within the field of computational sustainability, some notable examples are:

- wildlife migration corridor design
- species migration tracking
- forest fire prevention management
- sustainable forest planning

- invasive species control
- urban water management
- smart power grid design

Real world domains often present challenges that are not considered when solving a purely theoretical problem. This has been true for the research presented in this thesis. Real problems have messy, even conflicting data and values with goals that are rarely absolute. When multiple levels of planning are involved, one has to expect that their results are just one input into a more complex decision making process which will not be modelled. Thus, the optimal solution may not be accessible. The goal is to improve performance in a stochastic world with an imperfect model where no one knows for sure what the 'correct' answer is. This kind of research also provides an exciting opportunity to influence decision making about important and challenging questions facing society.

Chapter 9

Conclusion

Our original research question asked how we can perform planning in the kinds of spatiotemporal planning problems that arise in environmental domains, such as forestry. Put another way, how can we learn a high value policy when there are decisions at thousands of locations while trying to maximize a complex, non-local value model? Developing better methods for solving these planning problems could yield enormous benefits for society, the economy and the environment. In exploring answers to these questions, this thesis has defined spatiotemporal planning problems as factored MDPs, defined equilibrium landscape policies that can be used for representing spatially interrelated actions and presented a policy gradient algorithm for planning in spatiotemporal problems. The need for a spatial landscape policy motivated an exploration of the distributions represented by cyclic causal networks and the development of a novel approach to representation and inference for these models.

The spatiotemporal planning problem in this thesis is represented as a factored MDP where states are factored not only into features but also into cell locations. Actions are also factored into cell locations. A spatial landscape policy was defined by combining local, conditional distributions over actions at each cell to create a joint distribution over landscape actions. This policy was used in a policy gradient planning algorithm and first evaluated using a simple forestry simulation for dynamics. The effectiveness of sharing

policy parameters across locations versus maintaining separate parameters for each location were compared. It was shown that simple, abstract policies with shared parameters for all locations can perform better than overly detailed policies with parameters for each location.

A new planning algorithm, Equilibrium Policy Gradient planning (EPG), was defined which can learn an improved policy in the presence of a non-local value model and use an external simulation for its transition dynamics. An equilibrium landscape policy is defined as a Markov chain where local, causal policies centred on each cell provide the transition dynamics. Gibbs sampling is used to sample landscape actions from this distribution and provide estimates of the marginal distribution for each cell. A single Markov chain can be used to approximate the gradient of this policy, on each stored trajectory, for use in a Natural Actor Critic planning algorithm. This approach of defining a spatial stochastic policy as an equilibrium distribution which is then optimized using policy gradients is a novel one for spatiotemporal planning as far as we are aware.

The EPG algorithm was evaluated on different value functions using an existing forestry simulation with good results. The algorithm can learn policies that account for spatial correlations amongst actions using a small number of parameters. It produced sustainable harvest policies by balancing conflicting components in the value model. The kinds of policies EPG produces should provide insight, for policy makers using it, into the relationship between different cutting strategies and sustainability of the forest. The policy can be interpreted directly by analyzing parameters that weight the relative importance of different features on local decisions. The policy can also be used to produce visualizations which show the actions advised by the policy over time or as a probability map over possible actions at each location.

The Equilibrium Policy Gradient planning algorithm demonstrates a useful way to perform approximate planning for exponentially large spatiotemporal planning problems. Planning in these problems has, in the past, often been restricted either to very small sizes or has required strong assumptions of independence between action locations. EPG can provide a way to bal-

ance off these two extremes and find approximate solutions to very large problems in a feasible manner.

The Natural Actor-Critic algorithm on which EPG is based is an active area of research; new advances in actor-critic algorithms can be applied directly to improving the results presented here. Similarly, the sampling and estimation methods used in the EPG algorithm are independent of the planning problem. Future studies could compare which sampling and estimation methods are most effective for improving the planning performance of the algorithm.

The ability to use external, existing simulations is an important feature of EPG that allows reuse of expert domain knowledge in the form of simulations without needing to fully understand or integrate with those simulations. While integrating with an existing simulation, we found that some simulation tools are in fact simulation-planners which do not conform with the standard idea of a simple state-action transition model and need to be treated differently to integrate with existing planning techniques. This is one of the benefits of expanding applications of AI research into new, real world domains such as environmental planning; this kind of unexpected mismatch will often occur and lead to new ideas and new solutions.

The need for an interpretable representation for a landscape policy that could express spatial correlations between locations motivated us to consider cyclic causal networks. We found that the structure of a cyclic causal network can be used to build a latent variable structure which can represent, or approximate, the equilibrium distribution of the causal network. This Equilibrium Bayesian Network (EBN) model has free parameters which can be learned iteratively using inference on the original and latent variables. In some cases, this method can compute the exact marginal distributions from the equilibrium distribution using exact inference on the EBN. EBNs generalize the standard Structural Equation Model approach to causal modelling to permit models with cyclic correlations between variables.

The EBN method uses the existing structure of the causal network at each step of inference and defines new latent variables to represent the equilibrium distribution of queries on the variables. If the latent variables are

fully connected, the model is equivalent to inference in a DBN. The structure over the latent variables has a natural relation to queries of the marginal probability over subsets of the variables. We have shown that the latent model structure can be defined in such a way that inference on particular marginal queries is exact, although at the expense of greater computational complexity. When less structure is used, the latent model defines an approximation related to the structure of the causal model.

9.1 Future Work

There are several directions we can see for improving the results presented in this thesis and exploiting new possible research directions. Some good first steps would be to perform more evaluations of our approach on larger forestry planning problems as well as on other spatiotemporal planning problems such as forest fire management, infectious disease control and urban planning. We have focused on the problem of sustainable forest management. However, our algorithm is general enough to be applied to other spatiotemporal planning domains. Other opportunities for research directions are outlined in the following sections.

9.1.1 Variation Over Time

In this thesis we have assumed that the same policy is used for all time periods. While the policy depends on the context at each time and location it is possible that policies may need to change more fundamentally at different stages in the future. Climate change and sustainability constraints both make this likely. A current goal might be to move ecological systems towards (or away from) particular states, whereas once the system has entered (or safely avoided) that state in the future, the approach to planning could change. For example, dealing with the Mountain Pine Beetle infestation in the near term requires a focus on salvaging dead trees and restoring a balanced forest, but at some point in the future the policy will fundamentally change to a less aggressive stance. One simple way to achieve this type of policy would be to define separate policy parameters at different time periods. A number of way points could be created at different points in the

future which have their own independent policy parameters. The parameters for the intervening years between two way points could be a weighted average of the parameters for those two way points. This would allow the policy to represent the fact that expectations about forest growth as well as maintenance priorities can change over time.

9.1.2 Cell Clustering

The spatial landscape policy used in Chapters 6 and 7 is a spatially stationary policy, maintaining a single set of parameters for all locations. This allows a general policy to be learned, but when sampling and computing rewards during decision making each cell is still treated individually. This approach could be altered to cluster cells with common properties together, somewhat like a soft version of the strata model described in Chapter 2. Since the clusters that cells belong to would change based on actions (e.g. a recently harvested cell has different properties than a cell with mature trees) it would be important for clusters to be defined in feature space rather than being a fixed set of cells. Separate policy parameters could then be maintained for each cluster of cells during planning. Interesting problems here would include learning the optimal number of clusters and dynamically reclustering the landscape for different time periods.

9.1.3 Hierarchical Parameters

Another approach, which could be implemented separately or in tandem with cell clustering is a hierarchical policy parametrization. A general set of parameters would apply to all cells and more specific parameters could be defined for any subset of cells based on their features. During policy gradient planning, the current process of computing $\nabla_\theta \mathcal{V}^\pi$ and then updating the policy parameters could be augmented by new operations which split or merge the policy parameters. Splits and merges could be explored to build a decision tree representing a hierarchy with each node being an equilibrium landscape policy on a restricted set of cells. These splits and merges of cells could happen across spatial dimensions, feature space or time periods. For example, this could enable learning that:

- A new set of policy parameters are needed 20 years in the future once some sustainability target is reached.
- Certain regions of the forest near towns need a completely different policy formulation than other areas.
- Some value of a certain feature, such as age or tree species, defines a split between two set of cells which can be treated by different policies.

The existing policy structure used by EPG already accounts for actions that must be taken in response to different conditions in the cell or other cells. Splitting policies would allow a new level of flexibility by allowing different conditional distributions, producing two policies which respond to the same conditions in an entirely different way.

9.1.4 Cyclic Causal Distributions

The discussion of cyclic causal networks leaves open several interesting questions. In some networks the sample ordering used to construct the EBN impacts the equilibrium distribution. It's an open problem to bound how much a marginal distribution could vary for different sample orderings.

9.1.5 Filtering in DBNs

The EBN model discussed in Chapter 5 could also be used to perform approximate inference in filtering problems where the maringal probabilities of some variables are tracked over time conditioned on changing observations. The latent model would allow a rich approximation structure that could be compared to other approximation methods such as the BK method of Boyen and Koller [1998]. The directed structure of the latent EBN model should allow more fine grained control over which conditional relationships are modelled in the approximation than other methods based on seperation of cliques.

9.1.6 Policy Interpretation and Visualization

To enable automated planning systems can be a constructive part of existing planning process it is necessary to be able to interpret the meaning of a

policy and justify its advice. As discussed in Section 7.4 there are interesting questions about how to visually represent spatial landscape policies and equilibrium landscape policies. One straightforward approach would be to visualize the probability distributions for each cell overlaid onto a GIS map of the domain so that the user can see what actions are being advised by the policy. Ideally, the user could interact with an optimized policy by intervening visually at specific locations, harvesting one cell for example. The visual representation of the probabilities would be updated by recomputing the equilibrium again.

9.1.7 Parallelization

A major next step from an implementation point of view would be improving the speed of the algorithm by taking full advantage of the opportunities for parallelization. This could lead to significant improvements in speed and in the number of iterations that can be spent learning the policy.

In general, Stages I and II in the `EPG` algorithm can be carried out independently, each using the latest output from the other. Stage I can even be run in multiple parallel processes, simulating many trajectories, while Stage II computes the gradient relative to the currently available history.

Stage I generates new trajectories using the latest policy parameters and stores the trajectories. Meanwhile, Stage II updates the policy using the gradient computed against the total set of simulated trajectories. This results in new policy parameters that consider all of the simulated scenarios seen so far. The gradient computations for each time step within of Stage II(i) can also be run in parallel.

Further performance gains may come from considering parallelization during sampling; there are two ways this could do done. The first approach uses blocking as described by Besag [1975] where cells are partitioned into independent sets. Gibbs sampling proceeds on the partitions, conditioning all cells in the current partition on the actions for cells in all other partitions. The cells within a partition can all then be sampled in parallel. The simplest model of this is a checkerboard pattern for cells laid out in a grid. In highly irregular spatial domains, such as forestry, there would likely need to be

more than two partitions and many partitions may contain only a very small number of cells. In the landscape used for our experiments in Chapter 7 the cells would be partitioned into over 30 partitions with the vast majority of cells contained within the three largest partitions. Parallel sampling of these partitions could lead to significant performance gains since sampling time is one of the dominant sources of complexity of EPG coming from both the trajectory simulation and gradient estimation stages of the algorithm.

The second parallelization approach would be to partition the landscape into clusters of connected cells based on distance. For each cluster, the equilibrium distribution of the actions would be sampled assuming all the other cells in the landscape outside the cluster are fixed. Clusters of cells that are not adjacent could be sampled in parallel as with the blocking approach. But this method allows the possibility of using the exact equilibrium policy for each cluster. The exact equilibrium could be computed using a deterministic method such as the EBN method from Chapter 5. This approach would sacrifice some the global consistency of computing the equilibrium of the full landscape policy but it might model enough local interaction for planning purposes and could potentially be much faster due to the reduced need for sampling.

Bibliography

R. Agarwal, M. Meehan, and D. O'Regan. *Fixed Point Theory and Applications*. Cambridge University Press, 2001. → pages 82

J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983. → pages 137

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998. → pages 45

B. C. Ministry of Forests. *Forest Practices Code of British Columbia Visual Impact Assessment Guidebook*. Forest Practices Branch, Victoria, BC, 2005. → pages 16

C. Bailey-Kellog and F. Zhao. Qualitative spatial reasoning: extracting and reasoning with spatial aggregates. *AI Magazine*, 24:47–60, 2004. → pages 137

E. Z. Baskent and S. Keles. Spatial forest planning: A review. *Ecological Modelling*, 188:145–173, 2005. → pages 1, 20, 25, 27, 28

J. Baxter and P. L. Bartlett. Direct gradient-based reinforcement learning (invited). In *Proceedings of the International Symposium on Circuits and Systems*, pages III–271–274, 2000. → pages 41

J. Baxter and P. L. Bartlett. Infinite-horizon gradient-based policy search. *Journal of Artificial Intelligence Research*, 15:319–350, 2001. → pages 132

R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957. → pages 36, 37, 38, 39

D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27:819–840, November 2002. → pages 48, 134

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Nashua, NH., 2001. → pages 31

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dymanic Programming.* Athena Scientific, Nashua, NH., 1996. → pages 39

J. Besag. Statistical Analysis of Non-Lattice Data. *The Statistician*, 24(3): 179–195, 1975. → pages 147

N. Best, S. Richardson, and A. Thomson. A comparison of Bayesian spatial models for disease mapping. *Statistical Methods in Medical Research*, 14(1):35–59, Feb. 2005. → pages 2

P. Bettinger, J. Sessions, and K. Boston. Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. *Silva Fennica*, 36(2):561–581, 2002. → pages 26, 27

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999. → pages 48, 132

X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Fourteenth Annual Conference on Uncertainty in AI (UAI)*, number 42, page 33, 1998. → pages 135, 146

X. Boyen and D. Koller. Exploiting the architecture of dynamic systems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999. → pages 136

M. Boyland, J. Nelson, and F. L. Bunnell. A test for robustness in harvest scheduling models. *Forest Ecology and Management*, 207(1-2), 2005. → pages 17

P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues.* Springer, 1999. → pages 71, 75, 76, 80, 93

X. Cao. A basic formula for online policy gradient algorithms. *IEEE Transactions on Automatic Control*, 50(5), May 2005. → pages 41, 132

X. Cao and H. Chen. Perturbation realization, potentials, and sensitivity analysis of markov processes. *IEEE Transactions on Automatic Control*, 42:1392–1382, 1997. → pages 131

A. Cauchy. *Oeuvres 2*, volume III. 1821. → pages 82

B. C. Chamberlain and M. J. Meitner. Automating the visual resource management and harvest design process. *Landscape and Urban Planning*, 90(1-2):86 – 94, 2009. → pages 29

A. Clements, N. Lwambo, L. Blair, U. Nyandindi, G. Kaatano, S. Kinung'hi, J. P. Webster, A. Fenwick, and S. Brooker. Bayesian spatial analysis and disease mapping: tools to enhance planning and implementation of a schistosomiasis control programme in tanzania. *Tropical Medicine & International Health*, 11(4):490–503, 2006. → pages 2

C. Coburn and A. Roberts. A multiscale texture analysis procedure for improved forest stand classification. *International Journal of Remote Sensing*, 25(20):4287–4308, 2004. → pages 12

Council of Forest Industries. The forestry industry in BC: Economic statistics. http://www.cofi.org/forest_industry_BC/economic_statistics.htm, 2007. → pages 10

M. Crowley and D. Poole. Policy gradient planning for environmental decision making with existing simulators. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, Special Track on Computational Sustainability and AI. (AAAI-11)*, San Francisco, 2011. → pages

M. Crowley, B. Boerlage, and D. Poole. Adding local constraints to Bayesian networks. In Z. Kobti and D. Wu, editors, *Canadian AI*, volume LNAI 4509, pages 344–355, Montreal, 2007. Springer-Verlag. → pages

M. Crowley, J. Nelson, and D. Poole. Seeing the forest despite the trees: Large scale spatial-temporal decision making. In *Proceedings of the Twenty-Fifth Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, 2009. → pages

G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963. → pages 24

A. Darwiche. *Modeling and Reasoning with Bayesian Networks.* Cambridge University Press, 2009. → pages 136

D. Dash. Restructuring dynamic causal systems in equilibrium. In *10th Int. Workshop on AI and Stats*, pages 81–88, 2005. → pages 75

E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, Palo Alto, CA, 1990. → pages 137

L. S. Davis, K. Johnson, P. S. Bettinger, and T. E. Howard. *Forest Management: To Sustain Ecological, Economic, and Social Values*. Series in Forest Resources. McGraw Hill, 4th edition, 2001. → pages 14, 18, 20, 21, 23, 124

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989. → pages 34, 71, 132

R. Dechter. Bucket elimination : A unifying framework for probabilistic inference. In E. Horvitz and F. Jensen, editors, *Proceeding of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 211–219, 1996. → pages 33

M. Eng, A. Fall, J. Hughes, T. Shore, B. Riel, and P. Hall. Provincial level projection of the current mountain pine beetle outbreak. Technical report, NRC-CFS-PFC, 2004a. → pages 12, 17

M. Eng, A. Fall, J. Hughes, T. Shore, B. Riel, and P. Hall. Provincial level projection of the current mountain pine beetle outbreak. Technical report, NRC-CFS-PFC, 2004b. → pages 12

A. Fall, T.L.Shore, L. Safranyik, W. Riel, and D. Sachs. Integrating landscape-scale mountain pine beetle projection and spatial harvesting models to assess management strategies. In T.L.Shore, J.E.Brooks, and J.E.Stone, editors, *Mountain Pine Beetle Symposium: Challenges and Solutions*, pages 114–132. NRC-CFS-PFC, 2003. → pages 23

F. M. Fisher. A Correspondence Principle for Simultaneous Equation Models. *Econometrica*, 38(1), 1970. ISSN 00129682. → pages 88

N. Forsell, F. Garcia, and R. Sabbadin. Reinforcement learning for spatial processes. In *18 th World IMACS / MODSIM Congress*, pages 755–761, 2009a. → pages 29, 134

N. Forsell, P. Wikström, F. Garcia, R. Sabbadin, K. Blennow, and L. Eriksson. Management of the risk of wind damage in forestry: a graph-based Markov decision process approach. *Annals of Operations Research*, pages 1–18, Feb. 2009b. → pages 29, 134

C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1):199–227, 1992. → pages 137

D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. Combining spatial and temporal logics: Expressiveness vs. complexity. *Journal of Artificial Intelligence Research*, 23:167–243, 2005. → pages 137

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984. → pages 35

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986. → pages 26

C. Glymour and P. Spirtes. Latent variables, causal models and overidentifying constraints. *Journal of Econometrics*, 39(1-2):175–198, 1988. → pages 88

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, USA., 1989. → pages 26

C. Gomes. Computational sustainability: Computational methods for a sustainable environment, economy, and society. *The Bridge, National Academy of Engineering*, 39(4), Winter 2009. → pages 139

C. Guestrin, S. Venkataraman, and D. Koller. Context-specific multiagent coordination and planning with factored MDPs. In *American Association for Artificial Intelligence*, 1996. → pages 48

C. Guestrin, D. Koller, and R. Parr. Multiagent Planning with Factored MDPs. In *Advances in Neural Information Processing Systems 14*, 2001. → pages 133

C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *ICML*, pages 227–234, 2002. → pages 133

J. Y. Halpern. Axiomatizing Causal Reasoning. *Journal of Artificial Intelligence Research*, 12:202–210, 2000. → pages 65, 67

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970. → pages 36

N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Siam, Philadelphia, 1996. → pages 46

J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Harbor, MI, USA., 1975. → pages 26

R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960. → pages 38

Y. Iwasaki and H. A. Simon. Causality and model abstraction. *Artificial Intelligence*, 67(1):143–194, May 1994. → pages 68

F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, 1994. → pages 33

A. Jiang, K. Leyton-Brown, and N. Bhat. Action-graph games. *Games and Economic Behavior*, 71(1):141–173, January 2011. → pages 138

K. N. Johnson and H. Sheurmann. Techniques for prescribing optimal timber harvest and investment under different objectives–discussion and synthesis. *Forestry Science*, (Monograph 18), 1977. → pages 21

S. Kakade. A natural policy gradient, 2002. → pages 45

B. Kent, B. B. Bare, R. C. Field, and G. A. Bradley. Natural resource land management planning using large-scale linear programs: the USDA forest service experience with FORPLAN. *Operations Research*, pages 13–27, 1991. → pages 25

K. Kersting and K. Driessens. Non-parametric policy gradients: A unified treatment of propositional and relational domains. In *ICML*, 2008. → pages 41

J. P. Kimmins et al. Possible forest futures: Balancing biological and social risks in mountain pine beetle epidemics. In T. Shore, J.E.Brooks, and J. Stone, editors, *Mountain Pine Beetle Initiative*, Working Paper 2005-11, pages 33–40, Victoria, Canada., 2005. NRC-CFS-PFC. → pages 28

S. Kirkpatrick, C. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983. → pages 26

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, London, 2009a. → pages 94, 95

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, London, 2009b. → pages 34

M. Kurttila. The spatial structure of forests in the optimization calculations of forest planning: a landscape ecological perspective. *Forest Ecology Management*, 142:129–142, 2001. → pages 25

K. Lai, C. Gomes, M. Schwartz, K. McKelvey, D. Calkin, and C. Montgomery. The steiner multigraph problem: Wildlife corridor design for multiple species. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence, Special Track on Computational Sustainability and AI. (AAAI-11)*, 2011. → pages 17

G. Liu, S. Han, X. Zhao, J. D. Nelson, H. Wang, and W. Wang. Optimisation algorithms for spatially constrained forest planning. *Ecological Modelling*, 194(4):421–428, April 2006. → pages 27

C. Lockwood and T. Moore. Harvest scheduling with spatial constraints: a simulated annealing approach. *Canadian Journal of Forest Research*, 23 (3):468–478, 1993. → pages 27, 139

K. Lowell. Effects of adjacent stand characteristics and boundary distance on density and volume of mapped land units in the boreal forest. *Plant Ecology*, 143:99–106, 1999. → pages 12

O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems". In *Proceeding of the 16th National Conference on Artificial Intelligence*, 1999. → pages 134

R. Mateescu, R. Dechter, and K. Kask. Tree approximation for belief updating. In *18th National Conference on Artificial Intelligence*, pages 553–559, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. → pages 33, 76

A.-H. Mathey and J. Nelson. *Designing Green Landscapes*, chapter Decentralized Forest Planning Models - a Cellular Automa Framework, pages 167–183. Springer, 2007. → pages 17, 28, 134

F. S. Melo and M. Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757 – 1789, 2011. → pages 133

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953. → pages 36

T. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI '01)*, pages 362–369, 2001. → pages 136

R. Neal. On deducing conditional independence from d-separation in causal graphs with feedback (research note). *Journal of Artificial Intelligence Research*, 12:87–91, 2000. → pages 88

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999. → pages 23

J. Pazis and R. Parr. Generalized value functions for large action sets. In *Proceedings of the 25th international conference on Machine learning*, 2011. → pages 133

J. Pearl. *Probabilistic Reasoning in Intelligent Systems:Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988. → pages 32, 76, 136

J. Pearl. Aspects of graphical models connected with causality. In *49th Session of the International Statistics Institute*, 1993. → pages 34

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition edition, 2009. → pages 8, 34, 65, 68, 71, 75, 90

J. Pearl and R. Dechter. Identifying independencies in causal graphs with feedback. In *12th Conference on Uncertainty in AI*, pages 420 – 426, 1996. → pages 88

J. Peters, S. Vijayakumar, and S. Schaal. Natural actor critic. In J. G. et. al., editor, *European Conference on Machine Learning*, volume 3720 of *Lecture Notes in Computer Science*, pages 280–291. Springer Verlag, Berlin, 2005. → pages 45

A. Pfeffer. Approximate separability for weak interaction in dynamic systems. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI '06)*, 2006. → pages 136

W. B. Powell. Merging AI and OR to solve High-Dimensional stochastic optimization problems using approximate dynamic programming. *INFORMS Journal on Computing*, 22(1):2–17, 2010. → pages 131

T. Pukkala and M. Kurttila. Examining the performance of six heuristic optimisation techniques in different forest planning problems. *Silva Fennica*, 39(1):67–80, 2005. → pages 25, 27, 28

M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1994. → pages 36

M. L. Puterman and M. C. Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):pp. 1127–1137, 1978. → pages 39

C. R. Rao. *Linear statistical inference and its applications*. Wiley, 1965. → pages 94

T. Richardson. The geometry of turbo-decoding dynamics. *Information Theory, IEEE Transactions on*, 46(1):9 –23, jan 2000. → pages 136

M. Riedmiller, J. Peters, and S. Schaal. Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 254–261, 2007a. → pages 45

M. Riedmiller, J. Peters, and S. Schaal. Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 254–261, 2007b. → pages 41, 43, 45, 53, 58

M. Schmidt and K. Murphy. Modeling discrete interventional data using directed cyclic graphical models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*, pages 487–495, Arlington, Virginia, United States, 2009. AUAI Press. → pages 88

J. Sessions, K. Johnson, et al. Methodology for simulating forest growth, fire effects, timber harvest and watershed disturbance under difference management regimes. In *Sierra Nevada Ecosystem Project: Final Report to Congress*, volume II. Davis: University of California., 1996. → pages 21, 25

R. H. Strotz. Interdependence As a Specification Error. *Econometrica*, 28 (2), 1960. → pages 68, 88

R. H. Strotz and H. O. A. Wold. Recursive vs. Nonrecursive Systems: An Attempt at Synthesis (Part I of a Triptych on Causal Chain Systems). *Econometrica*, 28(2), 1960. → pages 63, 68

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. → pages 37, 39

R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pages 1057–1063. MIT Press, 2000. → pages 8, 41, 42, 131

C. Szepesvári. *Algorithms for Reinforcement Learning*. Number 9 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2010. → pages 40

K. R. Walter. Design and development of a generalized forest management modeling system: Woodstock,. In *Proceedings of the International Symposium on Systems Analysis and Management Decisions in Forestry*, pages 190–196. www.remsoft.com, 1993. → pages 24

L. Weaver and N. Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference*, pages 538–545, Seattle, WA, 2001. Morgan Kaufman. → pages 43

R. J. Whitehead and G. L. Russo. "Beetle-proofed" lodgepole pine stands in interior BC have less damage from MPB. Technical report, NRC-CFS-PFC, 2005. → pages 15, 17

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(2):229–256, 1992. → pages 41, 42

J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *International Joint Conference on Artificial Intelligence – Distinguished Lecture track*, 2001. → pages 136

N. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 171–178, 1994. → pages 33, 76

Appendix A

Integrating with a Simulator-Planner

The EPG algorithm on page 110 uses an episodic simulation, `sampleSimPlanner`, to provide the dynamics for the MDP. The simulator used for the experiments in Chapter 7 was the FSSAM simulator-planner described in Section 2.4.2. Integrating with a simulator-planner raised challenges which would not have arisen if we had continued using only custom simulations of the forest dynamics.

We had expected that existing forestry simulations would closely resemble the classical idea of a stochastic transition function. This would be a black box that takes in a state and an action and returns a new state conforming to some straightforward stochastic model. The simulators we looked at were ATLAS and FSSAM. Each was primarily meant to be used manually and performed their own optimization after input was received from the user. Both simulators integrate the action selection step with the state transition step to some degree and thus are simulation-planners according to our definition in Section 2.4.1.

FSSAM was chosen for the evaluation due to its detailed model of forest development, the fact that it is actively used for planning by the BC Forest Service and the availability of a large scale data set for simulating a real forest domain. As described in Section 2.4.2, FSSAM simulates an entire

trajectory at once, but it has the facility to call an external policy to 'advise' the cut selection process by providing the order in which cells are considered for action.

To integrate `EPG` with the FSSAM simulation-planner, `sampleSimPlanner` was implemented as a wrapper function, to interact with FSSAM and an equilibrium landscape policy. `sampleSimPlanner` runs an instance of FSSAM and at each time period, receives the next state of the forest which is sent to the function `sampleSpatialPolicy` (shown on page 95). `sampleSpatialPolicy` samples a new landscape action $\mathbf{a}$ from the policy and acquires an estimate of the marginal distribution, $\mathbf{M}[a_c]$ for each cell c.

Algorithm 10: `sampleSimPlanner` $(\mathbf{s}^0, \theta)$ - This algorithm shows an existing simulator-planner (see Section 2.4.1) that has been modified to connect to an equilibrium policy.

for $t \in \mathrm{T}$ **do**
  `// Policy Phase`
  $\mathbf{a}, \mathbf{M}$ = `sampleSpatialPolicy` $(\mathbf{s}^t, \theta)$
  `cellOrder`, `blockedCells` = `sortAndBlock` $(C, \mathbf{a}, \mathbf{M})$
  `// Constraint and Transition Phase`
  $\mathbf{s}^{t+1}$ = `FSSAMTransition` (`cellOrder`, `blockedCells`)
$k = \langle \mathbf{s}^0, \mathbf{a}^0, \mathbf{s}^1, \mathbf{a}^1, \ldots, \mathbf{s}^{\mathrm{T}}, \mathbf{a}^{\mathrm{T}} \rangle$
return k

The function `sortAndBlock` uses the output of `sampleSimPlanner` to produce two vectors with the following properties:

- `cellOrder`- contains the cell c if a_c = `Cut`. The ordering of cells in `cellOrder` is determined by marginal probability $\mathbf{M}$ for each cell taking the sampled action. So if $\mathbf{M}[a_c] > \mathbf{M}[\mathbf{a}_d]$ then $c < d$ in `cellOrder`.
- `blockedCells`- contains the cells that are blocked from being selected by the simulator for cutting in this time period. Any cell where a_c = `NoCut` are blocked.

After the ordering is determined, the remaining processes of the simulation-

planner, represented by `FSSAMTransition`, are carried out unmodified. After the final time period is completed, the entire simulated trajectory is returned.