Dynamic reserve site selection under contagion risk of deforestation

[1]**Sabbadin, R.,** [1]**C-E. Rabier,** **and** [2]**D. Spring**

[1]INRA de Toulouse, Unite de Biométrie et Intelligence Artificielle
BP 52627 - 31326 CASTANET-TOLOSAN cedex - FRANCE
E-Mail: sabbadin@toulouse.inra.fr
[2]Australian Centre for Biodiversity,
School of Biological Sciences, PO Box 18,
Monash University, Clayton, Victoria 3800 - AUSTRALIA
E-Mail: Daniel.Spring@sci.monash.edu.au

Keywords: *Reserve site selection, Spatial controlled processes, Dynamic programming, Reinforcement learning*

EXTENDED ABSTRACT

Creating networks of protected nature reserves is the primary means of reducing biodiversity loss. The principle focus of the reserve design literature is on determining which sites to reserve to maximise the number of species conserved. To each site is attached species which become conserved when the site is reserved. A good reservation policy is one that conserves as many species as possible. Until recently, most site selection models have been static: they assumed that sites threatened by development would be reserved immediately after the optimal plan is determined. This rarely occurs in practice, often because there are insufficient funds available when the optimal plan is determined, or because not all sites are immediately available to be reserved. But once postponed, the reservation decisions might never be implemented if the targeted sites become developed first. Recently, dynamic models based on stochastic dynamic programming (SDP) have been proposed to solve the reserve selection problem. In these models, unreserved sites can become irreversibly developed each year with a given probability and only a limited number of sites can be reserved each year, due to budget constraints. The problem is to design a dynamic reservation policy that results in the maximum expected number of species conserved at the end of the problem horizon. These models are used to investigate the importance of the timing of selections in conservation programs. Existing models consider a random development pattern where development probability for each site is independent of the development status of neighbouring sites. However, it is more likely that deforestation occurs as a “contagion” process, in which deforestation in a region begins when extensions to road networks make the region economically accessible.

In this paper we propose improved SDP algorithms that make use of a graph representation of the sites network to address the reserve site selection problem under this spatially explicit form. First, we present an already existing model based on SDP and we propose a new, exact, dynamic programming algorithm, with which theoretical and experimental complexities are assessed. Unfortunately, this exact method can only be applied to small problems (less than ten sites), when real-world problems may have hundreds or thousands of sites. So, instead of exact methods, heuristic methods have been proposed in order to solve the reserve selection problem. In this paper we provide a general framework for describing such heuristic methods. Our main contribution is then to propose a new heuristic method, based on a *parameterised reinforcement learning* algorithm. This method allows us to compute a heuristic function by performing and exploiting many simulations of the deforestation process. We show that this method can be applied to problems with hundreds of sites, and we show experimentally that it outperforms the classical heuristic methods in terms of the average number of species which can be conserved.

1 INTRODUCTION

The primary means of reducing biodiversity loss is to create networks of conservation reserves. In most cases, the establishment of a reserve network is a gradual, accretive process, comprising a sequence of land acquisitions through time. One of the reasons for this is that not all sites are available for purchase at the same time Meir et al. (2004); another reason is that funding for site acquisitions at any given time is insufficient to acquire all sites Costello and Polasky (2004). Conservation organisations that build reserve networks over an extended period of time must contend with the risk that sites will be developed before they can be reserved Costello and Polasky (2004); Meir et al. (2004).

Recently, dynamic models based on stochastic dynamic programming (SDP) have been used to solve reserve selection problems of this type Meir et al. (2004); Costello and Polasky (2004). In these models, unreserved sites are irreversibly developed each year with a given probability, but only a limited number of sites can be reserved each year, because of budgetary or site-availability constraints. The problem is to design a dynamic reservation policy that results in the maximum expected number of species conserved at the end of the problem horizon. These models are used to investigate the importance of the timing of selections in habitat conservation programs. Existing models consider a random development pattern, in which development probability for each site is independent of the development status of neighbouring sites. However, it is more likely that development will occur as a “contagion” process, beginning when new roads make regions economically and logistically accessible Laurance et al. (2004). In this paper we propose improved SDP algorithms which make use of a graph representation of the sites network to address the reserve site selection problem under this spatially explicit form.

2 A MODEL FOR DYNAMIC RESERVE SITE SELECTION PROBLEMS UNDER CONTAGION RISK OF DEFORESTATION

2.1 State variables

In Costello and Polasky (2004), the following model, based on habitat suitability is proposed for dynamic reserve site selection. Assuming that there exist J sites indexed by $j = 1, 2, \dots, J$ and I species indexed by $i = 1, 2, \dots, I$, a $J \times I$ matrix A is given, where an element A_{ji} equals 1 if site j is suitable for species i, and 0 if not. At a given time period t, any site j can be in one of the three following states : *developed*, *reserved* or *unreserved*. It is assumed that a species i exists in site j if and only if site j is not developed (i.e. is in state *reserved* or *unreserved*). Thus, the state S_t of sites can be unambiguously described by the means of two of the three vectors D_t, R_t, U_t where $D_t(j) = 1$ means that site j is developed, and $D_t(j) = 0$ means that it is not. R_t and U_t model whether sites are reserved or unreserved. It is clear that for any site j, exactly one of $D_t(j), R_t(j), U_t(j)$ equals one, and the two others equal zero. Thus we define $S_t = (D_t, R_t)$.

2.2 Control variable and process dynamics

The state of sites will evolve over time under the influence of two types of factors: controlled and uncontrolled factors.

- Controlled factor. At any time period, it is possible to select one unreserved site for reservation, thus changing its state from *unreserved* to *reserved*[1].
- Uncontrolled factor. At any time period t, it is assumed that any unreserved site j which is not selected for reservation can become developed at the end of the period with a known probability p_j. In the most general case, the probability of development of each site j can depend on the whole current pattern of development of the sites.

So, at each time step t, a selection action is chosen, consisting in a site number $a_t \in \{1 \dots J\}$, selected for reservation. Then, if $U_t(a_t) = 1$ (site number a_t is unreserved), we get $R_{t+1}(a_t) = 1$ and $U_{t+1}(a_t) = 0$. Then, development can occur, and we consider it as a contagion process : the probability that an undeveloped site j becomes developed depends on the development status of *neighbour* sites of j. The neighbourhood relation between sites is represented by a symmetric matrix G such that $G(j, j') = 1$ if sites j and j' are neighbours, and $G(j, j') = 0$ if not. $G(j, j) = 0$ for all sites by definition. Then, we define the neighbourhood of a site j as $N(j) = \{j' \in 1 \dots J, G(j, j') = 1\}$. The development probabilities of a site j which is currently undeveloped are: $p_j(S_{t+1}(j) = D | S_t(j) = U, S_t(N(j)), a_t)$ where $S_t(N(j)) = \{S_t(j'), j' \in N(j)\}$. The global transition probabilities between states are defined as:

$$Pr(S_{t+1} | S_t, a_t) = \prod_{j, S_t(j)=U} p_j(S_{t+1}(j) | S_t(j) = U, S_t(N(j)), a_t)$$

$$\text{and } \delta(S_{t+1}, S_t) \text{ if } \nexists j, S_t(j) = U.$$

At this point, it should be noticed that development

[1]In Costello and Polasky (2004) the possibility for selecting several sites, under a global budget constraint is modelled, however, in the practical cases studied, only one site at a time can be reserved. In this paper we adopt this simplification from the beginning, but the more general case could be considered as well in our approach

and reservation are assumed to be irreversible : a *developed* or *reserved* site remains in this state forever. Thus, the development / reservation process will always end in an *absorbing state* in which no unreserved site persists. Such an absorbing state will be reached in a number of time steps bounded by the number of sites since at each time step one undeveloped site becomes irreversibly reserved.

2.3 Reserve selection policy, objective function

We have described the stochastic controlled process model of the evolution of global states. Let us now describe the objective function of the control problem. The objective of a reserve selection problem is of course to minimise species losses, or equivalently to maximise the number of species present in reserved sites when the process has reached an absorbing state. Our goal is to find a *policy* π assigning to every possible states S_t a site to reserve. Such a policy should be defined so as to maximise the expected value of the number of species reserved when an absorbing state of the process is reached. ¿From now on, we give up the subscript t in the notations of the state and action variables for sake of simplicity since i) the process is assumed to be stationary (transitions and rewards do not depend on time) and ii) it can be shown (see Puterman (1994)) that optimal policies are in this case stationary. Let us now define a *reward function* $r(S,a)$ as the number of additional species which are protected when site a is reserved in state S:

$$\forall S = (R,D), a, r(S,a) = \\ |\{i, (A_{ai} = 1) \wedge (\nexists j, R(j) = 1 \wedge A_{ji} = 1).\}|$$

$(A_{ai} = 1) \wedge (\nexists j, R(j) = 1 \wedge A_{ji} = 1)$ means that species i is in site a, and does not belong to any site already reserved.

Now, let us consider a *trajectory* τ, that is an alternate sequence of states and actions, starting in an arbitrary state S^0 and ending in an absorbing state S^k : $\tau = (S^0, a^0, S^1, a^1, \ldots, S^{k-1}, a^{k-1}, S^k)$. We define the *value* $V(\tau)$ of such a trajectory τ as the number of species eventually protected at the end of the trajectory. Thus, $V(\tau)$ is exactly the number of species protected in state S^k. The following equality can be easily shown, which will be used in the dynamic programming solution method for the reserve selection problem :

$$\forall \tau = (S^0, a^0, S^1, a^1, \ldots, S^{k-1}, a^{k-1}, S^k), \\ V(\tau) = \sum_{i=0}^{k-1} r(S^i, a^i).$$

A fixed policy π does not define a single trajectory τ, when applied in a start state S, but rather a probability distribution over a set of possible trajectories. The value $V_\pi(S)$ of this policy is defined as the expected number of new species which can be protected by applying π, from start state S :

$$V_\pi(S) = E[V(\tau)|S,\pi] \tag{1}$$

where $E[V(\tau)|S,\pi]$ is defined over the set of possible trajectories generated by policy π applied in initial state S. In the next Section, we are going to show how to compute this value and how to find π maximising $V_\pi(S)$ for any possible initial state S.

3 AN EXACT DYNAMIC PROGRAMMING SOLUTION METHOD

We first define the following subsets of the global states space:

$$\forall k = 0, \ldots, J, \mathcal{U}_k = \\ \{S = (R,D), |\{j, \max(R(j), D(j)) = 0\}| = k\}.$$

$\mathcal{U}_k$ is the set of states in which exactly k sites remain undeveloped. Obviously, any state $S \in \mathcal{U}_0$ is an absorbing state, for which no site can be reserved (and no new species protected). The value of such states should be zero :

$$\forall S \in \mathcal{U}_0, V(S) = 0.$$

Now, given a fixed policy π, the value $V_\pi(S)$ of this policy, defined in equation 1 can be computed recursively over all states :

Algorithm 1: Policy evaluation

```
Data: $< S, \pi, r, p >$
Result: $V_\pi$
begin
    for $S \in \mathcal{U}_0$ do $V_\pi(S) = 0$   for $k = 1 \ldots J$ do
        for $S \in \mathcal{U}_k$ do
            $V_\pi(S) = r(S, \pi(S)) + \sum_{S' \in \mathcal{U}_0 \cup \ldots \cup \mathcal{U}_{k-1}} p(S'|S, \pi(S)) \cdot V_\pi(S')$
    return $V_\pi$
end
```

This recursive computation of $V_\pi(S)$ is made possible since the number of unreserved sites can only decrease whenever a selection action is applied. Thus, we can start by computing the value of states with no site unreserved, then for states with only one unreserved site, then two, etc. In this way, states $S(R,D)$ are visited at most once.

It can then be shown by classical arguments from stochastic dynamic programming Puterman (1994) that there exists a *dominating policy* π^*, such that $\forall S, V_{\pi^*}(S) \geq V_\pi(S)$. Such an optimal policy, as well as its value in every state can be computed exactly, using the following dynamic programming algorithm :

Algorithm 2: Optimal policy computation

```
Data: $< S, \pi, r, p >$
Result: $\{\pi^*, V_{\pi^*}\}$
begin
    for $S \in \mathcal{U}_0$ do $V_{\pi^*}(S) = 0$   for $k = 1 \ldots J$ do
        for $S \in \mathcal{U}_k$ do
            $V_{\pi^*}(S) = \max_{a\ undeveloped}[r(S,a) + \sum_{S' \in \mathcal{U}_0 \cup \ldots \cup \mathcal{U}_{k-1}} p(S'|S,a) \cdot V_{\pi^*}(S')];$
            $\pi^*(S) = \arg\max_{a\ undevelop}[r(S,a) + \sum_{S' \in \mathcal{U}_0 \cup \ldots \cup \mathcal{U}_{k-1}} p(S'|S,a) \cdot V_{\pi^*}(S')];$
    return $\{\pi^*, V_{\pi^*}\}$
end
```

The following complexity result can be shown :

Proposition 1 *Space complexity of Algorithm 2 is in $O(J4^J)$, while its space complexity is in $O(3^J)$.*

Proof: First, consider the inner loop of the algorithm. A max and an $argmax$ operations are performed over exactly k actions (there are k undeveloped sites). In addition, a summation is performed over the 2^{k-1} possible successor states (each of the $k-1$ sites remaining undeveloped after the current site to reserve is chosen can become developed or stay undeveloped). So, the complexity of the inner loop is in $O(k2^{k-1})$. Now, one can check that $\mathcal{U}_k$ contains exactly $C_J^k 2^{J-k}$ possible states : there are C_J^k possibilities to chose $k\ undeveloped$ sites among J sites, and there are 2^{J-k} possible combinations of $reserved$ and $developed$ sites over the remaining $J-k$ sites. Finally, the outer loop is performed for $k = 0$ to J, so that the overall complexity of the algorithm is in $O(\sum_{k=0}^J C_J^k 2^{J-k}(k2^{k-1}))$. But it can be shown that

$$\sum_{k=0}^{J} C_J^k 2^{J-k}(k2^{k-1}) = J2^{2J-2} = J4^{J-1},$$

hence the overall complexity is in $O(J4^J)$. Concerning the space complexity, it is easily seen that the algorithm only stores two values for each of the 3^J states, so that obviously space complexity is in $O(3^J)$. $\square$

¿From these complexity results, it follows that the exact computation of optimal policies for large reserve selection problems is not feasible, except for really small problems (less than ten sites). So, in the following Section we will introduce a heuristic-based approach to the reserve selection problem. We will present two simple heuristic methods proposed by Costello and Polasky (2004). Then, in Section 5, we will present another kind of heuristic method, which is adaptive, and makes use of simulations to compute the heuristic function value. We will see that this family of methods outperforms the simpler heuristic methods in terms of quality of the returned policies, and is still applicable for very large problems (hundreds of sites).

4 HEURISTIC SOLUTION METHODS

In the previous Section it has been shown that the computation of an exact optimal policy is infeasible, but for small problems. There are two sources of difficulties for this exact computation, which are :

- the computation of the exact optimal value function, V_{π^*} and
- the computation of $\pi^*(S)$ which involves a sum over the 2^{k-1} possible successors of S.

We will overcome the first limitation by computing a simple heuristic approximation of the optimal value function, while the second limitation will be overcome by summing over a sample of the set of successor states. These two “tricks” will allow to compute heuristic reserve selection policies, which performance can be in turn estimated by using Monte Carlo simulation.

4.1 Heuristic approximation of the value function

Recall that once the optimal value function V_{π^*} is known, we get an optimal policy π^* by :

$$\forall S, \pi^*(S) = \arg \max_{a\ undeveloped\ in\ S}[r(S,a) + \sum_{S'} p(S'|S,a) \cdot V_{\pi^*}(S')].$$

Now, when it is too costly to compute V_{π^*} and when a heuristic approximation $\tilde{V}$ is available, we can compute an approximately optimal policy $\tilde{\pi}$:

$$\forall S, \tilde{\pi}(S) = \arg \max_{a\ undeveloped\ in\ S}[r(S,a) + \sum_{S'} p(S'|S,a) \cdot \tilde{V}(S')]. \quad (2)$$

Costello and Polasky (2004) proposed two approximation methods for computing a reserve selection policy, which can be cast into this heuristic framework. Namely, they called them *myopic* and *informed myopic* policies. They correspond to the two following choices for the heuristic value function $\tilde{V}$:

- *Myopic*. In this case, $\tilde{V}(S') = 0, \forall S'$, so that $\tilde{\pi}$ is defined by

$$\forall S, \tilde{\pi}(S) = \arg \max_{a \ undeveloped \ in \ S} r(S, a).$$

- *Informed myopic*. In this case, $\tilde{V}(S') = \max_{a \ undeveloped \ in \ S'} r(S', a)$, so that equation 2 becomes

$$\forall S, \tilde{\pi}(S) = \arg \max_{a \ undeveloped \ in \ S} [r(S, a) + \max_{a' \ undeveloped \ in \ S'} \sum_{S'} p(S'|S, a') \cdot r(S', a')].$$

4.2 Sample-based computation of a heuristic policy

Now, let us assume that a heuristic function $\tilde{V}$ has been chosen. The heuristic policy $\tilde{\pi}$ remains to be computed through equation 2. But computing $\tilde{\pi}(S)$ for a given S, requires to compute $\sum_{S'} p(S'|S, a,) \cdot \tilde{V}(S')$ over the whole set of possible successors of S, which size is $2^{n-|D|-|R|}$, which can be too large, if many sites remain undeveloped in S. To overcome this limitation when the number of undeveloped sites is too large, $\tilde{\pi}$ is computed on line (i.e. $\tilde{\pi}(S)$ is only computed when state S is encountered) by the following equation instead of equation 2. n_{simul} successor sites S'_i are drawn at random, according to the law $p(S'|S, a)$ and :

$$\forall S, \tilde{\pi}(S) = \arg \max_{a \ undeveloped \ in \ S} [r(S, a) + \frac{1}{n_{simul}} \cdot \sum_{i=1}^{n_{simul}} \tilde{V}(S'_i)].$$

In this way, a policy is never explicitly computed and stored, but rather computed on line and only the heuristic function $\tilde{V}$ needs to be stored.

4.3 Monte Carlo estimation of heuristic policies

The computation of a heuristic-derived policy $\tilde{\pi}$ can be onerous, due to the need to explore the whole set of possible successors of a given state S. The estimation of the value of any policy π can also be prohibitively costly. As for the computation of $\tilde{\pi}$, it can be computed through simulation :

$$\tilde{V}_\pi(S) = \frac{1}{n_{simul}} \cdot \sum_{i=1}^{n_{simul}} V(\tau_i) \qquad (3)$$

where τ_i is a randomly generated trajectory obtained by applying π in S. $\tilde{V}_\pi(S)$ is of course an unbiased estimation of $V_\pi(S) = E[V(\tau|S, \pi)]$.

In the following Section, we are going to show a method improving in practice the method based on the myopic heuristic, which is the only one applicable to very large problems (we observed that on reserve selection problems with contact development, informed myopic does not significantly improve the performance of the myopic heuristic, but is computationally more expensive). This improved heuristic method uses a parameterised representation of the heuristic function $\tilde{V}$ and the parameters are automatically tuned and optimised through repeated simulations.

5 PARAMETERISED REINFORCEMENT LEARNING SOLUTION METHOD

Reinforcement learning is a set of simulation-based methods which allow for the solution of large-scale Markov Decision Problems Bertsekas and Tsitsiklis (1996), such as the reserve site selection problem we are interested in. In this framework, the optimal value function V_{π^*} is approached by a (linear) parameterised value function V_{ε^*} which is computed through repeated simulations of trajectories. Then, a policy π_{ε^*}, greedy with respect to V_{ε^*} is computed (using a sample-based method). π_{ε^*} approaches the optimal policy π^*. The value of π_{ε^*} in the initial state (every site undeveloped) is then estimated using Monte Carlo methods, and compared to the heuristics previously described.

5.1 Parameterised linear approximation of the optimal value function

For very large SDP problems, such as the reserve selection problem, when the number of sites is large, it is not convenient to compute the exact optimal value function V_{π^*} in tabular form. It may be more reasonable to look for an approximate, parameterised, value function V_{ε^*}, which can be expressed much more concisely than V_{π^*} itself. A linear approximation of V_{π^*} is often used, where an approximation of V_{π^*} is searched in the set of parameterised value functions of the form

$$V_\varepsilon(S) = \varepsilon(1)\psi_1(S) + \ldots + \varepsilon(k)\psi_k(S).$$

The $\varepsilon(i), i \in \{1, \ldots, k\}$ are parameters which will be computed by simulation and the ψ_i are arbitrarily given real-valued functions called *features* Bertsekas and Tsitsiklis (1996).

Then, the objective of feature-based reinforcement learning algorithms is to compute a parameters vector ε^* such that V_{ε^*} is a reasonable approximation of V_{π^*}. The general way is to use a simulation of the controlled process in order to compute a sequence of parameters vectors (ε_n), in the form

$$\varepsilon_{n+1} = \varepsilon_n + \Delta(S_n, a_n, S_{n+1}, r(S_n, a_n))$$

where $\Delta(S_n, a_n, S_{n+1}, r_n(S_n, a_n))$ is a correction factor computed from the output of the current simulation trial. The most common implementation of the above principle is the *gradient descent* method, where updates take the form Bertsekas and Tsitsiklis (1996)

$$\varepsilon_{n+1} = \varepsilon_n + \alpha_n(R_n - V_{\varepsilon_n}(S_n))\nabla_{\varepsilon_n} V_{\varepsilon_n}(S_n).$$

where R_n is a direct estimation of the value of V_{π^*} drawn from the current trial and past experience. A simple such estimation consists in using

$$R_n = \max_a r(S_n, a) + V_{\varepsilon_n}(S_{n+1}).$$

Now, one simple case of *features* is of particular interest Tsitsiklis and Van Roy (1996). This is the one in which the ψ_n take their values in the set $\{0, 1\}$. In this case, the above equation simplifies into

$$\begin{aligned}&\forall i \in 1, \dots, k\\ &\varepsilon_{n+1}(i) = \varepsilon_n(i) + \alpha_n(\max_a\{r(S_n, a) + V_{\varepsilon_n}(S_{n+1})\}\\ &\qquad\qquad - V_{\varepsilon_n}(S_n))\psi_i(S_n).\end{aligned}$$

This will be the parameters update function which we will use to solve the reserve selection problem.

5.2 Features in the reserve selection problem

Let us first describe the parameterised approximation of the optimal value function in the reserve selection problem. In this problem, we choose to use the following J features: $\psi_i(S) = 1$ if site i is not developed, and $\psi_i(S) = 0$ if it is developed, for $i = 1 \dots J$. Thus, $\psi_i(S) = \psi_i(S(i))$ only depends on the state of site i, and not on the global state of the problem. In addition, concerning the learning rate parameter α_n, which should simply decrease to 0 as n grows, we choose to define it as $\alpha_n = \frac{1}{n(a)}$, i.e. the number of time site a has been reserved so far, during the learning phase of the algorithm.

5.3 Computation of an approximately optimal policy

The following steps are performed in order to choose which site to reserve, when a new state is encountered.

- irst of all, $\varepsilon^* = \lim_{n\to+\infty} \varepsilon_n$ is computed.
- $\tilde{V} = V_{\varepsilon^*}$ is chosen as the heuristic function.
- Then, for any given state S encountered *on line*, site $\tilde{\pi}(S)$ computed through equation 8 is chosen for reservation.

Such a policy can be estimated by applying procedure 3 to the initial (every sites undeveloped) state of randomly generated problems. A comparison of the simple heuristic-based solutions described in the previous Sections and of the RL-based heuristic, is shown in the next Section.

6 EMPIRICAL RESULTS

6.1 Benchmark problems

We analysed empirically the results of the three above described methods (exact, myopic-heuristic and parameterised reinforcement-learning) on a randomly generated set of reserve site selection problems of various sizes. Problems were randomly generated, using the following set of parameters :

- J is the chosen number of sites,
- I is the number of considered species,
- δ is the *degree* of the graph, i.e. the maximum number of neighbours for a given node,
- TS is the maximum number of *threatened sites*, i.e. sites which can become developed without having any developed neighbour site,
- SS is the maximum number of suitable sites for any given species.

The outputs of the generator are the following:

- the neighbourhood graph matrix G,
- the site / species matrix A,
- the development diffusion probabilities used for building the development probabilities for each site j ($p_{diff}(j) \in \mathcal{U}[0.3; 0.5]$),
- the development probabilities for threatened sites j ($p_{dev}(j) \in \mathcal{U}[0.2; 0.3]$).

6.2 Small problems

For small problems ($J \leq 10$), the exact method described in Section 3 could be applied, and we checked experimentally the CPU time needed for solving each problem (Figure 1). The parameters values we used were the following : $J \in \{4 \dots 10\}; I = 15; \delta = 4; SS = 3$ and $TS = 2$. The computation times were obtained by averaging over 30 randomly generated problems for each value of J.

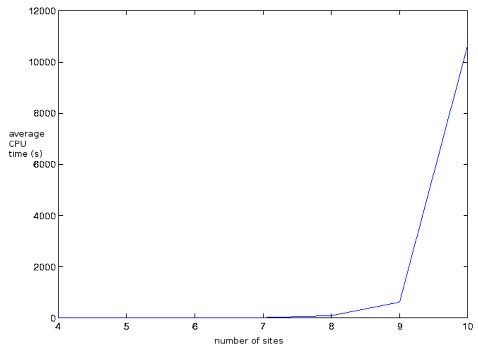

Figure 1. Empirical computation time for the exact method

6.3 Large problems

For larger problems ($J \leq 100$), we compared the RL and myopic methods in terms of species losses (Figure 2). The parameters values were : $J \in \{30, 40, 50, 60, 70, 80, 90, 100\}; I = J + 50; \delta = 4; SS = 3$ and $TS = 6$. For each configuration we generated randomly one problem over which we performed 1000 pairs of trajectories (one induced by the myopic heuristic, the other by the RL method). Figure 2 shows the average loss of species experienced in each configuration for the two methods. It can be observed that in average,

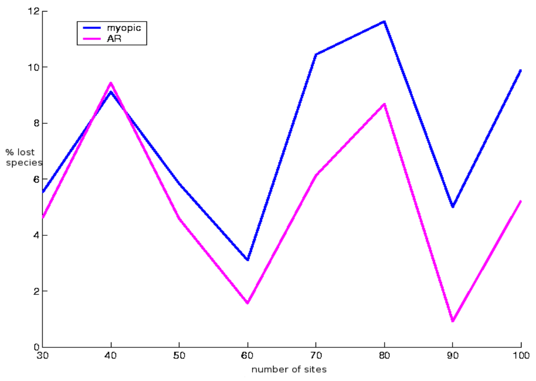

Figure 2. Empirical estimation of species losses : Myopic heuristic and RL

for problems of size 60 or more sites, the average percentage of species lost is around 8% for the myopic heuristic, and around 4.5% for the RL method. Thus, the RL method allows to reduce the number of lost species by more than 40% !

6.4 Very large problems

For very large problems ($J \leq 1000$) it becomes really difficult to assess the value of policies generated by RL methods, since we need to compute on-line the RL policy, using sampling (equation 11), after a vector of optimal parameters ξ^* has been obtained. Thus, computing a single trajectory for a given parameter is a bit costly, and it is not realistic to evaluate by a Monte Carlo method the policy π_{ξ^*}. However, it is of course realistic to apply such a policy for very large problems, since the time between two decision steps is important. Furthermore, the CPU time needed for the computation of ξ^* does not increase too quickly, as Figure 3 shows.

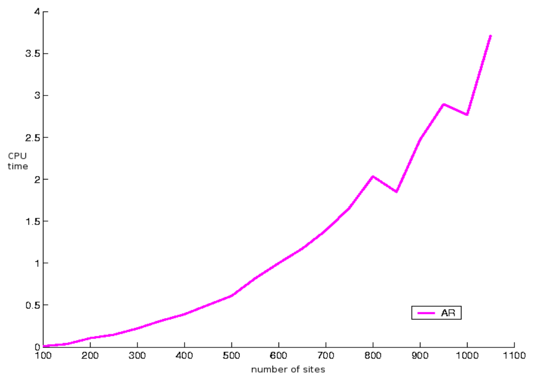

Figure 3. Empirical estimation of ξ^* computation time for very large problems

7 CONCLUDING REMARKS

In this paper we described several methods for solving the dynamic reserve site selection problem, initially described in Costello and Polasky (2004). We first described a new, exact, dynamic programming algorithm, applicable to small problems, and we showed its theoretical and empirical complexity. Then, we presented a general framework for heuristic selection methods, and showed that the selection methods proposed in Costello and Polasky (2004) fit into this framework. Finally, we proposed a new *parameterised reinforcement learning* method, which could be seen as an on-line heuristic method, and we showed experimentally that it improved significantly the already existing heuristic methods in terms of quality of the result, and that it could be applied to really large problems, such as the ones that can be encountered in practice. Our next step is to apply the latter method to a real example in Costa Rica forests for which we have historical data for the past twenty five years (development process).

8 REFERENCES

Bertsekas, D. P. and J. N. Tsitsiklis (1996). *Neuro-Dynamic Programming*. Belmont, Massachussetts: Athena Scientific.

Costello, C. and S. Polasky (2004). Dynamic reserve site selection. *Resource and Energy Economics 26*, 157–174.

Laurance, W., A. Albernaz, P. Fearnside, H. Vasconcelos, and L. Ferreira (2004). Deforestation in amazonia. *Science 304*(5674), 1109–11.

Meir, E., S. Andelman, and H. Possingham (2004). Does conservation planning matter in a dynamic and uncertain world? *Ecology Letters 7*, 615–622.

Puterman, M. L. (1994). *Markov Decision Processes*. New York: John Wiley and Sons.

Tsitsiklis, J. N. and B. Van Roy (1996). Feature-based methods for large-scale dynamic programming. *Machine Learning 22*, 59–94.