Two Approximate Dynamic Programming Algorithms for Managing Complete SIS Networks

Martin Péron
1) School of Mathematical Sciences, Queensland University of Technology
Brisbane, Queensland 4000, Australia
2) Land and Water, CSIRO
Ecosciences Precinct, Dutton Park, Queensland 4102, Australia
martinperon@outlook.com

Peter L. Bartlett
Computer Science Division and Department of Statistics, University of California
Berkeley, CA, USA
peter@berkeley.edu

Kai Helge Becker
Mathematical Optimization Department, Zuse Institute Berlin
Berlin, Germany
becker@zib.de

Kate Helmstedt
School of Mathematical Sciences, Queensland University of Technology
Brisbane, Queensland 4000, Australia
kate.helmstedt@qut.edu.au

Iadine Chadès
Land and Water, CSIRO
Ecosciences Precinct, Dutton Park, Queensland 4102, Australia
iadine.chades@csiro.au

ABSTRACT

Inspired by the problem of best managing the invasive mosquito *Aedes albopictus* across the 17 Torres Straits islands of Australia, we aim at solving a Markov decision process on large Susceptible-Infected-Susceptible (SIS) networks that are highly connected. While dynamic programming approaches can solve sequential decision-making problems on sparsely connected networks, these approaches are intractable for highly connected networks. Inspired by our case study, we focus on problems where the probability of nodes changing state is low and propose two approximate dynamic programming approaches. The first approach is a modified version of value iteration where only those future states that are similar to the current state are accounted for. The second approach models the state space as continuous instead of binary, with an on-line algorithm that takes advantage of Bellman's adapted equation. We evaluate the resulting policies through simulations and provide a priority order to manage the 17 infested Torres Strait islands. Both algorithms show promise, with the continuous state approach being able to scale up to high dimensionality (50 nodes). This work provides a successful example of how AI algorithms can be designed to tackle challenging computational sustainability problems.

KEYWORDS

Markov decision process; Susceptible-Infected-Susceptible networks; Aedes albopictus; Approximate dynamic programming; Invasive species; Optimal management; Computational sustainability

COMPASS '18, June 2018, CA, USA

ACM ISBN 978-1-4503-5816-3/18/06…$15.00
https://doi.org/10.1145/3209811.3209814

ACM Reference Format:
Martin Péron, Peter L. Bartlett, Kai Helge Becker, Kate Helmstedt, and Iadine Chadès. 2018. Two Approximate Dynamic Programming Algorithms for Managing Complete SIS Networks. In *COMPASS '18: ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS), June 20–22, 2018, Menlo Park and San Jose, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209811.3209814

1 INTRODUCTION

Markov decision processes (MDPs) are a mathematical framework designed to optimize sequential decisions under uncertainty given a specific objective [1, 27]. MDPs can be solved in polynomial time by a method called stochastic dynamic programming [13]. However, in many real-world applications the states describing the system are factored. That is, states are naturally defined as a combination of sub-states. MDPs with such states are called factored MDPs. Sub-states can correspond to different features of the system [12], individuals in a population [30], spatial locations in a network [3] or products in an inventory problem [26]. An essential aspect of factored MDPs is that their number of states grows exponentially when the number of sub-states increases. So, since stochastic dynamic programming requires listing all reachable states [27], too many sub-states make stochastic dynamic programming intractable. This issue has been termed the curse of dimensionality [1].

There exist some exact MDP solvers tailored to solve factored MDPs, e.g. SPUDD [12]. SPUDD consists of using algebraic decision diagrams to represent policies and value functions, grouping together states that have the same value or optimal action (see also [31]). This approach works well when many sub-states are conditionally independent and poorly otherwise.

In this paper, we aim at optimizing management decisions on a particular type of factored MDP called Susceptible-Infected-Susceptible (SIS) network. In an SIS network, each sub-state represents a node in an interconnected network that can be either susceptible or infected [3]. SIS networks are commonly used to model the spread of infectious disease or parasites in epidemiology [28, 30], meta-population dynamics of threatened, or

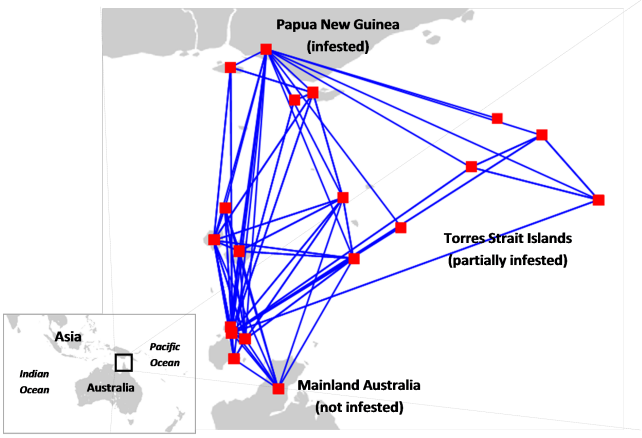

Figure 1: The Torres Strait Islands. Connections between islands depict the possibilities of transmission of the mosquitoes towards susceptible islands. Low transmission probabilities are not shown for readability.

invasive species in ecology [3, 20] or computer viruses in computer science [14, 21]. Inspired by a real case study, the management of the Asian tiger mosquito *Aedes albopictus* in Australia [22], we aim at exploiting this particular structure to solve highly connected large size SIS-MDPs, thus providing good policies on large networks to decision makers and circumventing the curse of dimensionality.

1.1 Case study: managing invasive Aedes albopictus

The Asian tiger mosquito, *Aedes albopictus*, is a highly invasive species and a vector of several arboviruses that affect humans, including chikungunya and dengue viruses. These invasive mosquitos were first detected in the Torres Strait Islands, Australia, in 2005 [29] where they persist today despite ongoing management effort. The $N = 17$ inhabited islands constitute potential sources for the introduction of *Aedes albopictus* into mainland Australia through numerous human-related pathways between the islands and towards north-east Australia (Figure 1).

Local eradication of the mosquito is possible through management actions on islands such as treating containers and mosquitoes with diverse insecticides. After eradication, re-infestation can occur from connected infested islands. Since budget is limited, not all islands can be treated simultaneously. The objective is to select islands to manage to maximize the expected time before the mainland becomes infested. Past attempts modeled this problem as an MDP and used stochastic dynamic programming (policy iteration) to find the optimal policy [22]. However, the approach failed to circumvent the curse of dimensionality. Only 13 out of the 17 Torres Strait Islands were accommodated, providing incomplete recommendations to managers. The main motivation of this paper is to provide an approach to accommodate all 17 Torres Strait Islands.

To do so, we have identified two noteworthy properties of this system. First, the network is 'complete', i.e. every node can be infested from any other node of the network. Consequently, local optimization approaches such as graph-based MDPs [6, 18], which only consider potential infestations from a small subset of neighboring nodes, are not well suited to this problem. Second, since local eradication is difficult to achieve and transmission rates are low, the probability for each sub-state (node) to change (either from susceptible to infested or vice versa) is small. This implies that the MDP state at the next timestep will likely be similar to the current state, i.e. a small number of sub-states are likely to change. The two approximate dynamic programming approaches we propose exploit these properties.

1.2 Approximate approaches

In the last decade, several approaches have been explored to solve large factored MDPs, with multiple applications in computational sustainability [6, 7, 18, 20]. Generally speaking, these approaches can be classified into three groups [26], all of which are relevant to our case study.

First, simulation-optimization methods consist of evaluating a number of policies through simulations and selecting the best one [34] (see also [16] in conservation biology). These approaches do not anticipate what might happen in the future [26], which is appropriate for our case study problem because states do not change frequently. Alternative approaches use cascade models to capture SIS dynamics, but do not involve sequential decisions [32].

Second, rolling horizon procedures (roll-out) use a prediction of the near-future to save on potentially costly long-term predictions [18]. Typical approaches include model predictive control and Monte Carlo tree search [10]. Roll-out procedures have been used in conservation biology to solve SIS-MDPs that are large but much more weakly connected than the Torres Strait Island system [18, 19]. Finally, some hindsight optimization approaches can help optimize decisions on large networks, but with a focus on exponentially large action spaces [36].

Third, approximate dynamic programming (ADP) approaches explicitly estimate the values of states to derive optimal actions. For example, mean-field approximation algorithms [10, 20, 23] and approximate linear programming methods [6] approximate the value function by decomposing it into a sum of the values of each node. The value function is updated through local optimization, for example, in our case, assuming that each node is only connected to a limited number of neighbors. Therefore, these approaches are not suited to highly connected networks, e.g. some have been reported to "*work best when nodes have fewer than five neighbors*" [20].

Inspired by these three classes of approaches, we introduce two new approximate approaches to address large and highly connected SIS-MDP networks. Our first approach is a simplification of Bellman's equation where only a small subset of the future states are considered. We demonstrate that this approach comes with some performance guarantee and is less computationally complex than stochastic dynamic programming. However, its complexity is still exponential in the number of sub-states. In contrast, our second approximate approach is a more radical approximation that runs in linear time in the time horizon and in quadratic time in the number of nodes, but has no performance guarantees. We assess our algorithms on our case study and compare their solutions to

SPUDD when possible [12], the reference exact algorithm to solve factored MDPs.

2 MATERIAL AND METHODS

2.1 Markov decision processes

Markov decision processes (MDPs) are mathematical frameworks for modeling sequential decision problems where the outcome is partly stochastic and partly controlled by a decision-maker [1]. A MDP is defined by five components $< S, A, P, r, C >$ [27] : (i) a state space S, (ii) an action space A, (iii) a transition function P, (iv) an immediate rewards function r and (v) a performance criterion C.

The decision-maker aims to direct the process towards rewarding states. From a given state s, the decision-maker selects an action a and receives a reward $r(s, a)$. At the next time step, the system transitions to a subsequent state s' with probability $P(s'|s, a)$. The performance criterion C specifies the objective (e.g. maximize or minimize a sum of expected future rewards), the time horizon (finite or infinite), the initial state s_0 and whether there is a discount rate (γ). Here, we deal with a discounted infinite time horizon, where we maximize:

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0]. \tag{1}$$

A policy π describes which decisions are made in each state, i.e. $\pi : S \rightarrow A$. Solving an MDP means finding an optimal policy π^* that satisfies, in our case:

$$\pi^* = \arg\max_{\pi} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))|s_0]. \tag{2}$$

Exact algorithms to solve MDPs include linear programming, value iteration, and policy iteration [33]. We choose to use value iteration, because its simplicity makes it easy to adapt to approximately solve large factored MDPs.

2.2 Value iteration

Value iteration requires the introduction of a value function V, defined on all states s. The value $V(s)$ corresponds to the sum of future rewards one can expect, starting from the state s. The value function is unknown at the start of the algorithm, and it is customary to start with $V = 0$. The value function V is repeatedly improved, for each state, with Bellman's equation:

$$V'(s) = \max_{a \in A} \Big[r(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \Big]. \tag{3}$$

Once this is evaluated for all states $s \in S$, we set $V := V'$ and the process repeats until some termination condition is met, which can either consist of a maximum number of iterations (our choice in this paper) or a threshold ϵ under which the maximum difference between V to V' must fall (see [27]). The output policy is defined as:

$$\pi(s) = \arg\max_{a \in A} \Big[r(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \Big]. \tag{4}$$

Provided the reward is non-negative, the sequence of V is monotonic and guaranteed to converge [33]. The outputs of value iteration are a policy and a value function, and the value function is guaranteed to be within $\frac{2\gamma\epsilon}{1-\gamma}$ of the optimal value function with the ϵ termination criterion [33]. We now describe Susceptible-Infected-Susceptible networks and how they can be cast into MDPs.

2.3 Susceptible-Infected-Susceptible (SIS) networks

SIS networks are used to model spatial systems where a species can spread over a network [3]. Each node in the network can be either infested (the terminology for invasive species) or susceptible (i.e. at risk of being infested). The species can infest new nodes by spreading. Infested nodes can be cured and re-infested.

Numbering each node from 1 to N, we denote by s_i the status of node number i: $s_i = 1$ if node i is infested, 0 otherwise. A transmission probability matrix p_{ji} describes the probability for mosquitoes to be transmitted from any infested node j to any susceptible node i. The probability for node i to remain 'susceptible' is then given by:

$$Pr(s'_i = 0|s_i = 0) = \prod_{j \neq i} (1 - s_j p_{ji}), \tag{5}$$

where s'_i is the next state of node i. So, the probability to transition from 'susceptible' to 'infested' is $1 - \prod_{j=1}^{N} (1 - s_j p_{ji})$.

All nodes are able to be managed with a sub-action. The effectiveness of a sub-action implemented on node i is denoted by a_i. It is defined as the probability of locally eradicating the mosquitoes over one time step, which implies:

$$Pr(s'_i = 1|s_i = 1) = 1 - a_i. \tag{6}$$

In this paper, we address two common management objectives for SIS models: eradication and containment. In the eradication objective, the goal is to maximize the number of susceptible nodes [3], so the reward is defined as:

$$r(s, a) = \sum_{i=1}^{N} (1 - s_i). \tag{7}$$

In the containment objective, the goal is to prevent the species from reaching a node i [30], so the reward can be defined as:

$$r(s, a) = 1 - s_i = \begin{cases} 0 \text{ if node } i \text{ is infected;} \\ 1 \text{ otherwise,} \end{cases} \tag{8}$$

if node i is to be protected.

2.4 SIS-MDPs

Sequential decision problems on SIS networks can be cast into MDPs as follows (we call the resulting MDP an SIS-MDP). Each state s describes the situation on all nodes and is of the form $s = (s_1, s_2, \ldots, s_N)$. The transition function is defined as $P(s'|s, a) = \prod_{i=1}^{N} Pr(s'_i|s_i, a)$. Any dynamic programming approach, including value iteration, policy iteration or SPUDD, can then be applied to solve these SIS-MDPs [33].

The main issue with this approach is that the number of states is $|S| = 2^N$, which is computationally prohibitive when N grows. For the Torres Strait mosquito network, only up to 13 nodes have been reported to be tractable [22]. In practice, one can distinguish three causes of intractability, called curses of dimensionality [26]. The first curse of dimensionality is the exponential number of states itself, which is prohibitive because the value function must

be updated for each state (Eq. (3)). Further, each of these updates hinges upon a sum over future states (r.h.s. in Eq. (3)), which is equally prohibitive: this is the second curse of dimensionality. The third curse of dimensionality is the exponential number of actions, which does not apply in our case study because the total number of actions is limited by a budgetary constraint (a maximum of three islands can be managed simultaneously). Driven by this case study, we introduce two new approximate approaches to address the first two curses of dimensionality in SIS-MDPs.

2.5 First approximate approach: the *Neighbor* algorithm

The *Neighbor* algorithm is a modified version of value iteration (Algorithm 1). As mentioned in the Introduction, in our case study, the probability for each sub-state or node to change over one time step is low. So, future states will likely differ from the current state by a handful of nodes at most. Indeed, denoting p as an upper bound of the probability that any node changes, the probability that all nodes change is less than p^N. Also, the expected number of node switches is less than Np, and since the node switches are independent, it is unlikely that many more than Np nodes switch. This insight is the basis for the *Neighbor* algorithm.

The *Neighbor* algorithm[1] consists of approximating Bellman's equation (Eq. (3)) for each state by limiting the number of sub-states $K \in \{0, \dots, N\}$ that can change over the next time step. By using the Kronecker delta ($\delta_{s'_i s_i} = 1$ when $s'_i = s_i$ and 0 otherwise), this can be formulated as the constraint $\sum_{i=1}^{N} \delta_{s'_i s_i} \geq N - K$ on future states $s' \in S$ (Lines 6-7, Algorithm 1). When K is set to N, the *Neighbor* algorithm is equivalent to the value iteration algorithm. When K is less than N, fewer future states are accounted for in the calculation of the future expected values than in the standard value iteration. This simplification decreases the computational complexity (see Proposition 2) but also decreases the precision (see Proposition 1).

In addition, the number of iterations is set to the variable H (Line 2). Similarly to K, this variable can be tuned depending on the desired precision and computational complexity of the algorithm (Propositions 1 and 2). In our experiments, we chose $H = 10$ and $K = 4$, for example. The policy returned by the *Neighbor* algorithm will be evaluated on the real, complete problem during the simulations.

We aim to find an upper bound for the error incurred by the *Neighbor* algorithm (Algorithm 1) as opposed to the optimal value iteration. To do so, we denote by:

- π_N the policy returned by the *Neighbor* algorithm;
- V_π the exact value function of any policy $\pi : S \to A$, i.e.

$$V_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) | s_0 = s]; \tag{9}$$

  Note that V_{π_N} is the *exact* value function of the policy π_N, and likely differs from the approximate value function V computed in the *Neighbor* algorithm.

- p is the maximum probability a node will change in one time step:

$$p := \max\left(\max_{a \in A, 1 \leq i \leq N} a_i, \max_{1 \leq i \leq N} (1 - \prod_{j \neq i} (1 - p_{ji})) \right) \tag{10}$$

- $R_{max} := \max_{s \in S, a \in A} r(s, a)$ the maximum reward (we assume all rewards are nonnegative);

Proposition 1. *We assume that $K \geq Np$. We have:*

$$||V_{\pi^*} - V_{\pi_N}||_\infty \tag{11}$$

$$\leq \frac{R_{max} \gamma^H}{1 - \gamma} + \frac{R_{max} \gamma \exp\left(-2 \frac{(K+1-Np)^2}{N}\right)}{(1 - \gamma)^2} \tag{12}$$

Proof. See Appendix. □

This proposition shows that increasing K (the maximum number of changing sub-states) or H (the number of iterations) will reduce the loss incurred when implementing the policy π_N returned by the *Neighbor* algorithm instead of the optimal policy π^*.

Proposition 2. *The Neighbor algorithm runs in $O(H 2^N |A| N^K)$ operations, as opposed to $O(H 4^N |A|)$ for value iteration.*

Note that the number of actions $|A|$ will likely depend on N as well. Here, because the number of actions grows only polynomially due to the budgetary constraint, we focus on the number of states (first and second curses of dimensionality).

Proof. The first three for-loops of the algorithm are over H iterations, 2^N states and $|A|$ actions. The number of times the last for-loop is computed equals

$$\sum_{k=0}^{K} \binom{N}{k} = \sum_{k=0}^{K} O(N^k) = O(N^K) \tag{13}$$

□

This proposition shows in particular that increasing the value of K or H will increase the computational complexity of the *Neighbor* algorithm. Taken together, Propositions 1 and 2 show that one can trade off performance and computational expense by varying the parameters K and H. The complexity is still exponential in the number of current states but not in the number of future states. The second curse of dimensionality is circumvented[2]. This algorithm should run faster than value iteration, but still falls prey to the first curse of dimensionality. Our second approximate algorithm avoids this caveat.

The *Neighbor* algorithm is related to approximate dynamic programming [26] or approximate value iteration [33]. The difference with these classes of algorithms is that the *Neighbor* algorithm does not use an approximate representation of the value function such as a linear approximation. Instead, the approximation occurs in the probabilities involved in Bellman's equation (Eq. (3)). Also, the *Neighbor* algorithm uses expected value calculations instead of sampling, which is common in reinforcement learning [2, 35].

[1] The terminology *Neighbor* does not refer to nearby islands or nodes (geographically), but to MDP states that are similar. In this regard, this algorithm shares similarities with reduced MDP approaches (see [24] for an example).

[2] For increasing values of N, K needs to be increased to ensure that $K \geq Np$ is satisfied, which also increases the complexity. However, we show in Appendix B that the number of future states computed by the *Neighbor* algorithm is negligible compared to that of value iteration when N grows to infinity.

Algorithm 1 Neighbor(K, H)

1: **Initialization**: $V(s) = 0\ for\ all\ states\ s \in S$
2: **for** $iter = 0 : H - 1$ **do**
3: **for** $s \in S$ **do**
4: **for** $a \in A$ **do**
5: $Q(a) = r(s, a)$
6: **for** $s' \in S, \sum_{i=1}^{N} \delta_{s'_i s_i} \geq N - K$ **do**
7: $Q(a) = Q(a) + \gamma P(s'|s, a)V(s')$
8: $\pi(s) = \arg\max_{a \in A} Q(a)$
9: $V'(s) = \max_{a \in A} Q(a)$
10: $V(s) = V'(s)\ for\ all\ states\ s \in S$
Output: Policy π_N

2.6 Second approximate approach: the *Continuous* algorithm

Our second approach, which we refer to as the *Continuous* algorithm, is an online algorithm (Algorithm 2): it only provides an action to implement in the current state. Thus, it avoids listing the states altogether and overcomes the first curse of dimensionality. This stands in contrast with the first approximate approach and dynamic programming, which return the entire policy for all states before implementation. The *Continuous* algorithm is a rollout algorithm, i.e. the values associated to different actions are evaluated through simulations over a moving time horizon of fixed duration H_c [18, 33].

As in the first approximate approach, this new approach is based on the observation that the probability for sub-states or nodes to change over one time step is small. As a consequence, the future MDP state will likely be similar to the previous state, if not identical. This implies that the same action will likely be applied multiple times. Thus, one can compare different actions by assuming that the action chosen will never change in the future: this establishes a first approximation (Lines 1-3, Algorithm 2). Then, the binary sub-state of each node, i.e. $s_i \in \{0, 1\}$ corresponding to susceptible or infested, is replaced by its (continuous) probability of infestation, i.e. $s_i \in [0, 1]$. Treating discrete entities as continuous in an SIS context is common in continuous time [4] but is not common when optimizing decisions. When s_i was binary, the calculations of future infestation probabilities were written as

$$\begin{cases} Pr(s'_i = 1|s_i = 1) = 1 - a_i, \\ Pr(s'_i = 1|s_i = 0) = 1 - \prod_{j=1}^{N}(1 - p_{ji}s_j), \end{cases} \tag{14}$$

where p_{ji} is the probability of transmission from j (if infested) to i. It can now be adapted to these continuous sub-states as follows (Line 6):

$$s'_i = s_i(1 - a_i) + (1 - s_i)(1 - \prod_{j=1}^{N}(1 - p_{ji}s_j)). \tag{15}$$

These continuous sub-states establish a second approximation. They are considerably faster to calculate than the probability of each of the 2^N combination of sub-states because the number of operations is quadratic in the number of nodes instead of exponential. However, these estimates are based on the probability of infestation of sub-states instead of using the precise conditional probabilistic relations between sub-states. Over many iterations, these estimates will diverge from the discrete case.

Similarly to the discrete case, we define the reward in the continuous case as follows: for the eradication objective, the reward at each time step is $\sum_{i=1}^{N}(1 - s_i)$, which represents the average number of susceptible nodes to maximize. For the containment objective, the reward is $1 - mainland$, i.e. the probability that the mainland is infested. These rewards are used to calculate $Q(a)$, the cumulative 'score' of action a (Line 5). Note that Line 7 applies to the containment objective only. At the end of the rolling horizon, the action with maximum score is selected (Line 9).

Proposition 3. *The Continuous algorithm runs in $O(|A||N|^2 H_c)$ operations.*

Proof. For each of the $|A|$ actions and H_c iterations, the sub-state of each of the N nodes is updated by multiplying $N-1$ numbers. □

Algorithm 2 Continuous(s)

1: **for** $a \in A$ **do**
2: **Initialization**: $Q(a) = 0$, $mainland = 0$, $(s_1, \ldots, s_N) := s$
3: **for** $iter = 0 : H_c - 1$ **do**
4: $Q(a) = Q(a) + \gamma^{iter} r(s, a)$
5: **for** $i = 1 \rightarrow N$ **do**
6: $s'_i = s_i(1 - a_i) + (1 - s_i)(1 - \prod_{j=1}^{N}(1 - p_{ji}s_j))$
7: $mainland = mainland + (1 - mainland)(1 - \prod_{i=1}^{N}(1 - p_{i,mainland}s_i))$ // containment case only
8: $s_i = s'_i\ for\ all\ 1 \leq i \leq N$
9: $a = \arg\max_{a \in A} Q(a)$
10: **Output**: Action a

The complexity of this approach is polynomial in the problem size, which is a considerable improvement as compared to stochastic dynamic programming. It circumvents both curses of dimensionality.

2.7 Performance evaluation

We can evaluate the performance of each algorithm through simulations by implementing the recommended action at each time step (Algorithm 3). Note that this is much faster for the first approach because the algorithm computes the policy for all states before the simulations, potentially at a very high one-off computational cost (that is, it is an offline algorithm). In contrast, the algorithm *Continuous* only outputs an action for one given state, so it needs to be re-run at every time step for the updated observed state (it is an online algorithm).

2.8 Framing the case study as an SIS-MDP

We will aim to find the optimal management of *Aedes albopictus*. This decision problem is modeled as an SIS-MDP in which:

- The observable component $s \in S$ specifies the presence or absence of the mosquitoes across the $N = 17$ islands ($|S| = 2^N + 1 = 131,073$). The term '+1' corresponds to an

Algorithm 3 EvaluatePolicy(s_0)

$t = 0$
for $i = 1 : nSimulations$ **do**
  Initialization: $s = s_0$, $mainland = 0$ // all islands are initially infested
  while $mainland = 0$ **do**
    $a = \pi(s)$ or $a = Continuous(s)$
    $mainland := DrawMainlandState(s, a)$
    $s := DrawState(s, a)$
    $t := t + 1$
Output: Average time: $t/nSimulations$

absorbing state representing the presence of mosquitoes in the mainland.

- Each action $a \in A$ describes which islands should be managed and the type of management (light or strong). Due to a budgetary constraint, only up to three islands can be managed simultaneously. The set A only contains the combinations of management actions that satisfy this constraint.
- The transition probabilities $T(s, a, s')$ accounts for the possible local eradications and transmissions between islands. In accordance with [22], we investigate two transmission rates: fast and slow.
- In the case of a containment objective, the reward $r(s, a)$ equals 0 if the mainland is infested and 1 otherwise; in the case of an eradication objective, the reward is the sum of susceptible islands — the mainland is disregarded;
- In the containment case, γ should ideally be 1 so the expected cumulative reward (value) equals the expected time before the infestation of Australia in years. For ease of comparison with SPUDD, we set $\gamma = 0.99$ for containment and $\gamma = 0.95$ for eradication.

3 RESULTS

We show the average result, standard deviation and computational time on various problem instances for both the *Neighbor* and *Continuous* algorithms (Table 1) on 10,000 simulations. We set $H = 10$, $K = 4$ and $H_c = 10$ as these parameters achieved a satisfying trade-off between computation time and performance. We compare their performances to SPUDD (version 3.6.2). We run our algorithms with 10 islands (for which the optimal value was calculated using the classic stochastic dynamic programming algorithm policy iteration in [22]), 17 islands (the full-scale problem) and a hypothetical network of 50 islands (to test scalability), and with different transmission parameters (high, low and random) and management objectives (eradication and containment). On 17 islands, SPUDD cannot accommodate a highly connected network, so for tractability we allow, for each island, transmissions only from the five islands with the highest transmission probabilities.

Both of our proposed algorithms are tractable for up to 17 islands for all objectives and transmissions. However, SPUDD runs out of memory or time, for the eradication objective and the random transmissions. The eradication objective is difficult to solve because all islands contribute to the rewards and thus to the value function. In contrast, the containment objective is simpler because keeping a few key islands susceptible might be enough to achieve a good performance. As for the random transmission case, it is harder to solve because the islands have random attributes. The 'low' and 'high' transmission probabilities are easier, e.g. Thursday and Horn islands have relatively high transmission probabilities to and from other islands and all solvers manage to rapidly identify them as management priorities.

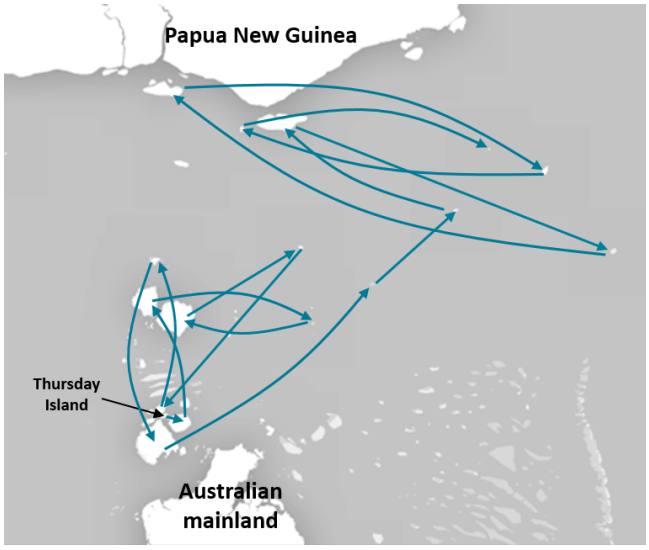

Figure 2: The prioritization ranking shown on the Torres Strait Islands. The recommendation is to start by managing Thursday Island and follow the arrows if mosquitoes are successfully eradicated. As a general rule of thumb, islands that are closer to the mainland are to be managed with priority. Other factors such as effectiveness of management actions and closeness to other Torres Strait Islands also account for this ranking.

In a limited system of 10 islands all solvers perform near-optimally. This shows that both our approximate algorithms are able to provide good policies and suggests that they might perform well on larger problems as well.

When all 17 islands are included, the three solvers obtain the same value with high transmission probabilities, with SPUDD being much faster than the *Neighbor* algorithm. However, SPUDD underperforms with 'low' transmission probabilities. This is because we limited SPUDD to only consider infestation from a maximum of five islands to ensure tractability. Under these conditions, SPUDD outputs a policy tree depending on a handful of islands. Therefore, it makes no recommendations about other islands, resulting in a loss of performance. Our approximate approaches outperform SPUDD and also show more robustness to the parameters of the problem.

Finally, only the *Continuous* algorithm is tractable with 50 islands, because it overcomes both curses of dimensionality and has a polynomial complexity (it would also be tractable for much larger problems for the same reasons, but we did not confirm this through experiments). In contrast, the *Neighbor* algorithm overcomes only one of those issues and still has an exponential complexity.

Instance	Neighbors $H = 10$, $K = 4$		Continuous $H_c = 10$		SPUDD	
#islands / transmission objective ($\|S\|/\|A\|$)	Value ± 95% confidence	Time	Value ± 95% confidence	Time	Value ± 95% confidence	Time
10 / high / containment (1,025/276) **Optimal: 13.6** [22]	**13.5 ± 0.3**	60 s	**13.6 ± 0.3**	2,441 s	**13.5 ± 0.3**	4,631 s
10 / low containment (1,025/276) **Optimal: 65.1** [22]	**64.3 ± 1.4**	59 s	**64.9 ± 1.4**	4,406 s	**65.2 ± 1.4**	10,546 s
17 / high / containment (131,073/1,123)	**12.4 ± 0.3**	172,222 s	**12.8 ± 0.3**	12,160s	**12.6 ± 0.3** (*)	207 s (*)
17 / low / containment (131,073/1,123)	54.9 ± 1.2	183,572 s	**57.2 ± 1.2**	20,151 s	54.9 ± 1.2 (*)	955 s (*)
17 / random / containment (131,073/1,123)	**16.4 ± 0.4**	158,260 s	**16.8 ± 0.4**	13,449 s	Out of memory	
17 / high / eradication (131,073/1,123)	**94.6 ± 0.3**	156,544 s	**95.0 ± 0.3**	57,100 s		Out of time
17 / low / eradication (131,073/1,123)	**124.7 ± 0.4**	156,479 s	**125.1 ± 0.4**	29,378 s		Out of time
17 / random / eradication (131,073/1,123)	78.5 ± 0.4	172,258 s	**79.8 ± 0.4**	64,242s	Out of memory	
50 / random containment (2.25×10^{15}, 23, 376)	Out of memory		**5.4 ± 0.3**	164,225 s (1,000 simulations)	Out of memory	

Table 1: Average values, 95% confidence intervals and computational times of the two approximate approaches and SPUDD on 10,000 simulations. Best values are shown in bold. (*) SPUDD was run on an approximate version of the 17-island problem for tractability (see main text). For the offline solvers (the *Neighbor* algorithm and SPUDD), the computational time shown does not include the simulation running time, which is very short. In contrast, the *Continuous* algorithm is online, so the computational time is only that of the simulations (no preprocessing). The memory is set to 2GB, which is the maximum memory supported by SPUDD. The computational time limit was set to 1 week (604,800s).

Since the *Continuous* algorithm performs well, we show in Table 2 and in Figure 2 which islands should be managed in priority in the containment case according to this approach. Under the containment objective, it recommends managing Thursday, Horn and Mulgrave Islands with priority if they are infested. These islands are highly populated and close to mainland Australia and therefore have the highest probability of directly transmitting mosquitoes to the mainland. This matches the recommendations found in [22].

4 DISCUSSION

In this manuscript, we aimed to solve an MDP on a large and fully connected SIS network. Given the intractability of stochastic dynamic programming, we propose two new approximate approaches based on the observation that the transition probabilities for each node is low. The first approach is a modified version of value iteration where only those future states that are similar to the current state are accounted for, with provable performance guarantees. This drastically reduces the computational time of Bellman's equation at little cost on the quality of the policy. The second approach goes further by modeling the sub-states comprising the MDP states as continuous instead of binary, with an adapted Bellman's equation.

Both approaches solve all versions of this case study, which policy iteration and SPUDD could not. The *Neighbor* algorithm solves the second curse of dimensionality on future states. The *Continuous* algorithm also circumvents the first curse of dimensionality on current states. Both approaches could handle completely connected SIS-MDP networks of size at least 17 (*Neighbors*) and 50 nodes (*Continuous*). While it was not possible to evaluate loss of optimality on the 17-island problem because it is intractable for established techniques, our algorithms achieved near-optimal performance on the 10-island problem. The lower performance of SPUDD is not surprising as SPUDD takes advantage of conditional independence between sub-state variables [12]. In our problem, all sub-state variables are conditionally dependent because we are dealing with a complete network.

Although both of our new proposed approaches share some similarities, they also differ on several points. The advantage of the *Neighbor* algorithm is that it accounts for a small amount of sub-state changes (K), while the *Continuous* algorithm does not. Additionally, the *Neighbor* algorithm can trade-off computational time for policy quality by increasing/decreasing the number of iterations (H) or the number of following states (K). The extreme case, i.e. setting the number of changes allowed to the total amount of nodes ($K = N$), is equivalent to performing value iteration given H is large enough. It is an offline algorithm, which is easier to communicate to managers since the solution is calculated once. The *Continuous* algorithm is online and each simulation only takes a few seconds to run. In our case this approach is fast and outperforms the *Neighbor* algorithm. However, it comes with no performance

Ranking	Island name	Ranking	Island name
1	Thursday	10	Coconut
2	Horn	11	Yorke
3	Mulgrave	12	Saibai
4	Sue	13	Murray
5	Banks	14	Talbot
6	Yam	15	Darnley
7	Hammond	16	Mt Cornwallis
8	Jervis	17	Stephens
9	Prince of Wales		

Table 2: Priority ranking of the 17 Torres Strait Islands for the *Continuous* algorithm with the containment objective with both low and high transmission probabilities. At each time step, only the two or three infested islands with highest ranking are managed, due to a limited budget. Note that this ranking is not unique: when Thursday, Horn and Mulgrave islands are susceptible, the *Continuous* algorithm recommends managing Sue, Yam and Jervis islands with 3 light managements. However, if Sue island becomes susceptible, the *Continuous* algorithm recommends managing Banks and Yam islands with strong and light management respectively. It is then unclear whether Banks is more of a priority than Jervis. Nevertheless, the prioritization ranking we present is accurate for most islands and provides a good idea of which islands should be managed first.

guarantees and has the disadvantage of not accounting for a change of action in the future: it might perform poorly on systems that rely on changing actions significantly within a short time. Finally, the *Neighbor* algorithm retains exponential complexity in the number of nodes in the network while the *Continuous* algorithm is polynomial.

This work provides many avenues for future research. First, we have developed our equations for SIS-MDPs, however the *Neighbor* algorithm could be applied to more general factored MDPs with only minor changes. Second, the *Continuous* algorithm works well when sub-states do not change frequently. To apply this algorithm efficiently to more difficult cases, it may be necessary to allow actions to change in the future. This might be achieved while keeping computational complexity down by considering actions for the most likely future states, using for example smart sampling techniques. Third, the *Neighbor* algorithm could be converted to an online form by only considering states that can be reached from the current state. This would avoid running the entire algorithm prior to simulations. Also, we have built this algorithm as approximate version of value iteration but it would be interesting to design and evaluate a policy iteration version. Finally, we acknowledge that there are many algorithms that would be appropriate in this context, e.g. Reinforcement Learning [35]. However, they do not natively assume or exploit that sub-states do not change frequently. Tailoring these reference algorithms for this property may lead to considerable computational savings.

This work could be applied in multiple fields. There are many environmental spatial problems requiring effective MDP solvers on highly connected networks. Examples include management of forestry at risk of wind damage [7], adaptive management of migratory birds under sea level rise [17] or control of invasive mammals [9] or invasive weed [5]. Other non-ecological interconnected systems would also benefit from this work. For example, system administrators try to keep as many machines as possible running in a network [25] or in maximizing the reliability of information in a military sensor network [8].

Inspired by the problem of best managing the invasive mosquito *Aedes albopictus* in Australia, we aimed at solving a Markov decision process on large Susceptible-Infected-Susceptible (SIS) networks that are highly connected. Current exact approaches are intractable for these types of networks. We have proposed two approximate algorithms that can tackle such large-scale problems and achieve promising results, and we have provided some theoretical insights about their performances. Although our two approximate approaches are not guaranteed to be optimal, the resulting policies can still be used as an initial policy or a basis for comparison with other algorithms.

5 ACKNOWLEDGMENTS

This research is supported by an Industry Doctoral Training Centre scholarship (MP). We thank Sam Nicol and Dan Pagendam for their valuable feedback.

REFERENCES

[1] Richard Bellman. 1957. Dynamic Programming. *Princeton University Press* (1957).
[2] Dimitri P. Bertsekas and John N. Tsitsiklis. 1995. Neuro-Dynamic Programming: An Overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference On*, Vol. 1. IEEE, 560–564.
[3] Iadine Chadès, Tara G. Martin, Sam Nicol, Mark A. Burgman, Hugh P. Possingham, and Yvonne M. Buckley. 2011. General Rules for Managing and Surveying Networks of Pests, Diseases, and Endangered Species. *Proceedings of the National Academy of Sciences of the United States of America* 108 (2011), 8323–8328.
[4] Peter G. Fennell, Sergey Melnik, and James P. Gleeson. 2016. Limitations of Discrete-Time Approaches to Continuous-Time Contagion Dynamics. *Physical Review E* 94, 5 (2016), 052125.
[5] Jennifer Firn, Tracy Rout, Hugh Possingham, and Yvonne M. Buckley. 2008. Managing beyond the Invader: Manipulating Disturbance of Natives Simplifies Control Efforts. *Journal of Applied Ecology* 45 (2008), 1143–1151. https://doi.org/10.1111/j.1365-2664.2008.01510.x
[6] Nicklas Forsell and Régis Sabbadin. 2006. Approximate Linear-Programming Algorithms for Graph-Based Markov Decision Processes. *Frontiers in Artificial Intelligence and Applications* 141 (2006), 590.
[7] Nicklas Forsell, Peder Wikström, Frédérick Garcia, Régis Sabbadin, Kristina Blennow, and Ljusk Ola Eriksson. 2011. Management of the Risk of Wind Damage in Forestry: A Graph-Based Markov Decision Process Approach. *Annals of Operations Research* 190 (2011), 57–74.
[8] Duncan Gillies, David Thornley, and Chatschik Bisdikian. 2009. Probabilistic Approaches to Estimating the Quality of Information in Military Sensor Networks. *Comput. J.* 53, 5 (2009), 493–502.
[9] Kate J. Helmstedt, Justine D. Shaw, Michael Bode, Aleks Terauds, Keith Springer, Susan A. Robinson, and Hugh P. Possingham. 2016. Prioritizing Eradication Actions on Islands: It's Not All or Nothing. *Journal of Applied Ecology* 53, 3 (2016), 733–741.
[10] Christopher Ho, Mykel J. Kochenderfer, Vineet Mehta, and Rajmonda S. Caceres. 2015. Control of Epidemics on Graphs. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference On*. IEEE, 4202–4207.
[11] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.
[12] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. 1999. SPUDD: Stochastic Planning Using Decision Diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 279–288.

[13] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. 1995. On the Complexity of Solving Markov Decision Problems. Morgan Kaufmann Publishers Inc., 394–402.

[14] Alun L. Lloyd and Robert M. May. 2001. How Viruses Spread among Computers and People. *Science* 292, 5520 (2001), 1316–1317.

[15] László Lovász, József Pelikán, and Katalin L. Vesztergombi. 2003. *Discrete Mathematics.* Springer, Secaucus, NJ.

[16] Marissa F. McBride, Kerrie A. Wilson, Michael Bode, and Hugh P. Possingham. 2007. Incorporating the Effects of Socioeconomic Uncertainty into Priority Setting for Conservation Investment. *Conservation Biology* 21, 6 (2007), 1463–1474.

[17] Sam Nicol, Olivier Buffet, Takuya Iwamura, and Iadine Chadès. 2013. Adaptive Management of Migratory Birds under Sea Level Rise. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence.* AAAI Press, Beijing, China, 2955–2957.

[18] Sam Nicol and Iadine Chadès. 2011. Beyond Stochastic Dynamic Programming: A Heuristic Sampling Method for Optimizing Conservation Decisions in Very Large State Spaces. *Methods in Ecology and Evolution* 2 (2011), 221–228. https://doi.org/10.1111/j.2041-210X.2010.00069.x

[19] Sam Nicol, Iadine Chadès, Simon Linke, and Hugh P. Possingham. 2010. Conservation Decision-Making in Large State Spaces. *Ecological Modelling* 221, 21 (2010), 2531–2536.

[20] Sam Nicol, Regis Sabbadin, Nathalie Peyrard, and Iadine Chadès. 2017. Finding the Best Management Policy to Eradicate Invasive Species from Spatial Ecological Networks with Simultaneous Actions. *Journal of Applied Ecology* (2017).

[21] Romualdo Pastor-Satorras and Alessandro Vespignani. 2001. Epidemic Spreading in Scale-Free Networks. *Physical review letters* 86, 14 (2001), 3200.

[22] Martin Péron, Cassie C. Jansen, Chrystal Mantyka-Pringle, Sam Nicol, Nancy A. Schellhorn, Kai Helge Becker, and Iadine Chadès. 2017. Selecting Simultaneous Actions of Different Durations to Optimally Manage an Ecological Network. *Methods in Ecology and Evolution* 8, 10 (2017), 1332–1341.

[23] Nathalie Peyrard and Régis Sabbadin. 2006. Mean Field Approximation of the Policy Iteration Algorithm for Graph-Based Markov Decision Processes. *Frontiers in Artificial Intelligence and Applications* 141 (2006), 595.

[24] Luis Enrique Pineda and Shlomo Zilberstein. 2014. Planning Under Uncertainty Using Reduced Models: Revisiting Determinization. In *ICAPS.*

[25] Pascal Poupart. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes.* Ph.D. Dissertation. University of Toronto, Toronto.

[26] Warren B. Powell. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Vol. 703. John Wiley & Sons, Inc., New York, NY, USA.

[27] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., New York, NY, USA.

[28] Olivier Restif and Jacob C. Koella. 2003. Shared Control of Epidemiological Traits in a Coevolutionary Model of Host-Parasite Interactions. *The American Naturalist* 161, 6 (2003), 827–836.

[29] Scott A. Ritchie, Peter Moore, Morven Carruthers, Craig Williams, Brian Montgomery, Peter Foley, Shayne Ahboo, Andrew F. Van Den Hurk, Michael D. Lindsay, and Bob Cooper. 2006. Discovery of a Widespread Infestation of Aedes Albopictus in the Torres Strait, Australia. *Journal of the American Mosquito Control Association* 22 (2006), 358–365.

[30] Faryad Darabi Sahneh, Fahmida N. Chowdhury, and Caterina M. Scoglio. 2012. On the Existence of a Threshold for Preventive Behavioral Responses to Suppress Epidemic Spreading. *Scientific reports* 2 (2012).

[31] Scott Sanner and David McAllester. 2005. Affine Algebraic Decision Diagrams (AADDs) and Their Application to Structured Probabilistic Inference. In *IJCAI,* Vol. 2005. 1384–1390.

[32] Daniel Sheldon, Bistra Dilkina, Adam N. Elmachtoub, Ryan Finseth, Ashish Sabharwal, Jon Conrad, Carla P. Gomes, David Shmoys, William Allen, and Ole Amundsen. 2012. Maximizing the Spread of Cascades Using Network Design. *arXiv preprint arXiv:1203.3514* (2012).

[33] Olivier Sigaud and Olivier Buffet. 2010. *Markov Decision Processes in Artificial Intelligence.* John Wiley & Sons, Inc., New York, NY, USA.

[34] James C. Spall. 2005. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* Vol. 65. John Wiley & Sons.

[35] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning.* MIT Press.

[36] Shan Xue, Alan Fern, and Daniel Sheldon. 2014. Dynamic Resource Allocation for Optimizing Population Diffusion. In *Artificial Intelligence and Statistics.* 1033–1041.

A PROOF OF PROPOSITION 1

We prove the following proposition:

Proposition 1. *We assume that $K \geq Np$. We have:*

$$||V_{\pi^*} - V_{\pi_N}||_\infty \tag{16}$$

$$\leq \frac{R_{max}\gamma^H}{1-\gamma} + \frac{R_{max}\gamma \exp\left(-2\frac{(K+1-Np)^2}{N}\right)}{(1-\gamma)^2} \tag{17}$$

Let us denote by V_π^N the value as calculated in the the *Neighbor* algorithm (Algorithm 1) of any policy $\pi : S \to A$, i.e. $V_\pi^N(s) = Q(\pi(s))$ for each $s \in S$. Let us first prove the following lemma.

Lemma 1. *For any policy $\pi : S \to A$ and any state $s \in S$, we have:*

$$0 \leq V_\pi(s) - V_\pi^N(s) \tag{18}$$

$$\leq \frac{R_{max}\gamma^H}{1-\gamma} + \frac{R_{max}\gamma \exp\left(-2\frac{(K+1-Np)^2}{N}\right)}{(1-\gamma)^2} \tag{19}$$

Proof.

$$V_\pi(s) - V_\pi^N(s) \tag{20}$$

$$= \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r(s_t, \pi(s_t))|s_0 = s\right] \tag{21}$$

$$- \mathbb{E}\left[\sum_{0\leq t\leq H-1, \sum_{i=1}^N \delta_{s_{t+1,i}s_{t,i}} \geq N-K} \gamma^t r(s_t, \pi(s_t))|s_0 = s\right] \tag{22}$$

$$= \mathbb{E}\left[\sum_{t\geq H} \gamma^t r(s_t, \pi(s_t))|s_0 = s\right] \tag{23}$$

$$+ \mathbb{E}\left[\sum_{0\leq t\leq H-1, \sum_{i=1}^N \delta_{s_{t+1,i}s_{t,i}} < N-K} \gamma^t r(s_t, \pi(s_t))|s_0 = s\right] \tag{24}$$

This sum is nonnegative because all rewards are nonnegative, which proves the first inequality of the lemma. For the second inequality, we can write:

$$V_\pi(s) - V_\pi^N(s) \tag{25}$$

$$\leq R_{max}\mathbb{E}\left[\sum_{t\geq H} \gamma^t |s_0 = s\right] + \tag{26}$$

$$R_{max}\mathbb{E}\left[\sum_{t\geq 0, \sum_{i=1}^N \delta_{s_{t+1,i}s_{t,i}} < N-K} \gamma^t |s_0 = s\right] \tag{27}$$

$$= \frac{R_{max}\gamma^H}{1-\gamma} + \tag{28}$$

$$R_{max}\left(\sum_{t\geq 0} \gamma^t - \mathbb{E}\left[\sum_{t\geq 0, \sum_{i=1}^N \delta_{s_{t+1,i}s_{t,i}} \geq N-K} \gamma^t |s_0 = s\right]\right) \tag{29}$$

Recall that $\sum_{i=1}^N \delta_{s_{t+1,i}s_{t,i}}$ is the number of sub-states (nodes) that do not change from time step t to time step $t+1$. With s_t fixed and s_{t+1} following the Markov process, this sum is a random variable. It is equivalent to a binomial distribution with N independent experiments, each with a probability of success of $1-p$ at least.

So, the probability that a state s_t and its successor s_{t+1} satisfy $\sum_{i=1}^{N} \delta_{s_{t+1,i} s_{t,i}} \geq N - K$ is, by Hoeffding's inequality [11], no less than

$$P_K := 1 - \exp\left(-2\frac{(K+1-Np)^2}{N}\right) \tag{30}$$

This result is based on our assumption that $K \geq Np$. It implies that $(P_K)^t$ is a lower bound of the probability that all states from s_0 to s_t satisfy this property, which are the states involved in the second sum of Eq. (29). So,

$$V_\pi(s) - V_\pi^N(s) \tag{31}$$

$$\leq \frac{R_{max}\gamma^H}{1-\gamma} + R_{max}\left(\sum_{t\geq 0}\gamma^t - \sum_{t\geq 0}\gamma^t (P_K)^t\right) \tag{32}$$

$$= \frac{R_{max}\gamma^H}{1-\gamma} + R_{max}\left(\frac{1}{1-\gamma} - \frac{1}{1-\gamma P_K}\right) \tag{33}$$

$$\leq \frac{R_{max}\gamma^H}{1-\gamma} + \left(\frac{R_{max}\gamma(1-P_K)}{(1-\gamma)^2}\right) \tag{34}$$

$$= \frac{R_{max}\gamma^H}{1-\gamma} + \frac{R_{max}\gamma \exp\left(-2\frac{(K+1-Np)^2}{N}\right)}{(1-\gamma)^2} \tag{35}$$

which terminates the proof of the lemma. □

Then we have, for each state $s \in S$:

$$0 \leq V_{\pi^*}(s) - V_{\pi_N}(s) \tag{36}$$

$$\leq \left[V_{\pi^*}(s) - V_{\pi^*}^N(s)\right] + \left[V_{\pi^*}^N(s) - V_{\pi_N}^N(s)\right] + \left[V_{\pi_N}^N(s) - V_{\pi_N}(s)\right] \tag{37}$$

The second term between brackets is non-positive because the policy π_N is optimal with regard to the value function V^N. The first and third terms between brackets are bounded by the right and left inequalities in Lemma 1 respectively, which yields:

$$V_{\pi^*}(s) - V_{\pi_N}(s) \tag{38}$$

$$\leq \frac{R_{max}\gamma^H}{1-\gamma} + \frac{R_{max}\gamma \exp\left(-2\frac{(K+1-Np)^2}{N}\right)}{(1-\gamma)^2} \tag{39}$$

This terminates the demonstration of Proposition 1.

B FUTURE STATES IN THE *NEIGHBOR* ALGORITHM

In this section, we show that, under the assumption $p < 1/2$, the number of future states computed by the *Neighbor* algorithm is negligible compared to that of value iteration when N grows to infinity, for any desired precision. Let us denote by $F(N)$ the number of future states computed by the *Neighbor* algorithm with N nodes. Since the number of future states computed by value iteration is 2^N, we want to find an upper bound on $F(N)/2^N$.

Proposition 1 shows that the loss of performance due to restricting the future states through the variable K is at most:

$$\frac{R_{max}\gamma \exp\left(-2\frac{(K+1-Np)^2}{N}\right)}{(1-\gamma)^2} \tag{40}$$

So, we can ensure that this loss is below any given threshold $\rho > 0$ by setting

$$K = Np - 1 + \sqrt{\frac{N \log\left(\frac{R_{max}\gamma}{(1-\gamma)^2\rho}\right)}{2}} \tag{41}$$

$$= Np - 1 + C\sqrt{N}, \tag{42}$$

with the notation $C = \sqrt{\frac{\log\left(\frac{R_{max}\gamma}{(1-\gamma)^2\rho}\right)}{2}}$.

Theorem 5.3.2 in [15] provides probabilistic bounds on coin tosses, which can be related to bounds on sums of binomial coefficients. This yields the following upper bound on $F(N)$:

$$F(N) \leq 2^{N-1} e^{-\frac{(N-2K-2)^2}{4(N-K-1)}} \tag{43}$$

So,

$$\frac{F(N)}{2^N} \leq \frac{1}{2} e^{-\frac{(N-2K-2)^2}{4(N-K-1)}} \tag{44}$$

Further, we have

$$\frac{(N-2K-2)^2}{4(N-K-1)} \geq \frac{(N-2K-2)^2}{4N} \tag{45}$$

$$= \frac{(N-2(Np-1+C\sqrt{N})-2)^2}{4N} \tag{46}$$

$$= \frac{(\sqrt{N}(1-2p)-2C)^2}{4} \xrightarrow{N\to\infty} \infty \tag{47}$$

$$\tag{48}$$

under the assumption that $p < 1/2$. This implies

$$\frac{F(N)}{2^N} \xrightarrow{N\to\infty} 0, \tag{49}$$

i.e. the number of future states computed by the *Neighbor* algorithm is negligible compared to that of value iteration when N grows to infinity.