Policy Gradient Planning for Environmental Decision Making with Existing Simulators

Mark Crowley and **David Poole**
University of British Columbia
crowley@cs.ubc.ca poole@cs.ubc.ca

Abstract

In environmental and natural resource planning domains actions are taken at a large number of locations over multiple time periods. These problems have enormous state and action spaces, spatial correlation between actions, uncertainty and complex utility models. We present an approach for modeling these planning problems as factored Markov decision processes. The reward model can contain local and global components as well as spatial constraints between locations. The transition dynamics can be provided by existing simulators developed by domain experts. We propose a landscape policy defined as the equilibrium distribution of a Markov chain built from many locally-parameterized policies. This policy is optimized using a policy gradient algorithm. Experiments using a forestry simulator demonstrate the algorithm's ability to devise policies for sustainable harvest planning of a forest.

Introduction

In many environmental and natural resource planning domains, an action needs to be taken at every point in a large landscape at multiple time periods. Automated planning in these domains is challenging because they can have enormous state spaces and action spaces, there are spatial interactions between actions, there is much uncertainty, and there are rich utility functions. Additional challenges arise because the best models of the dynamics, those that have been painstakingly built by researchers and practitioners, are often deterministic and do not conform to the sorts of models that are usually studied by AI researchers.

Our goal here is to bridge the gap between concrete spatial planning in environmental domains and the planning and modelling techniques used in AI. Our solution provides a flexible model for planning in large spatial domains which satisfies many real world requirements for use by planning practitioners.

We define a landscape policy in terms of local, parameterized policies. A distribution over global actions is defined by the equilibrium of a Markov chain built from these local policies. This approach allows the policy to account for spatial correlations between actions at different locations. It is also provides an understandable, local justification for actions that can be used by practitioners on the ground when implementing the policy.

We motivate and demonstrate the effectiveness of our approach in a forestry planning domain using policy gradient planning to improve the policy parameters relative to a provided reward model. Many existing modelling techniques used in forestry planning have difficulty dealing with spatial interactions among decisions and with uncertainty in the state or dynamics of a system. For the forestry planning community our approach suggests a way towards richer modelling and planning than is currently the norm.

Spatial Planning as an MDP

We assume a landscape is divided into cells that have spatial relations among them (including the neighbors relation). Let C be the set of all cells. Let S be the set of states of a single cell and A the set of actions that can be taken in a cell. We perform planning for time periods t up to time T of an inifinite horizon planning problem.

A spatial planning problem is a factored Markov decision process(MDP) (Puterman 1994; Boutilier, Dean, & Hanks 1999) $\langle \mathbf{S}, \mathbf{A}, r, \Delta \rangle$ where:

- $\mathbf{S}$, the set of *landscape states*, is S^{C}, the set of functions from C into S. A landscape state is denoted $\mathbf{s}$.
- $\mathbf{A}$, the set of *landscape actions*, is A^{C}, the set of functions from C into A. A landscape action is denoted $\mathbf{a}$. The action at a particular cell c is denoted a_c.
- $r(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1})$ is a reward function which returns the reward received by starting in landscape-state $\mathbf{s}^t$, then taking action $\mathbf{a}^t$ and ending up in state $\mathbf{s}^{t+1}$ at the next time period.
- $\Delta(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1})$ is the dynamics which specifies the probability of transitioning from landscape state $\mathbf{s}^t$ to state $\mathbf{s}^{t+1}$ given landscape action $\mathbf{a}^t$.

A trajectory is a series of states and actions over the considered planning horizon, $k = \langle \mathbf{s}^{0:\mathrm{T}}, \mathbf{a}^{0:\mathrm{T}-1} \rangle = \langle \mathbf{s}^0, \mathbf{a}^0, \ldots, \mathbf{a}^{\mathrm{T}-1}, \mathbf{s}^{\mathrm{T}} \rangle$. The total discounted return for a trajectory is

$$R(k) = \sum_{t=0}^{\mathrm{T}-1} \gamma^t r(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1}) \quad (1)$$

where $\gamma \in [0,1]$ is a discount factor.

Example Domain : Forestry Planning

To see the difficulty of planning in this type of problem, consider an example from the domain of forestry planning. Forestry planning is the problem of deciding how to sustainably manage forests while balancing complex economic, ecological and societal values. Decisions need to be made about management options for thousands or hundreds of thousands of forest locations each year. For our testing we use a forest landscape of 1880 cells, part of which is shown in figure 1.

Sustainable plans need to be devised over a horizon of centuries. Plans are always being revised based on new data, improved models and changing societal values or regulations. This means that implementation of the actions advised by a plan may only actually be carried out for the first one or two years. However, this does not mean the future can be ignored while planning; the future must be considered to avoid long-term damaging outcomes.

Actions 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. Other actions include treating trees for disease, building roads or doing nothing at all. Actions are generally carried out in one year intervals with the actual implementation of the year's plan taking many months.

The space of possible landscape states and actions is huge. Consider a relatively small spatial planning problem with 1000 cells, 10 binary features per cell and binary actions. The number of possible landscape actions is $2^{1000} \approx 10^{300}$ while the number of landscape states is $(2^{10})^{1000} \approx 10^{100000}$. Clearly any method that relies on enumerating states and actions is impractical.

Features Features describing the state of a cell could include : area of the cell, volume of the lumber, elevation, climatic zone, distribution of tree species, average age of trees in the cell, level of pest infestation and the number of dead and living trees. The boundaries of cells are defined so that the features apply fairly uniformly throughout the area of a cell. Cell features can also include aggregate features derived from the states and actions of other cells. Some example aggregate features for a cell are: the number of neighboring cells being cut, the total pest infestation rate of neighboring cells, the size of the largest neighbor, the current fraction of the harvest target being cut across the landscape and the age distribution of cells of within a region.

Rewards Forestry planners need to consider how to optimize a wide variety of objectives including harvest yields and shortfalls, risk of fire and pest outbreaks, costs of building roads, the market value of lumber, employment levels in different communities, recreational uses of forests as well as satisfying ecological constraints.

These objectives are often embedded only in reports and studies. Some objectives are considered only by strategic planners setting targets for a region while others are only considered by those carrying out planning on the ground. Rarely are all the relevant components of an objective model written down in one location. Boyland, Nelson, & Bunnell (2005) provide a good overview of some planning methods and objectives used in forestry planning.

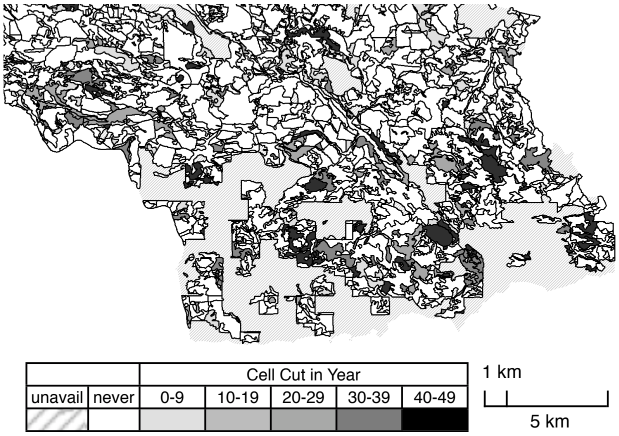

Figure 1: Part of a region of forest in British Columbia used for testing. Each colour indicates cells cut at some point in each 10 year period for a run of the simulator. The policy used is the AVR policy resulting in a sustainable cut level.

A challenging aspect of natural resource planning problems is the existence of spatial constraints over what action can be performed. We express these constraints as part of the reward model. Two such constraints commonly used are adjacent cutting constraints and biodiversity age bounds. The adjacent cutting constraint is simply a requirement that adjacent cells not be cut in the same year or within a "greenup" period of several years. This is an attempt to reduce the ecological impact of clear-cutting. We demonstrate this constraint in the reward model in our experiments. Some other common spatial constraints that we do not look at here include biodiversity age bounds and connectivity constraints. Biodiversity age bounds are targets for what portion of the forest must fall within a particular age classification. There might be a constraint that at least 30% of the forest is old growth trees (over 200 years old) or that no more than 25% is less than 10 years old. This type of constraint could be applied globally or applied at multiple levels of abstraction leading to different outcomes in each case. Mathey & Nelson (2007) provide a good description of these constraints in more detail. Spatial constraints could also model requirements about which different types of connectivity are valued. This could include managing firebreaks or ensuring that a policy maintains a connected migration corridor for wildlife.

Dynamics Many simulations of the dynamics for forestry domains are painstakingly developed by expert practitioners, but they are often not designed with automated planning in mind. Simulators are used by human planners to search through possible plans and their outcomes. These simulators

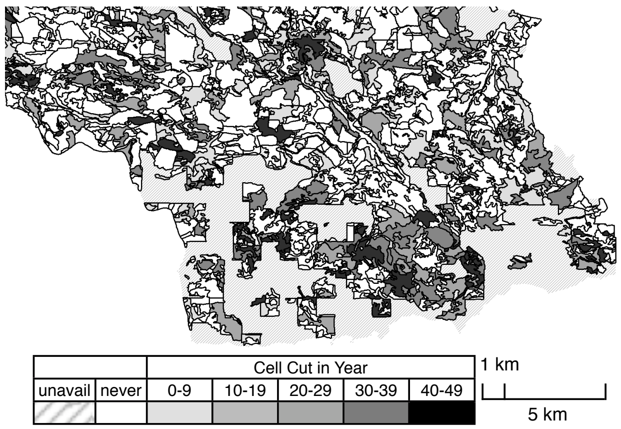

Figure 2: Part of a region of forest in British Columbia used for testing. Each colour indicates cells cut at some point in each 10 year period for a run of the simulator. The policy used is the `HVR` policy resulting in an unsustainable cut level.

automate some tasks that are difficult or repetitive to perform manually. They are often the best models of the domain and so should be used for decision making.

The particular simulator used in our evaluation (described in more detail later) is a forest harvest simulator called the *Forest Service Spatial Analysis Model (FSSAM)*[1] which is used by the British Columbia Forest Service to analyze the impact of different harvest levels on sustainable ecological and economic goals. We treat FSSAM as a (mostly) black box which provides the dynamics Δ for our MDP.

Simulators used in forestry often have essentially aspatial dynamics, modelling the growth of each stand of trees independently as this is a reasonable approximation for trees alone. FSSAM can also model the predicted growth of pests, such as the Mountain Pine Beetle, which can spread across the landscape and are a significant factor in forest development. For the experiments presented here this feature has been disabled.

Our Approach

The challenge is to develop a general spatial planning algorithm for environmental and natural resource problems. Distributions over actions at different locations can be correlated by spatial rewards, such as spatial constraints, and spatial dynamics, such as pest spread. Any representation of the policy needs to model these correlations. The scale of the domain makes any kind of state based optimization method infeasible so we need to use a factored approach. The dynamics are also inaccessible so analytical methods which require an explicit form for the dynamics will not be available to us.

We address these problems by using *policy gradient (PG) planning* (Sutton *et al.* 2000) to optimize a stochastic spatial policy which models the the spatial correlations explicitly. PG algorithms attempt to optimize a parameterized policy by following the gradient of its expected value and do not require an analytical formulation for the transition dynamics. The policy we optimize represents spatial correlations in terms of an equilibrium distribution over local, interacting policies. This allows for a small number of interpretable parameters with an equilibrium semantics for the distribution.

The Equilibrium Spatial Policy

A spatial policy is composed of interacting local cell policies. Each local cell policy definition answers the question: *"What action would you take at this location if you could wait until all the actions at other locations were already decided."* The landscape policy is then built from local cell policies.

The Cell-Policy

A *cell policy*, $\pi_c(a_c|a_{-c},\mathbf{s},\theta)$, is a distribution defined by parameters θ over the actions a_c at cell c, conditioned on the landscape state $\mathbf{s}$ and actions at all other cells a_{-c}. Intuitively, π_c models the probability the policy would choose action a_c if c were the 'last' cell to be decided after the actions for all other cells, a_{-c}, had been decided.

We assume a set $\mathscr{F}$ of features such that for each $f \in \mathscr{F}$, the value $f_c(a_{-c},\mathbf{s})$ can depend on the actions from other cells as well as the states of all cells. Feature values are normalized to fall within [0,1]. We model a cell-policy as a log-linear distribution over the actions at cell c given the state of the cell and other cells. The policy parameters $\theta_f(a)$ associate a weight with each combination of cell-actions and features f. [2] A potential function ψ combines these policy weights with each feature and action for a single cell:

$$\psi(a_c,a_{-c},\mathbf{s},\theta) = \sum_f \theta_f(a_c) f_c(a_{-c},\mathbf{s}) \qquad (2)$$

The cell-policy is then:

$$\pi_c(a_c|a_{-c},\mathbf{s},\theta) = \frac{\exp(\psi(a_c,a_{-c},\mathbf{s},\theta))}{\sum_{b_c \in A}\exp(\psi(b_c,a_{-c},\mathbf{s},\theta))} \qquad (3)$$

The policy parameters are used across all cells, and so are *spatially- stationary*; each cell depends on the states and actions of its surroundings using the same function. The identity of the cell is not used.

The Landscape Policy

The cell policy above is a cyclic definition with each cell depending on the actions at other cells. Thus (3) does not *directly* define the conditional probability of an action at a cell given the actions at other cells. Instead, these local cell

[1] http://www.barrodale.com/bcs/timber-supply-model

[2] In the binary case is it not necessary to model a policy parameter for each combination of feature and action, the same expressiveness could be achieved with one parameter per feature. We need $\mathscr{F} \times |-1|$ parameters for the general case if we do not want to impose any kind of ordering on cell actions. We choose to add the additional action to provide a uniform structure for the policy model. This may aid practitioners interpreting the policy directly as we show in table 2.

policies are interpreted as the transition model of a Markov chain.

To define the Markov chain, we use a sample ordering which is a total ordering of the cells. Samples of actions at different steps in the Markov chain will be indexed as $\mathbf{a}^\tau$ or with other letters while actions at different time periods during the planning will always be indexed with t. Each sample step, τ, goes through all of the cells according to the sample ordering and uses (3) to update the action for cell c given the current actions for all other cells. This can be interpreted as partitioning the cells into c_-, those before c in the sample ordering, and c_+, those after c, so that (3) becomes $\pi_c(a_c|a^\tau_{c-} \cup a^{\tau-1}_{c+}, \mathbf{s}, \theta)$.

The equilibrium of the Markov chain defines the *landscape policy*, $\pi(\mathbf{a}|\mathbf{s},\theta)$, the distribution over landscape actions $\mathbf{a} \in \mathbf{A}$. To simplify notation we treat $\mathbf{s}$ and θ as fixed for now and we recursively define the κ-step lookahead probability:

$$p_\kappa(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^\tau) =$$
$$\begin{cases} 1.0 & \text{if } \kappa = 0 \\ \prod_c \pi_c(a^\tau_c | a^\tau_{c-} \cup a^{\tau-1}_{c+}) & \text{if } \kappa = 1 \\ \sum_{\mathbf{a}^{\tau-\kappa+1} \in \mathbf{A}} p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa+1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) & \text{otherwise} \end{cases}$$

The equilibrium of the Markov chain in terms of p_κ is then:

$$\pi(\mathbf{a}^\tau) = \lim_{\kappa \to \infty} p_\kappa(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^\tau) \qquad (4)$$

The equilibrium can also be represented in a recursive form which we will use for defining the gradient:

$$\pi(\mathbf{a}^\tau) = \sum_{\mathbf{a}^{\tau-1}} \pi(\mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \qquad (5)$$

We can sample from the equilibrium using Gibbs sampling following the sample ordering. Because all of the probabilities from (3) are greater than zero, this Markov chain is *ergodic*, having a nonzero probability of reaching all landscape actions. Ergodic Markov chains are guaranteed to converge to a unique equilibrium distribution as $\tau \to \infty$ (Bremaud 1999).

Crowley, Nelson, & Poole (2009) define the landscape policy as the direct product of local cell policies similar to (3) rather than as an equilibrium. That approach focuses on the impact of different parameterizations on the quality of the output but did not adequately represent spatial interdependencies.

Spatial Policy Gradient Planning

We assume we are given an initial state $\mathbf{s}^0$ representing the current state of the world. We begin with a set of parameters θ (usually randomly selected) and an empty set of trajectories K. The spatial policy gradient planning algorithm involves two interacting processes which are iterated until convergence of the gradient or some maximum time is reached.

I Generate sample trajectories :

  i Generate $\mathrm{k} = \langle \mathbf{s}^{0:\mathrm{T}}, \mathbf{a}^{0:\mathrm{T}-1} \rangle$ using the simulator as implementation of the dynamics $\Delta(\mathbf{s}^t, \mathbf{a}^t, \mathbf{s}^{t+1})$ 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 gradient relative to a single trajectory: For each trajectory $\mathrm{k} \in \mathrm{K}$ compute the combined gradient of the current policy for each state and action encountered at each timestep in k:

$$\nabla_\theta \log \pi(\mathbf{a}^\mathrm{k}|\mathbf{s}^\mathrm{k}, \theta) = \sum_t \nabla_\theta \log \pi(\mathbf{a}^{\mathrm{k},t}|\mathbf{s}^{\mathrm{k},t}, \theta) \qquad (6)$$

  where $\mathbf{a}^{\mathrm{k},t}$ and $\mathbf{s}^{\mathrm{k},t}$ are the landscape action and state at time t from the trajectory k. How to estimate the gradient $\nabla_\theta \log \pi(\mathbf{a}^{\mathrm{k},t}|\mathbf{s}^{\mathrm{k},t}, \theta)$ is described in the next section.

  ii Combine gradients from multiple trajectories: The expected policy value, $\mathscr{V}^\theta$, weights the total return, R(k), received for each trajectory by the probability of that trajectory under the current policy. In practice we choose a subset $\mathrm{H} \subseteq \mathrm{K}$ containing trajectories with the highest $R(k)$. The gradient of $\mathscr{V}^\theta$ is:

$$\nabla_\theta \mathscr{V}^\theta \approx \frac{1}{|\mathrm{H}|} \sum_{\mathrm{k} \in \mathrm{H}} \mathrm{R}(\mathrm{k}) \nabla_\theta \log \pi(\mathbf{a}^\mathrm{k}|\mathbf{s}^\mathrm{k}, \theta) \qquad (7)$$

  iii Update policy parameters:

$$\theta' = \theta + \lambda \nabla_\theta \mathscr{V}^\theta \qquad (8)$$

  where λ is a learning rate.

The subset H has a fixed size and serves the purpose of ensuring the computation time for the gradient estimate doesn't grow without bound as the number of trajectories grows. It also allows us to focus on the high reward trajectories to improve the policy more directly.

One of the attractive properties of this algorithm for large problems is that its components can be carried out in parallel. Multiple instances of Step I can be carried out independently from a seperate instance of Step II. Step I generates a new trajectory using the latest policy parameters and stores the trajectory. Meanwhile, Step II updates the policy using the gradient computed against the set of simulated trajectories. This results in new policy parameters that consider all of the simulated trajectories scenarios seen so far. When estimating the gradient, each time period within II(i) can also be run as seprate parallel computations.

Gradient of Policy

To complete our algorithm we need the gradient of the landscape policy from equation (6). This can be worked out by taking the gradient of the recursive form of the policy equilibrium from (5). After applying the recurrence ω times and grouping terms this becomes:

$$\nabla_\theta \pi(\mathbf{a}^\tau) =$$
$$\sum_{\mathbf{a}^{\tau-\omega}} \nabla_\theta \pi(\mathbf{a}^{\tau-\omega}) p_\omega(\mathbf{a}^\omega, \mathbf{a}^\tau) + \sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \pi(\mathbf{a}^{\tau-\kappa})$$
$$\sum_{\mathbf{a}^{\tau-\kappa+1}} \nabla_\theta p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) \quad (9)$$

where κ iterates over the lengths of Markov chains from 1 up to a maximum length of ω.

Note that the gradient in equation (9) is cyclic, requiring the gradient for all possible actions $\mathbf{a}^{\tau-\omega} \in \mathbf{A}$ in order to compute the gradient for $\mathbf{a}^\tau$. We can eliminate this cycle by using (4) to note that as $\omega \to \infty$

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

Thus, the first term in (9) can be removed, leaving us with the following approximation for finite ω:

$$\nabla_\theta \pi(\mathbf{a}^\tau) \approx \sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \pi(\mathbf{a}^{\tau-\kappa})$$
$$\sum_{\mathbf{a}^{\tau-\kappa+1}} \nabla_\theta p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^\tau) \quad (10)$$

The remaining term we need is the gradient of the transition probability, $\nabla_\theta p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau)$. The parameters of the local cell policy θ are broken down into action and feature components α and f.

$$\begin{aligned}&\nabla_{\alpha f} \log\, p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) \\ &\quad = \sum_c \nabla \log \pi_c(a_c^\tau | a_{c-}^\tau \cup a_{c+}^{\tau-1}) \\ &\quad = \sum_{c \in \mathrm{C}} g(\alpha, a_{c-}^\tau, a_{c+}^{\tau-1}) f(\alpha, a_{c-}^\tau, a_{c+}^{\tau-1}) \quad (11)\end{aligned}$$

where

$$g(\alpha, a_{c-}^\tau, a_{c+}^{\tau-1} f) = \begin{cases} 1 - \pi_c(\alpha | a_{c-}^\tau \cup a_{c+}^{\tau-1}) \text{ if } \alpha = a_c^\tau \\ -\pi_c(\alpha | a_{c-}^\tau \cup a_{c+}^{\tau-1}) \text{ if } \alpha \neq a_c^\tau \end{cases}$$

A full derivation of a similar policy can be found in Crowley, Nelson, & Poole (2009). The gradient (11) can be included into the policy gradient in (10) by using the fact that $\nabla f(x) = f(x) \nabla \log f(x)$.

The gradient is essentially gathering the expected contributions from many different cells and combining them. Each parameter is modified in proportion to how often the associated action occurred at different locations and at different times weighted by the likelihood of that outcome for some length of the Markov chain. So, parameters are increased in proportion to their consistency with the given trajectory and decreased in proportion to their mismatch with the trajectory.

The overall effect in the planning algorithm in equations (7) and (8) is to alter the policy to make the good trajectories more likely and the bad trajectories less likely.

Estimating the Gradient Computing the gradient approximation in (10) is difficult since it involves sums over exponential numbers of landscape actions and the true equilibrium distribution of the policy. To estimate this we generate samples, $\mathbf{a}^{0:\omega}$, from a single Markov chain and store the transition probabilities $p_1(\mathbf{a}^i, \mathbf{a}^{i+1})$ for samples $i \in [0, \omega)$ as well as computing $p_1(\mathbf{a}^i, \sigma)$ for each sample; this provides inputs from chains of different lengths all ending in σ. We add together all the gradients of these sample chains so that our gradient estimator is:

$$\nabla \hat{\pi}(\sigma) = \sum_{i=0}^{\omega-1} p_1(\mathbf{a}^i, \sigma) \nabla_\theta \log p_1(\mathbf{a}^i, \sigma) + \quad (12)$$
$$\sum_{\kappa=1}^{\omega-1} \left[\sum_{i=0}^{\omega-\kappa-1} p_1(\mathbf{a}^{i+\kappa}, \sigma) \nabla \log p_1(\mathbf{a}^i, \mathbf{a}^{i+1}) \right.$$
$$\left. \prod_{j=0}^{\kappa-1} p_1(\mathbf{a}^{i+j}, \mathbf{a}^{i+j+1}) \right]$$

We use this estimate of the gradient divided by the estimate of $\hat{\pi}(\sigma)$ from the chain to provide an estimate of the gradient needed in (6) as $\nabla \log \pi(\sigma) \approx \frac{\nabla \hat{\pi}(\sigma)}{\hat{\pi}(\sigma)}$

This method provides estimates of the relative direction of the gradient for each parameter after a small number of steps (50-100) compared to the length of runs for sampling the equilibrium (1000s). For increased stability we also replace the sum in (7) with the natural gradient method described by Riedmiller, Peters, & Schaal (2007).

The FSSAM Simulator

For our experiments we model the dynamics of forest growth using the FSSAM simulator which is used by human planners to explore the effects of different yearly harvest targets. FSSAM takes as input an initial forest state and the desired harvest level for each year in cubic meters. At each time period the simulator uses a deterministic policy to sort the cells, by age or volume, for example. It's cut selection module then attempts to cut enough cells to achieve the desired harvest target for that year while not violating any enabled constraints. Tree growth and other processes are then simulated forward to determine the state of the forest in the following year.

The FSSAM simulator uses detailed knowledge of forest growth patterns and is a tool that practitioners know and trust. We integrate FSSAM into our planning algorithm by interfacing with its cut selection module and using it as a (mostly) black box for the dynamics of the MDP.

Integrating with the Simulator

To integrate our planner with FSSAM we provide an interface to the cut selection module of the simulator to call our policy whenever it needs an action sample. The simulator provides the current state of the forest in its simulation to our policy. MCMC is then run to sample a landscape action $\mathbf{a}$ from the equilibrium of the policy. For cells where $a_c = \texttt{Cut}$ we use the MCMC estimate of the conditional probability of that cell action as the sorting key for cell c.

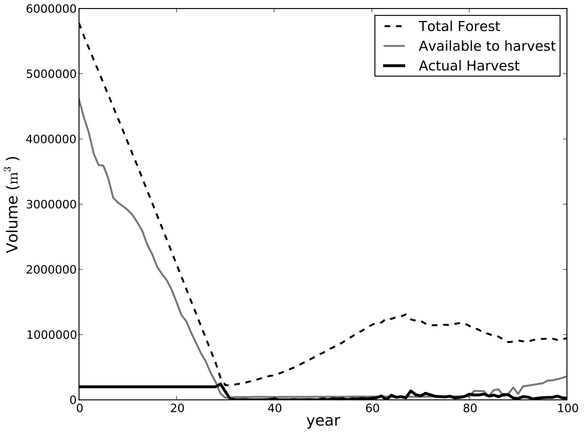

Figure 3: Example run of FSSAM simulator with an unsustainable fixed harvest level of 200,000 m^3 per year. Cells which are too young to be cut or in reserve areas will not be available for harvest.

We return to the simulator a list of all the cells in descending order with the actions having the highest confidence coming first. Cells where $a_c = \texttt{NoCut}$ in our sample are placed at the end of the order and are blocked from being selected by the simulator for cutting.

Evaluation

Figure 3 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.

The goal is to set a harvest level that maintains a healthy forest ecosystem while providing a steady utilization of an important natural resource. While harvest targets do not need to be completely uniform year to year it undesirable to have wild swings in either the amount of lumber harvested or the overall size of the forest.

Setting a fixed target that is lower could avoid the collapse seen here but might underutilize a resource that is an important part of the economy. Given a very low, fixed target it may also be difficult to satisfy spatial constraints using a greedy allocation method.

Experimental Setup

We demonstrate our algorithm on the forest model described earlier by considering the goal of finding regular harvest levels that can be sustained over decades without compromising overall forest health.

We focus on the first 50 years of planning since by that time it can already be clear the harvest level is unsustainable. 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. There are 1880 cells in the forest model we are using.

We use the following set of features:

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

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

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

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

Here we see different types of features which model local state information (features 1 and 2), state information from spatial neighbors (feature 3) and action information from spatial neighbors (feature 4). The initial policy is set with uniform weighting of all features at a moderate/low cut level.

We define three reward models:

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

Each reward model also provides positive reward for the total harvested volume and penalizes adjacent cutting. For runs of the FSSAM simulator we enable adjacency constraint checking.

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 policy from our algorithm was generated after about 18 hours of runtime using 500 MCMC steps for each action sample and 15 gradient update steps running the gradient estimator for 100 samples. The algorithm consults the top 5 stored trajectories in addition to the most recently generated trajectory so $|\mathrm{H}| = 6$.

Results

We ran our planning algorithm using each of the three reward models to generate a policy. The total harvest summed over the entire period for each policy plus two fixed target levels are provided in table 1.

	Total Harvest Volume (m^3)
HAVR	817,456
AVR	1,765,900
HVR	4,915,846
FSSAM-200k	5,000,000
FSSAM-100k	6,038,651

Table 1: Comparison of the Total Harvested volumes summed over 50 years for each policy.

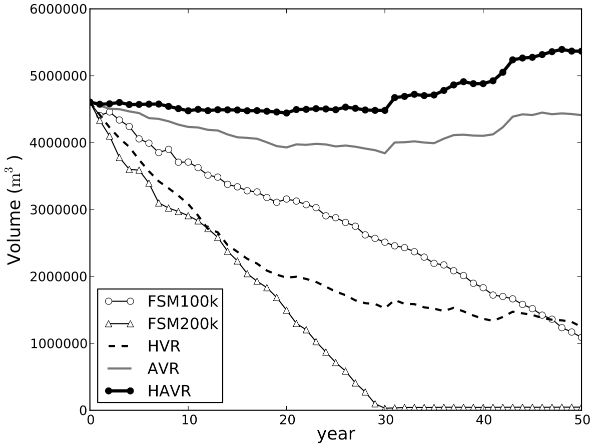

Figure 4: Comparison of total forest volume that is available for harvest under each policy over 50 years.

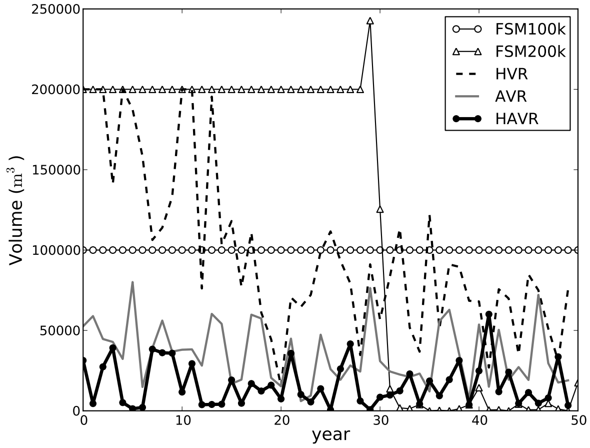

Figure 5: Comparison of total harvested volume under each policy over 50 years.

Figure 4 shows the impact of each of these policies on the overall size of the forest. We can see clearly the unsustainable direction of the two fixed targets of 100,00m^3 and 200,000m^3. The policy obtained by targeting a regular harvest volume using HVR comes in somewhere between these fixed targets with the maintainable size of the forest beginning to flatten out after 50 years.

Using the AVR and HAVR reward functions we learn two policies that attempt to maintain a regular total forest size. Both achieve a sustainable forest volume. The policy using AVR has no penalties for an irregular harvest so it achieves its goal by planning larger periodic larger cuts followed by years of low level cutting. The policy using HAVR on the other hand is trying to optimize for both harvest and total forest regularity and ends up cutting slightly less as a result.

Figures 1 and 2 show the cells which were chosen for harvest by the AVR and HVR policies showing actions within each ten year period with the same colour. The resulting harvest volumes of lumber per are shown in figure 5.

A fixed harvest target near the mean volume cut by the HAVR policy, around 30,000m^3, will yield a similar forest profile. While in this simple case it may not be too difficult to manually search for such a sustainable harvest target, the complexity of this search increases with each new constraint and dimension added to the model. Our policy model and algorithm provide a general way to explore these tradeoffs by specifying values and meaningful features rather than directly tuning the harvest level directly.

Looking at the final policy parameters for the policies in Table 2 we can also see how the weights might be interpreted directly. Each weight describes the correlation between the value of the feature and probability of taking the associated action. 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. We can also determine if a feature is relevant for planning if feature weight does not change over many runs then it is not relevant for planning.

$\theta_f(a)$ Parameters - Initial Values

Action	Age	Max AV	AnyAdj	Volume
0	1.0	1.0	1.0	1.0
1	5.0	5.0	5.0	5.0

$\theta_f(a)$ Parameters - HVR Reward

Action	Age	Max AV	AnyAdj	Volume
0	-2.98	-3.32	-2.03	-3.70
1	0.95	1.64	2.24	2.24

$\theta_f(a)$ Parameters - AVR Reward

Action	Age	Max AV	AnyAdj	Volume
0	1.10	-1.28	-1.75	-3.58
1	9.27	10.13	9.20	10.24

$\theta_f(a)$ Parameters - HAVR Reward

Action	Age	Max AV	AnyAdj	Volume
0	-3.42	-2.77	1.45	-2.63
1	7.75	6.69	6.60	6.13

Table 2: Example of final policy parameters after 10 gradient updates for HVR , AVR and HAVR reward functions. Action 0 is Cut and 1 is NoCut

Related Work

There has been work on general planning approaches that apply to the types of spatial domains we address here. Powell (2010) provides a good summary connecting the various methods used for planning in operations research and artificial intelligence, some of which can deal with rich, factored MDPs. They propose a way to use linear programming to solve very large resource planning problems which may be applicable to our domain. Guestrin, Lagoudakis, & Parr (2002) demonstrate a model based reinforcement learning approach for solving factored MDPs. This is a powerful technique but requires exact inference on the value function so is limited to small problem sizes.

In the particular domain of forestry planning Mathey & Nelson (2007) present a local optimization algorithm for finding a single, approximately optimal plan in the presence of spatial constraints such as biodiversity constraints. Forsell et al. (2009a; 2009b) point out the need for scalable, model-free planning methods that can take advantage of existing simulators for natural resource planning problems. They compare a number of linear programming and reinforcement learning approaches on forestry planning problems using Graph-based Markov decision process (GMDP) models, another form of factored MDP. They use GMDPs to address the problem of minimizing wind damage to trees 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 a policy iteration process.

Our approach attempts to addresses the same need without requiring extensive domain knowledge to engineer an efficient problem representation.

Conclusions

We have demonstrated an approach for casting the complex policy space of spatial planning into a locally-parameterized, spatially-stationary policy. The algorithm requires domain experts to provide the dynamics through (existing) simulators, a set of features to model the relevant state information for the policy and a reward model describing what they value.

An important challenge in environmental planning is to develop methods for exploring, debugging and implementing policies for sustainable planning in complex spatial environments. These methods need to enable different stakeholders in the decision making process to more clearly define their values, explore the outcomes of different policies and understand what a given policy is advising to do and why. The approach shown here contributes to addressing this challenge and hopefully will lead to further discussion in the AI community into how to use the tools at our disposal to make progress in these very challenging planning domains.

Acknowledgements

We would like to thank the anonymous reviewers who provided several helpful comments which helped us to improve the focus of this paper. We would also like to thank the Ministry of Forests, Lands and Natural Resource Operations of the Province of British Columbia for access to their FSSAM tool and datasets. This research was conducted with the assistance of a grant to David Poole from the Natural Sciences and Engineering Research Council of Canada.

References

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and compuatational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Boyland, M.; Nelson, J.; and Bunnell, F. L. 2005. A test for robustness in harvest scheduling models. *Forest Ecology and Management* 207(1-2).

Bremaud, P. 1999. *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues.* Springer.

Crowley, M.; Nelson, J.; and Poole, D. 2009. Seeing the forest despite the trees: Large scale spatial-temporal decision making. In *Proceedings of the 25th Annual Conference on Uncertainty in Artificial Intelligence*.

Forsell, N.; Wikström, P.; Garcia, F.; Sabbadin, R.; Blennow, K.; and Eriksson, L. 2009b. Management of the risk of wind damage in forestry: a graph-based Markov decision process approach. *Annals of Operations Research* 1–18.

Forsell, N.; Garcia, F.; and Sabbadin, R. 2009a. Reinforcement learning for spatial processes. In *18th World IMACS / MODSIM Congress*, 755–761.

Guestrin, C.; Lagoudakis, M.; and Parr, R. 2002. Coordinated reinforcement learning. In *ICML*, 227–234.

Kersting, K., and Driessens, K. 2008. Non-parametric policy gradients: A unified treatment of propositional and relational domains. In *ICML*.

Mathey, A.-H., and Nelson, J. 2007. Decentralized forest planning models - a cellular automa framework. In Gadow, K., and Pukkala, T., eds., *Designing Green Landscapes*. Springer. 167–183.

Powell, W. B. 2010. Merging AI and OR to solve High-Dimensional stochastic optimization problems using approximate dynamic programming. *INFORMS Journal on Computing* 22(1):2–17.

Puterman, M. 1994. *Markov Decision Processes*. New York: John Wiley & Sons.

Riedmiller, M.; Peters, J.; and Schaal, S. 2007. Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 254–261.

Sutton, R. S.; Mcallester, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 12, 1057–1063. MIT Press.