Using Equilibrium Policy Gradients for Spatiotemporal Planning in Forest Ecosystem Management

Mark Crowley

Abstract—Spatiotemporal planning involves making choices at multiple locations in space over some planning horizon to maximize utility and satisfy various constraints. In Forest Ecosystem Management, the problem is to choose actions for thousands of locations each year including harvesting, treating trees for fire or pests, or doing nothing. The utility models could place value on sale of lumber, ecosystem sustainability or employment levels and incorporate legal and logistical constraints on actions such as avoiding large contiguous areas of clearcutting. Simulators developed by forestry researchers provide detailed dynamics but are generally inaccesible black boxes. We model spatiotemporal planning as a factored Markov decision process and present a policy gradient planning algorithm to optimize a stochastic spatial policy using simulated dynamics. It is common in environmental and resource planning to have actions at different locations be spatially interelated; this makes representation and planning challenging. We define a global spatial policy in terms of interacting local policies defining distributions over actions at each location conditioned on actions at nearby locations. Markov Chain Monte Carlo simulation is used to sample landscape policies and estimate their gradients. Evaluation is carried out on a forestry planning problem with 1880 locations using a variety of value models and constraints.

Index Terms—Markov Decision Processes, Reinforcement Learning, Machine Learning, Policy Gradient Planning, Computational Sustainability, Ecosystem Management, Forestry Planning, Optimization

1 Introduction

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 utilities about different outcomes in the world.

As an example, the spatiotemporal planning problem in *Forest Ecosystem Management* [3] is to formulate a policy for deciding whether to cut trees, perform maintenance activities or leave the trees to grow in each one of thousands of locations in the forest, called *cells*, each year. These decisions may need to optimize economic, logistical, regulatory and ecological goals over a horizon of decades or even centuries. These goals could include improving some measure of the overall health of the forest and wildlife; maximizing the revenue from sold lumber; ensuring enough workers and machinery are available to carry out plans; adhering to government regaulations; or maintaining a spatial particular pattern of harvested areas in a landscape.

We address two specific challenges that arise in spatiotemporal planning. The first challenge is that these problems often exhibit correlations across space and time which makes exact solution with traditional methods intractable. We propose using direct policy search methods and a new kind of spatial policy representation to address this complexity.

The second challenge is that in complex planning domains such as forest ecosystem management, the dynamics of the system are often best defined by sophisticated simulation models developed by domain researchers for their own purposes. Using existing simulators is challenging because they may be difficult to modify or access in an analytical form. Generating such simulations can be expensive, so we need to treat simulation data as a precious resource to be used as effectively as possible. We treat the dynamics as a black box that takes actions and states as input and provides a future state of the world as output.

A major goal of any decision support system is to focus the expertise of decision makers onto modelling their domain and defining their values rather than spending time modifying algorithms. This means focussing on modelling features which describe the important aspects of the domain for making decisions and value models which distinguish which outcomes are better than others. A decision support system can then use these features and values to search for a *policy* that is optimal, or at least has a high expected value. A policy in simply a function which takes as input a description of the world and outputs an action to follow or a distribution over actions. The output policy from a decision support system can serve as a suggestion to the decision maker, usually as an input into a broader process planning. Our experiments show how the specified reward model can influence the resulting policy, providing feedback to decision makers about their utilities and their possible

• *M. Crowley is with the Department of Electrical and Computer Engineering, Oregon State University, Corvallis, OR, 97331.*
E-mail: see http://markcrowley.ca/contact

outcomes.

To this end we define an interpretable landscape policy which can represent spatial interdependence between locations and be used for spatiotemporal planning via policy gradient planning. This approach is flexible enough to deal with correlated actions across space as well as 'flow constraints' which tie together actions across time. This paper extends previous work from [13] providing a detailed derivation of the entire algorithm in the appendicies. Appendix C describes and analyses in detail a novel sampling algorithm for estimating the gradient of a Markov chain represented by local parametrized conditional functions such as the cell policy used here. The experiments section includes more extensive experiments and results on the effectiveness of the algorithm in achieving a sustainable dynamic policy in comparison to a fixed harvest schedule as is commonly used in forestry.

Our approach contains three main components: a parametrized stochastic policy for defining actions at a single location in the landscape; an equilibrium landscape policy defined as the equilibrium distribution of a Markov chain defined by the local policy; and a policy gradient algorithm for sampling actions, estimating gradients and improving the landscape policy based on simulated experiences.

The algorithm is evaluated on a sustainable forestry planning problem with 1880 cells using an existing forest simulator in use by the British Columbia Forest Service.

2 Forest Ecosystem Management

This paper uses as a domain example Forestry Ecosystem Management as it is practised in British Columbia, Canada. The total areas considered for planning can range from thousands to millions of hectares. The smallest division of forest for our purposes is a *cell* which can range in size from 1-50 ha. A common problem instance in forestry planning would need to deal with thousands of cells. Each cell has a number of features describing the state of the forest within that cell. Some examples of commonly used features of a cell are:

- Area of cell (1-50ha)
- Volume of trees in cell (0-100,000m^3)
- Dominant tree species (pine, spruce, balsam)
- Presence of tree species (boolean)
- Number of trees (0-50,000)
- Dominant Tree Age (0-250 years)
- Presence or severity of pests (boolean or low/med/high)
- Elevation of cell (0-2000m)
- Bio-climatic zone (the climate classification representing the expected weather conditions in an entire region of the province)

Figure 1 shows a sample area of forest near Kelowna, BC, where the colors indicate the dominant age class of trees in the cell.

The boundaries of each cell are often chosen to enclose a region of fairly homogenous feature values. This means the size and shape of a cell can be very irregular. In the example we use, the number of adjacent neighbours to a cell can range from 2 to 30. Developing methods for defining the boundaries for cell polygons is a difficult and active area of research in its own right [11] so we assume boundary polygons are included as input with other landscape data.

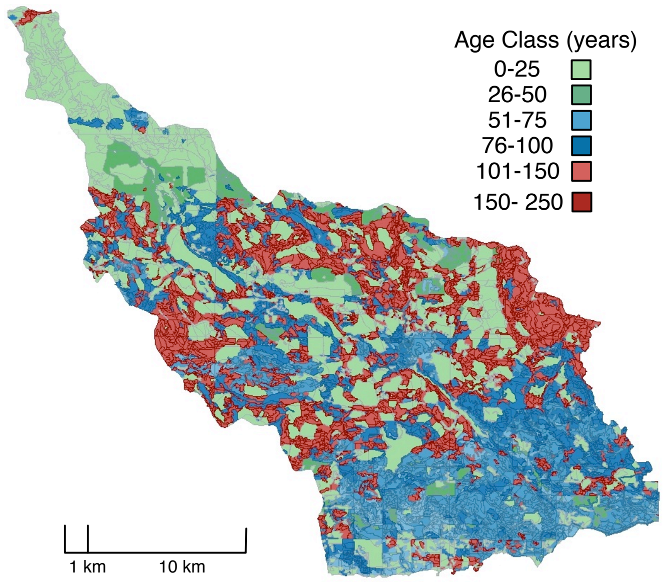


Fig. 1. A region of forest in British Columbia with 1880 cells. Each color indicates the dominant age class, in years, of trees in the cell.

There are a small set of actions available at each cell in each year. For our purposes we model binary actions where a cell can be *clear-cut*, which involves cutting all the trees and replanting, or no action is taken and the forest will be allowed to grow for another time period, which will be the case for the vast majority of cells. In general, other actions could be taken such as thinning of tree density, salvaging of recently dead trees or single tree treatments against pests or disease [41].

Information about the exact utilities and constraints that make up the value model of decision makers in forest management are often hard to obtain in practice as they are usually embedded in scattered reports and studies. See [9] for a good overview of some objectives used in forestry planning. Some types of constraints commonly used in forestry planning are:

- *Annual Allowable Cut (AAC)* : The total volume cut each year may not exceed the AAC set for each region.
- *Even flow* constraint: The volume of timber harvested or the total forest population 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.
- *Green-up* constraint: Cells that have been recently harvested and replanted are in "green-up" phase

and may not be cut for some fixed number of years.

- *Spatial Cutting* constraints: Restrictions on the spatial arrangement of allowed cuts. This is used to limit impact on wildlife habitats and ecosystem health. (e.g. *adjacent cutting constraint* requiring that a cell not be cut while any of its neighbours are being cut or are in green-up).
- *Viewsheds* : Restrictions on the visibility of cuts from populated areas [2].
- *Road Access* : The areas to be harvested must have the necessary road access to carry out the plan.
- *Biodiversity age bounds* : Targets for percentage of the forest which must fall within a specified feature range such as "At most 30% of the forest be old growth" [25].
- *Connectivity* constraints : Requirement that spatially distant habitats for migrating animals have connected corridors of natural landscape [15].

The dynamics of the Forest Management process are well defined by numerous detailed simulations researchers have developed in their domain through extensive study of tree growth patterns, forest ecosystem life-cycles and the epidemiology of pests. The simulators developed by forest researchers include state features describing each cell in the forest and can model the dynamics of tree birth, growth and death, replanting of young trees after clearcutting, killing of trees by pests or fire and the spread of these disturbances over space and time.

One common use of these simulators is to allow human planners to search through different parametrizations manually and view the resulting plan. This allows them to compare outcomes for different paramters or perform sensitivity analysis on existing policies [14]. Because of this, the simulators are often not designed to be accessible programatically or as a simple analytical formula; they need to be treated as a black box of the problem dynamics. In our experiments we use the *Forest Service Spatial Analysis Model (FSSAM)* which is used by the British Columbia Forest Service to analyse the impact of setting AAC targets on sustainable ecological and economic goals.

3 Markov Decision Processes

A *Markov Decision Process (MDP)* [5], [31] 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.
- $r : \mathbf{S} \times \mathbf{A} \rightarrow \Re$: is a reward function which returns the expected immediate reward received by starting in state $\mathrm{s} \in \mathbf{S}$, then taking action $\mathbf{a} \in \mathbf{A}$.
- $\mathcal{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$: is a dynamics model which specifies the probability of transitioning from state s to state s' given action $\mathbf{a}$.

A *trajectory* k is a series of states and actions $\langle \mathrm{s}^0, \mathbf{a}^0, \mathrm{s}^1, \mathbf{a}^1, \ldots \rangle \in \mathrm{K}$. When it is not clear from context we will slightly abuse notation by using k to also index the states and actions in the kth trajectory as in $\mathbf{s}^{k,t}$ and $\mathbf{a}^{k,t}$. The length of the trajectory is called the *planning horizon*. Since there is no particular goal state in our problem we define a discounted infinite horizon planning problem where in practice we consider only up to T steps in the future.

In forestry planning problems, the reward is often best described in the form of the *total expected return*, $\mathcal{R}(k)$, for an entire trajectory such as when we need to account for changes over time as with the even flow constraints discussed in Section 2. Thus we will define reward by the *discounted return* of a trajectory. This is a function, $\mathcal{R} : \mathrm{K} \rightarrow \Re$ defined as $\mathcal{R}(k) = \sum_t \gamma^t r(\mathbf{s}^t, \mathbf{a}^t)$ for discount factor $\gamma \in [0, 1]$ which models the fact that current rewards are worth more than future rewards.

3.1 Scale Considerations

3.1.1 Actions

There are a small set of available actions per cell but $\mathbf{A}$ defines the set of all possible joint, or landscape, actions which is enormous. 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.

3.1.2 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. Some specific features F we want to define for each cell:

- 1:Volume (100-100,000m^3) - the total available volume of trees in the current cell
- 2:Age (0-250 years old) - the average age of the dominant group of trees in the cell
- 3:Maximum adjacent volume (100-100,000m^3) (MaxAV) - the volume of trees in the largest adjacent cell which is available for cutting
- 4:Adjacent cut flag (0,1) (AnyAdj) - true if any adjacent cell is being cut under the current landscape action

Features can model cell state information (features 1 and 2), cell state information from spatial neighbours (feature 3) and cell action information from spatial neighbours (feature 4). The third feature introduces some complexity since it requires aggregate properties to be computed for each cell. The fourth feature introduces is essentially cyclical, mutually tying simultaneous actions at multiple locations together.

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 look to Policy Gradients for planning since this will avoid the need to enumerate states and actions. These types of state features and actions require additional beyond traditional MDPs, this is described in detail in the next section.

3.2 Spatiotemporal Planning using Factored MDPs

The exponential number of states and actions presents challenges for representation since all the states and actions cannot be enumerated. It also presents a challenge for defining a policy with a reasonsable number of parameters which can be optimized efficiently. We address both these problems using a factored representation for states, actions and policies. Each location has a state describing the conditions, an action to be taken and a policy function which defines a distribution over these actions given the local state. We begin by defining this facotrization formally as a *factored MDP* and then deal with the more challenging problem of defining truly spatial factored policies in section 4 and using those for policy gradient (PG) planning.

Let C be the set of all cells. Each cell, $c \in C$, 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. Variables denoted in **bold** indicate a factored state or action that contains information for each cell. We now define a *factored Markov decision process* $\langle \mathbf{S}, \mathbf{A}, r, \mathcal{T} \rangle$ where r and $\mathcal{T}$ are as before but we modify the state and actions definitions:

- $\mathbf{S}$ is S^C, the set of *landscape states*. A particular landscape state is denoted as $\mathbf{s} \in \mathbf{S}$ and the state at a particular cell c is denoted s_c. A cell state, s_c, is defined by a set, F, of features.
- A feature is a function $f_c(\mathbf{a}_{-c}, \mathbf{s}) : C \times \mathbf{A} \times \mathbf{S} \to [0, 1]$. Note, that the features $f_c(\mathbf{a}_{-c}, \mathbf{s})$ can potentially include information about actions and states at all other cells $\mathbf{a}_{-c}$, but in practice they will be limited to actions and states of cells in the spatial 'neighbourhood' of c.
- $\mathbf{A}$ is A^C, the set of *landscape actions*. A particular landscape action is denoted as $\mathbf{a} \in \mathbf{A}$ and the action at a particular cell c is denoted $a_c \in A$.

The factored form of this MDP is similar to other factored models such as the factored MDP described by [8] as well as the multi-agent factored MDP in [17].

3.3 Solving Factored MDPs with Policy Gradient Planning

Solving an MDP generally involves finding a policy that maximizes the expected value .

$$\mathcal{V}^{\pi}(\mathbf{s}) = \mathbb{E}_{\pi}\left[\mathcal{R}(k)\right] \tag{1}$$

There are a wide range of methods in the field of *Reinforcement Learning (RL)* [37] for solving MDPs using only generated experiences and rewards to guide learning, see [39] for a recent overview. 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 which corresponds well with the reward models and simulated dynamics we have access to in ecosystem management problems. One of the most general RL methods is *Generalized Policy Iteration*, also called the *Actor-Critic* method[37]. 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.

Direct policy search algorithms skip the evaluation step entirely and directly modify the policy. Essentially, these algorithms learn a policy by focussing on *how to act* in a given state rather than the *value of acting* in every possible state. This is especially useful for solving very large MDPs where the value function cannot be respresented. *Policy gradient(PG)* planning is a direct policy search approach implementing Generalized Policy Iteration. PG planning contains two main steps: *generating* samples from the policy and *updating* the policy. Instead of evaluating the full value of the current policy, only a *direction* is computed in which to update the policy parameters iteratively based on simulated experience. Policy gradient planning is an active area of research with early work by [42] with his REINFORCE method and more recently by many others [38], [33], [21], [4].

To begin, we define a *stochastic parametrized policy* as a function, $\pi : \mathbf{S} \times \Theta \to \delta(\mathbf{A})$, from states to a distribution over actions given some parameters $\theta \in \Theta$. The parameters of the policy are a ρ-tuple of real numbers.

Given a stochastic policy and the dynamics model we can express the probability of a trajectory as:

$$p(k|\theta) = p(\mathbf{s}^0) \prod_{t=1}^{\mathrm{T}} \mathcal{T}(\mathbf{s}^t|\mathbf{s}^{t-1}, \mathbf{a}^{t-1}) \pi(\mathbf{a}^{t-1}|\mathbf{s}^{t-1}, \theta) \tag{2}$$

We want to look at problems where the reward is only available after the entire trajectory is complete, so we have $\mathcal{R}(k)$ but not $r(\mathbf{s}^t, \mathbf{a}^t)$. To achieve this, the value function from (1) can be restated as

$$\mathcal{V}^{\pi} = \sum_{k \in \mathcal{K}} p(k|\theta) \mathcal{R}(k) \tag{3}$$

where $\mathcal{K}$ is the set of all possible trajectories for the fixed start state $\mathbf{s}$.

From the value function in equation (3) we can see that:

$$\begin{aligned} \nabla_{\theta} \mathcal{V}^{\pi} &= \nabla_{\theta} \sum_{k \in \mathcal{K}} p(k|\theta) \mathcal{R}(k) \\ &= \sum_{k} p(k|\theta) \nabla_{\theta} \log p(k|\theta) \mathcal{R}(k) \end{aligned} \tag{4}$$

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

Next we need to compute the gradient of the trajectory probability from (2), $\nabla_\theta \log p(k|\theta)$ which can be simplified as

$$\begin{aligned}\nabla_\theta \log p(k|\theta) = \\ \nabla_\theta \log p(\mathbf{s}_0) + \nabla_\theta \sum_{t=1}^{\mathrm{T}} \log \mathcal{T}(\mathbf{s}^{k,t}|\mathbf{a}^{k,t-1}, \mathbf{s}^{k,t-1}) \\ + \nabla_\theta \sum_{t=1}^{\mathrm{T}} \log \pi(\mathbf{a}^{k,t-1}|\mathbf{s}^{k,t-1}, \theta) \\ = \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \end{aligned} \quad (5)$$

since 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.

The final gradient of the value function is thus:

$$\nabla_\theta \mathcal{V}^\pi = \sum_{k\in\mathcal{K}} p(k|\theta)\mathcal{R}(k) \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \quad (6)$$

$$\approx \frac{1}{|\mathrm{K}|} \sum_{k\in\mathrm{K}} \mathcal{R}(k) \sum_t \nabla_\theta \log \pi(\mathbf{a}^{k,t}|\mathbf{s}^{k,t}, \theta) \quad (7)$$

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

The basic PG algorithm proceeds as follows: the inputs are an initial state s^0, a transition model $\mathcal{T}$ and initial policy parameters θ. The algorithm proceeds with two interacting processes which are iterated until convergence of the gradient or some maximum time is reached. The first process repeatedly generates new trajectories k using the current policy parameters θ and computes the new gradient $\nabla_\theta \mathcal{V}^\theta$. The second process updates the latest gradient to update the policy parameters:

$$\theta' = \theta + \lambda \nabla_\theta \mathcal{V}^\theta \quad (8)$$

where λ is a learning rate that controls the size of the policy update steps. These steps can be carried out alternatively or in parrallel. PG planning can be shown to converge to locally optimal policies[38] but generally performs well globally if the initial policies are reasonable.

One problem that arises in PG planning is the high variance of the gradient estimate based on a small number of trajectory samples. There are various methods used to reduce this variance, one of the most robust of which is the *Natural Actor-Critic* algorithm[28], [34] which is the basis of our approach. The algorithm follows the natural gradient[1] of the policy which has been shown to minimize the variance when updating the policy.

4 Factored Spatial Policies

The remaining challenge now is to define a stochastic policy representation which can model all the types of features from section 3.1.2 including the cyclical fourth type where actions at one location are relevant to actions at other locations at the same timestep. We would also like the policy to use a relatively small number of parameters so it can be initialized and interpreted by humans users and that is appropriate for optimizing using PG planning. We achieve this by extending the stochastic policy defined previously into a policy with two parts, a cell policy and a landscape policy, in order to correspond with the spatial structure of the domain.

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 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. The actions for other cells are set as interventions rather than observations. Using Pearl's causal notation [27] this would be written as $\pi_c(a|do(\mathbf{a}_{-c}), \mathbf{s}, \theta)$, see Chapter 5 of [12] for further discussion of this.

A local cell policy will be defined as a log-linear distribution where the features can model the cyclic dependency of an action at one location on the actions at other locations. Using the local features for a cell and the policy parameters for each feature and action we can define a potential function for each cell:

$$\psi(a, \mathbf{a}_{-c}, \mathbf{s}, \theta) = \sum_f \theta[f, a] f_c(\mathbf{a}_{-c}, \mathbf{s}) \quad (9)$$

This potential function is then used to define a distribution over actions for a single cell as a log-linear distribution:

$$\pi_c(a|\mathbf{a}_{-c}, \mathbf{s}, \theta) = \frac{\exp(\psi(a, c, \mathbf{a}_{-c}, \mathbf{s}, \theta))}{\sum_{b\in A} \exp(\psi(b, c, \mathbf{a}_{-c}, \mathbf{s}, \theta))} \quad (10)$$

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 in policy parameters.

4.1 The Landscape Policy

A *landscape policy* is a distribution over the joint actions taken at all locations in the landscape. In order to define a meaningful landscape policy we need to understand how the cell policies interact. The cell policy π_c is a cyclic definition with each cell depending on the actions at other cells. Thus, (10) does not *directly* define a conditional distribution of an action at a cell given the distribution of actions at other cells. One natural interpretation of a cyclic system such as this is as the equilibrium of a Markov chain [36].

Given a sample ordering, a total ordering of the cells, 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

1. Throughout this section 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.

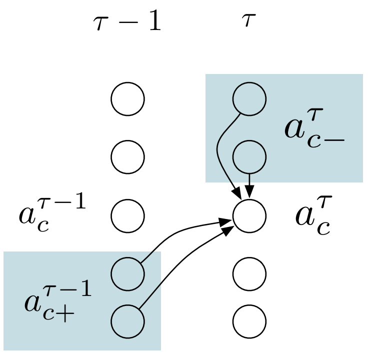


Fig. 2. Visualization of a single sample update in the Markov chain via Gibbs sampling. For an action a_c, the actions at other cells are split into two partitions, $\mathbf{a}_{-c}^{\tau} = \mathbf{a}_{c-}^{\tau} \cup \mathbf{a}_{c+}^{\tau-1}$, based on the sample ordering.

using equation (10). 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 $\mathbf{a}_{-c}^{\tau}$ are split into two partitions as shown in Figure 2. The partitions are the cells before c in the sample ordering, $\mathbf{a}_{c-}^{\tau}$, and the cells after c in the sample ordering, $\mathbf{a}_{c+}^{\tau-1}$. Thus the Markov chain transition probability $\pi_c(a_c^{\tau}|\mathbf{a}_{-c}^{\tau}, \mathbf{s}, \theta)$ based on equation (10) is now in fact $\pi_c(a|\mathbf{a}_{c-}^{\tau} \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta)$.

The equilibrium of the Markov chain can be expressed in terms of the κ-step landscape transition probability $p_{\kappa}^{\theta}(\mathbf{b}, \mathbf{a}, \mathbf{s})$, which is the probability of moving from landscape action $\mathbf{b}$ to landscape action $\mathbf{a}$ in κ steps for fixed policy parameters and state. For conciseness we will often drop the state, parameters and partitions here using simply $\pi_c(a_c^{\tau}|\mathbf{a}_{-c}^{\tau})$ and $p_{\kappa}(\mathbf{b}, \mathbf{a})$.

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:

$$p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^{\tau}) = \prod_{c} \pi_c(a_c^{\tau}|\mathbf{a}_{-c}^{\tau}) \tag{11}$$

For analysis, 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:

$$\Pi(\mathbf{a}^{\tau}) = \mathbb{E}_{\mathbf{a}^{\tau-1}\in\mathbf{A}} \left[p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^{\tau})\right] \tag{12}$$
$$= \sum_{\mathbf{a}^{\tau-1}\in\mathbf{A}} \Pi(\mathbf{a}^{\tau-1}) p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^{\tau}) \tag{13}$$

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 (13) can be extended back κ steps from $\mathbf{a}^{\tau}$:

$$\Pi(\mathbf{a}^{\tau}) = \mathbb{E}_{\mathbf{a}^{\tau-k}} \left[p_{\kappa}(\mathbf{a}^{\tau-k}, \mathbf{a}^{\tau})\right] \tag{14}$$

The probabilities in (14) 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 [10]. 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:

$$\Pi(\mathbf{a}^{\tau}) = \lim_{\kappa\to\infty} p_{\kappa}(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau}) \tag{15}$$

4.2 Estimating the Equilibrium Distribution

For the purposes of sampling and estimating the gradient we will sometimes need an estimator of the equilibrium distribution. The conditional probabilities, $\pi_c(a_c^{\tau}|\mathbf{a}_{-c}^{\tau})$, of the most recently sampled actions can be used to produce an empirical estimator for the marginal distribution over the actions at a single cell a_c summing the Gibbs counts for each action over $\mathcal{M}$. To improve stability we additionaly use a simple form of Rao-Blackwellization[32] by weighting the samples by their conditional probability over some number of previous step ancestors for cell c. The complexity of this algorithm, `sampleStepGibbs`, is $O(C\mathcal{M})$ since it makes an update to each cell and this is performed for $\mathcal{M}$ samples.

5 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 equation (13) to yield the following result (see Appendix A for a full derivation):

$$\nabla_{\theta}\Pi(\mathbf{a}^{\tau}) = \sum_{a^{\tau-\omega}} \nabla_{\theta}\Pi(\mathbf{a}^{\tau-\omega}) p_{\omega}(\mathbf{a}^{\tau-\omega}, \mathbf{a}^{\tau}) + \sum_{\kappa=1}^{\omega} \sum_{\mathbf{a}^{\tau-\kappa}} \Pi(\mathbf{a}^{\tau-\kappa}) \sum_{\mathbf{a}^{\tau-\kappa+1}} \nabla_{\theta} p_1(\mathbf{a}^{\tau-\kappa}, \mathbf{a}^{\tau-\kappa+1}) p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1}, \mathbf{a}^{\tau}) \tag{16}$$

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 (16) 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 (15), 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 (16) can be removed, leaving the following approximation for finite ω:

$$\begin{aligned}\nabla_\theta\Pi(\mathbf{a}^\tau) \approx \sum_{\kappa=1}^{\omega}\sum_{\mathbf{a}^{\tau-\kappa}}\Pi(\mathbf{a}^{\tau-\kappa})\sum_{\mathbf{a}^{\tau-\kappa+1}}\nabla_\theta p_1(\mathbf{a}^{\tau-\kappa},\mathbf{a}^{\tau-\kappa+1})\\ p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1},\mathbf{a}^\tau) \qquad (17)\end{aligned}$$

5.1 Gradient of the Cell Policy

The remaining term needed to estimate the gradient in (17) is the gradient of the one-step transition probability, $\nabla_\theta p_1^\theta(\mathbf{a}^\kappa,\mathbf{a}^{\kappa+1})$, for some pair of consecutive samples in the Markov chain. This is done by working out the gradient with respect to a particular policy parameter $\theta[\alpha,f]$ of the log transition model(see Appendix B for a full derivation):

$$\begin{aligned}&\nabla_{\alpha f}\log\, p_1^\theta(\mathbf{a}^{\tau-1},\mathbf{a}^\tau)\\ &\quad=\sum_c\nabla_{\alpha f}\log\pi_c(a_c^\tau|\mathbf{a}_{-c}^\tau,\mathbf{s},\theta)\\ &\quad=\sum_c g(a_c^\tau,\alpha,\mathbf{a}_{-c}^\tau,\mathbf{s},\theta)f_c(\mathbf{a}_{-c}^\tau,\mathbf{s}) \qquad (18)\end{aligned}$$

where

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

5.2 The Combined Gradient

The gradient of the landscape policy $\nabla_\theta\Pi(\mathbf{a}^\tau)$ from equation (17) can now be derived using $\nabla\ f(x) = f(x)\nabla\log\ f(x)$ on the gradient from (26).

$$\begin{aligned}\nabla_\theta\Pi(\mathbf{a}^\tau) &\approx \sum_{\kappa=1}^{\omega}\sum_{\mathbf{a}^{\tau-\kappa}}\Pi(\mathbf{a}^{\tau-\kappa})\\ &\qquad\sum_{\mathbf{a}^{\tau-\kappa+1}}p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1},\mathbf{a}^\tau)\nabla_\theta p_1^\theta(\mathbf{a}^{\tau-\kappa},\mathbf{a}^{\tau-\kappa+1})\\ &=\sum_{\kappa=1}^{\omega}\sum_{\mathbf{a}^{\tau-\kappa}}\Pi(\mathbf{a}^{\tau-\kappa})\sum_{\mathbf{a}^{\tau-\kappa+1}}p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1},\mathbf{a}^\tau)\times\\ &\qquad p_1(\mathbf{a}^{\tau-\kappa},\mathbf{a}^{\tau-\kappa+1})\nabla_\theta\log p_1^\theta(\mathbf{a}^{\tau-\kappa},\mathbf{a}^{\tau-\kappa+1})\\ &=\sum_{\kappa=1}^{\omega}\sum_{\mathbf{a}^{\tau-\kappa}}\Pi(\mathbf{a}^{\tau-\kappa})\sum_{\mathbf{a}^{\tau-\kappa+1}}p_1(\mathbf{a}^{\tau-\kappa},\mathbf{a}^{\tau-\kappa+1})\\ &\quad p_{\kappa-1}(\mathbf{a}^{\tau-\kappa+1},\mathbf{a}^\tau)\sum_{c\in C}g(a_c^{\tau-\kappa+1},\alpha,\mathbf{a}_{-c}^{\tau-\kappa+1},\mathbf{s},\theta)f_c(\mathbf{a}_{-c}^{\tau-\kappa+1},\mathbf{s})\end{aligned} \qquad (19)$$

5.3 Estimating the Gradient

Since the distribution $\Pi(\mathbf{a})$ is unknown, (19) 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:

$$\begin{aligned}&\nabla_\theta\Pi(\mathbf{a}^\tau)\\ &\approx\sum_{\kappa=1}^{\omega}\mathbb{E}_{\mathbf{a}^{\tau-\kappa}\to\mathbf{a}^\tau}\Bigg[\sum_{c\in C}g(a_c^{\tau-\kappa+1},\alpha,\mathbf{a}_{-c}^{\tau-\kappa+1},\mathbf{s},\theta)\\ &\qquad\qquad f_c(\mathbf{a}_{-c}^{\tau-\kappa+1},\mathbf{s})\Bigg] \qquad (20)\end{aligned}$$

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, `gradEstimate` using stochastic simulation can be used to perform this estimation (20). `gradEstimate` runs a single Markov chain of length ω and collects statistics for each state and action encountered to estimate its likelihood as well as using the conditional probabilities to compute the local policy gradients for $g(.)$ in (20). The complexity of a single gradient estimate using this algorithm is $O(\omega C+\omega^2)$. The algorithm is described more fully in Appendix C.

6 The Equilibrium Policy Gradient Planning Algorithm

The *Equilibrium Policy Gradient* (`EPG`) algorithm, shown in Figure 9, is a policy gradient planning algorithm that brings together the components for policy sampling, gradient estimation and simulation.

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

- `sampleSimPlanner` : $\mathbf{S}\times\Theta\to\mathcal{K}$ - 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 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 `sampleStepGibbs` algorithm
- $\mathcal{R}:\mathcal{K}\to\Re$: a reward function returning the expected, discounted reward for a trajectory $k\in\mathcal{K}$.

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.

Stage II first selects a subset of the current trajectories to use for computing the gradient. `topTrajectories`

Algorithm: $\texttt{EPG}(s_0, h, \texttt{sampleSimPlanner}, \mathcal{R}, \mathcal{M}, \omega)$

```
$\mathrm{K} = \emptyset$;    $\theta = [\texttt{random}]^{F \times A}$    $G_\theta = [0]^{h \times |\theta|}$
repeat
  $k = \texttt{sampleSimPlanner}(\mathbf{s}_0, \theta, \mathcal{M})$ {Stage I}
  $K = K \cup k$
  $H = \texttt{topTrajectories}(\mathrm{K}, \mathcal{R}, h)$ {Stage II}
  for all $k \in H$ do
    $G_\theta[k] = \sum_t \texttt{gradEstimate}\ (\mathbf{s}^{k,t}, \mathbf{a}^{k,t}, \theta, \omega)$
  end for
  $[\delta\theta \quad \mathcal{V}^\pi] = \texttt{naturalGradient}\ (G_\theta, \mathcal{R}(H))$
  $\theta = \theta + \lambda\delta\theta$
until termination condition reached
return $\theta$
```

Fig. 3. Pseudocode for the Equilibrium Policy Gradient algorithm.

chooses the h trajectories in K with the highest reward; the set H stores this history. `gradEstimate` then estimates the gradient of the policy over all trajectories in H with respect to the current policy parameters θ.

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 natural gradient[1] 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[28]. The `naturalGradient` function computes the natural gradient of the value function as shown in [34] 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.

6.1 Complexity

At a high level the EPG algorithm has two stages: sampling a new trajectory using the current policy and updating the policy based on the natural gradient of previously sampled trajectories. For a planning run with $\mathbf{K}$ iterations, using a history of size h to estimate the gradient, the complexity of EPG in terms of these two stages is:

$$O(\texttt{EPG}) = \mathbf{K}(O(I)) + \mathbf{K}h(O(II))$$
$$O(I) = O(\texttt{sampleSimPlanner})$$
$$O(II) = O(\texttt{gradEstimate}) + O(\texttt{naturalGradient})$$

In more detail, the EPG algorithm is composed of three functions with the following complexities:

$$O(\texttt{sampleSimPlanner}) \propto \mathrm{T}(\mathbf{Z} + \mathcal{M}CAF) + \mathcal{R}$$
$$O(\texttt{gradEstimate}) \propto \mathrm{T}(\omega C + \omega^2)$$
$$O(\texttt{naturalGradient}) \propto (FA)^2 h + \frac{(FA)^3}{3}$$

where $\mathbf{Z}$ is the complexity of a single step of the domain simulator.

This makes the full complexity of EPG:

$$O(\texttt{EPG}) = \mathbf{K}h\mathbf{T}\omega^{\mathbf{2}} + \mathbf{K}h\mathbf{T}\omega\mathbf{C} + \mathbf{K}h^2(FA)^2 + \mathbf{K}h\left(\frac{(FA)^3}{3}\right) + \mathbf{K}(\mathrm{T}(\mathbf{Z} + \mathcal{M}CAF) + \mathcal{R})$$

Note that this complexity could be improved by running different parts of the algorithm in parallel 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. However one element beyond our control is the dependence on Z the speed of the external simulator.

7 Evaluation

We evaluated the EPG algorithm 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. Using the `EPG` algorithm these targets are defined by three different reward models.

7.1 Reward Models

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

Each reward model also provides positive reward for the total harvested volume and has a spatial constraint penalizing adjacent cutting. For each cell, `AdjCut` is 1 whenever in the current landscape action the cell c is cut and some adjacent cell is also cut, indicated by the `AnyAdj` feature for c being true. The adjacency penalty `AdjPen` is the average over all timesteps of the sum of `AdjCut` for all cells.

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:

$$\texttt{AvailableDev} = \sqrt{\frac{1}{T}\sum_t (\texttt{AvailVolume}_t - \mu_{AV})^2}$$

$$\texttt{HarvestDev} = \sqrt{\frac{1}{T}\sum_t (\texttt{HarvestVolume}_t - \mu_H)^2}$$

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

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

where the constants $w_{AV} = .3, w_H = 1$ were set through trial and error to adjust for the different scales between the available and harvest volumes. The constant $w_{ADJ} = 500$ is set to make one adjacent cut violation roughly equivalent to one standard deviation from the mean harvested volume.

- 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

7.2 Experimental Setup

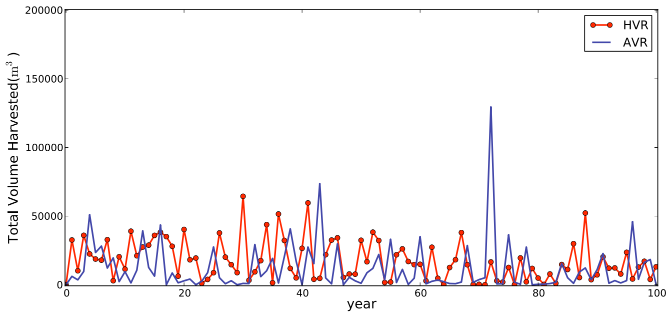
There are 1880 cells in the forest model we are using and the planning horizon will be 100 years, which allows enough time for regrowth of trees cut early on. Actions will be binary, $A = \{\texttt{Cut}, \texttt{NoCut}\}$, where Cut removes all trees and replants new trees and NoCut puts off any activity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

A sustainable policy will need to cut relatively little each year compared to the size of the available forest. 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 $\theta[f, \texttt{NoCut}] = 5.0$ 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. To save computation, the algorithm consults the top 5 stored trajectories in addition to the most recently generated trajectory so that 6 trajectories are used for computing the gradient at any time, that is $h = 6$.

7.3 Analysis



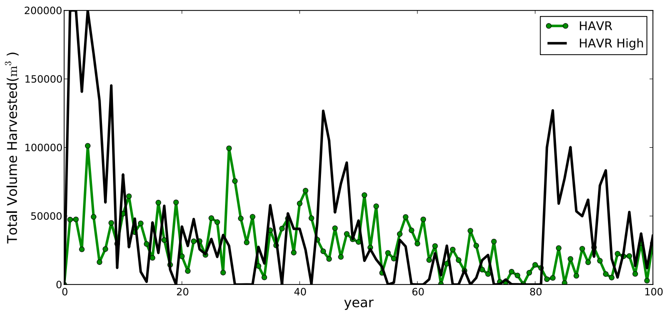


Fig. 4. Harvest flows of policies computed using the AVR, HVR, HAVR and HAVR-High value models.

We analyse the behaviour of the algorithm by looking at a set of typical policies resulting from using each of the value models. 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 4 shows the volumes actually harvested under the four policies. The first thing we notice is that the harvest level for all four policies are quite irregular. Some variation in yearly harvest is unavoidable since the policies are not specifying partial cuts of cells and the number of cells being cut is quite small, in the range of 5 to 30 cells at each time step. So there are only so many discrete harvest levels that are available. Also, for all policies but HVR, which has the lowest harvest variance by far (see table 1), variance of the

Policy Under Value Model	std(Harvest)	std(Available Forest)	Total Harvest Volume (m^3)	Mean Yearly Harvest (m^3)
AVR	18,065	411,085	1,208,827	12,088
HVR	14,422	248,920	1,751,652	17,517
HAVR	20,309	224,212	2,923,499	29,235
HAVR-High	46,699	212,070	3,973,170	50,986

TABLE 1
Each reward model results in a harvest policy, shown are the standard deviation from the mean for the harvest and available forest flows, total volume harvest over 100 years and the mean yearly available forest volume.

total available forest is directly penalized. In essence, we are exchanging regularity in harvest level for regularity in the total mature, unreserved forest population. The result of this can be seen clearly in Figure 5 where the dashed lines show the available harvest level for a simple fixed harvest policy using the FSSAM simulator which uniformly harvests the mean yearly harvest of the associated learned policy.

Comparing the behaviour of the three reward models to each other recall that the difference between the three models is the penalty applied for deviation from an even flow for the harvest level or the available forest level. 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 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.

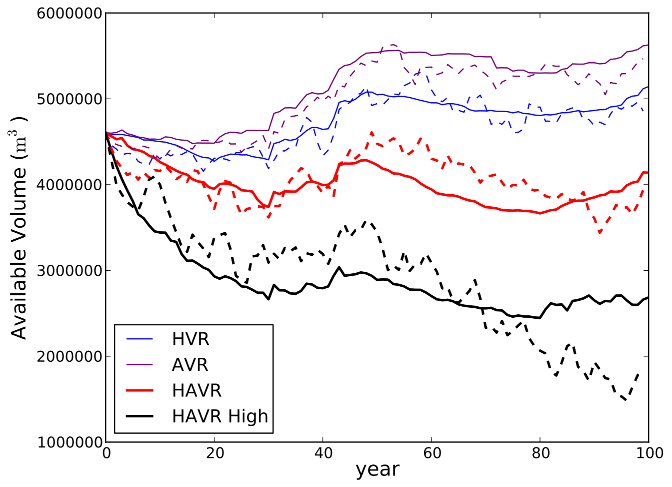


Fig. 5. Available forest volume under policies computed using the AVR, HVR and HAVR value models. Dotted lines show available forest level with a fixed yearly harvest equal to mean yearly harvest of learned policies.

Figure 5 shows the available volume in each year in a 100 year period 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. Each of the learned policies are smoother compared to a fixed harvest level corresponding to the mean yearly harvest of each policy, displayed as a dotted line of the same color in figure 5.

Comparing the learned policies to each other we see that HVR has a more regular available volume profile than AVR even though AVR penalizes deviation from irregular available flow (see Table 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, 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 (see Table 1). HAVR-High performs a very large harvest approximately every 40 years, which corresponds roughly with the time for some trees to grow to maturity in this simulation. The value gained from an increase in the total volume harvested and high regularity of the available forest for most of the period compensates 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 on 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.

$\theta_f(a)$ 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 2
Initial parameter settings for and final policy paramters of the four learned policies.

8 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.

The parameters of these spatial policies can be interpreted directly given a causal interpretation of the conditional distributions of the local cell policies. Table 2 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 (10). Each weight describes the correlation between the value 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 means 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.

The map in Figure 6 shows another example of a way to visualize a policy. This map shows the cells cut under the `HAVR-High` policy; cells cut in each ten-year period have the same color. A policy maker could look at these maps to get an idea of what each policy does over time and to refine their value model or explore specific sets of actions in simulation or submitted into an interactive planning tool. Richer visualizations for these kinds of policies are an open area of future research.

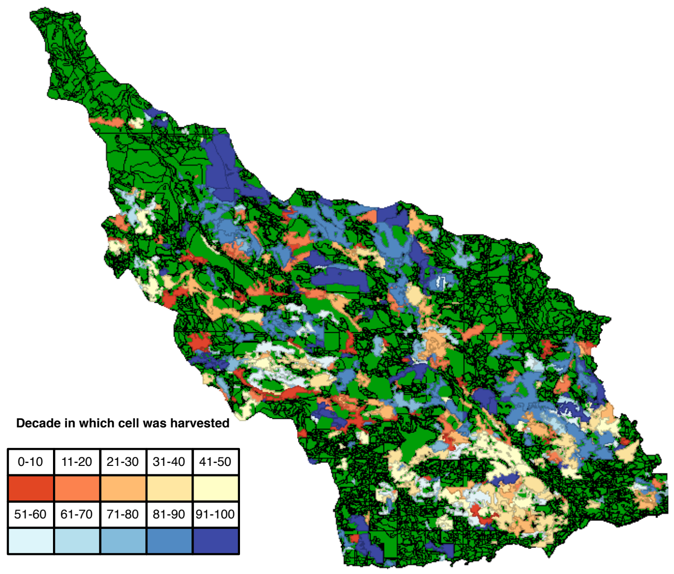


Fig. 6. Map of Harvest using `HAVR-High` policy. Each color indicates in which decade from the starting year the cell was cut.

9 Related Work

One of the broad goals of this research has been to find ways to use technology to free decision makers to think more about values rather than optimization methods. There is a vast literature on optimization methods in Artificial Intelligence, Operations Research and other fields to choose from. When choosing an optimization algorithm to use for a spatiotemporal problem, one of the difficult computational trade-offs to consider is between the number of locations where actions must be taken and the correlation between the actions at those locations. Figure 7 attempts to capture in an informal map the relative applicability of some common optimization methods to large spatiotemporal planning problems along this trade-off. Distance from the top indicates

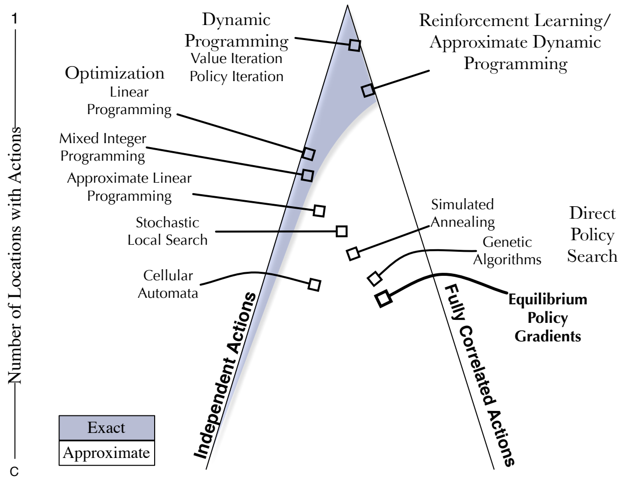


Fig. 7. Informal map of relative power of some common optimization algorithms on spatiotemporal planning problems. Assuming a fixed statespace size the algorithms are mapped out qualitatively according to their relative ability to solve problems with an increasing number of locations (vertical) and the amount of correlation there is between actions due to spatial constraints, rewards and dynamics (distance from either line). Note that for a single location correlation between actions is meaningless and the total number of actions in the MDP increases exponentially as C increases down the graph.

an algorithm which can optimize policies over larger action spaces. The distance of the square from the right edge indicates how relatively correlated those actions can be without causing huge performance problems for the algorithm. The detailed overview of Powell[29] inspired the layout of this map. All these problems can be represented as MDPs where the goal is to choose optimal actions measured against some value model, although researchers in different applied fields often do not use the language of MDPs. Mathematical programming methods can optimize policies for very large numbers of action variables but must assume that the value of an action relates linearly to the number of actions taken. *Linear programming (LP)* can be used to solve MDPs directly[31] but in Forestry LP is commonly used in a more direct way[14] by dividing the landscape into independent areas called “strata” with uniform features and common actions[20]. This type of model makes true spatial constraints difficult to express[22]. *Mixed Integer Programming (MIP)* is another mathematical optimization approach which has been used to model planning for treatment against forest fire by learning a simplified spatial fire spread model and optimizing a set of linear constraints to minimize fire spread[40]. An interesting analysis of optimal spatial actions for the single year problem of forest fire treatment by fragmenting a graph representing spatial spread paths is shown in [35].

Stochastic local search (SLS) methods which iteratively improve and eventually may find optimal solutions are also commonly used in forestry under the general name *meta-heuristics*[3], [30]. *Simulated Annealing (SA)*[23] is the most general of these, allowing many variables to be altered at once to make non-local, and even suboptimal, leaps in action space. This is useful since it can choose the action for each location independently at random and thus search for optimal joint landscape actions. SA can deal with spatial structure quite well but is very inefficient since it is essentially unguided search [7].

We have already discussed the field of *Reinforcement Learning (RL)* which deals with finding optimal policies from experience by iteratively improving an estimate of a value model for visiting any state and taking any actions under the current policy which is itself generally derived from the current value model. In this formulation the action is the full joint action taken at all locations or by many agents and thus any complex relationships between locations, action and states is expressed in the value model. Unfortunately the size of this full value model is exponential in the number of locations in spatiotemporal planning problems.

Our approach falls into the class of *direct policy search* methods which further abstract the action via a parametrized policy function. This policy is optimized through gradient descent of the parameters of the policy weighted against the value obtained when acting under that parametrization. Another popular such method is *genetic algorithms (GA)*. Genetic algorithms have been looked at in forestry and while they can handle some spatial constraints they are very slow since they lack graident information and are guiding their search towards the goal of “reproductive robustness” of the policy parameters as well as optimality[30].

The reward functions that we define here include a flow constraint which has similarities to the mean-variance performance measures discussed by Manor and Tsitsiklis in [24]. They showed that optimizing MDPs for such problems is NP-hard as it requires a model of the history. The EPG algorithm can function in this domain by directly improving the policy and by using stored trajectories. Further research needs to be done into how these results impact planning in sustainability domains where such performance measures are common.

9.1 Factored Planning Approaches

There are many different approaches to research on factored MDPs. [18] defines a *multi-agent factored MDP* which distributes actions to many agents and models each agent’s contribution via local value functions using a graphical coordination graph to model interaction between agents. This has a similar structure to our model, although we do not represent full value functions explicitly. In [19] a model-based reinforcement learning approach for solving factored MDPs is demonstrated that performs very well against other algorithms. However, it requires exact inference on the value function, so it is limited to small problem sizes.

[26] describes 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 uses a small MDP that can be solved exactly, they describe some ways to scale it which could be promising.

[16] compares a number of LP and RL approaches on forestry planning problems using a factored MDP model called *Graphical MDPs*. They point out the need for scalable, model-free planning methods which 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.

A *Cellular Automata* approach to forestry planning is presented in [25] that uses local optimization to find a single, approximately optimal plan in the presence of spatial constraints. This approach uses local conditioning on actions as we do but EPG does not attempt to coorinate actions between locations beyond this.

10 Conclusions and Future Work

The EPG planning algorithm presented here 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 balance off these two extremes and find approximate solutions to very large problems in a feasible manner.

Our method 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.

EPG learns policies that account for spatial correlations amongst actions using a small number of parameters. It produced sustainable harvest policies in a harvest simulation by balancing conflicting components in the value model. Used on a larger scale these kinds of policies could provide insight for policy makers into the relationship between different cutting strategies and sustainability of the forest. For the Forest Ecosystem Management community our approach suggests a way towards richer modelling and planning than is currently the norm. However, in order to be widely usable a number of other advances will be needed, which provides challenges for Computer Science research.

Automated spatiotemporal planning for sustainable ecosystem and resource management problems is still a wide open area and holds many exciting research challenges. Nonstationary policies that adapt over time or have independent components in different epochs are needed to deal with climate change and fluctuating regulations. One way to achieve this is through a hierarchical policy modelled as a general set of parameters applied to all cells and more specific parameters defined for any subset of cells based on their features.

Another challenging and largely unexplored area is how to represent spatial policies in a way that is interpretable and useful. The policy parameters θ of our policy can be interpretted as weights relating features and actions. But richer methods will be needed as the complexity of policies and the size of problems grow.

In terms of performance, a major advantage of direct policy search algorithms such as EPG is that they are easy to parrallelize. 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.

Further performance gains may come from considering parallelization during sampling. For example, blocking could be used as described by Besag[6] to partition cells into independent sets. Gibbs sampling would proceed 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. 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 EPG algorithm is general enough to be applied to other spatiotemporal planning domains. We are currently working on using it for a forest fire domain and invasive species problems which have spatially correlated dynamics as processes spread across space. There are important contributions to be made by applying this or similar approaches to other important real world spatiotemporal planning problems such as infectious disease control, agriculture or fisheries management and urban planning. Any domain where a huge number of actions need to be taken across space by taking into account large data, statistics and simulations over time there is a spatiotemporal planning problem that could be addressed. These spatiotemporal planning problems are often some of the most important decisions being made in our society and we now are beginning to have the tools available to provide assistance to human planners in dealing with these problems more thoroughly than ever before. So, devoting effort to developing better methods for solving these planning problems could yield enormous benefits for society, the economy and the environment.

References

[1] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[2] B. C. Ministry of Forests. *Forest Practices Code of British Columbia Visual Impact Assessment Guidebook*. Forest Practices Branch, Victoria, BC, 2005.

[3] Emin Zeki Baskent and Sedat Keles. Spatial forest planning: A review. *Ecological Modelling*, 188:145–173, 2005.

[4] 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.

[5] R. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[6] J. Besag. Statistical Analysis of Non-Lattice Data. *The Statistician*, 24(3):179–195, 1975.

[7] P. Bettinger, John Sessions, and K. Boston. Eight heuristic planning techniques applied to three increasingly difficult wildlife planning problems. *Silva Fennica*, 36(2):561–581, 2002.

[8] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

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

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

[11] C.A. Coburn and A.C.B. Roberts. A multiscale texture analysis procedure for improved forest stand classification. *International Journal of Remote Sensing*, 25(20):4287–4308, 2004.

[12] Mark Crowley. *Equilibrium Policy Gradients for Spatiotemporal Planning*. Ph.d. thesis, University of British Columbia, 2011.

[13] Mark Crowley and David 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.

[14] Lawrence S. Davis, K.Norman Johnson, Peter S. Bettinger, and Theodore E. Howard. *Forest Management: To Sustain Ecological, Economic, and Social Values*. Series in Forest Resources. McGraw Hill, 4th edition, 2001.

[15] Marvin Eng, Andrew Fall, Josie Hughes, Terry Shore, Bill Riel, and Peter Hall. Provincial level projection of the current mountain pine beetle outbreak. Technical report, NRC-CFS-PFC, 2004.

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

[17] C. Guestrin, S. Venkataraman, and Daphne Koller. Context-specific multiagent coordination and planning with factored MDPs. In *American Association for Artificial Intelligence*, 1996.

[18] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent Planning with Factored MDPs. In *Advances in Neural Information Processing Systems 14*, 2001.

[19] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, pages 227–234, 2002.

[20] 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.

[21] Kristian Kersting and Kurt Driessens. Non-parametric policy gradients: A unified treatment of propositional and relational domains. In *ICML*, 2008.

[22] J. P. Kimmins et al. Possible forest futures: Balancing biological and social risks in mountain pine beetle epidemics. In T.L. Shore, J.E.Brooks, and J.E. Stone, editors, *Mountain Pine Beetle Initiative*, Working Paper 2005-11, pages 33–40, Victoria, Canada., 2005. NRC-CFS-PFC.

[23] S. Kirkpatrick, C.D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[24] Shie Mannor and John Tsitsiklis. Mean-Variance Optimization in Markov Decision Processes. *arXiv:1104.5601 [cs.LG]*, April 2011.

[25] Anne-Helene Mathey and John Nelson. *Designing Green Landscapes*, chapter Decentralized Forest Planning Models - a Cellular Automa Framework, pages 167–183. Springer, 2007.

[26] Jason Pazis and Ronald Parr. Generalized value functions for large action sets. In *Proceedings of the 25th international conference on Machine learning*, 2011.

[27] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition edition, 2009.

[28] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor critic. In J. Gama et. al., editor, *European Conference on Machine Learning*, volume 3720 of *Lecture Notes in Computer Science*, pages 280–291. Springer Verlag, Berlin, 2005.

[29] Warren 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.

[30] 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.

[31] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1994.

[32] C. R. Rao. *Linear statistical inference and its applications*. Wiley, 1965.

[33] 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, 2007.

[34] Martin Riedmiller, Jan Peters, and Stefan 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, 2007.

[35] David B Shmoys and Gwen Spencer. Approximation algorithms for fragmenting a graph against a stochastically-located threat. In *Workshop on Approximation and Online Algorithms*, 2011.

[36] Robert 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.

[37] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[38] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay 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.

[39] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Number 9 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2010.

[40] Yu Wei, Douglas Rideout, and Andy Kirsch. An optimization model for locating fuel treatments across a landscape to reduce expected fire losses. *Canadian Journal of Forest Research*, 38(4):868–877, April 2008.

[41] Roger J. Whitehead and Glenda L. Russo. ”Beetle-proofed” lodgepole pine stands in interior BC have less damage from MPB. Technical report, NRC-CFS-PFC, 2005.

[42] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(2):229–256, 1992.

Mark Crowley Mark Crowley completed his Ph.D. in Computer Science at the University of British Columbia in Vancouver, Canada in 2011. His research focusses on planning in large scale spatiotemporal domains under the existence of uncertainty and spatial constraints especially in fields within Computational Sustainability. He has looked at was compact representation of spatial policies as the equilibria of cyclic causal models and how to perform inference on those models. He also carried out research on shielding the unwanted effects of interventions used to model constraints in Bayesian networks. He received his B.A. in Computer Science from York University in Toronto, Canada and his M.Sc from University of British Columbia.

Appendix A
Derivation of the 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 equation (13) to yield the following result[12]:

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

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:

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

Appendix B
Derivation of the Cell Policy Gradient

The remaining term needed to estimate the gradient in (17) is the gradient of the one-step transition probability, $\nabla_\theta p_1^\theta(\mathbf{a}^\kappa, \mathbf{a}^{\kappa+1})$, for some pair of consecutive samples in the Markov chain. This is done by working out the gradient with respect to a particular policy parameter $\theta[\alpha, f]$ of the log transition model:

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[2].

$$\begin{aligned}
\nabla_{\alpha f} \log\ p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^\tau) &= \nabla_{\alpha f} \log \prod_c \pi_c(a_c^\tau | \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}) \\
&= \sum_c \nabla_{\alpha f} \log \pi_c(a_c^\tau | \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1})
\end{aligned} \tag{21}$$

2. In order to simplify the derivation we will shorten $\psi(a_c^\tau, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta)$ to $\psi(a_c^\tau)$ when the full context is not necessary.

Then we need gradient of the cell log-policy π_c.

$$\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) \\
&= \nabla_{\alpha f} \log \left(\frac{\exp(\psi(a_c^\tau))}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau))} \right) \\
&= \nabla_{\alpha f} \psi(a_c^\tau) - \nabla_{\alpha f} \log \sum_{b_c \in A_c} \exp(\psi(b_c^\tau)) \\
&= \nabla_{\alpha f} \psi(a_c^\tau) - \frac{\nabla_{\alpha f} \sum_{d_c \in A_c} \exp(\psi(d_c^\tau))}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau))} \\
&= \nabla_{\alpha f} \psi(a_c^\tau) - \frac{\sum_{d_c \in A_c} \nabla_{\alpha f} \exp(\psi(d_c^\tau))}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau))} \\
&= \nabla_{\alpha f} \psi(a_c^\tau) - \\
&\qquad \frac{\sum_{d_c \in A_c} \exp(\psi(d_c^\tau)) \nabla_{\alpha f} \psi(d_c^\tau)}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau))}
\end{aligned} \tag{22}$$

The definition of ψ in (9) 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:

$$\begin{aligned}
&\nabla_{\alpha f} \psi(a_c^\tau, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) = \\
&\begin{cases} f_c(\mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}) & \text{if } \alpha = a_c^\tau \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{23}$$

Using (23) and the definition of π_c in (10) then (22) reduces to:

$$\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) - \\
&\qquad \frac{\exp(\psi(a_c^\tau, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta)) f_c(\mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s})}{\sum_{b_c \in A_c} \exp(\psi(b_c^\tau, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta))} \\
&= \nabla_{\alpha f} \psi(a_c^\tau, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) - \\
&\qquad \pi_c(\alpha_c^\tau | \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) f_c(\mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s})
\end{aligned} \tag{24}$$

The gradient in (24) has two cases due to (23).
if $a_c^\tau = \alpha$ then:

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

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

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

We can combine both cases under one notation:

$$\begin{aligned}
&\nabla_{\alpha f} \log\ \pi_c(a_c^\tau | \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) \\
&\qquad = g(a_c^\tau, \alpha, \mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) f_c(\mathbf{a}_{c-}^\tau \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s})
\end{aligned} \tag{25}$$

where

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

The gradient of the log transition model in (21) can now be expressed as:

$$\begin{aligned}\nabla_{\alpha f} &\log p_1(\mathbf{a}^{\tau-1}, \mathbf{a}^{\tau}) \\ &= \sum_c \nabla_{\alpha f} \log \pi_c(a_c^{\tau} | \mathbf{a}_{c-}^{\tau} \cup \mathbf{a}_{c+}^{\tau-1}) \\ &= \sum_c g(a_c^{\tau}, \alpha, \mathbf{a}_{c-}^{\tau} \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s}, \theta) f_c(\mathbf{a}_{c-}^{\tau} \cup \mathbf{a}_{c+}^{\tau-1}, \mathbf{s})\end{aligned} \tag{26}$$

Appendix C
Sampling Algorithm for Estimating the Gradient of the Equilibrium Policy

When computing the gradient, a fixed landscape 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 8; the chains ending in $\mathbf{a}^5$ are highlighted.

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

$$\begin{aligned}\nabla\hat{\Pi}(\sigma) = &\sum_{\tau=0}^{\omega-1} p_1(\mathbf{a}^\tau, \sigma) \nabla_\theta \log p_1(\mathbf{a}^\tau, \sigma) + \\ &\sum_{\kappa=1}^{\omega-1} \Bigg[\sum_{\tau=0}^{\omega-\kappa-1} p_1(\mathbf{a}^{\tau+\kappa}, \sigma) \nabla_\theta \log p_1(\mathbf{a}^\tau, \mathbf{a}^{\tau+1}) \\ &\prod_{i=0}^{\kappa-1} p_1(\mathbf{a}^{\tau+i}, \mathbf{a}^{\tau+i+1}) \Bigg]\end{aligned} \tag{27}$$

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 (27) would match the expectation (20).

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 the following page, 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 components and add them to the existing estimates for all starting points of the chain. Thus, all the terms in (27) 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:

$$\begin{aligned}O(\texttt{gradEstimate}) &= 2\omega C + \sum_{\tau=0}^{\omega} \tau \\ &= 2\omega + \frac{\omega(\omega+1)}{2} \\ &= O(\omega C + \omega^2)\end{aligned} \tag{28}$$

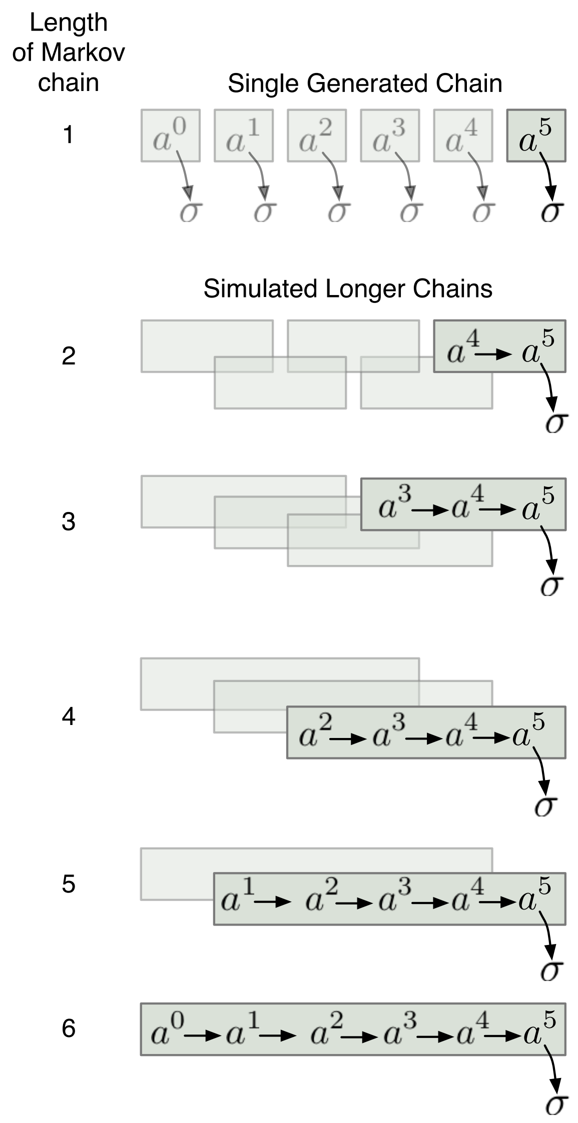

Length of Markov chain
Single Generated Chain
1
a^0 a^1 a^2 a^3 a^4 a^5
σ
Simulated Longer Chains
2
$a^4 \to a^5$
3
$a^3 \to a^4 \to a^5$
4
$a^2 \to a^3 \to a^4 \to a^5$
5
$a^1 \to a^2 \to a^3 \to a^4 \to a^5$
6
$a^0 \to a^1 \to a^2 \to a^3 \to a^4 \to a^5$

Fig. 8. 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.

Algorithm: gradEstimate$(\sigma, \mathbf{s}, \theta, \omega)$

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

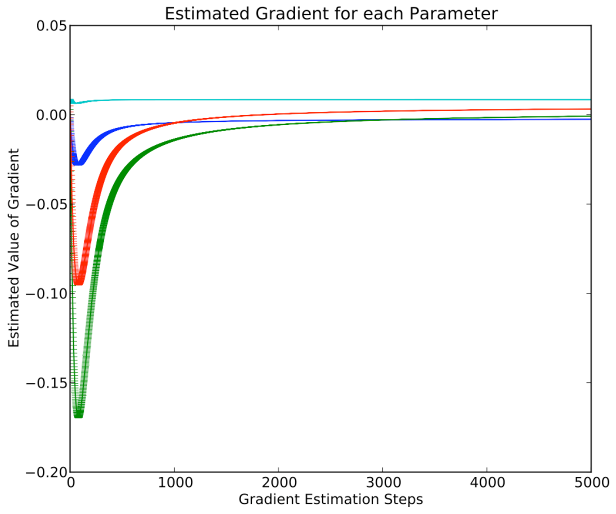
Fig. 9. Estimate gradient of policy for a given state and action.

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.

C.1 Evaluation of the Gradient Estimation Algorithm

While we cannot compare our gradient estimate to the exact gradient for the full size planning problem we performed some analysis to valide the consistency of the estimates coming out of the algorithm. We looked at two states withing a planning run with 1880 cells, binary actions and four independent policy 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 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.



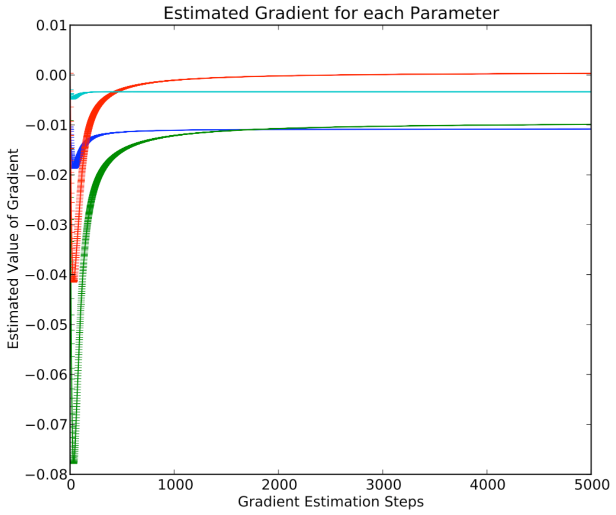


Fig. 10. Two examples of 10 runs of the gradient estimation algorithm on the same state and action. The width of the lines indicate the variance at each sample step amongst the samples.

Figure 10 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.