Maximizing the Spread of Cascades Using Network Design

Daniel Sheldon[*], **Bistra Dilkina, Adam Elmachtoub**[†], **Ryan Finseth, Ashish Sabharwal, Jon Conrad, Carla Gomes and David Shmoys**
Cornell University, Ithaca, NY

Will Allen, Ole Amundsen and Buck Vaughan
The Conservation Fund, Arlington, VA

Abstract

We introduce a new optimization framework to maximize the expected spread of cascades in networks. Our model allows a rich set of actions that directly manipulate cascade dynamics by adding nodes or edges to the network. Our motivating application is one in spatial conservation planning, where a cascade models the dispersal of wild animals through a fragmented landscape. We propose a mixed integer programming (MIP) formulation that combines elements from network design and stochastic optimization. Our approach results in solutions with stochastic optimality guarantees and points to conservation strategies that are fundamentally different from naive approaches.

1 Introduction

Many natural processes — such as the diffusion of information in a social network or the spread of animals through a fragmented landscape — can be described as a network diffusion process or *cascade*. For example, an individual who buys a new product or adopts a new technology may trigger similar behavior in friends; if this process gains momentum it can cascade through a significant portion of a social network. The study of cascading behavior in networks is most familiar in social or epidemiological settings [3, 4, 8, 12]. However, a similar framework called *metapopulation modeling* also exists in ecology to describe the occupancy pattern of habitat patches in a fragmented landscape [9].

One would often like to intervene to steer the course of a cascade toward some goal, e.g., to maximize its spread through the network. However, typical interventions (e.g., choosing where to initiate a cascade) do not *directly* affect the cascade outcome; the cascade is a complex stochastic process that depends only *probabilistically* on the choices made. Hence there is great uncertainty as to the actual effect of any intervention. In this paper, we contribute a new optimization framework to maximize the expected spread of a cascade under a very general class of interventions.

The problem of maximizing the spread of a cascade is of great practical import. In the social network setting, Domingos and Richardson asked the question of which individuals to target to maximize the effectiveness of a viral marketing strategy [6]. Kempe, Kleinberg and Tardos later showed that in the context of several different cascade models, this problem — selecting the set of k individuals to initiate a new cascade that will maximize its eventual spread — is a submodular optimization problem [10]. Submodularity implies that it can be solved to near optimality by a greedy approach.

In a conservation setting, one would also like to maximize the spread of a cascade (i.e., the spread of a conservation-target species through the landscape) given a limited budget for management intervention. In this case, the management tools consist of augmenting the network of habitat patches through conservation or land acquisition. Unlike the social network example, the initial locations for the cascade are predetermined by the current spatial distribution of the species, which is difficult to manipulate. Moreover, metapopulation models predict that long-term population dynamics are determined by properties of the landscape much more so than initial occupancy [17].

Motivated by this application, we introduce a much more general optimization framework for cascades. In our model, one may choose from a rich set of *management actions* to manipulate a cascade: in addition to choosing where to initiate the cascade, these actions may also intervene directly in the network to change cascade dynamics by adding nodes. The model is gen-

[*]Present affiliation: Oregon State University
[†]Present affiliation: Massachussets Intitute of Technology

eral enough to also capture adding edges, or increasing the local probability of propagating the cascade.

The objective function is no longer submodular with respect to this more general decision space, so optimization becomes more difficult. We formulate the problem as mixed integer program (MIP) that combines elements from deterministic network design problems and stochastic optimization.

One of our main computational tools is *sample average approximation* (SAA): we find a solution that is optimal in hindsight for a number of *training cascades* that are simulated in advance. The SAA may overfit for a small set of training cascades, but converges to the true optimum with increasing training samples, and provides a statistical bound on the optimality gap. Moreover, because the set of training cascades are known prior to optimization, SAA allows significant computational savings compared with other simulation-based optimization methods.

In addition to the MIP-based SAA approach, we contribute a set of preprocessing techniques to reduce computation time for SAA and other algorithms that repeatedly reason about a fixed set of training cascades. Our method compresses each training cascade into a smaller cascade that is identical for reasoning about the effects of management actions; in our instances, resulting in a small fraction of the original size. Our experiments show that preprocessing greatly reduces running times both for the SAA approach and two greedy baselines adapted from [10] and [13].

We apply our model to a sustainability problem that is part of an ongoing collaboration with The Conservation Fund to optimize the conservation of land to assist in the recovery of the Red-cockaded Woodpecker (RCW), a federally listed rare and endangered species. Unlike heuristic methods that are often used in conservation planning, our method directly models desired conservation outcome. For the RCW problem, we find solutions with stochastic optimality guarantees demonstrating conservation strategies that are fundamentally different from naive approaches.

2 Problem Statement

We begin by stating the generic optimization problem for *progressive* cascades; these serve as the foundation for all of our modeling. Specifics of our conservation application — which uses a variant called a *non-progressive cascade* — are given in Section 2.3.

A progressive cascade is a stochastic process in a graph $G = (V, E)$ that begins with an initial set of active nodes which proceed to activate new nodes over time, according to local activation rules among neighbors, until no more activations are possible. Given a limited management budget and a set of target nodes, we wish to select a set of *management actions* that affect node activations to maximize the expected number of target nodes that become active.

Let $\mathcal{A} = \{1, \ldots, L\}$ be a set of management actions, and let c_ℓ be the cost of action ℓ. Let $\mathbf{y}$ be a *strategy* vector that indicates which actions are to be taken: $\mathbf{y}$ is a 0-1 vector where $y_\ell = 1$ if and only if action ℓ is taken. Strategy $\mathbf{y}$ results in a particular cascade process based on the specification of the cascade model and management actions. Let $\{X_v(\mathbf{y})\}_{v \in V}$ be random variables capturing the outcome of a cascade under strategy $\mathbf{y}$ — the variable $X_v(\mathbf{y})$ is 0 or 1 indicating whether or not v is activated. Let B be the budget, and let $\mathcal{T}$ be a set of target nodes. The formal problem statement is

$$\max_{\mathbf{y}} \sum_{v \in \mathcal{T}} E[X_v(\mathbf{y})] \;\; \text{s.t.} \sum_{\ell=1}^{L} c_\ell y_\ell \leq B. \tag{1}$$

In the remainder of this section we discuss the details of the cascade model and management actions.

2.1 Cascade Model

Our model is the *independent cascade model* of [8,10]. Consider a graph $G = (V, E)$ with *activation probabilities* p_{vw} for all $(v, w) \in E$. Notationally, we sometimes write $p(v, w)$ for clarity, and adopt the convention that $p_{vw} = 0$ for $(v, w) \notin E$.

A (progressive) cascade proceeds is a sequence of *activations*. To begin, each node in a given source set $\mathcal{S}$ is activated. When any node v is *first* activated, it has a single chance to activate each neighbor w. It succeeds with probability p_{vw} independent of the history of the process so far. If the attempt succeeds and w was not already active, then w becomes newly activated, and will have a chance to activate its neighbors in subsequent rounds. If the attempt fails, v will never attempt to activate w again. Pending activation attempts are sequenced arbitrarily.

In their proofs, Kempe, Kleinberg and Tardos [10] argue that the following procedure is an equivalent, albeit impractical, way of simulating a cascade. For each edge $(v, w) \in E$, flip a coin with probability p_{vw} to decide whether the edge is *live*. Then the set of activated nodes are those that are reachable from $\mathcal{S}$ by live edges. This is tantamount to simulating the cascade, but flipping the coins for each possible activation attempt in advance. Let the subgraph G' consisting of live edges be called the *cascade graph*. The SAA method presented in Section 4 will optimize over a fixed set of cascade graphs that are simulated in advance.

Non-progressive cascades. In a progressive cascade, an activated node remains so forever (even though it does not attempt further activations of its neighbors). This is appropriate for social settings where a person adopts a new technology or buys a new product only once. However, patch occupancy in a metapopulation is *non-progressive*: empty habitat patches may become occupied and then unoccupied again many times, and occupied patches may colonize others during any time step, not only upon their first activation. To model this, suppose that all activations are batched in rounds corresponding to discrete time steps, and that an active node i has probability β_i of becoming inactive during each time step.

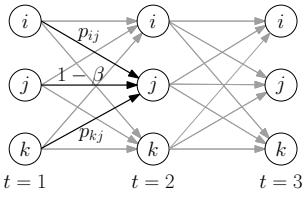


Figure 1: Cascade in layered graph

Kempe et al. [10] showed how to reduce this to the progressive case by representing a non-progressive cascade in graph $H = (V, E)$ as a progressive cascade in the *layered graph* $G = (V^T, E')$, where nodes are replicated for each time step, and the nodes at time t connect only to those at time $t+1$ (see Figure 1). Let $v_{i,t} \in V^T$ represent the node $i \in V$ at time t. The non-progressive cascade in H is equivalent to a progressive cascade in G with probabilities:

$$p(v_{i,t}, v_{j,t+1}) = p_{ij}, \quad p(v_{i,t}, v_{i,t+1}) = 1 - \beta_i,$$

and $p(v, w) = 0$ for all other $v, w \in V^T$. We think of this as node i at time t activating *itself* at time $t+1$ with probability $1-\beta_i$, and failing to do so with probability β_i so that it becomes inactive. (However it will remain active if it is also activated by a neighbor in the same time step).

2.2 Management Actions

Management actions in our model consist of predefined sets of nodes that may be added to the network for a cost. Consider the problem of augmenting a graph G_0 on vertex set V_0 to optimize for the spread of cascades. Let V_ℓ be the set of nodes purchased by action ℓ, and let $V = V_0 \cup \left(\bigcup_{\ell=1}^{L} V_\ell\right)$ be the complete set of vertices. Let G be the corresponding graph on vertex set V — we assume that activation probabilities p_{vw} for each $v, w \in V$ are known input parameters.

Given a strategy $\mathbf{y}$, a cascade may proceed through nodes that are part of the resulting network, either because they belong to V_0 or because some action was taken to purchase the node. Let $V(\mathbf{y}) = V_0 \cup \left(\bigcup_{\ell:y_\ell=1} V_\ell\right)$ be the set of nodes purchased by $\mathbf{y}$, and let $G(\mathbf{y})$ be the corresponding subgraph of G. An outcome $\{X_v(\mathbf{y})\}$ encodes the nodes activated by a cascade in graph $G(\mathbf{y})$.

A model that purchases sets of nodes is sufficiently general to model other interesting management actions such as the purchase of edges or sources. For example, to purchase edges, modify the graph as follows: replace edge (v, w) by two edges (v, e) and (e, v), where e is a new node representing the edge purchase. Let $p'(v, e) = p(v, w)$, $p'(e, w) = 1$. Then the cascade proceeds from v to w in the new graph with probability p_{vw}, only if node e is purchased. Similar ideas can be used to model actions that purchase sources, so that the submodular optimization problem of [10] can be seen as a special case of our model.

2.3 Conservation Application

Here we describe how this model of cascades and management actions relates to metapopulations and a conservation planning problem. A metapopulation model is a non-progressive cascade that describes the occupancy of different habitat patches over some time horizon T. Node $v_{i,t}$ represents habitat patch i at time t. For $i \neq j$, the activation or *colonization probability* p_{ij} represents the probability that an individual from patch i will colonize unoccupied patch j in one time step. The *extinction* probability β_i is the probability that the local population in patch i goes extinct in one time step. We assume that colonization and extinction probabilities are specified in advance, although in practice they are very difficult to know precisely.

Patches are grouped into non-overlapping *parcels* $\mathcal{P}_1, \ldots, \mathcal{P}_L$; some are already conserved, and the others are available for purchase at time 0. For each unconserved parcel $\mathcal{P}_\ell$, there is a corresponding management action with node set V_ℓ containing the nodes $v_{i,t}$ for all patches $i \in \mathcal{P}_\ell$ and for all t. The set V_0 consists of nodes representing patches in parcels that are already under conservation. In other words, the target species may only occupy patches that fall within conserved parcels, and the land manager may choose additional parcels to purchase for conservation. The fact that the species may not exist outside of conserved parcels is a simplification, but there is flexibility in defining "conserved", which is adequate for our purposes.

The sources consist of nodes $v_{i,0}$ such that patch i is occupied at the beginning of the time horizon. The target set is $\mathcal{T} = \{v_{i,T}\}$, i.e., the objective is to maximize the expected number of occupied patches at the end of the time horizon.

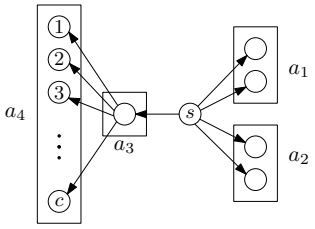
2.4 Non-Submodularity

Our problem shares the same objective as the influence-maximization problem of [10]. However, with respect to our expanded set of actions, the objective is not submodular. A set function f is submodular if for all $S \subseteq T$ and for all m, the following holds:

$$f(S \cup \{m\}) - f(S) \geq f(T \cup \{m\}) - f(T).$$

This is a diminishing returns property: for any m, the marginal gain from adding m to a set T is no more than the gain of adding m to a subset S of T.

It is easy to see that submodularity does not hold for the expanded set of actions in our problem, and that the greedy solution can perform arbitrarily worse than optimal. This is true because an instance may contain a high payoff action that is only enabled by first taking a low payoff action.



S	$f(S)$
$\{a_1, a_2\}$	4
$\{a_3\}$	1
$\{a_4\}$	0
$\{a_3, a_4\}$	$c+1$

Figure 2: Example of non-submodularity.

Consider the example in Figure 2, with unit cost actions and where all activation probabilities are equal to 1. Action a_4 has payoff of c, but only if action a_3 is also taken. Hence, the marginal gain of adding a_4 to $\{a_3\}$ is greater than that of adding a_4 to the empty set, so the objective is not submodular. For a budget of 2, the greedy algorithm will select $\{a_1, a_2\}$ for a total reward of 4, but the optimal solution $\{a_3, a_4\}$ has objective value $c + 1$.

3 Related Work

Cascades. Cascade optimization has also been considered by Leskovec et al. and Krause et al. in their work on optimal sensor placement for outbreak detection in networks [11,13]. They seek to find an optimal selection of nodes to *detect* cascades that occur in a network, and show that this problem is also submodular so can be approximated well by a greedy approach. They make improvements to the greedy approach to: (1) provide bounds in the case of non-uniform sensor costs and (2) greatly improve performance by introducing the CELF algorithm, which makes the same selections as a naive greedy algorithm with many fewer function evaluations by taking advantage of submodularity to avoid unnecessary evaluations. The analysis of the greedy approximation for submodular functions is due to Nemhauser et al. [16].

Stochastic Optimization. Stochastic models have long been considered to be significantly harder computational problems. Unlike in the deterministic case, stochastic linear programs are computationally demanding (e.g., it is easy to show that even in extremely simple settings these problems are $\#P$-hard [7]). It is only in the last two decades that substantial attention has been paid to solving stochastic integer programs as well (see, e.g., the survey of Ahmed [2]).

The Sample Average Approximation (SAA) has been instrumental in recent advances. The survey of Shapiro [18] outlines the range of convergence results for SAA that can be proved for a wide swath of stochastic optimization problems. The SAA also yields surprising strong approximation algorithm results, for both structured linear and integer programming problems, as surveyed by Swamy and Shmoys [19]. This approach has recently been applied to other large-scale stochastic combinatorial optimization problems, such as was done in the work of Verweij et al. [21].

4 Methodology

In this section, we describe our method to solve the cascade optimization problem (1). A major challenge is the fact that the objective itself is very difficult to compute. Even for a fixed strategy $\mathbf{y}$, the problem of computing $E[X_v(\mathbf{y})]$ for a single node v is equivalent to the $\#P$-complete *s-t reliability problem* of computing the probability that two terminals remain connected in a graph with random edge failures [20].

4.1 Sample Average Approximation

The SAA method [18, 21] is a method for solving stochastic optimization problems by sampling from the underlying distribution to generate a finite number of scenarios and reducing the stochastic optimization problem to a deterministic analogue. In our setting, instead of maximizing the expected value in (1) directly, the SAA method maximizes the empirical average over a fixed set of samples from the underlying probability space. It is important to note that, for any strategy $\mathbf{y}$, the random variables $X_v(\mathbf{y})$ can be measured in a common probability space defined over subgraphs of G. In other words, to simulate a cascade in $G(\mathbf{y})$, one can first flip coins for all edges in G to construct a subgraph G' (the cascade graph), and then compute reachability in $G'(\mathbf{y})$.

Let $G'_1, \ldots, G'_N \subseteq G$ be a set of *training cascades* produced in this fashion, and let $\xi_v^k(\mathbf{y})$ be a deterministic

value that indicates whether or not node v is reachable in $G'_k(\mathbf{y})$. The sample average approximation of (1) is

$$\max_{\mathbf{y}} \frac{1}{N} \sum_{k=1}^{N} \sum_{t \in \mathcal{T}} \xi_v^k(\mathbf{y}) \;\; \text{s.t.} \sum_{\ell=1}^{L} c_\ell y_\ell \leq B. \qquad (2)$$

To encode this as a MIP, we introduce reachability variables x_v^k to represent the values $\xi_v^k(\mathbf{y})$, and a set of linear constraints that enforce consistency among the $\mathbf{x}$ and $\mathbf{y}$ variables such that they have the intended meaning. To match our application, we use the following formulation tailored to non-progressive cascades, for which G'_k is guaranteed to be acyclic, though this is not a fundamental limitation.

$$\begin{aligned}
\max_{\mathbf{x},\mathbf{y}} \frac{1}{N} \sum_{k=1}^{N} \sum_{t \in \mathcal{T}} x_v^k & \\
\text{s.t.} \sum_{\ell=1}^{L} c_\ell y_\ell &\leq B \\
x_v^k &\leq \sum_{\ell \in \mathcal{A}(v)} y_\ell, \qquad \forall v \notin V_0, \forall k \quad (3) \\
x_v^k &\leq \sum_{(u,v) \in E_k} x_u^k, \qquad \forall v \notin \mathcal{S}, \forall k \quad (4) \\
0 \leq x_v^k &\leq 1 \\
y_\ell &\in \{0,1\}
\end{aligned}$$

Consider a fixed $\mathbf{y}$. We say that node v is *purchased* if $\sum_{\ell \in \mathcal{A}(v)} y_\ell \geq 1$. The constraints (3) and (4) together imply that $x_v^k > 0$ only if there is a path in G'_k from the sources $\mathcal{S}$ to v consisting of purchased nodes. In this case, the constraints are redundant and x_u^k may be set to the upper bound of 1 for all nodes u on the path. If there is no path to v, then an inductive argument shows that x_v^k must be equal to 0. Otherwise, by (4), there must be some node u such that $x_u^k > 0$ and $(u, v) \in E_k$. By induction, we can build a reverse path from v comprised of nodes w such that $x_w^k > 0$. Such a path must end at a source (recall that G'_k is acyclic), contradicting the fact that v is not reachable.

Relationship to Network Design. After sampling the training cascades, our problem is a deterministic *network design problem*: which sets of nodes should be purchased to connect the most targets? Network design is one of the most well-studied classes of deterministic combinatorial optimization problems. Traditionally, these problems arise in applications such as telecommunication networks and supply chains, where there is a given input graph that specifies potential routing links (and nodes) that can be purchased to provide a specified quality of service, and the aim is to do this at minimum cost. Our formulation is similar to standard flow-variable based network design MIPs (e.g., see Magnanti & Wong [15]), but since the input graphs are acyclic and purchases are made on nodes instead of edges, we can utilize a more compact formulation without any edge variables.

Algorithm 1: The SAA procedure.

Input: M samples of N training cascades, N_{valid} validation cascades, N_{test} testing cascades

1 Solve M independent SAA problems of size N to produce candidate solutions $\mathbf{y}_1, \ldots, \mathbf{y}_M$, and upper bounds $\bar{Z}_1, \ldots, \bar{Z}_M$. Set $\bar{Z} = \frac{1}{M} \sum_{i=1}^{M} \bar{Z}_i$.

2 Choose the best solution $\mathbf{y}^*$ from $\mathbf{y}_1, \ldots, \mathbf{y}_M$ by re-estimating the objective value of each using N_{valid} independent validation cascades.

3 Compute $\underline{Z}(\mathbf{y}^*)$ using N_{test} independent testing cascades. The estimated upper bound on the optimality gap is $\bar{Z} - \underline{Z}(\mathbf{y}^*)$.

Bounding Sub-Optimality of SAA The SAA optimum converges to the true optimum of (1) as $N \to \infty$, but for small N the optimal value may be optimistic, and the solution sub-optimal.[1] Verweij et al. describe a methodology to derive statistical bounds on the quality of the SAA solution compared with the true optimum [21].

Let OPT be the true optimal value of problem (1), and let $\bar{Z}$ be the optimal value of the SAA problem (2) for a fixed set of N training cascades. For any solution $\mathbf{y}$ (typically the SAA optimum), let $\underline{Z}(\mathbf{y})$ be an estimate of the objective value of solution $\mathbf{y}$ for problem (1) made by simulating N_{test} cascades in $G(\mathbf{y})$. The bounds are based on the fact that

$$E[\underline{Z}(\mathbf{y})] \leq \text{OPT} \leq E[\bar{Z}]$$

The value $E[\bar{Z} - \underline{Z}(\mathbf{y})]$ is then an upper bound on the optimality gap $\text{OPT} - E[\underline{Z}(\mathbf{y})]$, and the random variable $\bar{Z} - \underline{Z}(\mathbf{y})$ an estimate of the upper bound. To reduce the variance of $\bar{Z}$, one can solve the SAA problem many times with independent samples and take $\bar{Z}$ to be the average of the upper bounds obtained in this way. This also gives many candidate solutions, of which the best is selected using validation samples. The overall procedure is specified in Algorithm 1. Note that if the SAA problem in line 1 is not solved to optimality, the upper bound $\bar{Z}_i$ is the best upper found during optimization, *not* the objective value of $\mathbf{y}_i$.

[1] This situation is analogous to overfitting a small training sample in machine learning.

4.2 Preprocessing

The SAA problem optimizes over a fixed set of training cascades that are sampled in advance. Because of this, it is possible to achieve significant computational savings by preprocessing the cascades to accelerate later computations.

We defined the training cascade G'_k to be a subgraph of G obtained by retaining each edge (u, v) independently with probability $p(u, v)$. However, this may be an inefficient representation of a cascade. For example, we can obtain a more compact representation by simulating the cascade forward from $\mathcal{S}$ in the natural fashion so that nodes and edges that are not reachable under *any* strategy are never explored. Such a simulation results in a graph that is equivalent for computing reachability from $\mathcal{S}$ under any strategy $\mathbf{y}$.

In general, during preprocessing of a cascade we will consider arbitrary transformations of the problem data for each training cascade — originally consisting of $(\mathcal{S}, \mathcal{T}, \mathcal{A}, G'_k)$ — to a more compact representation that preserves computation of the objective for any strategy $\mathbf{y}$. This is done separately for each training cascade resulting in parameters $\mathcal{S}_k$, $\mathcal{T}_k$, and $\mathcal{A}_k$ that are now specific to the kth training cascade; the MIP formulation is modified in the obvious way to accommodate this. We first introduce one generalization: let the *reward vector* $\mathbf{r}^k$ replace the target set $\mathcal{T}$ so that the objective is computed as $\sum_k \sum_{v^k} r_v^k x_v^k$. Initially, $r_v^k = 1$ for $v \in \mathcal{T}$, and 0 otherwise. There are two elementary preprocessing steps: *pruning* and *collapsing sources*.

Pruning. Generating the cascade graph via forward simulation from $\mathcal{S}$ excludes any nodes that are not reachable from $\mathcal{S}$. We additionally prune all nodes v with no path to $\mathcal{T}$.

Collapsing Sources. If there is a path from $\mathcal{S}$ to v consisting exclusively of nodes from V_0, then v will be reachable for any strategy $\mathbf{y}$. In this case, v is added to $\mathcal{S}_k$ as a new source. The set $\mathcal{S}$ is collapsed to a single source node s with an outgoing edge (s, v) for each $(u, v) \in E_k$ such that $u \in \mathcal{S}$.

In addition to these two techniques, we contribute a method to find additional sets of nodes that can be collapsed into singletons because the fates of all nodes in the set are tied together — i.e, for any strategy, whenever one node in the group is reachable, all are. The simplest example of this is a strongly connected component consisting of nodes purchased by the same action(s).

Let $u \Rightarrow v$ denote the following situation: for any strategy $\mathbf{y}$, if node u is reachable, then node v is also reachable. There are two basic cases that guarantee this:

1. $(u, v) \in E$ and $\mathcal{A}(u) \subseteq \mathcal{A}(v)$. I.e., u links to v, and whenever u is purchased v is also purchased.
2. $(v, u) \in E$ and $\nexists w : (w, u) \in E$. I.e., the *only* link to u comes through v; hence, any path that reaches v must go through u.

The relation $u \Rightarrow v$ is clearly transitive, so to find all pairs such that $u \Leftrightarrow v$, we build a graph with directed edges corresponding to the two basic cases above and compute its strongly connected components (SCCs). Once we have computed the SCCs, we form the quotient graph by collapsing each strongly-connected component C into a single node and update the edge set accordingly. We must also update the other parameters of the problem. Let v_C be the node corresponding to component C. Then: (i) v_C becomes a source if any node in C was originally a source, (ii) the reward of v_C is equal to the sum of rewards for the original nodes in C, and (iii) the new action set of v_C is $\mathcal{A}(v_C) = \bigcap_{u \in C} \mathcal{A}(u)$. To justify (iii), note that any strategy under which all of C is reachable must purchase every node in C: the set of actions that does this is exactly $\bigcap_{u \in C} \mathcal{A}(u)$.

These preprocessing steps may be repeated until no additional progress is made and we have obtained a reduced cascade. Because preprocessing results in training cascades that are completely equivalent for reasoning about the objective values of strategies, they may be used in conjunction with *any* algorithm.

4.3 Greedy Baselines

In our experiments, we use two greedy algorithms adapted from [10] and [13] as baselines. Each starts with an empty set of actions, and repeatedly adds the "best" action that does not violate the budget. The algorithm GREEDY-UC (for *uniform cost*) chooses the action that results in the greatest increase in objective value. The algorithm GREEDY-CB (for *cost-benefit*) chooses the action with the highest ratio of increase in objective value to cost.

In each step, the increase in objective value is evaluated for each possible action by simulating N cascades. For our problem instances, it is prohibitively time consuming to simulate new cascades for each objective function evaluation. Instead, we reuse a set of N presampled training cascades as in the SAA method. Simulations in pre-sampled cascades require many fewer edge explorations, because only live edges from the original cascade are considered. We may also apply our preprocessing techniques in this case. The incremental nature of the greedy algorithms allow for an *additional* optimization: after action ℓ is selected in some step, every subsequent strategy will include action ℓ. Hence

we may modify the problem by moving the nodes of V_ℓ to V_0 so that they become part of the "original" graph to be augmented, and then repeat preprocessing. The accumulated speedup from these optimizations are quite significant: approximately 1000x for our instances.

5 Experiments

Our application is part of an ongoing collaboration with The Conservation Fund to optimize land conservation to assist the recovery of the Red-cockaded Woodpecker (RCW), a federally listed rare and endangered species. RCW are a "keystone species" in the southeastern US: they excavate tree cavities used by at least 27 other vertebrate species [1]. However, habitat degradation has led to severe population declines, and existing populations are highly fragmented [5]. As a result, habitat conservation and management are crucial to the continued viability of RCW.

We use the RCW recovery problem as a test bed for the computational approaches developed in this work. The scientific and political issues surrounding endangered species are complex, and great care must be taken when developing models that predict their fate. Although we use real data for this study, some parameter choices and assumptions have not been thoroughly verified with respect to RCW ecology. Hence one should not draw specific conclusions about RCW conservation planning from the particular results presented here. However, our computational results show that the model is robust to many different parameter settings, so it may be applied to a variety of specific conservation problems. Coupled with a biologically verified diffusion process, our model could be a key tool that provides decision makers with information about balancing options and resource constraints.

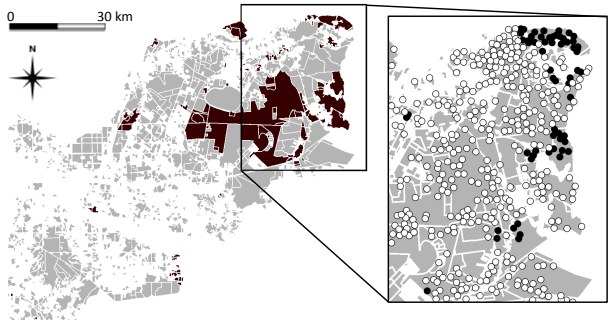


Figure 3: The study area. Left: spatial layout of parcels; dark parcels are conserved. Inset: cirles indicate territories; filled circles are occupied. There are no occupied territories outside the inset area.

For the purposes of this computational study, we choose a study area in the southeastern United States suggested by the Conservation fund. The study area contains 443 non-overlapping parcels of land with area of at least 125 acres (the estimated minimum size to support a RCW territory). The cost to conserve a new parcel is equal to its assessed property value; already-conserved parcels are free.

RCWs live in small groups in a well-defined *territory* (i.e., patch) that is centered around a cluster of cavity trees where the individual birds roost [14]. The study contains 63 presently-occupied RCW territories (these determine the sources $\mathcal{S}$), the vast majority of which fall within already-conserved parcels.

We identify almost 2500 *potential* RCW territories satisfying minimum habitat and area requirements, using a 30 by 30 km habitat suitability raster of the study area that is calculated using land cover type, hurricane/climate change risk, and development risk. The fact that these territories are specified in advance and assumed to exist at time 0 is a simplification, but compatible with RCW ecology and management strategies. Because of restrictive requirements for cavity trees (live old-growth pines 80 to 150 years old) and significant time investments to excavate cavities (one to six years), the locations of territories are stable over time; it is far more common for an individual to fill a vacancy in an existing territory than to build a new one [14]. Further, it is a common management practice for land managers to drill or install artificial cavities to create new territories in actively maintained parcels [1].

The parcel composition can be seen on the left of Figure 3, while the location of active and potential RCW territories can be seen on the right of Figure 3. Given a specific budget, the goal is to decide which parcels to conserve in order to maximize the expected number of active territories at a 100 year time horizon.

To model the dispersal process among RCW territories, we utilize a simple parametric form for colonization probabilities that is based on previous theoretical work about meta-populations, and the parameter values are adapted to loosely match an individual-based model for the RCW [14]. In particular, the probability that some individual from an active territory i colonizes an unoccupied territory j in one year is computed according to Equation 5.

$$p(i,j) = \begin{cases} 1/C_i & d(i,j) \le r_0 \\ \alpha \exp(-\gamma \cdot d(i,j)) & d(i,j) > r_0 \end{cases} \quad (5)$$

When the distance from territory i to territory j is within the species foraging radius r_0, the probability of colonization is inversely proportional to C_i – the number of neighboring territories within distance r_0. For

territories j beyond the foraging distance, the probability of colonization is exponentially decaying with distance. In our study, the foraging radius is $r_0 = 3$ km, and the parameter values chosen are $\alpha = 0.1$ and γ =7.69e-4. The single-year extinction probability for territory i is $\beta_i = 0.29$.

5.1 Performance and Bounds

We evaluate the quality of the solutions obtained by the SAA approach versus the ones obtained by the greedy approaches at different budget levels. The results are reported in Figure 4(a). The SAA values are computed using the procedure specified in Algorithm 1. We use $M = 50$ training samples of $N = 10$ cascades, and solve the SAA problem for each. We then evaluate the expected number of active territories of each solution using $N_{valid} = 500$ validation cascades. The best solution is tested against $N_{test} = 500$ cascades and the final expected number of active territories is reported as the 'saa' value. We also report the stochastic upper bound 'saa-ub' on the objective function obtained from the training samples. The proximity of 'saa' and 'saa-ub' indicate the the SAA solutions are essentially optimal.

The results of the two versions of the GREEDY are reported for comparison. The greedy algorithms are run on $N = 100$ training cascades and then the solutions are re-evaluated using the same set of $N_{test} = 500$ test cascades as the SAA solution. The GREEDY-CB approach outperforms the GREEDY-UC approach. Although the SAA solutions are best, they are not significantly better than the greedy solutions. The upper bounds indicate that GREEDY is also nearly optimal for this instance. We will later see that the relative performance of SAA versus GREEDY can vary significantly with problem instance.

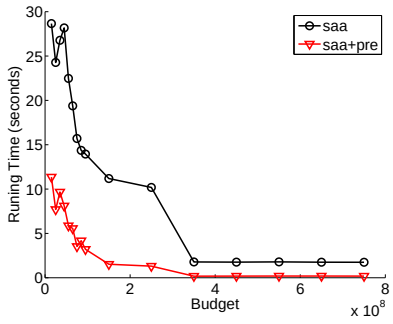
To evaluate whether $N = 10$ training cascades are sufficient, we evaluate the performance for different sample sizes at a budget level of 10% (approximately $400M). The results are presented in Figure 4(b). The gaps are quite small between the stochastic upper and lower bounds on the objective: for $N = 10$ the upper bound is 696.07 and the lower bound is 687.97, only a 1.16% gap. Moreover, the gap does not decrease significantly for larger sample sizes, indicating that $N = 10$ is a large enough sample size to obtain high quality SAA solutions. The error bars in Figure 4(b) represent 95% confidence intervals that are computed over the M training samples averaged to obtain the upper bound, and the N_{test} cascades for the lower bound. These indicate high confidence that SAA is close to the *true* optimum for the stochastic optimization problem. The number of active territories in different test cascades does not vary wildly as suggested by the tight error bars. This indicates that the solution generated by SAA is quite robust to different scenarios that might unfold in reality.

Error bars for Figure 4(a) are of similar size, but omitted because they are too small to be seen relative to the larger scale.

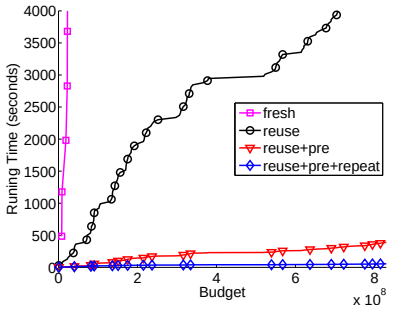
5.2 Preprocessing

Preprocessing is critical to the running time performance of our models. Figure 5(a) shows the average running time to solve a single instance of the SAA problem for time horizon $T = 20$ and $N = 10$. Preprocessing provides a 10x speedup.

These optimizations also provide siginificant speedups to the greedy algorithms, as illustrated in Figure 5 (b). The lines are labeled as follows: 'fresh' is the variant of the greedy algorithm that evaluates the marginal benefit of each action by simulates a fresh set of training cascades each time; 'reuse' is the variant where training cascades are simulated in advance as in the SAA; 'reuse+pre' includes also preprocessing the training cascades; and 'reuse+pre+repeat' reapplies the preprocessing step each time the greedy algorithm commits to a management action. The results clearly indicate runtime savings of: (1) generating training cascades in advance and (2) preprocessing cascades to reduce their size.



(a)



(b)

Figure 5: Run times with different levels of preprocessing: (a) SAA, (b) GREEDY.

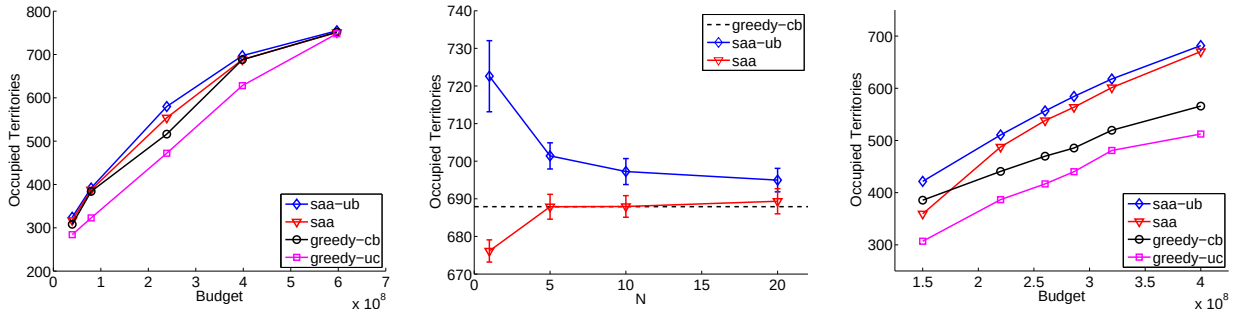


Figure 4: (a) Objective values for SAA and GREEDY, and SAA upper bound, for various budgets. (b) Upper and lower bounds on objective as a function of training size N. (c) Results for MOD instance, K=20.

6 Discussion

In Section 2.4, we discussed an example that showed that the greedy algorithm can perform arbitrarily badly with respect to optimal. However, in Figure 4, the GREEDY objective is often competitive with that of the SAA solution. This is partly due to the nature of our study area and data set, which is well suited to a myopic strategy such as greedy. The diffusion behavior of the RCW is such that short range colonizations between territories within the foraging radius are much more common than long range dispersal. As such, territories adjacent to currently occupied territories will always be the most appealing for a greedy approach. But often that might not be optimal. SAA is also constrained somewhat to buy territories connected to currently occupied territories so that they can be colonized by short-range hops. However, SAA is also capable of setting goals: it can build a path from the sources to another area if that area is highly favorable.

In our results, as expected GREEDY follows an incremental strategy where it builds outward from the initially occupied territories. However, in this instance, there are conserved parcels near the initially active territories that can support an increased population at no additional cost. This makes selecting the parcels near the source territories *highly* favorable and hence GREEDY approach constructs solutions that are similar to the ones obtained by SAA.

It is easy to imagine real conservation scenarios where this is not the case. We have modified the problem instance to demonstrate this point, by taking many of the parcels in the northeastern quadrant out of conservation and assigning them costs, and then making many of the parcels farther to the west conserved (i.e., these territories are available at no cost). This reflects a situation where an existing population is located at some distance from a reservoir of conservation land that is suitable for habitation, or could be made suitable by low-cost management strategies. All other details of the problem such as colonization probabilities and time horizon remain the same.

Similarly to the original instance, we evaluate performance of SAA as well as GREEDY-CB and GREEDY-UB across different budgets. The results are presented in Figure 4 (c). Here we can see a clear advantage of the SAA approach over the GREEDY approach.

Figure 6 illustrates the actual strategies of GREEDY-CB and SAA on this modified instance. We can see qualitatively different strategies between the two. In this case, GREEDY-CB continues to follow the myopic conservation strategy of building outward from the initial population, while SAA recognizes that there is a high payoff available by connecting to existing conservation land and builds a path in that direction.

7 Conclusion

In this work, we addressed the problem of optimal network design for maximizing the spread of cascades under budget restrictions. In particular, we address settings where the cascade is driven by a stochastic process and one needs to plan under uncertainty. Unlike other cascade optimization problems, this one cannot be provably approximated by a naive greedy approach; we propose a sample average approximation approach that provides stochastic optimality gap guarantees. We also develop preprocessing techniques that greatly reduce problem size. We evaluate our methodology on a computational problem that arises in the area of sustainability — spatial conservation planning for species population growth. The SAA approach scales well to this large real-world instance, and finds better solutions than the greedy baseline while proving near-optimality. Moreover, preprocessing techniques result in a dramatic runtime speedup for both SAA and the GREEDY algorithms. Most importantly, our

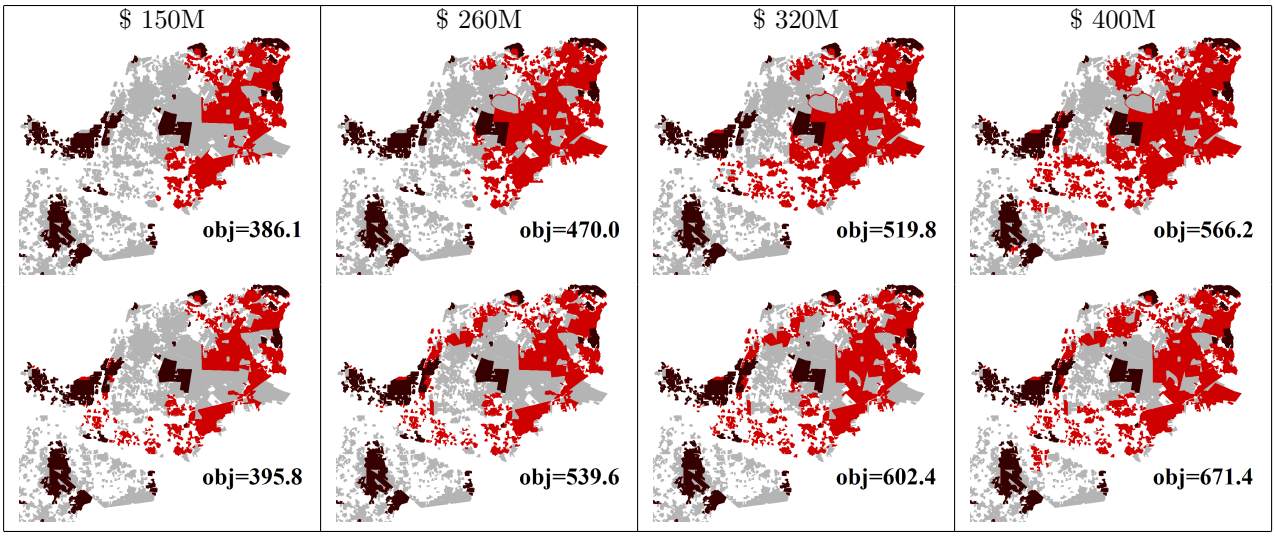


Figure 6: Illustration of conservation strategies for different budget levels obtained by (top row) GREEDY and (bottom row) SAA. The expected number of active territories is given for each solution.

results show that the optimal solutions generated by the SAA approach can be qualitatively different than the ones obtained by a myopic greedy approach. The methodology described here can be a useful strategic tool for conservation as well as in other areas.

References

[1] Red-cockaded woodpecker recovery plan. U.S. Fish and Wildlife Service, Southeast Region, Atlanta, GA, 2003.

[2] S. Ahmed. Introduction to stochastic integer programming. *COSP Stochastic Programming Introduction – http://stoprog.org*, 2004.

[3] R. Anderson and R. May. *Infectious diseases of humans: dynamics and control*. Oxford University Press, 1992.

[4] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):1, 2008.

[5] R. Conner, D. Rudolph, J. Walters, and F. James. *The Red-cockaded Woodpecker: surviving in a fire-maintained ecosystem*. Univ of Texas Press, 2001.

[6] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

[7] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. *Mathematical Programming*, 106(3):423–432, 2006.

[8] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

[9] I. Hanski. *Metapopulation ecology*. Oxford University Press, USA, 1999.

[10] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[11] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134:516–526, 2008.

[12] J. Leskovec, L. Adamic, and B. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

[13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[14] B. H. Letcher, J. A. Priddy, J. R. Walters, and L. B. Crowder. An individual-based, spatially-explicit simulation model of the population dynamics of the endangered red-cockaded woodpecker, picoides borealis. *Biological Conservation*, 86(1):1 – 14, 1998.

[15] T. Magnanti and R. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1, 1984.

[16] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, 1978.

[17] O. Ovaskainen and I. Hanski. Spatially structured metapopulation models: global and local assessment of metapopulation capacity. *Theoretical Population Biology*, 60(4):281–302, 2001.

[18] A. Shapiro. Monte Carlo sampling methods. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, Handbooks in Operations Research and Management Science Vol. 10, pages 353–426, 2003.

[19] C. Swamy and D. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. In *Proceedings of FSTTCS*, pages 5–19, 2006.

[20] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[21] B. Verweij, S. Ahmed, A. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24(2):289–333, 2003.