Stochastic model of lake system invasion and its optimal control: neurodynamic programming as a solution method

Alex Potapov,
Department of Mathematical and Statistical Sciences
and Centre for Mathematical Biology,
University of Alberta, Edmonton, AB, T6G 2G1 Canada
and Department of Biological Sciences, University of Notre Dame,
Notre Dame, Indiana
e-mail: apotapov@math.ualberta.ca

To appear in Natural Resource Modeling, 2009

Abstract

We develop a stochastic model for the process of spread of an aquatic invader in a lake system. The invader is transported along with recreation boats, and boats treatment allows one to control the invader spread. Optimization of invasion and control costs leads to a dynamic programming problem. However, standard stochastic dynamic programming (SDP) algorithms allow us to solve the problem for at most 13-14 lakes, which is significantly less than the size of systems arising in applications. We have developed a new solution technique, which is based upon the ideas of reinforcement learning and neurodynamic programming. This approach allows us to obtain approximate but reasonable control policy for essentially greater lake systems. The resulting numerical technique is comparatively easy to implement, and it can be applied to other spatially extended optimal control problems as well.

1 Introduction

Biological invasions are one of the important topics of modern ecology [27]. The number of successful invaders is small compared to the abundance of local species, but impact of some of them may be close to ecological catastrophe and result in a big economical damage [19, 13, 8, 16]. To diminish the damage, various control and prevention measures can be taken [14]. This in turn raises the problem of modeling and optimizing the related efforts. Such problems appeared to be close to other ones related to optimal resource management and bioeconomics [7, 5]. However, some spatially extended stochastic problems of optimal invader control appear to be
very complicated computationally. Below we consider one of such problems, which required developing of a special solution techniques.

A general scheme of a bioeconomic invasion model includes the following parts:

a) a model for the invader population dynamics;

b) a model for the invader spread;

c) a model for invader prevention and/or control (below we shall use the term “control” for both of them);

d) a cost/benefit functional, accounting for losses due to invasion and control costs;

e) optimization problem solver giving optimal allocation of the control efforts in space and time.

Below this scheme is implemented for a stochastic model of lake system invasion. Our model has been motivated by the spread of zebra mussels (Dreissena Polymorpha) [14, 4, 16] and spiny waterflea (Bythotrephes longimanus) [15] across North American lakes not connected by waterways. In both cases the main way of invader spread is along with the recreation boats [11, 12]. Both invaders bring damages to the ecosystems, zebra mussels cause serious economic problems as well [16]. In both cases the simplest prevention measure is to treat the boats transporting the invaders [18]. Deterministic models for controlling such invasion were considered in [20, 21]. However, most ecological processes are essentially stochastic, and hence a stochastic model may be more appropriate for specific applications, see e.g. [17]. An example of such a model for a lake system has been considered in [14]. However, the model considered there was solvable only for small lake systems containing at most \( N = 7 \) lakes.

The latter limitation is related to the methods of solving stochastic optimal control tasks rather than ecological constraints. The standard tool applied in such cases is stochastic dynamic programming (SDP) [2, 26]. However, the computational complexity of such an optimization task exponentially grows with the number of lakes \( N \) and the number of possible control actions. Roughly speaking, in our case the number of possible ways of invasion spread \( \sim 2^N \), the number of possible control regimes grows even faster, and SDP has to consider all of them simultaneously. This effect is known as a “curse of dimensionality”. Increasing power of computer several times one can increase \( N \) only insignificantly, while ecological applications require to consider systems of 50 and more lakes. The practical answer to this challenge in machine learning [26] is the following: let us develop a technique that gives us a solution reasonably close to optimum in reasonable time rather than the best solution in very long time (infinite from practical point of view). We apply some ideas of reinforcement learning and neurodynamic programming [2, 26] with certain modifications.

Therefore, the problem considered in this paper includes two tasks:

1) develop a stochastic model for optimal control of a lake system invasion;

2) develop a solution technique suitable for a bigger size lake system.

The model presented below is a Markov decision process with continuous controls. The optimized SDP code allows to solve it for \( N = 13 \div 14 \) lakes. For greater \( N \) we apply a combination of Monte-Carlo and neurodynamic programming approach. On several tests with \( N = 13, 30 \) and 50 the method have shown satisfactory performance in reasonable computer time. Since the solution technique does not use the specifics of the invasion model, one can expect that it can be applied for other problems of spatially extended management and control
models.

2 The model

In this section we describe the stochastic model of invasion in a lake system. It uses some concepts of similar deterministic models considered in [20, 21].

2.1 The dynamics of individual lake

We consider a system of \( N \) lakes. Each of them can be in two possible states, invaded and uninvaded. It is convenient to describe the state of lake \( i \) by a variable \( s_i \): \( s_i = 0 \) if the lake is uninvaded and \( s_i = 1 \) otherwise. We also introduce the state \( s \) of the whole system of \( N \) lakes

\[
s = \{s_1, s_2, \ldots, s_N\}.
\]

The total number of different states \( s \) is \( 2^N \).

The state of each lake \( s_i \) is updated each year. For each lake \( i \) there may be an incoming Poisson flow of invader with the intensity \( w_i \) propagules per year. The problem of determining the probability of invasion as a function of a number of arrived invaders has been analyzed by Jerde and Lewis [10]. They show that the probability that the invader population has not established after arrival of \( n \) individuals is

\[
p = \exp (-\alpha' n).
\]

Combining exponential and Poisson distribution [6] one obtains the probability that uninvaded lake \( i \) remains uninvaded in the next year as

\[
p_i (0 \to 0) = \exp (-\alpha w_i), \quad \alpha = 1 - \exp (\alpha'). \tag{1}
\]

The probability for the lake to be invaded is

\[
p_i (0 \to 1) = 1 - p_i (0 \to 0). \tag{2}
\]

Once the lake has been invaded (there is established population of the invader), it remains invaded in the future.

Since the transition probabilities depend on the invader flow, first we consider the model for \( w_i \).

2.2 Invader transport between the lakes

We assume that the major way of invader flow between the lakes is transport of recreational and fishing boats [12]. Let us assume that there is a flow of boats transported from lake \( j \) to lake \( i \) with the mean \( T_{ij} \) boats per year. If the lake \( j \) is invaded \( (s_j = 1) \), each boat can carry a certain number of invader individuals, the average number of invader per boat let us denote by \( \eta \). Then we assume that the invader transport from lake \( j \) to lake \( i \) has Poisson distribution with intensity \( \eta T_{ij} \) invaders per year.
It is convenient to set $T_{ii} = 0$, then total incoming invader flow for the lake $i$ has intensity

$$w_i = \eta \sum_{j=1}^{N} T_{ij} s_j.$$ 

Since $s_j = 0$ for uninvaded lakes, they do not contribute to $w_i$.

In a number of papers [5, 4, 15] the estimates of $T_{ij}$ have been obtained from boat survey data using approximation by gravity models [23].

In this paper we do not consider an external source of the invader. However, it can be introduced by adding one more invaded lake with connections to other lakes equal to the external transport.

### 2.3 Control of the invader and invasion losses

To reduce the amount of the invader it is possible to wash boats at infected lakes after use and/or at uninfected lakes before use. The result of washing can be described by a factor between 0 and 1 that shows how the number of propagules diminish after washing. Let the efforts for washing one boat at the lake $i$ be $x_i$. We assume that the proportional reduction of transported invader organisms is related to the effort as $\exp(-\kappa x_i)$ [20]. The exponential dependence arises because of assumption that the result of two successive independent washings with efforts $x_{ia}$ and $x_{ib}$ are equivalent to a single treatment with the effort $x_{ia} + x_{ib}$. The amount of the propagules transported by one boat from lake $j$ to lake $i$ after the washing diminishes from $\eta$ to $\eta \exp(-\kappa x_j - \kappa x_i)$, and the incoming flow intensity for the lake $i$ becomes $w_i = \eta \sum_j e^{-\kappa x_j} T_{ij} e^{-\kappa x_i} s_j$. The efforts per single boat we assume to be bounded,

$$0 \leq x \leq x_{\max},$$

and therefore the propagule transport cannot go down to zero.

We assume that at the washing checkpoints it may be hard to distinguish boats travelling from invaded to uninvaded lakes from all other boats, and therefore it is necessary to process all boats departing from or arriving to a certain lake. For an invaded lake ($s_i = 1$) it is necessary to process all departing boats. Their mean number per year is

$$D_i = \sum_{j=1}^{N} T_{ji}.$$ 

For an uninvaded lake ($s_i = 0$) on average it is necessary to process all arriving boats,

$$A_i = \sum_{j=1}^{N} T_{ij}.$$ 

We assume that the total values per year have negligible variability, and can be replaced by averages. Therefore, for all $N$ lakes total boat washing efforts per year can be written as

$$E(s) = \sum_{i=1}^{N} x_i \left( s_i D_i + (1 - s_i) A_i \right).$$
We assume that presence of the invader within lake \( i \) causes losses per year equal to \( g_i \). The cost of effort \( E \) we assume to be \( C_1 E + C_2 E^2 \). Therefore to obtain the expression for total losses per year at state \( s \) due to invasion is

\[
C(s) = \sum_{i=1}^{N} (s_i g_i + C_1 E(s) + C_2 E(s)^2).
\]

### 2.4 States of the \( N \)-lake system and transitions between them

Let us enumerate all possible states of the \( N \)-lake system. Totally uninvaded state is not interesting, since there is no invader spread at all, and the system can stay in it indefinitely. Other states we arrange such that first we enumerate all states with one invaded lake, then with two invaded lakes and so on. The last one is completely invaded state (all \( s_i = 1 \)) and its number is \( M = 2^N - 1 \). With this ordering the spread of invasion means steady increase of the state number. To avoid possible confusions, below we shall use small indices for numbering of individual lakes, e.g. \( g_i, i = 1, \ldots, N \), and capital indices for numbering the states of the \( N \)-lake system, e.g. \( s_I \) or \( x_I, I = 1, \ldots, M \). The components of state vectors, which are related to both state of the system and state of a lake, we shall denote by two indices, e.g. \( s_{I,i} \) shows whether the lake \( i \) at state \( I \) is invaded (=1) or not (=0). Similar notation will be used for flows \( w_{I,i} \) and control efforts \( x_{I,i} \). The expression for the incoming flow into lake \( i \) at state \( s_I \) is

\[
w_{I,i} = \eta \sum_{j=1}^{N} e^{-\kappa x_{I,i} - \kappa x_{I,j}} T_{ij} s_{I,j}.
\]

We assume that each year the lakes change their states independently from each other. That is, probability of the lake \( i \) to change its state depends only on its own present state \( s_{I,i} \) and its incoming flow \( w_{I,i} \), but does not explicitly depend on the states of the other lakes, and on their transitions at the same year. This assumption allows us to obtain the transition probabilities between the states of the whole system as the product of the transition probability for each lake:

\[
P(s_I \rightarrow s_K) = \prod_{i=1}^{N} p_i (s_{I,i} \rightarrow s_{K,i}),
\]

where transition probabilities for each individual lake are given by (1) and (2).

**Example.** Let us consider the system of \( N = 4 \) lakes. For such a system there are \( M = 2^4 - 1 = 15 \) states \( s_I = (s_{I,1}, s_{I,2}, s_{I,3}, s_{I,4}) \):

\[
\begin{align*}
N/U & (0, 0, 0, 0) & s_1 = (0, 0, 0, 1) & s_8 = (0, 1, 1, 0) & s_{12} = (1, 1, 0, 1) \\
s_1 = (1, 0, 0, 0) & s_5 = (1, 1, 0, 0) & s_9 = (0, 1, 0, 1) & s_{13} = (1, 0, 1, 1) \\
s_2 = (0, 1, 0, 0) & s_6 = (1, 0, 1, 0) & s_{10} = (0, 0, 1, 1) & s_{14} = (0, 1, 1, 1) \\
s_3 = (0, 0, 1, 0) & s_7 = (1, 0, 0, 1) & s_{11} = (1, 1, 1, 0) & s_{15} = (1, 1, 1, 1)
\end{align*}
\]

Suppose that at the year \( t \) the system is at the state \( s(t) = s_5 \) (lakes 1, 2 are invaded, lakes 3, 4 are uninvaded). What is the probability that at \( t + 1 \) it will be in the state \( s(t + 1) = s_{11} \)?
The transition $s_5 \rightarrow s_{11}$ implies that at the same time four events must take place: 1) lakes 1 and 2 remain invaded; 2) lake 3 become invaded; 3) lake 4 remains uninvaded. The incoming invader flow for uninvaded lakes are

$$w_{5,i} = e^{-\kappa(x_{5,1}+x_{5,i})}T_{i1} + e^{-\kappa(x_{5,2}+x_{5,i})}T_{i2}, \quad i = 3, 4.$$ 

Therefore, the transition probabilities for each individual lake are

1 : $p_1 (1 \rightarrow 1) = 1,$
2 : $p_2 (1 \rightarrow 1) = 1,$
3 : $p_3 (0 \rightarrow 1) = 1 - \exp(-\alpha w_{5,3}),$
4 : $p_4 (0 \rightarrow 0) = \exp(-\alpha w_{5,4}),$

and the probability of transition between

$$P(s_5 \rightarrow s_{11}) = (1 - e^{-\alpha w_{5,3}}) e^{-\alpha w_{5,4}}.$$ 

Note that there are nonzero probabilities of transition from state 5 only to states 5, 11, 12, and 15. The total control effort per year in state 5 is

$$E_5 = x_{5,1}D_1 + x_{5,2}D_2 + x_{5,3}A_3 + x_{5,4}A_4,$$

and the corresponding cost per year is

$$C(s_5) = g_1 + g_2 + C_1 E_5 + C_2 E_5^2.$$ 

2.5 Control policy $\pi$

When the lake system at a certain moment is at a certain state $s_I$, the manager has to decide, what control efforts $x_{I,i}$ have to be applied at each lake. The rule for making such a decision is called a policy. There are various types of policies: deterministic or random, depending only on the current state $s$, and depending on time as well. In this work we shall consider only deterministic policies.

Policy determines controls, $\pi = x(s(t), t)$, the controls determine the transition probabilities $P(s_I \rightarrow s_K)$, the latter determine future states and future costs. The problem of optimal management is to find the policy that minimizes current and weighted future costs.

2.6 What is optimized: average cost of invasion under the given policy

Because of stochasticity it is impossible to optimize one specific realization of the invasion history $s(t)$. Stochastic optimization problem is solved for average total present cost of invasion. For a certain realization of invasion process, that is the sequence of states $s(0)$, $s(1)$, $s(2)$, ..., $s(t)$, $s(t+1)$, ..., and selected control policy $\pi$ one can sum up the costs as

$$J_R(s(0), \pi) = C(s(0)) + \gamma C(s(1)) + \ldots + \gamma^{t-1} C(s(t-1)) + \gamma^t C(s(t)) + \ldots$$ (3)
Here $\gamma = e^{-\rho}$ is a discounting factor, and $\rho$ is a discounting rate used by economists to compare costs related to different moments of time [25]; typical values of $\rho$ are between 0.01 and 0.1. However, for fixed $s(0)$, and $\pi$ different realizations will in general give different sequences of states, and hence different values of $J$. For this reason optimization task is solved for the costs averaged over all possible sequences of states:

$$J(s(0), \pi) = \langle J_R(s(0), \pi) \rangle = \langle C(s(0)) + \gamma C(s(1)) + \ldots + \gamma^t C(s(t)) + \ldots, \rangle$$  \hspace{1cm} (4)$$

How many terms should one take in this sum? Usually managers are interested in the policy for a few years or a few tens of years, up to a certain time $T$ called “control horizon”. If $T$ is big enough, such that $\gamma^T \ll 1$, then $J(s(0), \pi)$ is practically insensitive to what occurs at $t > T$. Then one can account only for terms up to $T$ (effectively setting all costs for $t > T$ to zero), and the resulting policy will be practically independent of $T$, except for a few very last years. However, if $T$ is comparatively small, this approach gives policies strongly depending on $T$. If $T$ is very small, it is optimal not to control at all: anyway, before time $T$ the invasion will not spread essentially, and after $T$ all losses are neglected [20].

Therefore, to obtain a control policy that does not depend on $T$ one has to account for what happens beyond the control horizon. In economics this problem is solved by adding so-called terminal costs $C_T$. In ecology to estimate $C_T$ one has to consider future development of invasion, losses and control costs, that is basically the same problem. For this reason more convenient proves the following step: let us formally set $T = \infty$, solve the resulting control problem, and follow the obtained policy for a finite time. Due to discounting the control horizon will effectively be finite: as mentioned before, if $\gamma^t \ll 1$, costs arising later in time practically do not contribute to $J$. However, mathematically the infinite-horizon problem may be more convenient. For example, in this case the optimal policy typically does not depend explicitly on time [2], that is the controls $x$ depend only on $s(t)$. This allows to replace average $\langle \gamma C(s(1)) + \ldots + \gamma^t C(s(t)) + \ldots \rangle$ by $\gamma J(s(1), \pi)$: we would obtain the same value if we start at $t = 0$ from the state $s(1)$. Averaging can be written explicitly using the transition probabilities $P_{IK}$ between the system state. Eventually, the optimal policy that minimizes the total cost related to each state $s_I$ is a solution of Bellman equation [1, 2]

$$J(s_I) = \min_\pi \sum_{K=1}^{M} P_{IK} (C(s_I) + \gamma J(s_K)) .$$  \hspace{1cm} (5)$$

If one is looking for an explicitly time-dependent policy, as in case of finite-horizon problem, then Bellman equation has to be written for each time step, e.g. as in [14].

2.7 Problem formulation

Eventually we come to the following problem of optimal management. Find the optimal control policy for the system of $N$ lakes, or, equivalently, solve the corresponding infinite-horizon optimal control problem. The required policy $\pi$, that is the set of controls $x_{I,k}$ at each state, satisfies the Bellman equation (5).
For moderate $M$ (5) can be solved iteratively with the help of stochastic dynamic programming (SDP) methods [2, 26]. In this paper the method of asynchronous policy iteration and gradient descending in controls $x_{I,k}$ was used. Since SDP methods are standard and well described in the literature, we shall not consider their details here. Our goal is to bypass their constraints on the size of the lake system $N$. However, in this case (5) cannot be solved exactly, only approximate solutions can be found. Below we consider two such approaches. The first is a model simplification by considering a solvable one with averaged parameters. The second approach is an approximate minimization algorithm for the original model.

3 Simplified model: averaged system parameters and control switching

3.1 A model of $N$ identical lakes

Let us consider a simple case, when all lakes and all connections are identical, that is $g_i = g$, $T_{ij} = T$, $i \neq j$. Due to symmetry, control should depend only on the number of invaded lakes. For the given number of invaded lakes $K$ there should be the same control intensities $x$ at all invaded lakes, and the same control intensities $y$ at all uninvaded lakes. Therefore we can consider all states with the same $K$ as identical, and the problem reduces to one with only $N$ different states.

The incoming invader flow into any of $N-K$ uninvaded lakes is $w_K = K \eta T \exp (-\kappa (x + y))$, and the probability for any lake to stay uninvaded is $p = e^{-\alpha w_K}$. Therefore the probability of transition to a state with $K + k$ invaded lakes is

$$P_{K,K+k} = C_{N-K}^k (1 - e^{-\alpha w_K})^k e^{-\alpha w_K (N-K-k)}.$$

The $k$ lakes to be invaded can be chosen from $N-K$ uninvaded lakes in $C_{N-K}^k$ different ways, the remaining $N-K-k$ lakes stay uninvaded. The control efforts are $E_K (x, y) = K (N-1) Tx + (N-K)(N-1)Ty$. Since the effect of control depends only on $z = x + y$, the efforts $E_K (x, y)$ can be minimized while keeping $z$ constant: it is optimal to choose $x \geq 0$, $y = 0$ for $K < N/2$, $x = 0$, $y \geq 0$ for $K > N/2$. Therefore we obtain a simple switching rule, similar to one in deterministic model [20]: if less than half of the lakes are invaded, control departing boats only at invaded lakes, otherwise control arriving boats only at uninvaded lakes. As $K$ passes the value $K/2$, there should be control switching from invaded to uninvaded lakes. The optimization problem can be formulated only in terms of $z_K$. The resulting Bellman equation is

$$J_K = \min_z \left\{ K g + \min (K, N-K) (N-1) T z + \gamma \sum_{k=0}^{N-K} P_{K,K+k} (z) J_{K+k} \right\},$$

(6)

$K = 1, ..., N-1$. Since the fully invaded system does not need any control, $z = 0$ for $K = N$ and

$$J_N = \frac{Ng}{1-\gamma}.$$
The system (6) can be solved only numerically even in the absence of control, but the solution can be obtained for any reasonable $N$.

3.2 Comparison of simplified and full models

For any model with different $g_i$ and $T_{ij}$ we can average its parameters and associate with it a simplified model. Below we make comparison of results for a full model and its associated simplified one.

The idea of control switching appeared to be useful for the original control task as well. For small $N$, when the full problem can be solved with the help of SDP, it appears that restriction of control to only one type of lakes makes the numerical solution significantly more stable, and practically does not affect the optimal cost values. The switching rule is generalized as follows.

For full inhomogeneous problem let us denote the total traffic between all invaded lakes by $T(i \rightarrow u)$, the traffic from invaded lakes to uninvaded ones by $T(i \rightarrow u)$, and the traffic between uninvaded lakes by $T(u \rightarrow u)$. Then the control at invaded lakes each year has to process $T(i \rightarrow u) + T(i \rightarrow i)$ departing boats, and the control at uninvaded lakes $T(i \rightarrow u) + T(u \rightarrow u)$ arriving boats. It is natural to switch the control from invaded to uninvaded lakes when $T(i \rightarrow u) + T(i \rightarrow i) \geq T(i \rightarrow u) + T(u \rightarrow u)$ or when

$$S_w = \frac{T(i \rightarrow i)}{T(i \rightarrow i) + T(u \rightarrow u)} \geq \frac{1}{2}.$$  

(7)

This switching rule has been used in the results presented below.

We found numerically optimal control efforts for a number of full models, and corresponding average models. Examples are shown in Fig. 1, the details are given in the next section. There is no good way to compare the results graphically, so we have collected some statistics, which are presented in the next section as well, along with other numerical results. For $K = 1$ the control according to the simplified model on average provides about 60% higher costs than the optimal control. For greater $K$ the difference was smaller. On the other hand, an arbitrarily chosen control can give 10-100 times higher costs. Therefore, the simplified model does not provide a very good approximation to the optimal control, but may be useful to (a) quickly provide a coarse estimate for the system as a whole and (b) check the performance of other methods: one may expect several tens of percent enhancement compared to the simplified model control.

4 Approximate solving

When $M$ is too large, the Bellman equation becomes unsolvable even numerically. Its solution requires to store $M$ values of $J$, and $\sim M^2$ operations to calculate the expressions involving $M \times M$ matrix $P_{IK}$. Since $M = 2^N - 1$, this imposes serious restrictions on $N$. We attempted to find an approximate algorithm that could be applied for bigger $N$. 

9
4.1 Existing possibilities

Markov decision processes that are impossible to solve exactly arise in many areas, e.g. in machine learning. Typical approach to the problem is to develop an approximate solution technique. A variety of methods are described in [2, 26]. Their common feature is that they do not solve the Bellman equation. Instead they evaluate the cost (or value) of each state from numerous realizations of the system’s trajectories. The existing policy is updated according to the performance of the system during the previous realizations. For example, if we know possible next states of the system and have some approximate estimates of their costs, then we modify the policy to go to the best next state with higher probability. This bunch of methods is named “reinforcement learning”.

When the number of states is too big, it becomes impossible to estimate the cost function for all states. Then a fitting function \( J(s, w) \) is used: it takes as input characteristics of the state \( s \), and generates the estimate of the corresponding cost. The number of fitting parameters \( w_j \) is usually much less than the number of different states. The algorithms of estimating the costs of states are modified to obtain the best values of the fitting parameters [2, 26]. The policy again can be improved by choosing the best next state. One of the possible choices for the functional form of \( J(s, w) \) is a multilayer neural network, and this is why the corresponding method has been called “neurodynamic programming”.

The described techniques work well when the set of possible control actions is not big. In our case the actions are continuous, and even if we use a set of \( k \) discrete values, the total number of combinations to analyze may be about \( k^{N/2} \) (control at \( \frac{1}{2}N \) lakes, \( k \) possibilities at each). Therefore, we come to the same problem as before, and the solution is also similar: instead of searching for the best action use a fit for the optimal action at each state: \( s \rightarrow x \). This means that we introduce another function \( x(s) = \Phi(s, v) \) with a number of fitting parameters \( v \). The corresponding approximation is called “action network” [2].

However, introduction of an action network creates next problem: how to obtain the best fitting parameters \( v \), that is how to train this action network? The only recommendation in [2] is to minimize the r.h.s. of Bellman equation as a function of \( v \). And we have started right from the attempt to avoid using this equation!

In this situation we have found two ways to bypass using the huge sums in Bellman equation. First, it appears that if we use for \( J(s, w) \) a polynomial in \( s \), then the summation over all states in the Bellman eq. can be done analytically, and its r.h.s. becomes a similar polynomial in probabilities of invasion for each lake, see Appendix A for details. Then the algorithm of training the action network from [2] can be implemented. However, the method did not always converge to a good solution, and sometimes even diverged. For this reason we have tried a totally different approach, which worked much better in our case.

4.2 Approximate solution algorithm

The successful algorithm uses Monte-Carlo estimates of the state cost \( J \) and an action network for generating the controls. Its main parts are the following.

1) Control approximation. Instead of evaluating all \( N \) controls \( x \) at the given state \( s_I \)
at once, we did a fit for a single lake and apply it $N$ times. The inputs contained the state $s$, a few characteristics of the state such as $S_w$ (7), and the characteristics of the lake: its incoming/outcoming traffic and losses $g_i$. The output is the single value $x_{I,i}$, the control at the given lake when the system is at state $s_I$. We have tried two types of approximation: 3-layer neural network [3, 2] and a bounded polynomial approximation. Both of them have shown satisfactory performance. Technical details about the approximations are in Appendix B. Below the set of fitting parameters we denote by $v$ (weights of the neural network or coefficients of the polynomial).

2) **What is minimized** *(target cost $J_{tgt}$)*. What is required for practice is to minimize the total invasion cost (4) at current system’s state $s_0 = s(0)$. If this state is known, it can be used in calculations. If it is unknown, then average over a number of most likely initial states can be done. Then we fix some value of $v$. To evaluate $J$ (4), we evaluate a number of values of $J_R$ (3): we set the initial state $s(0)$ and then simulate the invasion process to obtain a trajectory $s(t)$. This step is repeated a number of times giving several trajectories $s^{(k)}(t)$. For each of them we evaluate the total discounted cost $J^{(k)}_R(s(0), v)$, and then average it to get the estimate of $J(s(0), v)$ (4). This is the estimate of the total invasion cost for the initial state $s(0)$ and the control policy corresponding to current fitting parameters $v$. If necessary, it is averaged over several initial state $s(0)$, which gives the target cost $J_{tgt}(v)$. The number of realizations of the trajectory should be chosen to provide required accuracy of $J_{tgt}(v)$, in our experiments we used the relative accuracy 3%. Better accuracy required essential increase of computations, worse accuracy was bad for the optimization.

3) **Optimization step**. To find a control policy close enough to the optimal we need to find minimum of $J_{tgt}(v)$. The main difficulty lies in the fact that we need to minimize a stochastic function of parameters $v$: averaging that has been done at the previous step reduces magnitude of the noise but does not exclude it. A good review of stochastic optimization techniques can be found in [24]. In brief, there are two basic approaches:

a) Methods of stochastic gradient. This is a generalization of the simplest deterministic minimization technique, the gradient descending. The estimates of gradient for a stochastic function are inaccurate due to the presence of noise. The stochastic gradient technique does not ensure that every algorithm’s step brings improvement, but eventually the minimum is reached as the number of steps goes to infinity provided some conditions are satisfied [24]. There are numerous successful applications of the method. However, in our case it appeared to converge extremely slowly. The reason is in the general properties of neural network approximations: they contain many plateaus where $J_{tgt}(v)$ remains almost constant [3], and most of the differences are due to noise. This property creates difficulties even in case of deterministic minimization [3]. In presence of stochasticity the stochastic part of the gradient is dominating, and the minimization process practically stops.

b) Various versions of random search [24, 22]. In this case a series of randomly selected values of $v$ is chosen, and one looks for $v$ giving the lowest $J_{tgt}(v)$. Most of the random search algorithms are very easy to implement. Their main difficulty is exponential growth of the number of the trial points to ensure that the answer lies close enough to the true optimum. For this reason, theoretically a good convergence of the algorithm is guaranteed only for a small dimension of $v$: 2–5 fitting parameters. However, the property of neural
network approximations mentioned above, which prevents application of stochastic gradient minimization, makes the random search applicable. This happens because the minimum quite often appears to be in one of the mentioned plateaus. Since the plateau is almost flat, it is not necessary to come very close to the true optimal $v$ to obtain a good value of $J_{tgt}(v)$ close enough to the optimal, and hence to obtain a feasible control.

Experiments show that in most cases 5000 to 50000 steps of random search are able to provide good enough minimum of $J_{tgt}(v)$ with the accuracy close to 5%. This result is better and is obtained faster than with the help of other methods. We tried several modifications of random search [24, 22], including simulated annealing. In our tests the so-called “hide and seek” method [22] with zero “annealing temperature” have shown the best performance: we set an initial guess, try points within a certain distance from it, and when find better value, move to it and continue to search for a next improvement in its neighborhood, and so on. Nonzero annealing temperature implies that sometimes nonimproving steps are accepted, but our experiments did not show any improvement of the method performance due to this. To avoid local minima, the main advantage of the annealing methods, we restart the minimization in case there is no significant improvement during long enough period. The details of the algorithms are presented in Appendix B.

4.3 How optimal is approximate solution: tests of the method

The action network tremendously reduces the number of parameters that has to be determined. The total number of controls in all states of our problem is about $\frac{1}{4}N2^N$ (1/4 appears because control is implemented only at one type of lakes). At the same time, the number of fitting parameters $v_i$ is proportional to $\sim N$, varying from 2 $(N + 6)$ to 8 $(N + 8)$ in our experiments. The ratio is proportional to $2^N$, and there are no theorems, which guarantee that such tremendous reduction in complexity should give a solution close to the optimum. Experiments presented in the literature, however, show that a much lower-dimensional approximations can still give good results. Most of the published examples are related to approximations of value function, and one of the most impressive is neural network-based program playing backgammon, see [26] and references therein.

To test performance of the approximate method we have done simulations for a number of 13-lakes systems. $N = 13$ is close to the limit for SDP implementation: one SDP computation runs for a few hours on AMD Opteron or Intel Core 2 processors, while for $N = 14$ it takes about a day. Therefore $N = 13$ allows us to gain statistics of the method performance. On the other hand, at $N = 13$ the number of individual controls $\frac{1}{4}N2^N$ is 140 to 600 times greater than the number of approximating parameters that we used, and we hope that the effects of significant complexity reduction should manifest themselves.

A comprehensive study for all possible problem parameter values is beyond our capabilities. However, the control regimes typically belong to one of the three types: a) $x = 0$ at most states, this typically happens when control is expensive (big $C_1$, $C_2$), or not efficient ($\kappa$ is too small), or losses $g_i$ are small; b) $x = x_{\text{max}}$ in most states, this happens when control is very cheap or very efficient, or losses are big; c) most $x_{i,j}$ take intermediate values. The cases (a) and (b) are not very interesting since quite simple approximations may do the job. The case (c)
is most challenging for the action network, and for testing the method performance we chose parameters that fit into this category.

Calculations for real ecological systems are scheduled for the future work, and here all parameters are taken in nondimensional units. Lakes in the model systems are randomly allocated within a square of the size $10 \times 10$. Each lake is randomly assigned an attractivity “mass” $0 < m_i < 10$ and the value of losses $10 \leq g_i \leq 10^5$. Then the coefficients $T_{ij}$ are calculated from a symmetric gravity model $T_{ij} = m_i m_j d_{ij}^{-2}$, where $d_{ij}$ is the distance between lakes $i$ and $j$. The maximum control effort $x_{\text{max}} = 12$, the cost parameters $C_1 = 1, C_2 = 0.1$. Other parameters were $\alpha = 1, \kappa = 0.5, \gamma = 0.95$ (25 lake systems), $\alpha = 1, \kappa = 2, \gamma = 0.5$ (25 lake systems), $\alpha = 3, \kappa = 2, \gamma = 0.5$ (25 lake systems), and $\alpha = 3, \kappa = 0.5, \gamma = 0.99$ (25 lake systems).

For each system we obtained the following test data:

1) The optimal policy, that is the SDP solution $x_I^*$ and $J_I^*$.

2) The best approximate policy after 50000 steps of random search, that is the best set of fitting parameters $\bar{v}$ and the corresponding controls $\bar{x}_I = \Phi (s_I, \bar{v})$ at every state. Then we have obtained the exact cost of each state under this control $\bar{J}_I$. This calculation has been done for several approximating functions $\Phi$: 2-layer bounded polynomial network of the second order (BP2) and 3-layer neural networks with 2, 4, and 8 neurons in the hidden layer (NN2, NN4, NN8).

3) The best policy for simplified model using averaged $T = \langle T_{ij} \rangle$ and $g = \langle g_i \rangle$. Then using the obtained uniform controls $\bar{x}_{I,i} = x_{K(I)}$, where $K$ is the number of invaded lakes at the state $s_I$, we calculate the cost of each state $\bar{J}_I$ under this policy.

Figure 1 shows two examples (out of 100) of optimal control for the full model and for the simplified one. It can be seen that the difference between simplified and full models may be essential.

To characterize the quality of the approximate solution we calculate the relative errors

$$\xi_{I} = \frac{|\bar{J}_{I} - J_{I}^*|}{J_{I}^*}, \quad \xi_{J,I} = \frac{|\bar{J}_{I} (\bar{v}) - J_{I}^*|}{J_{I}^*}, \quad \xi_{x,I,i} = \frac{|\Phi (s_{I}, \bar{v}) - x_{I,i}^*|}{x_{\text{max}}}.$$ 

In our experiments $J_{I,g}$ was the average cost over all states with a single invaded lake. For the simplified model control $x_K$ we use

$$\tilde{\xi}_{I} = \frac{|\bar{J}_{I (\bar{v})} - J_{I (\bar{v})}^*|}{J_{I (\bar{v})}^*}, \quad \tilde{\xi}_{J,I} = \frac{|\bar{J}_{I} - J_{I}^*|}{J_{I}^*}, \quad \tilde{\xi}_{x,I,i} = \frac{|\bar{x}_{I,i} - x_{I,i}^*|}{x_{\text{max}}}.$$ 

Fig. 2 shows averages of $\bar{\xi}_{I,I}$ and $\bar{\xi}_{J,I}$ over all states with the same number of invaded lakes for four different approximating architectures. Horizontal solid line shows the average of $\langle \bar{\xi}_{I}\rangle$ over all states, horizontal dashed line shows $\langle \bar{\xi}_{J,I}\rangle$, averaged over all states and all lake systems.

To compare approximate and simplified solution, the only check available for bigger $N$, we used

$$\delta_{I} = \frac{\bar{J}_{I (\bar{v})} - \bar{J}_{I (\bar{v})}}{J_{I (\bar{v})}}, \quad \delta_{J,I} = \frac{\bar{J}_{I} (\bar{v}) - J_{I}^* (\bar{v})}{J_{I}^* (\bar{v})}.$$
Figure 1: Two examples of comparison of SDP results for full and simplified models. Panels (a), (b) correspond to one model lake system, panels (c), (d) to the other. \( K \) is the number of invaded lakes. Notation: (a), (c) black dots represent total cost for each state related to the cost of fully invaded state vs the number of invaded lakes. Solid line is the average cost over all states with the given \( K \). Gray dots show cost of each state for control \( \bar{x}_{I,i} \) according to the simplified model. (b), (d) black dots - optimal controls for the full model, solid line shows averaged control for given \( K \), circles show optimal controls for simplified model.

showing the improvement of the approximate solution compared to the simplified model.

The averaged numbers are given in Table 1. It shows that approximate solutions provide much better approximation for the target cost compared to the simplified model: the relative error is 4–5% vs 57%. However, the average error for the simplified model appears to be smaller than for the approximate methods. This can be explained as an effect of discounting. All our simulations begin in the states with a single invaded lake. Due to control, the system can typically stay in this state for quite a long time. Therefore, contribution of more invaded states is discounted, and the states with big \( K \) may practically not contribute at all to \( J_{tgt}(\bar{v}) \). This means that better approximation is more important for small \( K \). Big errors at the states with greater \( K \) are discounted. The consequence of this observation is that after some time, when a few new invasions happen in the modeled system, it may be necessary to repeat the calculations for the new current state to obtain more accurate control estimates.

The best results were obtained for approximations NN4 and BP2, see Fig. 2 and Table 1. However, the latter has only 38 fitting parameters compared to 85 for NN4. Approximation NN8 theoretically should be able to provide better approximation than NN4. Probably for 169 fitting parameters it is necessary to perform more steps of random search to obtain better results.
Figure 2: Test results for 100 model lake systems with \( N = 13 \) lakes and 4 different approximations for action network \( \Phi(s, v) \): (a) bounded polynomial, BP; (b) 3-layer neural network with 2 hidden neurons, NN2; (c) neural network with 4 hidden neurons, NN4; (d) neural network with 8 hidden neurons, NN8. Black circles show the mean relative error \( \langle \bar{\xi}_{J,I} \rangle \) averaged over all states with the given number of invaded lakes \( K \) and over all lake systems. Black vertical bars show the standard deviation for \( \langle \bar{\xi}_{J,I} \rangle \). Gray circles and bars show the same values for the controls \( \tilde{x}_{I,i} \) for the corresponding simplified models, \( \langle \bar{\xi}_{J,I} \rangle \) and its standard deviation. Solid horizontal line shows \( \langle \bar{\xi}_{tgt} \rangle \) averaged over all lake systems. Horizontal dashed line shows \( \langle \bar{\xi}_{J,I} \rangle \), averaged over all states and all lake systems (this value is given in Table 1).

4.4 Application of the approximate method for greater \( N \)

After testing the approximate method, we applied it to 4 model systems of 30 lakes and 4 systems of 50 lakes. For calculations we used only approximation BP2, which allowed much faster calculations than 3-layer neural networks. It also required less fitting parameters \( v_i \), 72 for \( N = 30 \) and 112 for \( N = 50 \). Calculations for one system took 6-8 hours of a parallel code on quad-core Intel Q6600 processor. There are no exact solutions to compare, so we did only the comparison between approximate solution and simplified model control for the target cost, the value \( \delta_{tgt} \) (8). For 30-lake systems the values were \( \delta_{tgt} = 0.54, 0.43, 0.78, \) and 0.64 with the average 0.60. For 50-lake system \( \delta_{tgt} = 0.43, 0.38, 0.47, \) and 0.12 with the average 0.35. This shows that for bigger lake systems the method allows us to obtain a control scheme that gives essentially better performance than the simplified model. We can conclude that the method may be useful in real spatial optimal control problems as well.
Table 1. Average test results for 100 lake systems of $N = 13$ lakes

<table>
<thead>
<tr>
<th></th>
<th>BP2</th>
<th>NN2</th>
<th>NN4</th>
<th>NN8</th>
<th>Simplified</th>
</tr>
</thead>
<tbody>
<tr>
<td>no of parameters $v_i$</td>
<td>38</td>
<td>43</td>
<td>85</td>
<td>169</td>
<td>N/A</td>
</tr>
<tr>
<td>$\langle \xi_{\text{tgt}} \rangle$</td>
<td>0.042±0.002</td>
<td>0.053±0.003</td>
<td>0.041±0.002</td>
<td>0.050±0.002</td>
<td>0.57±0.33</td>
</tr>
<tr>
<td>$\langle \xi_{j,l} \rangle$</td>
<td>0.24±0.90</td>
<td>0.31±2.51</td>
<td>0.19±0.64</td>
<td>0.25±1.47</td>
<td>0.18±0.41</td>
</tr>
<tr>
<td>$\langle \xi_{x,l,i} \rangle$</td>
<td>0.15±0.22</td>
<td>0.12±0.21</td>
<td>0.13±0.21</td>
<td>0.13±0.22</td>
<td>0.30±0.30</td>
</tr>
<tr>
<td>$\delta_{\text{tgt}}$</td>
<td>0.30±0.09</td>
<td>0.32±0.10</td>
<td>0.32±0.10</td>
<td>0.29±0.08</td>
<td>N/A</td>
</tr>
</tbody>
</table>

5 Discussion and conclusions

The results of this paper are of two types: those related to the model and to the solution technique. We briefly discuss both of them.

5.1 Stochastic and deterministic models of aquatic invasions

The problem of optimal control of aquatic invaders has been considered in a number of papers. Deterministic models developed in [20, 21] allowed us to study general properties of such problems and features of their optimal control, some of the results being analytical. As the most interesting findings we can mention the following. a) The importance of time horizon and/or terminal ecosystem value: small time horizon without accounting for terminal value diminishes the optimal level of control, sometimes it appears optimal not to control at all, see Section 2.6 for more details. b) The intensity of control depends on the trade-off between losses caused by invasion and cost of the control procedures. There is a critical value of losses below which it is always optimal not to control at all. Numerical calculations for the models in the present paper, as well as the results of [14], show validity of these conclusions for stochastic models as well. At the same time, in deterministic models it is possible to stop the spread of the invader provided there is Allee effect [21], while in stochastic models it is impossible.

Stochastic model considered in [14] had time horizon up to 25 years with no terminal cost. The model was quite different from considered here: it is developed for specific invader, zebra mussels, it includes more detailed population dynamics and economic estimates. At the same time, it did not consider spatial interactions of the lakes explicitly, and did not allow calculations for more than 7 lakes. The resulting Markov decision problem was solved by standard dynamic programming methods. The economic part of the model was based on economic data from power industry, and the model provided some predictions related to it.

The model considered in this paper is a generalization of the approach developed in [20, 21] to stochastic framework. Compared to [14], it has simpler population dynamics and less detailed economic part but more explicit mechanisms of spatial interactions and invader control. Simplified model of Section 3 is in fact the stochastic generalization of the macroscopic model developed in [20]. It similarly predicts the switching of control regimes from invaded to uninvaded lakes as invasion progresses. The control regimes presented in Fig. 1 have distinct similarities with those given in [20]. However the deterministic macroscopic model appears to be more informative since it admits analytical solutions.
Comparison of the full and simplified stochastic models shows that the latter model can provide control policy with costs about 57% higher than the optimal ones. At the same time this policy appears to be 10–100 times more efficient than an arbitrary policy. Therefore, in absence of other approaches, such a model may be useful even in practical problems. To find better controls one has to consider the full model. For practical ecological applications this requires overcoming technical difficulties, and a number of results for the full stochastic model is related with approximate solution techniques. The computational complexity of stochastic dynamic programming (SDP) exponentially grows with the number of lakes \( N \), and for \( N > 14 \) SDP becomes practically inapplicable. At the same time this spatially explicit optimal control problem cannot be solved with the help of linear programming technique as for models with linear controls [9].

5.2 Approximate solving of stochastic optimal control problem

We use the model problem as a testing framework for developing approximate optimal control techniques in ecological applications. The best results were obtained with the help of combining Monte-Carlo estimates of cost function with stochastic optimization technique. The method has the following advantages:

(a) simplicity of implementation: one needs only a simulator of the controlled process and a program generating controls for the given invasion state (action network);
(b) it becomes possible to obtain good control policies in cases where SDP is inapplicable;
(c) since Monte-Carlo estimates used for cost evaluation, it becomes possible to incorporate additional stochastic factors without big risk to make the model too complicated; one can implement features, for which it is hard to write an explicit expression for transition probabilities and hence to apply SDP;
(d) the method comparatively easily can be implemented on parallel computers.
(e) the accuracy of the method can be increased by increasing the number of steps of random search and by using more complicated approximating action network provided more powerful computers are available. There is enough flexibility to adjust the approximation technique to a specific problem.

The method shortcomings:
(a) the computing time still noticeably grows with the system size: for \( N = 30 \) and 50 calculations may require many hours of processor time. However the dependency on \( N \) is closer to polynomial rather than exponential;
(b) the accuracy strongly depends on the choice of the control approximation function (action network), so each new application may require some work on choosing the best approximation technique most appropriate for the problem in question;
(c) costs are well optimized for the target state and the states that occur in the nearest future, when the discount \( \gamma^t \) is not too small. For the states that are not likely to occur, or which contribution is strongly discounted, the estimate may be very coarse. Therefore, discounting creates a sort of “accuracy threshold”. Consequently, after some time, when the invasion reaches this threshold, the calculations have to be repeated to extend this threshold further.
Comparison with SDP solution for \( N = 13 \) shows that the approximate technique is capable of providing the control scheme with the total cost for the target state only a few percent higher than the optimal one. At the same time an arbitrary control policy may give costs that are 10–100 times greater.

Finally, the approximate solution method can be implemented for optimal control of other spatially extended systems. Examples of such problems can be found in ecology, resource management, disease spread and so on. We plan to develop such applications in future.

5.3 Acknowledgments

This work is a part of ISIS project and is supported by NSF grant DEB 02-13698. Author would like to thank M.A. Lewis and D. Lodge for useful discussions and anonymous referees for helpful suggestions.

6 Appendix A. Reduction of the sum in Bellman equation

For a big lake system the sum in Bellman equation (5) cannot be evaluated. However, if we apply a linear in \( w \) approximation for \( J_I = G(s_I, w) = \sum_{n=1}^{N_A} w_n \phi_n(s_I) \), where \( N_A \) is the number of basis functions \( \phi_n(s_I) \), then

\[
\sum_{K=1}^{M} P_{IK}(x_I) J_K = \sum_{K=1}^{M} P_{IK}(x_I) \sum_{n=1}^{N_A} w_n \phi_n(s_K) = \sum_{n=1}^{N_A} w_n \sum_{K=1}^{M} P_{IK}(x_I) \phi_n(s_K).
\]

Now let us assume that some \( \phi_n(s) \) depends only on \( k < N \) components of \( s \). Then the last sum for this \( n \) can be reduced from \( \sim 2^N \) to \( \sim 2^k \) terms. Let us split \( s_K \) into two parts, (a) components on which \( \phi_n(s) \) depends and (b) components on which it does not depend. Without loss of generality we may assume that \( \phi_n(s) \) depends on \( s_1, \ldots, s_k \), and does not depend on \( s_{k+1}, \ldots, s_N \). The transition probabilities are of the form

\[
P_{IK} = p_1(s_{I,1} \to s_{K,1}) p_2(s_{I,2} \to s_{K,2}) \ldots p_N(s_{I,N} \to s_{K,N}).
\]

Now let us fix \( s_1, \ldots, s_k \) and consider the subset \( \Omega(s_1, \ldots, s_k) \) of all possible states with these first \( k \) components fixed. Each \( \Omega \) contains \( 2^{N-k} \) states. For all terms from \( \Omega(s_1, \ldots, s_k) \) the factor \( \phi_n(s_K) \) has the same value. Therefore we can restructure our sum as a sum over all subsets \( \Omega_\alpha \) and the sum over \( K \) from each subset

\[
\sum_{K=1}^{M} P_{IK}(x_I) \phi_n(s_K) = \sum_{\Omega_\alpha} \sum_{K \in \Omega_\alpha} P_{IK}(x_I) \phi_n(s_K) = \sum_{\Omega_\alpha} \phi_n(s_{K,1}, \ldots, s_{K,k}) \sum_{J \in \Omega_\alpha} P_{IK}(x_I) =
\]

18
= \sum_{\Omega_\alpha} \phi_n (s_{K,1}, \ldots, s_{K,k}) \sum_{K \in \Omega_\alpha} p_1 (s_{I,1} \rightarrow s_{K,1}) p_2 (s_{I,2} \rightarrow s_{K,2}) \ldots p_N (s_{I,N} \rightarrow s_{K,N}) = \\
= \sum_{\Omega_\alpha} \phi_n (s_{K,1}, \ldots, s_{K,k}) p_1 (s_{I,1} \rightarrow s_{K,1}) \ldots p_N (s_{I,N} \rightarrow s_{K,N}) \times \\
\times \sum_{K \in \Omega_\alpha} p_{k+1} (s_{I,k+1} \rightarrow s_{K,k+1}) \ldots p_N (s_{I,N} \rightarrow s_{K,N}).

In the last sum, as \( K \) takes all values from \( \Omega (u_1, \ldots, u_k) \), \( s_{K,j} \) take the values both 0 and 1, and the sum can be factored as

\[
\sum_{K \in \Omega_\alpha} p_{k+1} (s_{I,k+1} \rightarrow s_{K,k+1}) \ldots p_N (s_{I,N} \rightarrow s_{K,N}) = (p_{k+1} (s_{I,k+1} \rightarrow 0) + p_{k+1} (s_{I,k+1} \rightarrow 1)) \ldots (p_N (s_{I,N} \rightarrow 0) + p_N (s_{I,N} \rightarrow 1)) = 1,
\]

because each bracket equals 1. Therefore,

\[
\sum_K P_{IK} (x_I) \phi_n (s_K) = \sum_{\Omega_\alpha} \phi_n (s_{K,1}, \ldots, s_{K,k}) p_1 (s_{I,1} \rightarrow s_{K,1}) \ldots p_N (s_{I,N} \rightarrow s_{K,N}),
\]

and for each \( \phi_i (s_1, \ldots, s_k) \) we have the number of terms equal to the number of different subsets \( \Omega (s_1, \ldots, s_k) \), that is \( 2^k \). The number of different functions depending on such \( k \)-tuplets is \( C_N^k \). For \( k \) small \( C_N^k 2^k \ll 2^N \).

One of the most obvious choices is polynomial form \( \phi_i (s_1, \ldots, s_k) = s_1 \times s_2 \times \ldots \times s_k \). Then in the sum over all subsets only the terms with \( s_1 = s_2 = \ldots = s_k = 1 \) remain, and therefore the sum turns into

\[
\sum_K P_{IK} (x_I) \phi_n (s_K) = p_1 (s_{I,1} \rightarrow 1) \ldots p_k (s_{I,k} \rightarrow 1).
\]

Suppose we have approximation of the order 2, for them there will be one basis function \( \phi_0 = 1 \), \( N \) basis functions \( \phi_i = s_i \), and \( N \cdot (N - 1) / 2 \) functions \( s_i s_j \) with \( j > i \). Note that because \( s_i \) take only values only 0 and 1, \( s_i^2 = s_i \), and hence we should exclude from our consideration all terms containing \( s_i^2 \). So eventually

\[
\sum_{K=1}^M P_{IK} (x_I) J_K = \sum_{K=1}^M P_{IK} (x_I) G (s_K, \mathbf{w}) = \\
w_0 + \sum_{i=1}^N w_i p_i (u_{Ii} \rightarrow 1) + \sum_{i=1}^{N-1} \sum_{j=i+1}^N w_{ij} p_i (u_{Ii} \rightarrow 1) p_j (u_{Ij} \rightarrow 1) = G (\mathbf{p}, \mathbf{w}).
\]

This sum contains only \( 1 + N \cdot (N + 1) / 2 \) terms instead of \( M \), which allows computations for relatively big \( N \). Similarly approximations of the order 3, 4, ... can be constructed. They also have the polynomial form \( G (\mathbf{p}, \mathbf{w}) \).

In our case this simplification did not lead to a useful algorithm for training of action network. However, it may be useful in some other problem.
7 Appendix B. Details of the approximate algorithm

7.1 Action network

Input data. The network should provide controls for the given system state $s$. Instead of generating a vector output, we decided to make a network that gives the control only at a single lake, but takes as its input more detailed information about the current state and the specific lake. Actually, we use two different networks, one for control at invaded lakes, the other for control at uninvaded lakes. The input data for both of them were identical.

The input data $\xi_1,...,\xi_n$ were the following:

$\xi_1 = S_w$, control switching parameter (7). For next 5 parameters we calculate $\eta_2$ to $\eta_6$ as

$$
\eta_2 = \frac{T (i \to u)}{T (i \to i) + T (u \to u)},
$$

see description of formula (7) for notation,

$$
\eta_3 = \frac{\sum s_i g_i}{N-1 \sum g_i},
$$

$$
\eta_4 = N^{-1} \sum s_i.
$$

Parameters $\eta_5$ and $\eta_6$ described specific lake $i$, for which the control approximation is done:

$$
\eta_5 = \frac{\sum_j T_{ji} (1 - s_j)}{D_i}
$$

(traffic from lake $i$ to uninvaded lakes with $s_j = 0$ divided by total outgoing traffic),

$$
\eta_6 = \frac{g_i}{N-1 \sum g_j}
$$

(proportion of losses caused by invasion of lake $i$). Then

$$
\xi_j = \frac{\eta_j}{1 + \eta_j}, \quad j = 2, ..., 6.
$$

Next $N$ parameters $\xi_7,..,\xi_{6+N}$ were the components of $s_j$ multiplied by 0.5 except for $s_i$, which is taken as is (to distinguish the current lake among others if it is invaded). This makes total $N + 6$ input parameters.

3-layered neural networks (NNi). We used a standard 3-layer architecture described e.g. in [3] with $N+6$ input neurons, $n_h = 2, 4, or 8$ middle (hidden) neurons, and one output neuron. The network output is multiplied by $x_{\text{max}}$ to obtain control in the range $[0, x_{\text{max}}]$. The total number of approximating coefficients $v_i$ is $(N+7) n_h + n_h + 1$. $N+7$ appears because there is one free additive parameter for each neuron’s input. To reduce the number of computations, for the sigmoid function we used piecewise linear function analogous to $\sigma (x)$ described below.
Bounded polynomial approximation (BP2). In this case we take function of the form

\[ f(\xi) = \sigma \left( v_0 + \sum_{j=1}^{N+6} v_j \xi_j + \sum_{j=3}^{N+6} v_{N+6+j} \xi_j^2 \right), \quad \sigma(x) = \begin{cases} 
\xi_{\text{max}}, & x \geq \xi_{\text{max}}, \\
x, & 0 \leq x < \xi_{\text{max}}, \\
0, & x < 0,
\end{cases} \]

which is similar to 2-layer neural networks with additional nonlinear terms. In this particular implementation it uses \(2(N + 6)\) fitting parameters. Its generalizations are quite obvious. One of its attractive features is the possibility to fit it to data quite easily (compared to complicated procedure of standard multilayer network training): just a few iterations of a standard polynomial fitting with modified error estimates. In early versions of our code we fitted this approximation to exact controls to check the potential capability of our architecture to approximate the controls. These experiments allowed us to select the function containing minimum number of fitting parameters.

### 7.2 Stochastic minimization algorithm

For stochastic minimization we used a non-annealing version of “hide and seek” algorithm (see e.g. [22]). For the given action network with \(n\) fitting parameters \(v_i\) we do the following.

1. Set an initial approximation \(v_0\), assign \(v = v_0\) and calculate \(J_{\text{tgt}}(v)\).
2. Generate a random \(n\)-dimensional vector \(d\) of unit length. The easiest way is to generate \(n\) normally distributed random numbers and normalize the resulting vector. In our experiments uniformly distributed numbers on \((-1,1)\) also worked well.
3. Generate a random step \(\lambda \in [0, \lambda_{\text{max}}]\). A proper choice of \(\lambda_{\text{max}}\) is very important for convergence rate, and it has to be adjusted experimentally. In our experiments good results were obtained with \(\lambda_{\text{max}} = 6\) for BP2 approximation and \(\lambda_{\text{max}} = 2.5\) for 3-layer neural networks.
4. Calculate \(J_{\text{tgt}}(v + \lambda d)\). If \(J_{\text{tgt}}(v + \lambda d) < J_{\text{tgt}}(v)\), assign \(v = v + \lambda d\), otherwise keep \(v\) the same.
5. If the prescribed number of algorithm steps is not performed, return to 2.

Our experiments have shown that if there is no essential improvement of \(J_{\text{tgt}}(v)\) during last 8000 steps, it is likely that the process is stuck at a local minimum, and it is better to restart the algorithm from step 1. However, this may be different for other problems, and experiments has to be made.

We have done 50000 minimization steps in the experiments described in this paper. Note that this algorithm can be easily parallelized: several trials in steps 2–4 can be done in parallel. For the experiments in this paper a parallel code with 4 concurrent threads has been used. On Intel Core 2 Quad E6600 2.4 GHz processor the acceleration was about 3.6 times. As a result, for \(N = 13\) one experiment typically took 1–2 hours for all kinds of action networks (it depended on the number of Monte-Carlo steps required to reach 3% accuracy of \(J_{\text{tgt}}(v)\)). For \(N = 50\) one experiment required about 6–8 hours with BP2 action network.

It is necessary to note that in this case very useful was big amount of cache in the Core 2 processor, 4MB per each pair of cores. This gave additional acceleration of computations almost 5 times compared to other processors with 512KB or 1MB cache.
References


