Scheduling Conservation Designs for Maximum Flexibility via Network Cascade Optimization

Citation	Xue, S., Fern, A., & Sheldon, D. (2015). Scheduling Conservation Designs for Maximum Flexibility via Network Cascade Optimization. Journal of Artificial Intelligence Research, 52, 331-360. doi:10.1613/jair.4679
DOI	10.1613/jair.4679
Publisher	AI Access Foundation
Version	Version of Record
Terms of Use	http://cdss.library.oregonstate.edu/sa-termsofuse

Journal of Artificial Intelligence Research 52 (2015) 331-360 Submitted 01/15; published 02/15

Scheduling Conservation Designs for Maximum Flexibility via Network Cascade Optimization

Shan Xue XUE@EECS.OREGONSTATE.EDU
Alan Fern AFERN@EECS.OREGONSTATE.EDU
School of EECS, Oregon State University
Corvallis, OR 97331 USA

Daniel Sheldon SHELDON@CS.UMASS.EDU
School of Computer Science, University of Massachusetts
Amherst, MA 01003, USA

Abstract

One approach to conserving endangered species is to purchase and protect a set of land parcels in a way that maximizes the expected future population spread. Unfortunately, an ideal set of parcels may have a cost that is beyond the immediate budget constraints and must thus be purchased incrementally. This raises the challenge of deciding how to schedule the parcel purchases in a way that maximizes the flexibility of budget usage while keeping population spread loss in control. In this paper, we introduce a formulation of this scheduling problem that does not rely on knowing the future budgets of an organization. In particular, we consider scheduling purchases in a way that achieves a population spread no less than desired but delays purchases as long as possible. Such schedules offer conservation planners maximum flexibility and use available budgets in the most efficient way. We develop the problem formally as a stochastic optimization problem over a network cascade model describing a commonly used model of population spread. Our solution approach is based on reducing the stochastic problem to a novel variant of the directed Steiner tree problem, which we call the set-weighted directed Steiner graph problem. We show that this problem is computationally hard, motivating the development of a primal-dual algorithm for the problem that computes both a feasible solution and a bound on the quality of an optimal solution. We evaluate the approach on both real and synthetic conservation data with a standard population spread model. The algorithm is shown to produce near optimal results and is much more scalable than more generic off-the-shelf optimizers. Finally, we evaluate a variant of the algorithm to explore the trade-offs between budget savings and population growth.

1. Introduction

Reserve site selection is a key problem in conservation planning in which planners select land regions to be designated as nature reserves, either to achieve general conservation goals such as preserving biodiversity, or to achieve specific goals such as supporting the recovery of an endangered species. In general, the problem is extremely complex as it involves reasoning about the interplay between uncertain population spread, uncertain future budgets, and other problem specific factors. In particular, properly assessing population spread involves reasoning about spatial aspects of landscapes such as their sizes, shapes, and connectivity. Further, the decision space is huge, consisting of all possible land investment combinations over time.

Given the above factors, it would be highly desirable for conservation practitioners to enhance their decision making via automated, or semi-automated, planning and scheduling algorithms. Unfortunately, this problem is beyond the scope of existing off-the-shelf stochastic planners and schedulers. This is largely due to the combination of enormous state and action spaces, the highly uncertain, exogenous dynamics, and the need for spatio-temporal reasoning. The main contribution of this paper is to make progress toward handling these complexities by studying a useful subproblem of conservation planning that can be used by practitioners on realistic scenarios. The general schema used to develop our algorithm is more widely applicable (see Section 7) and one that has not received significant attention in the AI community. Thus, we hope that this work will also inspire new specialized and general-purpose approaches for complex stochastic planning/scheduling problems.

Recently, Sheldon et al. (2010) studied a restricted, but still challenging, version of the conservation planning problem, which we will refer to as *upfront conservation design optimization*. In this problem, the planner is given an upfront budget and a stochastic *metapopulation model* (Hanski & Ovaskainen, 2000) that describes how the species under consideration will spread throughout a landscape of available habitat. In addition the system is given information about the costs of potential land parcels that are available for purchase and conservation. The objective is to select a set of land parcels to immediately purchase and conserve, subject to the budget constraint, that will maximize the spread of the population within a specified time horizon.

A key simplification present in this problem is that all land parcels are assumed to be purchased upfront with the currently available budget. An advantage of this simplification is that it allows for a reasonably efficient and near optimal solution approach (Sheldon et al., 2010). However, the upfront simplification limits the utility of the approach in a number of ways. First, conservation budgets generally arrive in increments over time, so it is unrealistic to purchase a large set of parcels in advance and restricting to a small set of parcels using the current budget may be very suboptimal in the long run. Moreover, it is often *unnecessary* to purchase parcels that are spatially remote from the current population until the species has spread enough to make them relevant to further population growth. Second, this upfront simplification requires planners to commit in advance to conservation strategies that may take many years to play out, which ignores the potential advantage of observing and responding to the stochastic outcomes of the population spreading process as it unfolds. For example, as the population spread is observed, it may be beneficial to divert money from failed subpopulations to purchase more parcels near thriving populations.

In contrast to upfront planning, an ideal approach would be *fully adaptive planning*, where, at regular decision epochs, the planner would make purchase decisions based on the most recent population and budgetary information. Unfortunately, no currently-available adaptive planning tools can scale to realistic conservation scenarios. This is due to the combination of an enormous state space (possible population and purchase configurations), enormous action space (possible subsets of land parcels to purchase), long horizons (tens to hundreds of years), and high degree of stochasticity in the population spread model.

Given the challenge of arriving at a fully adaptive solution, a main contribution of this paper is to introduce a problem that strikes an important middle-ground between the upfront and fully adaptive approaches. In particular, we consider *conservation design scheduling* for exploring the trade-off between future population and cost, where we are given an initial conservation design (i.e. a set of parcels to purchase) and are asked to schedule the purchase time of each parcel in a way that (1) achieves a population spread over the time horizon within an arbitrary tolerance of population loss, and (2) maximizes purchase flexibility by delaying the specified purchase time (i.e. purchase deadline) for each parcel as long as possible.

This problem formulation simplifies over the fully adaptive problem in a number of ways. First, the set of parcels to be purchased is provided as input, which removes this degree of freedom from the planning problem. Second, and more significantly, in order to select an action for the current time step, the general case of fully adaptive planning requires computing a policy that dictates what to do at each possible future contingency, or at least the reasonably likely ones. In contrast, the space of possible schedules, our focus here, is much smaller than the space of policies or even partial policies. This allows for a compact encoding of the scheduling problem, which does not appear possible for the problem of computing full, or even, partial policies. This distinction between the fully adaptive and scheduling setting is akin to the distinction between closed-loop and open-loop planning, where generally computing closed-loop plans is considered to be more difficult than open-loop plans for large stochastic problems.

A solution to the above scheduling problem yields a useful tool to conservation planners, who can first develop conservation designs that capture their own complex decision-making objectives, perhaps with optimization software, and then schedule the purchases to obtain the most efficient and cost-effective implementation of that design. The conservation planner then has the flexibility to purchase the parcels at any time before the schedule-specified deadlines, knowing that the population spread will not be hurt too much by such purchase delays.

In addition, our scheduling problem can potentially be used as a component of an adaptive planner. A common and successful approach for many adaptive planning problems is *replanning*, where at each decision epoch a non-adaptive plan is computed from the current state and its first actions are executed. Our work enables a replanning approach that at each decision epoch first computes an upfront design using existing work (e.g. Sheldon et al., 2010), and then computes a schedule and purchases only the parcels scheduled to be purchased immediately. This purchase strategy would spend the minimum amount of budget at each step while guaranteeing a limited loss in population spread.

In addition to introducing and formalizing the problem of conservation design scheduling, the second contribution of this paper is to develop a principled algorithm for solving it. The key idea is to apply the *Sample Average Approximation (SAA)* approach (Shapiro, 2003) in order to arrive at a novel deterministic optimization problem, for which we develop a principled solution with a motivation from its special case. In particular, when the approximated loss tolerance ratio is 0, our deterministic optimization problem is one of network cascade optimization, which we show is equivalent to a novel variant of the directed Steiner tree problem. In the traditional Steiner tree problem, graph edges are associated with costs, and the objective is to compute a Steiner tree with minimum cumulative edge cost. In our variant, the set-weighted directed Steiner graph problem, costs are associated with sets of edges (possibly non-disjoint) rather than individual edges. We show that this problem is computationally hard even under restrictions where the traditional problem admits an efficient solution. We then present an efficient primal-dual algorithm, which is guaranteed to compute both a feasible solution and a bound on the quality of the optimal solution. Then an early-stopping version of the algorithm provides a natural approach to explore the trade-off between future population and budget flexibility.

Our experiments on both real and synthetic data from a Red-cockaded Woodpecker conservation problem show that our primal-dual algorithm produces near optimal results and is much more scalable than standard optimization tools (CPLEX). We also show that the trade-off between population and budget allows for a flexibility of purchasing land parcels.

In what follows, Section 2 first presents related work, followed by our problem formulation in Section 3. Section 4 then shows how to reduce our subproblem to the set-weighted directed Steiner graph problem. Section 5 derives the corresponding primal-dual algorithm and a natural extension for the trade-off problem. Experiments are presented in Section 6. Finally we conclude and discuss future work.

2. Related Work

Previously, many different algorithms have been proposed to select reserve sites by formulating a numerical measure of reserve quality (together with the possible addition of constraints the reserve must satisfy) and then solving for the optimal set of sites under the proposed model (e.g. see the review article, Williams, Revelle, & Levin, 2005). Although the earliest reserve site selection algorithms largely ignored spatial considerations, many newer models incorporate spatial objectives or constraints directly into the optimization problems. Williams, Revelle, and Levin argue that a primary reason for the importance of spatial attributes is the fact that they capture properties of the landscape that are favorable for the underlying population dynamics, and that an important, but computationally difficult, research direction is to directly optimize with respect to a model for the population dynamics instead of using spatial attributes as a proxy. This is the direction that we are following in this paper by addressing the problem of spatial conservation planning with respect to a specific and widely adopted model of population dynamics.

A recent approach that explicitly reasons about a population dynamics model is the work of Sheldon et al. (2010) on the upfront conservation design problem, as described in Section 1. In order to cope with the stochasticity of the model, the popular *Sample Average Approximation* (SAA) approach was employed to transform the stochastic problem into a deterministic combinatorial optimization problem. This problem was then encoded as a Mixed Integer Program (MIP) and solved using state-of-the-art solvers. While that approach was able to solve reasonably large problems via various speedup techniques, the scalability is still limited to a relatively small number of "sample scenarios" used by SAA, which controls the accuracy of the approach. Kumar et al. (2012) have addressed this aspect of the approach. Lagrangian relaxation was used to decompose the SAA problem into independent subproblems that could each be solved in a practical time frame, possibly in parallel, by standard optimizers. This was shown to significantly reduce the runtime dependence on the number of SAA samples used.

Unfortunately, directly extending the above approach to compute multi-stage adaptive solutions, where the budget arrives in increments over time, does not seem practical. One attempt at this for two-stage problems was considered by Ahmadizadeh et al. (2010). They explore re-planning using a two-stage non-adaptive problem formulation, and find that it can indeed offer advantages over upfront planning. In their setting, the budget split and decision epochs are manually fixed. Unlike that work, our work explicitly separates the decision of which parcels to buy from the decision of when to buy them (the focus of this work), so that we may develop efficient special-purpose algorithms for the latter problem that scale much more easily to bigger problems and more stages.

There are several existing approaches that might be considered for a fully adaptive solution to the conservation problem. For example, the fully adaptive problem can be encoded as a Markov Decision Process (MDP), but the resulting state and action spaces would be far too big for state-of-the-art solvers. For instance, recent advances in solving large spatio-temporal MDPs (Crowley & Poole, 2011) require significant restrictions to the solution space, which are not acceptable in our application. As an existing approach for stochastic planning that has been successfully applied by Bent et al. (2004), Chang et al. (2000), Chong et al. (2000) and Yoon et al. (2008), Hindsight Optimization samples the future outcomes and optimistically estimates the state value based on the determined futures. However, when the action space is huge, this approach would have computational problems as current algorithms require enumeration of all the candidate actions when approximating the state value. Another approach would be to formulate the adaptive planning problem as a multi-stage stochastic integer program. However, the size of such a problem formulation scales exponentially with the number of stages, and the running time is already very costly for a single stage (Sheldon et al., 2010), or for a two-stage problem in a simpler setting that is not fully adaptive (Ahmadizadeh et al., 2010).

Recently, Golovin et al. (2011) proved that a simple greedy planning strategy provides near-optimal solutions in an adaptive conservation setting that at first appears similar to ours. However, in order to provide approximation guarantees, the authors restrict the population dynamics so that no spread occurs between distinct land parcels. While this may be a reasonable assumption for slow-moving species such as certain insects, which were the focus of that work, it ignores critical aspects of the population dynamics of highly-mobile animals such as birds, including the Red-cockaded Woodpecker on which our experiments are based.

3. Problem Formulation

In this section, we first introduce the basic terminology of conservation design planning and define our main stochastic optimization problem. Next, we describe how the *Sample Average Approximation* (SAA) is used to transform this problem into a deterministic optimization problem, which is the focus of the remainder of the paper.

3.1 Basic Concepts

We largely follow the formulation of Sheldon et al. (2010). Our conservation problems will involve a (large) land region of interest that is divided into *land parcels* that are the smallest land units available for purchase. Each parcel contains some number of distinct *habitat patches*, which are the atomic units in the population dynamics model and can either be occupied or unoccupied by the species of interest. For example, in the Red-cockaded Woodpecker problem considered in our experiments, habitat patches correspond to particular trees that have been prepared by humans (or existing birds) to facilitate nesting. Each parcel p has a cost $c(p)$, which denotes the cost of purchasing the land and restoring or conserving all of its habitat patches so that they are suitable for the species to occupy.

A *conservation design* is a set of parcels that are intended to be purchased and conserved. Given a conservation design D, a *purchase schedule* π for D is a mapping from parcels in D to purchase times in $\{0, 1, \ldots, H, \infty\}$, where H is the time horizon of interest and purchasing a parcel at time $t = \infty$ means this parcel is not going to be purchased. Thus the scheduler may choose not to purchase some parcels even though they are part of the design so as to realize the best tradeoff between budget flexibility and population spread. Although the species population dynamics have a yearly time step in our model (described below), the allowed purchase times (i.e. decision epochs) may be less frequent depending on the specific problem. An *upfront schedule* is one that assigns all parcels to purchase time $t = 0$.

It is worth noting that the purchase times specified by a schedule are best viewed as purchase deadlines. That is, we interpret the schedule as constraining the purchases to occur before or at the specified times. This view is justified by the fact that in the setup below, purchasing a parcel at an earlier time than specified will never result in worse population spread.

3.1.1 Population Dynamics Model

We use the same stochastic dynamics model as Sheldon et al. (2010), which is an instance of a popular *metapopulation model* from the ecology literature (Hanski & Ovaskainen, 2000). A patch a has two possible states at each time step, either unoccupied or occupied, and only conserved patches may be occupied. The population dynamics consists of two types of stochastic events. Colonization events occur when a population from patch a colonizes an unoccupied patch b, which happens with probability p_{ab}. Extinction events occur when a patch a that is occupied at time t becomes unoccupied at time $t + 1$, which happens with probability $1 - p_{aa}$. All events are independent. The details of the probabilities used in our experiments are given in Section 6.

The single-step colonization probability p_{ab} in our experiments typically decays with the distance between patches a and b, which encodes spatio-temporal dynamics in which populations slowly spread from a source population when new habitat is made available. Thus, in long-term planning for population spread, it is often unnecessary to purchase parcels that are distant from a source population at time $t = 0$, since the probability of the population spreading to such distant patches in the near future is negligible. By delaying such purchases until they become relevant to the design (i.e. the population has spread nearby), a conservation organization can use limited funds much more flexibly. However, it is non-trivial to decide how much to delay purchases so as not to harm the spread, since this decision depends very much on the spatio-temporal details of the population spread model. It is this decision that the optimization problem defined below is designed to make.

3.2 Stochastic Optimization Problem

Our problem statement will rely on two important concepts: 1) the *reward* of a schedule, and 2) the *flexibility* of a schedule. We first define these two concepts and then formulate the optimization problem in terms of them.

The reward of schedule π, denoted by $R(\pi)$, is a random variable that encodes the amount of population spread at time H, which is simply a count of the number of occupied patches at time H. It is easy to show in our model that the upfront schedule always achieves at least as much reward as any other schedule and thus maximizes the expected reward. Thus, we define the maximum expected reward as $R^* = E[R(\pi_{\text{upfront}})]$. Our optimization goal is to find a schedule π that almost achieves this optimal expected reward, —i.e. $E[R(\pi)] \geq (1-\varepsilon)R^*$ where ε is a positive real number and indicates the percentage tolerance of reward loss —but has maximum "purchase flexibility". We know that the upfront schedule achieves $(1-\varepsilon)R^*$, however, it requires commitment to all expenditures at the first time step and is thus the least flexible. Indeed, we now formalize the notion of flexibility in terms of expenditures over time.

Given any schedule π we can define its corresponding *cost curve* C_π to be a function from purchase times to accumulated cost, so that $C_\pi(t)$ is equal to the total cost of parcels purchased under π from time 0 up to and including time t. This curve is non-decreasing and provides a view of a schedule's spending profile over the time horizon. In particular, if the profile of cost curve C_{π_1} is never above that of C_{π_2}, i.e. the total expenditures of π_1 never exceed those of π_2, then we can say that π_1 offers more flexibility in terms of budget management compared to π_2 and should be preferred if all else is equal.

Now we define a surrogate cost function over schedules as

$$\text{cost}_f(\pi) = \sum_p c(p) \cdot f(\pi(p))$$

which is parameterized by a function f from times in $\{0, \dots, H, \infty\}$ to real numbers. We require $f(\infty) = 0$ for any f so that any parcel that is not purchased within the time horizon would not contribute to the surrogate cost. We can see that this surrogate cost function is simply a weighted sum of the parcel costs, where the weight is determined by f based on the parcel's purchase time. Although our algorithm can work with any real-valued function, we assume henceforth that f is strictly decreasing. There are two reasons for this. First, discounting future costs makes sense due to economic factors such as inflation. Second, intuitively, since f decreases with purchase times, minimizing with respect to cost_f would favor schedules that delay purchasing. In particular, if policy π_1 has a cost profile that is never greater than that of π_2, then π_1 will be assigned a lower surrogate cost when f is strictly decreasing.

If all parcels have positive costs, then the upfront schedule is the unique element that maximizes the surrogate cost, and the schedule that defers all purchases until time H gives the unique minimum as f is a strictly decreasing function. However, if we restrict to schedules that achieve at least reward $(1-\varepsilon)R^*$, the latter schedule will be excluded and there may no longer be a unique minimum.

We can now specify the problem of *conservation design scheduling*, which is to find a schedule π^* from the set of all possible schedules such that:

$$\pi^* \in \arg\min_{\pi} \text{cost}_f(\pi) \text{ s.t. } E[R(\pi)] \geq (1-\varepsilon)R^* \tag{1}$$

That is, out of all schedules that achieve reward $(1-\varepsilon)R^*$ we want to return one that is minimal in terms of its surrogate cost (i.e. it has maximal flexibility). Thus, ε controls the trade-off between flexibility and reward. In particular, using larger ε increases the set of feasible schedules and allows the potential for returning a more flexible schedule by sacrificing some reward.

Note that by varying the choice of f it may be possible to generate different solutions to Equation 1, each of which is *minimal* in the sense that no other feasible policy has a strictly lower cost curve. In our experiments, we use a simple discounted f given by $f(t) = \beta^t$ for a discount factor $\beta \in (0,1)$.

In practice, it is likely that a conservation manager will not have a particular value of ε in mind at design time. Rather, ε is best viewed as a parameter that will be varied in order to observe the different flexibility-reward trade-offs that are possible. The final selection of a schedule would then be based on an assessment of those possibilities.

Finally, it is worth noting that for $\varepsilon = 0$ (no reward approximation), the upfront solution will be the only feasible solution under typical population spread models. Thus, using $\varepsilon > 0$ is necessary for achieving any additional flexibility. This is because requiring a policy to achieve expected reward *exactly* R^* (i.e., $\varepsilon = 0$) requires it to make purchases to accommodate very unlikely outcomes of the population spread model that contribute a tiny, but positive, amount to the expected reward. For example, consider an outcome where the population jumps from its initial location to a very distant location in the first year, and then undergoes no further spread. This has vanishingly small, but positive, probability. The upfront schedule will support this population spread, since the distant location is purchased at the first step. Thus, any schedule that does not purchase the distant parcel in the first step will suffer a tiny loss in reward compared to the upfront schedule, so it does not achieve $\varepsilon = 0$. However, such a purchase will tend to be useless for the vast majority of the probability mass.

3.3 Deterministic Optimization Problem

The above optimization problem is stochastic in the sense that its constraint is defined in terms of an expectation over a complicated population spread distribution. This greatly complicates the direct solution of this problem. As in prior work on stochastic optimization of upfront schedules (Sheldon et al., 2010), we address this complication by converting the stochastic problem into an approximately equivalent deterministic optimization problem. This is done via the very common *Sample Average Approximation (SAA)* approach (see Shapiro, 2003 for a survey of some results). The key idea is to approximate a stochastic optimization problem using a collection of samples from the probability distribution, which are used to approximate expectations or probabilities via averages over samples.

In our problem formulation, each sample corresponds to a so-called *cascade scenario*, which is a particular realization of the population spread process over the time horizon. The main idea behind our application of SAA is to generate a set of such *cascade scenarios* from the probabilistic population spread model, and to approximate the expected reward of schedules as the average reward over the scenarios. The scenarios are combined into a single *scenario graph*, which is illustrated in Figure 1 and explained in detail in the remainder of this section.

More concretely, a cascade scenario is a layered graph, where layers correspond to time steps, with a vertex $v_{a,t}$ for each patch a and each time step t. For each pair of patches (a, b) and time step t, a coin is flipped with probability p_{ab} to determine if the directed edge $(v_{a,t}, v_{b,t+1})$ is present or not. If this edge is present and patch a is occupied at time t (through previous colonizations or non-extinctions), then patch b will be colonized and become occupied at time $t + 1$, as long as it is conserved. That is, the presence of edge $(v_{a,t}, v_{b,t+1})$ is interpreted as meaning that if a is occupied at time t and b is conserved at or before time $t + 1$, then b will be occupied at time $t + 1$ in the particular scenario. In this way, a cascade scenario graph encodes occupancy as reachability. In particular, assuming (for now) that all patches are conserved, then patch b is occupied at time t exactly when $v_{b,t}$ is reachable from a vertex $v_{a,0}$ corresponding to an initially occupied patch a.

To approximate the probabilistic spread model, we sample a set of N i.i.d. cascade scenarios $\{C_1, \ldots, C_N\}$, where we will denote the vertices in C_n by $\{v^n_{a,t}\}$. These scenarios are combined into a single *scenario graph*, which has an additional root vertex r with directed edges $(r, v^n_{a,0})$ to each vertex representing an initially occupied patch a. Figure 1 shows an example scenario graph that has three scenarios over a range of five time steps involving three patches a, b, and c, and two parcels, one containing a and b and the other containing just c. In this example, a is the only initially occupied patch and hence is connected at time step zero to the root node r across all three scenarios. Assuming that all parcels are conserved (i.e. purchased upfront), if a vertex is connected to the root node r, then the corresponding patch is considered to be occupied at the corresponding time in the particular scenario. This is because we defined r so that it can only be connected to other vertices through initially occupied patches.

Scenario graphs will be used in our work to estimate the reward of schedules as follows. Given a scenario graph, a schedule π is said to purchase node $v^n_{a,t}$ and all of its incoming edges if patch a is purchased no later than time t, that is, $\pi(p) \leq t$ where a belongs to parcel p. Thus, purchasing a parcel p at time t can be viewed as purchasing all vertices in the scenario graph, along with their incoming edges that involve patches in p that occur at layer t or later. This reflects the fact that once a patch is purchased and conserved, it is considered to be conserved and hence eligible for occupancy for the remainder of the time horizon. In Figure 1, an example schedule is shown that purchases parcel p_1 (containing a and b) at time 0, and parcel p_2 (containing c) at time 3. The vertices that are purchased by this schedule are shown in the shaded region and their (purchased) incoming edges are shown in bold.

We can now define under what conditions a vertex in a scenario graph is considered to be occupied given a schedule. Vertex $v^n_{a,t}$ becomes occupied under π if there is a path through *purchased* edges from r to $v^n_{a,t}$. We define the variable $X^n_\pi(a, t)$ to be equal to 1 if $v^n_{a,t}$ is occupied under π and 0 otherwise. In Figure 1, we have shaded in red the set of vertices that are occupied under the example policy. As an example, note that in scenario 3, the vertex $v^3_{c,3}$ is not occupied since there is no path from r to it through purchased edges. This is despite the fact that there is a path in the graph from r, since that path involves

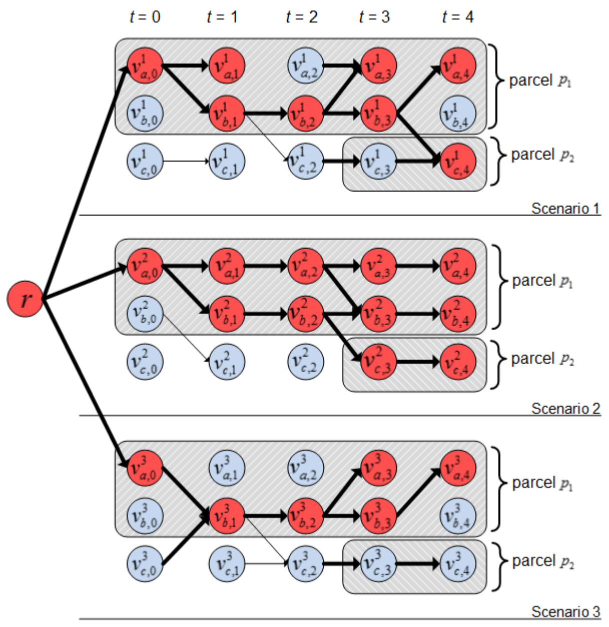

Figure 1: Example scenario graph ($N = 3$) for problem with parcels $p_1 = \{a, b\}$, $p_2 = \{c\}$. The schedule $(\pi(p_1) = 0, \pi(p_2) = 3)$ is also illustrated, using shaded boxes to indicate purchased nodes and heavy line weights to indicate purchased edges. Vertices representing occupied patches under this schedule are colored red.

some unpurchased edges. Note that the upfront schedule would purchase this node, since all vertices and edges would be considered to be purchased under that schedule.

The average reward of a schedule π relative to a scenario graph built from scenarios $\{C_1, \ldots, C_N\}$ is denoted as follows.

$$\hat{R}(\pi) = \frac{1}{N} \sum_{n=1}^{N} \sum_{a} X_{\pi}^{n}(a, H)$$

This is just the average across scenarios of the number of occupied patches at time H. In Figure 1 the average reward for the example schedule would be 2. A key property of scenario graphs is that as $N \to \infty$ we have that $\hat{R}(\pi)$ converges to $E[R(\pi)]$ for any fixed π. This implies that the set of schedules $\{\pi : \hat{R}(\pi) \geq (1 - \varepsilon)\hat{R}(\pi_{\text{upfront}})\}$ converges to the set $\{\pi : R(\pi) \geq (1 - \varepsilon)R^*\}$ as N grows, which is the set of policies that we wish to optimize flexibility over. Further, for any policy π, one can use standard probability concentration bounds (e.g. Chernoff bounds) to show that the event $|R(\pi) - \hat{R}(\pi)| \geq \epsilon$ has a probability mass that decreases exponentially fast as N grows. This suggests that only a relatively small number of scenarios are required to reliably obtain a tight approximation to the true expected reward of a policy. In practice, however, it is important to empirically validate that the approximation errors are reasonable for the number of scenarios used in the approximation.

The above motivates a deterministic SAA formulation of our original stochastic optimization problem (1) where flexibility is optimized subject to a constraint based on the empirical reward $\hat{R}$. That is, our deterministic problem is to solve:

$$\pi^* \in \arg\min_{\pi} \text{cost}_f(\pi) \text{ s.t. } \hat{R}(\pi) \geq (1-\hat{\varepsilon})\hat{R}^* \tag{2}$$

where $\hat{R}^* = \hat{R}(\pi_{\text{upfront}})$.

3.4 Overview of Solution Approach

Recall that for the stochastic optimization problem (1) setting $\varepsilon = 0$ resulted in an optimization problem that would typically have only the upfront schedule as a feasible solution. Rather, here, for the approximate SAA formulation, there will typically be non-upfront solutions that are feasible, even when using $\hat{\varepsilon} = 0$. This is because even for large (but practical) values of N, the set of scenarios used for the approximation will not tend to include highly unlikely scenarios, which need to be accounted for in the stochastic solution when using $\varepsilon = 0$. This observation motivates our solution approach for (2). In particular, in Section 4, we first consider the problem when $\hat{\varepsilon} = 0$, which turns out to be a new variant of the classic Steiner tree problem. We then derive an incremental primal-dual algorithm for the problem (Section 5) that can be used to approximately solve the $\hat{\varepsilon} = 0$ case or the $\hat{\varepsilon} > 0$ case through early stopping. Our experiments will show that this approach is able to provide significant flexibility with little loss in reward, with the flexibility-reward trade-off being controlled by $\hat{\varepsilon} > 0$.

4. Set-Weighted Directed Steiner Graph Formulation

As motivated above, here we focus on optimization problem (2) for the case when $\hat{\varepsilon} = 0$. That is, we must optimize flexibility subject to the constraint that we obtain the optimal empirical reward as measured by $\hat{R}$. In this section, we show how to formulate this problem as a novel variant of the Steiner tree problem.

4.1 Set-Weighted Directed Steiner Graph

From (2), we arrive at our final optimization problem for $\hat{\varepsilon} = 0$:

$$\pi^* \in \arg\min_{\pi} \text{cost}_f(\pi) \text{ s.t. } \hat{R}(\pi) = \hat{R}^*. \tag{3}$$

We can view this problem as a type of Steiner tree problem on the scenario graph. In particular, we say that any vertex at time $t = H$ is a *terminal vertex* if it is reachable from the root r, which is the set of nodes with $X^n_{\pi_{\text{upfront}}}(a, H) = 1$ and hence contribute to the upfront reward $\hat{R}^*$. The only way for π to satisfy the constraint $\hat{R}(\pi) = \hat{R}^*$ is to purchase a set of edges in the scenario graph that connect all of those target nodes to r. Thus, the constraint in Equation 3 corresponds to purchasing edges such that r has a path to each terminal, as in the Steiner tree problem.

As an example, consider again the scenario graph in Figure 1. The terminal nodes in this example are all nodes at layer $t = 4$ except for $v^1_{b,4}$ and $v^3_{b,4}$, which are the only two nodes that are not connected to r by a directed path. A schedule that satisfies the constraint

$\hat{R}(\pi) = \hat{R}^*$ must purchase edges such that all of these terminals are reachable from r. Note that for the example schedule of Figure 1, the set of purchased edges does not satisfy this constraint since there is no path of purchased edges to the terminal vertex $v^3_{c,4}$.

While our problem is very similar to the traditional Steiner tree problem, there is a significant difference. In the traditional problem, each edge is associated with a distinct weight and can be purchased individually, with the goal of connecting all terminals using a set of edges of minimum total weight (which always forms a tree). Rather, our situation is more complicated because we purchase parcels, which correspond to subsets of edges in the scenario graph. In particular, purchasing a parcel p at time t, which incurs cost $c(p)f(t)$ in Equation (3), corresponds to purchasing an edge set $E_{p,t}$ with cost $c(p)f(t)$ that contains all the edges $(u, v^n_{a,t'})$ that come from any vertex u and arrive at any vertex $v^n_{a,t'}$ with $a \in p$, $t' \geq t$ and $n \in \{1, \ldots, N\}$. Note that under this cost model, the total cost of edge sets purchased by a schedule π exactly equals our surrogate objective $\text{cost}_f(\pi)$.

From the above we see that our problem is an instance of a problem that we will call the *Set-weighted Directed Steiner Graph (SW-DSG)* problem, a novel variant of the Steiner tree problem, the goal of which is to select a set of vertices with minimal total cost in order to connect all the terminal vertices to the root. For the remainder of the paper we will discuss this problem in its general form to simplify notation. The input for SW-DSG is a directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with a single root vertex r, a set of terminal vertices $T \subseteq \mathbb{V}$, a set of M edge sets $\mathcal{E} = \{E_1, \ldots, E_M\}$ where each $E_s \subseteq \mathbb{E}$, and a non-negative cost c_s for each E_s. In particular, our conservation problem has edge sets $\mathcal{E} = \{E_{p,t}\}$ with $E_{p,t} = \{(u, v^n_{a,t'}) : (u, v^n_{a,t'}) \in \mathbb{E}, a \in p, t' \geq t, n \in \{1, \ldots, N\}\}$ and cost $c_{p,t} = c(p)f(t)$. A subset of $\mathcal{E}$ forms a Steiner graph if the union of the edges connect r to all vertices in T. The desired output is a minimum cost subset of $\mathcal{E}$ that forms a Steiner graph. Note that the optimal Steiner graph need not be a tree in SW-DSG, unlike in the traditional Steiner tree problem.

It is clear that SW-DSG is more general than the original deterministic optimization problem since the latter has a specific edge set structure. For instance, $E_{p,t_1} \subseteq E_{p,t_2}$ if $t_1 > t_2$. However, this structure does not make it an easier problem than SW-DSG. In the following sections, we prove that both problems are NP-complete and our primal-dual algorithm is the same for either setting. The special structure does not lead to any algorithmic advantages in deriving a primal-dual algorithm. Therefore, we mainly discuss the problem in the form of SW-DSG to simplify notation.

While the SW-DSG problem was motivated by our particular conservation application, it is relevant to other problems that have Steiner style objectives, but where the "edge resources" are best considered as groups. For example, Steiner trees are often used for the design of communication networks where edges correspond to existing or potentially new communication links. For situations where those links must be purchased as coherent sets (e.g. the communication infrastructure of different companies/organizations), our SW-DSG problem would be the appropriate formulation.

4.2 Computational Complexity

To our knowledge, the SW-DSG generalization of the Steiner tree problem has not been previously studied and hence we now consider its computational complexity.

The SW-DSG problem is a generalization of the traditional directed Steiner tree (DST) problem, which is known to be NP-complete (Hwang, Richards, & Winter, 1992). Further, under standard complexity assumptions, DST is hard to approximate by a factor better than $\log(|T|)$ (Charikar, Chekuri, Cheung, Dai, Guha, & Li, 1998). Note that these results hold even for acyclic directed graphs. There are a number of effective heuristic algorithms for DST (Drummond & Santos, 2009), with many of the most successful relying on shortest path computations as a subroutine. While shortest paths can be computed in edge weighted graphs efficiently, this turns out to not be the case for our set-weighted problem. In particular, note that the shortest path problem is a special case of DST (or SW-DSG) where there is a single terminal vertex. This problem turns out to be NP-Hard for SW-DSG, even when restricted to acyclic graphs and the special edge set structure shown in our original deterministic optimization problem, which is the case for the scenario graphs from our conservation problem.

Theorem 1. *The SW-DSG problem is NP-hard even when restricted to acyclic graphs with a single terminal vertex and the edge set structure in scenario graph.*

Proof. We prove the hardness by reducing the weighted set cover problem to the subclass of SW-DSG problems restricted to a scenario graph with one scenario and exactly one terminal. Note that here we consider the decision version of the SW-DSG problem, which asks if there is a feasible Steiner graph whose cost is less than a specified threshold C^*. An instance of the weighted set cover problem specifies a ground set of elements $S = \{e_1, \ldots, e_n\}$, a set $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets $S_j \subseteq S$, a cost C_j for each subset, and a cost bound C^*. The problem asks whether there is a collection $\mathcal{S}' \subseteq \mathcal{S}$ with total cost no more than C^* such that $\bigcup_{S_j \in \mathcal{S}'} = S$.

Given a set cover instance, we first describe how to construct a scenario graph as illustrated in Figure 2 and later describe the corresponding SW-DSG instance. The graph contains $2n$ layers, which alternate between *set layers* and *element layers* starting with a set layer (n layers of each, hence $2n$ layers). Each layer has m vertices labeled $S_1, \ldots, S_m$ to represent the sets in $\mathcal{S}$ and n vertices labeled $e_1, \ldots, e_n$ to represent the elements in S. In addition we include a root vertex r. Each vertex can also be seen as a parcel with a single patch. The edges in the graph only go from one layer to the immediate next layer as follows. The root vertex has an edge going to each S_j in the first set layer. For the ith element layer (i.e. layer $2i$ in the graph), we include an edge from a vertex with label S_j in the previous layer to a vertex with label e_i in the current layer whenever $e_i \in S_j$. Finally, vertex e_i at the ith element layer has an edge from it to each S_j in the next layer.

The corresponding SW-DSG (conservation problem) instance on this graph is specified as follows. The root node is r and the single terminal vertex is e_n at the final element layer. The edge sets are specified as follows, which is the same as the setting in a scenario graph. There are edge sets $E_{j,t}$ for each S_j at time t. In particular, $E_{j,t}$ contains every incoming edge that is from any vertex and to any vertex labeled as S_j at layers $t' \geq t$. We let the strictly decreasing $f(t)$ be sufficiently close to 1 for all t's. The cost of each $E_{j,t}$ is equal to $C_j \times f(t)$ where C_j is the cost of S_j in the original set cover problem. In other words, the cost of $E_{j,t}$ is almost C_j. Similarly, there are edge sets $E_{i,t}$ for each e_i at time t. We set their costs as 0. The cost threshold for the SW-DSG problem is equal to the threshold C^* of the set cover problem.

To see that this reduction is correct, consider the case where the resulting SW-DSG instance has a feasible solution. The solution provides a path from r to e_n through purchased edge sets that have total cost at most C^*.

Since the edge set $E_{i,t}$ for each e_i has zero cost, the edge set cost is the result of purchasing edge sets $E_{j,t}$. By the construction the path must go through a sequence of alternating element nodes and set nodes. In particular, the path must traverse each element node e_i for $i = 1, \ldots, n$. The only way for this to happen is to purchase for each of those e_i at least one edge leading to e_i from one of the immediately preceding S_j at layer $t \leq 2i$, which is only possible when $e_i \in S_j$. This can only happen by purchasing the corresponding edge set $E_{j,t}$, corresponding to S_j, which has a cost of (almost) C_j. From this we see that the collection of S_j corresponding to purchased edge sets must cover all of the elements and that their total cost is no more than C^*. Thus, the collection of sets is a solution to the set cover problem.

Conversely consider an instance of the set cover problem with a feasible solution. It is easy to verify that a feasible solution to the corresponding SW-DSG problem is to purchase edge sets $E_{j,1}$ corresponding to any S_j in the set cover solution. Combining the above we see that there is a feasible solution to the SW-DSG instance if and only if there is a feasible solution to the set cover instance.

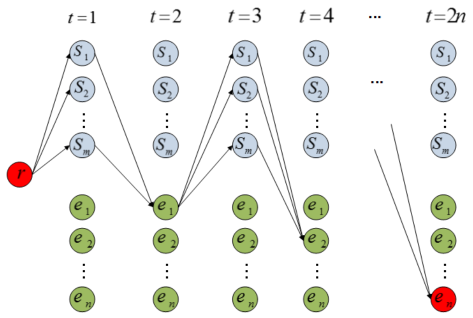

Figure 2: Description of the reduction from set cover to SW-DSG with a single terminal vertex and a scenario graph.

□

The above result proves that the shortest (or least cost) path problem is also NP-hard for SW-DSG, i.e. the problem of finding a least cost path when edges are purchased as sets. Thus, it is difficult to extend prior shortest-path-based heuristics for the Steiner tree problem to our problem. Given that SW-DSG is in NP, it is NP-complete. This motivates our derivation of an efficient heuristic solution approach in the next section, which computes both a feasible solution along with a bound on the cost of the optimal solution. Importantly this bound provides a sense of how good the computed solution is compared to the optimal.

5. Primal-Dual Algorithm

A potential solution approach to the SW-DSG problem is to encode it as a Mixed Integer Program (MIP), which is straightforward, and then to use an off-the-shelf MIP optimizer. While this approach produced non-trivial results for the upfront conservation problem (Sheldon et al., 2010), as our experiments will demonstrate, it does not scale well for our problem. A related approach could be to consider a rounding procedure for the MIP's LP-relaxation. While solving the LP-relaxation is easier than solving the MIP, our experiments show that the scalability of LP solvers is also poor for the problem sizes of interest to us. Instead, we exploit the MIP encoding in another way, by following the primal-dual schema (Vazirani, 2001) to derive a scalable algorithm that performs near optimally in our experiments. Our work can be considered as a non-trivial generalization of previous work (Wong, 1984), where the primal-dual schema was applied to DST. Moreover, an early-stopping version of our primal-dual algorithm provides a way to trade-off the schedule flexibility and reward ($\hat{\varepsilon} > 0$). Note that the primal-dual algorithms for SW-DSG and our original deterministic conservation problem only differ in the notations. In other words, the edge set structure in conservation problem does not offer further improvements for the algorithm. Thus in the following, we only present the approach for SW-DSG.

5.1 Primal-Dual Algorithm for SW-DSG

To apply the primal-dual schema, we start by giving a primal MIP for the SW-DSG problem along with the dual of its LP-relaxation in Figure 3. The primal MIP includes a binary variable $y(E_s)$ for each edge set in $\mathcal{E}$, which indicates whether E_s was purchased ($y(E_s) = 1$) or not ($y(E_s) = 0$). The objective of the primal is then simply the sum of these variables weighted by the costs of the corresponding edge sets. The Steiner graph constraint, requiring that all terminals be connected to the root node by purchased edges, is encoded using a standard network-flow encoding (lines 2–4) involving flow variables $x^k_{i,j}$. The flow variable $x^k_{i,j}$ encodes the flow on edge (i, j) destined for terminal k. The flow balance constraints (2) guarantee that one unit of flow is carried on a path from the root node r to terminal k.

The LP-relaxation of the primal simply replaces the integer constraints on the $y(E_s)$ variables with a positivity constraint. The dual of this relaxed problem (lines 6–9) includes dual variables u^k_i and $w^k_{i,j}$ corresponding to the primal flow constraints. Note that the constraint that one unit of flow leaves the root is implied by the other flow constraints. By omitting this constraint, one could simplify the dual by eliminating the u^k_r variables (or, equivalently, set $u^k_r = 0$ for all $k \in T$).

Given the primal and dual formulations of our problem, we can now apply the primal-dual schema for designing optimization algorithms. In particular, our primal-dual algorithm is iterative where each iteration increases the value of the dual objective and purchases a single edge set E_s, which corresponds to setting the primal variable $y(E_s) = 1$. The iteration stops when the purchased edges form a Steiner graph (i.e. the primal becomes feasible). The value of the dual objective at the end of the iteration serves as a lower bound on the optimal primal objective, which provides a worst-case indication of how far from optimal the returned solution is. At a high level, this algorithm is a simple greedy heuristic that continuously purchases the most beneficial edge set in order to build paths for an

$$\textbf{(Primal)} \quad \min \sum_{s=1}^{M} y(E_s) \times c_s, \quad \text{subject to:} \tag{1}$$

$$\sum_{(i,h)\in\mathbb{E}} x_{i,h}^k - \sum_{(j,i)\in\mathbb{E}} x_{j,i}^k = \begin{cases} 1, & \text{if } i = r \\ -1, & \text{if } i = k \\ 0, & \text{if } i \neq r, k \end{cases} ,k \in T, i \in \mathbb{V} \tag{2}$$

$$x_{i,j}^k \leq \sum_{s:(i,j)\in E_s} y(E_s),\ k \in T, (i,j) \in \mathbb{E} \tag{3}$$

$$x_{i,j}^k \geq 0,\ (i,j) \in \mathbb{E}, k \in T \tag{4}$$

$$y(E_s) \in \{0,1\} \tag{5}$$

$$\textbf{(Dual)} \quad \max \sum_{k\in T} (u_k^k - u_r^k), \quad \text{subject to:} \tag{6}$$

$$\sum_{k,(i,j)\in E_s} w_{i,j}^k \leq c_s, s \in \{1,\ldots,M\} \tag{7}$$

$$u_j^k - u_i^k - w_{i,j}^k \leq 0, k \in T, (i,j) \in \mathbb{E} \tag{8}$$

$$w_{i,j}^k \geq 0 \tag{9}$$

Figure 3: MIP for the SW-DSG problem and the corresponding dual LP of the MIP's LP-relaxation. The SW-DSG problem is defined by a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, a root vertex r, a set of terminal vertices T, and a set of edges sets $\mathcal{E} = \{E_s : s = 1, \ldots, M\}$, where each $E_s \subseteq \mathbb{E}$.

unconnected terminal. The primal-dual schema provides a principled way of incrementally computing this heuristic and at the same time computing a lower bound.

Algorithm 1 gives pseudo-code for the algorithm. The main data structure is an auxiliary graph $\mathbb{G}' = (\mathbb{V}, \mathbb{A})$ with the same vertices as the input graph $\mathbb{G}$. The auxiliary graph edge set $\mathbb{A}$ is initially empty and then each iteration adds the newly purchased edges $E_s \in \mathcal{E}$. The algorithm terminates when the edges in $\mathbb{A}$ form a Steiner graph. The edge sets used to construct this graph are then returned as the solution, following a pruning step that removes obviously redundant edge sets, which the algorithm can sometimes include during the iteration process.

In order to describe the algorithm in detail, we first introduce some terminology. Given a current auxiliary graph $\mathbb{A}$, we let $C(k)$ denote the set of all vertices that have directed paths to terminal node k via only edges in $\mathbb{A}$. Note that we consider k to be included in $C(k)$. Also, we define the *cut set* of a terminal node k, denoted by $\text{Cut}(k)$, to be the set of all edges (i, j) such that $j \in C(k)$ and $i \notin C(k)$. Intuitively, if k is not already reachable from the root, we know that at least one edge in $\text{Cut}(k)$ must be added to $\mathbb{A}$ in order to arrive at a Steiner graph.

Algorithm 1 Primal-Dual Algorithm for SW-DSG.

```
1: {**Inputs:** Graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, edge sets $\mathcal{E} = \{E_1, \ldots, E_M\}$, costs $\{c_1, \cdots, c_M\}$, terminals
   $T \subseteq \mathbb{V}$}
2: Initialize:
   $u_i^k = 0$, for each $k \in T, i \in \mathbb{V}$; $w_{i,j}^k = 0$, for each $(i,j) \in \mathbb{E}$, $k \in T$
   $\mathbb{G}' = (\mathbb{V}, \mathbb{A})$ with $\mathbb{A} = \emptyset$
   lowerBound $= 0$, solution $= \emptyset$
3: **while** $\mathbb{G}'$ is not a Steiner graph **do**
4:    Let $k$ be random vertex in $T$ not connected to $r$ in $\mathbb{G}'$
5:    $S = \{s \mid E_s \cap \mathrm{Cut}(k) \neq \emptyset, s \notin \text{solution}\}$
6:    $s^* = \arg\min_{s \in S} \Delta(s,k)$
      where $\Delta(s,k) = \left(c_s - \sum_{k' \in T, (m,n) \in E_s} w_{m,n}^{k'}\right) / |E_s \cap \mathrm{Cut}(k)|$
7:    $u_j^k = u_j^k + \Delta(s^*, k)$, for each $j \in C(k)$
8:    $w_{i,j}^k = w_{i,j}^k + \Delta(s^*, k)$, for each $(i,j) \in \mathrm{Cut}(k)$
9:    $\mathbb{A} = \mathbb{A} \cup E_{s^*}$
10:   lowerBound $=$ lowerBound $+ \Delta(s^*, k)$
11:   solution $=$ solution $\cup \{s^*\}$
12: **end while**
13: Pruning: solution $=$ solution $- \{s \mid \exists s' \in \text{solution}, E_s \subset E_{s'}\}$
```

The algorithm first initializes all dual variables to zeros and the auxiliary graph $\mathbb{A}$ to include all vertices and no edges. Each iteration then proceeds by first randomly selecting a terminal vertex k that is not connected to r in the auxiliary graph. At an intuitive level, the algorithm will then select an edge set E_s that contains a cutset edge of k according to a heuristic $\Delta(s,k)$ that is derived by applying the primal-dual schema. More concretely, the aim of each iteration is to raise the dual objective value by increasing the value of u_k^k while maintaining feasibility. Increasing u_k^k by itself will violate constraints of type (8) in the dual and lines 5 through 8 of the algorithm maintain feasibility by selecting an edge set E_{s^*} among those that intersects the cut set of k and then raising all variables corresponding to vertices in $C(k)$ and edges in $\mathrm{Cut}(k)$ by a value $\Delta(s^*, k)$ (including u_k^k). This is done in a way that causes the dual constraint of type (7) corresponding to edge set E_{s^*} to become tight. Since this constraint corresponds to primal variable $y(E_{s^*})$, the algorithm effectively sets $y(E_{s^*}) = 1$, indicating a purchase, by adding the edges in E_{s^*} to $\mathbb{A}$. The dual objective value at termination is the sum across iterations of $\Delta(s^*, k)$ and is returned as the lower bound.

The key property of our algorithm is that each iteration increases the dual objective, while also maintaining feasibility of the dual. This guarantees that at each iteration the dual objective value corresponds to a true lower bound on the optimal value of the primal.

Theorem 2. *Each iteration of the primal-dual algorithm produces a feasible dual solution with increased objective.*

Proof. As the base case, the initialization assigns all dual variables to be zeros, which is a feasible solution. Now suppose that iteration $q-1$ starts with a feasible solution $\{u_i^l, w_{i,j}^l\}$, which satisfies the dual constraints of type (7) and (8). Now if the algorithm terminates, we get a feasible solution. Otherwise let k be the terminal vertex selected. For all variables $\{u_i^l, w_{i,j}^l\}$ with $l \neq k$ the values are not changed, so (8) is satisfied. For the remaining variables with $l = k$, there are three cases. *Case 1:* For $j \notin C(k)$, the variables u_j^k and $w_{i,j}^k$ are unchanged, so they cannot contribute to a violation of (7) or (8). *Case 2:* For any edge (i, j) with both $j, i \in C(k)$, we increase both u_j^k and u_i^k by $\Delta(s^*, k)$ and continue to satisfy the corresponding constraint of (8). *Case 3:* For any cut set edge $(i, j) \in \text{Cut}(k)$, we increase u_j^k and $w_{i,j}^k$ by $\Delta(s^*, k)$ so that (8) remains satisfied. Since the $w_{i,j}^k$ for edges in the cut set are increased, we must ensure that constraints of type (7) do not become violated. The choice of $\Delta(s^*, k)$ made by the algorithm can be verified to never violate any of those constraints and makes at least one of them tight. □

After the main portion of the algorithm terminates, a pruning step is conducted to remove any edge set that is a subset of some other edge set in the solution, which decreases the total cost while maintaining feasibility. In particular, in the context of our conservation scheduling problem, the pruning step ensures that each parcel is purchased no more than once in the final solution. There are other more aggressive and computationally expensive pruning techniques that could also be used. For example, one could consider removing each one of the selected edge sets from the final solution and then test for feasibility. If the solution is still feasible, then the edge set can be eliminated. We did not find this more aggressive style of pruning to be necessary in our experiments.

5.1.1 Implementation and Running Time

Note that our pseudo-code stores and updates values for the u_j^k and $w_{i,j}^k$ dual variables. Then a naive implementation of the above algorithm would result in $O(|\mathbb{E}||T|)$ runtime for initialization as well as the computation per iteration, where $|\mathbb{E}|$ is the number of edges in the graph and $|T|$ is the number of terminals. This could be too much for SW-DSG problems with a large network such as the one in our conservation application. However, the algorithm is described in this way only for presentation purposes. It turns out that for the purposes of running the algorithm, it can be implemented significantly more efficiently. In particular, we only need to store and update the sum of corresponding $w_{i,j}^k$ values for each edge set (i.e. the sum term that appears inside of the definition of $\Delta(s, k)$ on line 6), and maintain the current objective value (stored as lowerBound in the pseudo-code), which is updated on line 10. Therefore, before the iterations only those $M + 1$ variables need to be initialized, where M is the number of edge sets and is much smaller than the size of the network. In our implementation the dominant computation per iteration is the computation of the cut set for the selected terminal k. We find the cut set by a backward traversal from terminal k toward the root. The time for this computation is acceptable when terminals are only connected to relatively small parts of the overall graph. This is the case in our conservation problem, where terminals are only connected to nodes in the same cascade and among those only ones that are spatially close enough to be reached. In other applications where terminals are possibly connected to a large portion of the graph, it may be preferable to incrementally maintain a cut set for every terminal at each iteration to reduce the computation. Then the memory needed is $O(C|T|)$ where C is the maximum size of a cut set and presumably $C \ll |\mathbb{E}|$. After getting the cut set, the algorithm takes $O(MC)$ time to identify the best edge set and update the solution.

5.2 Early-Stopping for Fractional Connection

In our primal-dual algorithm, the computation continues until all of the terminals in the scenario graph are connected by paths from the root. In the context of our conservation problem this corresponds to having no reward approximation loss ($\hat{\varepsilon} = 0$). Here we modify the above primal-dual algorithm to allow for reward approximation loss where $\hat{\varepsilon} > 0$. This case corresponds to only modifying the SW-DSG feasibility constraint to only require a fraction $1 - \hat{\varepsilon}$ of the terminals to be connected, leading to a natural way of exploring the trade-off between reward and flexibility.

Given the incremental, greedy nature of our primal-dual algorithm, which adds one edge set each iteration, a natural choice for this fractional connection problem is to stop the algorithm whenever at least a fraction $1 - \hat{\varepsilon}$ of the terminals are connected. While this basic early-stopping approach will lead to some improvement in the cost of the returned solution, compared to $\hat{\varepsilon} = 0$, the savings are often quite minimal. This is due to the fact that the primal-dual algorithm grows paths from the terminal nodes to the root and is unaware of the early-stopping condition. As a result, some of the paths that were being grown are never actually connected to the root at the point that the algorithm is stopped. These unconnected paths can be considered to be a waste of resources with respect to meeting the fractional connection constraint. Thus, to make this early-stopping algorithm viable, it is necessary to perform pruning on the early-stopping result. Our algorithm for fractional coverage then has two stages: 1) Generation, where early-stopping is used to generate an initial solution that meets the fractional connection constraint, and 2) Pruning, where the solution produced in stage 1 is pruned while maintaining the fractional connection constraint.

For the pruning stage we use a simple but effective greedy strategy. The idea is to iterate through the purchased parcels in the schedule returned by the early-stopping stage and to delay the purchase of each parcel as long as possible while ensuring that the number of connected terminal nodes is almost always within the required fractional connection tolerance.

We have found that for our conservation application, where the SW-DSG problem corresponds to a set of cascades, it is beneficial to prune using an independently generated and larger set of scenarios than those used to create the initial solution. This is analogous to using validation data to tune algorithm parameters in Machine Learning prediction problems, and it is beneficial for the same reasons. We found that pruning based on the original set of scenarios was often overly aggressive and hurt empirical performance due to over-fitting of the SAA scenarios. Since we can easily generate independent scenarios to estimate the true expected reward of the pruned policies, it is better to prune based on that criterion instead. Also, since the computational complexity of evaluating the reward of pruned policies is low compared with the SAA optimization, we can afford to use a larger set of scenarios. In particular, in our experiments we formed the initial schedules based on a set of 10 cascade scenarios and conducted the pruning step with respect to 40 cascade scenarios.

This approach for pruning can also be viewed as directly enforcing a threshold on the (independently estimated) expected reward from the original stochastic problem (Equation 1) instead of enforcing a threshold on the objective value of the SAA problem (Equation 2). Since we can't calculate the correct threshold value (the RHS of Equation 1) without knowing the true optimum R^* of the stochastic problem, we use the SAA optimum $\hat{R}^*$ in its place. The SAA optimum is a stochastic upper bound to R^*, so this is generally a conservative approach for enforcing Equation 1.

6. Experiments

In this section, we first evaluate our primal-dual algorithm by applying it to a real, full-scale conservation problem. Next, to verify the robustness of our approach to other problems, we present results using synthetic conservation data from a problem generator used in several recent studies. We focus the first two parts of the experimentation on the case of $\hat{\varepsilon} = 0$, which we will see provides substantial gains in flexibility. In order to explore the trade-off between flexibility and reward (population spread), at the end of this section, we evaluate the early-stopping approach for $\hat{\varepsilon} > 0$.

6.1 Evaluation of Primal-Dual Algorithm on Real Conservation Map

The real map we use is the same dataset as in prior work by Sheldon et al. (2010) on computing upfront conservation designs. The data is derived from a conservation problem involving the Red-cockaded Woodpecker (RCW) in a large land region of the southeastern United States that was of interest to The Conservation Fund. The region was divided into 443 non-overlapping parcels (each with an area of at least 125 acres) and 2500 patches serving as potential habitat sites. Parcel costs were based on land prices and some land parcels were already conserved and thus had cost zero. We use the same population spread model as Sheldon et al. (2010), which was based on individual-based models of the RCW. Since our approach requires a conservation design as input, we use the design computed by Sheldon et al. (2010) using a total budget constraint of \$320M. The map of the area is shown in the left cell of Figure 7, with parcels making up the design shaded green and free parcels (with cost 0) shaded grey; red '+' marks indicate initially occupied patches. Our method also requires specifying a strictly decreasing function for defining the surrogate cost function, for which we use $f(t) = \beta^t$ with $\beta = 0.96$. We found that the results are not very sensitive to the value of β.

6.1.1 Comparing to Optimal Solutions

Here we compare the solutions of our primal-dual algorithm to optimal solutions found using the CPLEX solver applied to a MIP encoding of the SW-DSG problem. The MIP encodings become very large as the horizon and number of scenarios increase. In particular, there are $443 \cdot H + 2500 \cdot H \cdot N$ variables and the number of constraints grows with the number of edges, in the cascade network, which becomes impractical as N and H grow. Since the optimal solver can't scale to large versions of the problem, we consider problems involving cascade networks with just 2 scenarios and horizons ranging from just 15 to 40 years. We also use CPLEX to compute solutions to the LP-relaxation of the MIP. The objective value returned for the LP provides an alternative approach to computing a lower bound on the optimal solution and thus is interesting to compare to our lower bound in terms of tightness and runtime. Since our primal-dual algorithm is stochastic due to the random selection of

	Cost (M$)		Lower Bound		Run Time (s)		
	MIP	PD	LP	PD	MIP	LP	PD
H = 15	126.8	126.2[2]	122.2	84.9	5.5	6.1[3]	0.9
H = 20	123.6	125.7	117.7	71.9	8.2	7.6	2.5
H = 25	117.6	121.4	104.7	61.5	28	10	9.0
H = 30	130.4	134.0	117.3	56.9	5126	15	11
H = 35		131.3	109.9	64.1		18	25
H = 40		127.5		59.7			45

Table 1: Comparison of Primal-Dual (PD) with MIP and LP.

terminal nodes, we report averages over 20 runs, noting that the standard deviations are negligible.

The first two data columns of Table 6.1.1 show the (surrogate) cost of the solutions found by CPLEX solving MIP and our algorithm (PD) for increasing horizons, where larger horizons correspond to larger problems. When a method fails to return a solution due to memory constraints no value is shown in the table. We see that for horizons where MIP is able to yield solutions by CPLEX, our algorithm produces solutions that have very similar costs (here lower cost is better). We also see that the MIP solver runs out of memory and is unable to return solutions for the 2 largest problem instances, which are already scaled down versions of the problem (small number of cascades and horizon).

The next two columns of Table 6.1.1 provide results for the lower bound computed by CPLEX solving the LP and by the PD algorithm. We see that the lower bound produced by the LP is significantly tighter than the bound produced by our algorithm. However, the LP cannot be solved by CPLEX for the largest problem, while our approach still yields a lower bound. Overall, though our lower bound is not as good as the LP (when it can be computed), it is generally within a factor of two of the optimal solution, which provides a non-trivial assurance about the quality of the returned solution for very large problems.

The final three columns of Table 6.1.1 present the time used by the approaches for each problem, where blank cells indicate that the method ran out of memory. Our algorithm is significantly faster than the MIP approach, which fails for the two largest problems, and is comparable with the LP approach, which only provides a lower bound and fails for the largest problem. This later result indicates that a solution based on LP-rounding would face difficulty, since even solving the LP for these large problems (40 time steps with 2500 patches each) is computationally demanding. An advantage of the primal-dual algorithm is that it avoids encoding the LP and rather works directly with a graph.

2. Here the PD cost is less than the “optimal” MIP cost returned by CPLEX. After investigating, we found that CPLEX correctly evaluates the solution it returns, but thinks that the solution is optimal when it is not. It appears that this issue is due to the small error tolerance allowed by the CPLEX solver.
3. Here MIP takes less time than LP. We think this is possibly because CPLEX uses different algorithms for LP and MIP. Especially, MIP is solved by branch-and-bound algorithm which uses modern features like cutting planes and heuristics, making CPLEX a powerful MIP solver.

6.1.2 Number of Cascades in the Scenario Graph

According to SAA, the optimal solution over a finite set of cascade scenarios will converge to true optimum with more scenarios. Previously only two cascades are used due to the poor scalability of CPLEX. Now we study the number of cascade scenarios we should use to ensure a good solution. Recall that as N increases, the original stochastic problem is approximated more accurately. Yet a larger N corresponds to more computation. More importantly, here with $\epsilon = 0$, a larger N leaves the schedule less space for flexibility. In the extreme case of $N \to \infty$, the only possible schedule is the upfront schedule that has minimal flexibility. To find a good value of N in practice, we study the primal-dual solutions with different number of cascade scenarios by validating the reward $R(\pi)$ that the each primal-dual schedule can achieve. Given that the population spread is stochastic, we compute the reward $\hat{R}(\pi)$ by running 20 simulations of the stochastic population spread model. Each simulation provides a reward value (number of occupied patches at the horizon) and we average the results. We do this for the schedules produced by our primal-dual algorithm and for the upfront schedule. Recall that the intention is to nearly match the reward of the upfront schedule. Figure 4 presents the results when the time horizon is $H = 20$. We observe that the primal-dual schedule achieves more and more reward when N increases, and the reward is converging towards the expected reward of the upfront schedule. We also see that 10 cascades is quite close to get the best performance and the rate of improvement is slowing down. Thus, the remainder of our experiments use 10 cascades for the SAA.

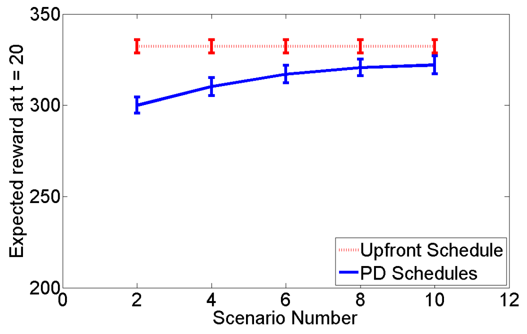

Figure 4: Rewards of PD solutions $w.r.t$ the number of cascade scenarios.

6.1.3 Quality of Conservation Schedules

We now evaluate our algorithm on problems of more realistic sizes. Here, we consider problems based on 10 cascades and horizons ranging from 20 to 100 years, which are well beyond the range approachable for the MIP and LP. The solution times for our algorithm ranged from 15 seconds for $H = 20$ to 29 minutes for $H = 100$.

First, we evaluate the average accumulated reward of the schedule returned by our method for each horizon in Figure 5. The average reward of the upfront schedule ranged from 332 for $H = 20$ to 615 at $H = 100$ and for all time horizons the primal-dual solution attained average reward at least 95.3% of optimal, with negligible error bars about the averages. The small gap indicates that for 10 scenarios the SAA approximation is quite good—the gap could be further reduced by increasing the number of scenarios.

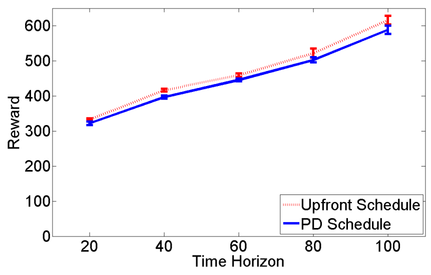

Figure 5: Rewards of PD schedules *w.r.t.* time horizon H.

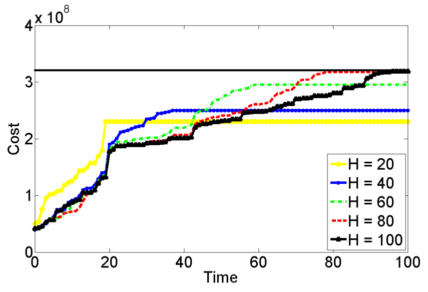

Figure 6: Cost curves of PD schedules for horizons 20 to 100.

Of course, we must also consider the cost curves corresponding to the schedules, since that is what affords the flexibility criterion of our problem. Figure 6 presents the cost curves for our schedules. Note that as defined in Section 3.2, a cost curve shows the (non-discounted) accumulated cost of a schedule over time. Then the cost curve for a schedule produced for horizon H will only increase until time H and then remain flat, reflecting that no purchases are made after that time. We see that for all horizons the cost curves show a fairly gradual increase in cost expenditures over time, indicating that the schedules are indeed providing a significant amount of flexibility regarding purchase times, particularly when compared to the upfront schedule, the cost curve of which is the flat black line in Figure 6 since all the parcels are purchased at time 0. In experiments not shown, we found that the cost curves vary by a small amount for different values of β, but the same general trend is present. Interestingly, in all curves there is a sudden jump in cost at around 20 years. To understand this in Figure 7 we show both the parcel purchases made by our schedule and the population spread on the map over the 100 year horizon. We see that at t = 20 the sharp increase in cost is due to the purchase of some relatively expensive and vast parcels in the southern part of the design. Looking at the population spread dynamics, it is apparent that those parcels are a critical gateway for ensuring reliable spread to the southwestern part of the design in later years. Delaying the purchase any longer significantly increases the probability that such spread does not occur, which our approach discovers.

Another interesting observation can be seen by comparing the expected population spread under the PD computed schedule and the expected population spread of the upfront schedule (Figure 7). The most striking difference between the spreads is seen at time steps $t = 20, 60, 80$ in the northeastern part of the map. For the upfront schedule the entire northeastern part is occupied in large part, while for the PD schedule there is a “hole” in the northeastern part near where the initial bird populations are located. Note, however, that this hole is finally occupied by the horizon of the problem ($t = 100$). At that time, the spread in the upfront and PD schedules are visually very similar, which agrees with the fact that their measured rewards were also similar. The reason for the difference in population spread is that the PD schedule delays the purchase of some of the northeastern parts of

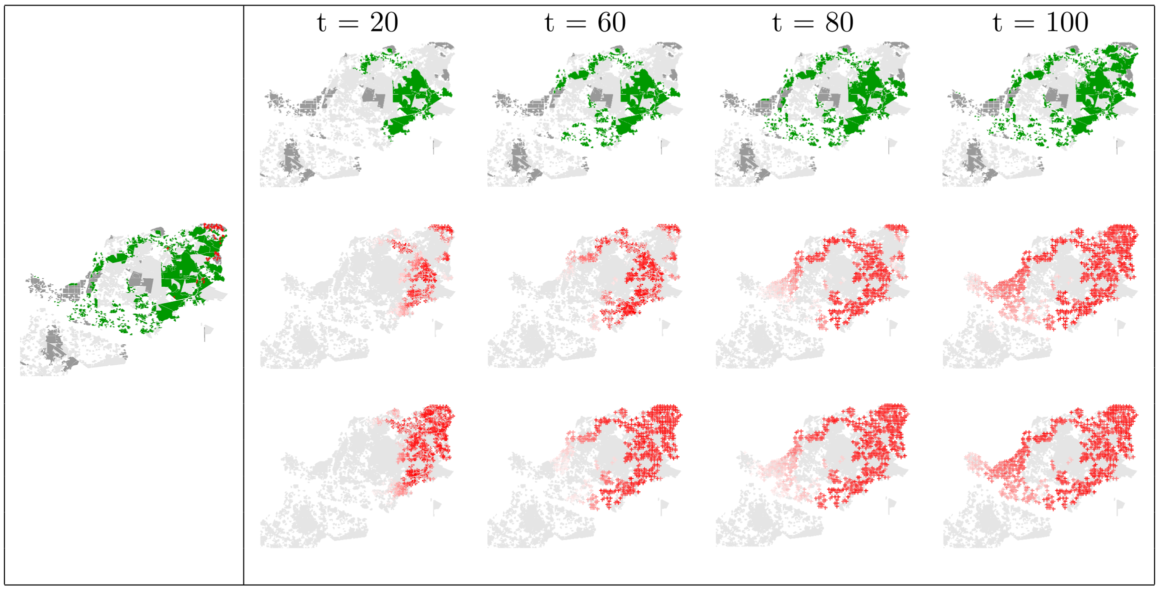

Figure 7: (Left) Original conservation design used for scheduling shown as green shaded parcels. Free (zero-cost) parcels are also shaded in dark grey and red '+' indicates initially occupied patches. (Right) The top row shows the parcels purchased (shaded green) by our PD schedule over a horizon of 100 years. The middle row shows the population spread over the same horizon for the schedule, where lighter red shading of a patch indicates a smaller probability of being occupied (as measured by 20 simulations). The bottom row shows the population spread of the upfront schedule over a horizon of 100 years.

the map near the initial bird population until about 20 years before the time horizon ends. From the population spread process of both schedules, we found that these parcels are very closely connected and hence can become occupied in a fairly short time if a bird population is nearby. This is apparent for the upfront schedule, where those areas are already occupied by $t = 20$. Thus, the purchase of such parcels can be delayed as long as there is enough time left for the population to spread over these landscapes. Therefore, the purchasing can be delayed not only for parcels far away from current population, but also for parcels that can be covered quickly and reliably. Note that such flexibility is mainly due to our definition of the reward function, which only takes the population at time H into account. If we count the population at every time step, presumably a good schedule would purchase the "hole" area very soon for more population.

6.2 Evaluation of Primal-Dual on Synthetic Maps

To evaluate the primal-dual algorithm more thoroughly, we randomly selected 10 synthetic maps generated and used in prior work (Ahmadizadeh et al., 2010). All the maps consist of the same region of 146 non-overlapping parcels and 411 patches, with different configurations of parcel costs and the initial population. For each map, we considered problems involving different conservation designs, where each design corresponded to the upfront solution when the budget is limited to a factor b of the total parcel cost of the map, where b ranged from 0.1 to 0.5. In this section, we present a very similar analysis as in Section 6.1 and show consistent results, indicating that our primal-dual algorithm is stable across different problems.

6.2.1 Comparing to Optimal Solutions

We first compare the upper and lower bounds returned by our primal-dual algorithm to the optimal objective values of MIP and LP as computed using CPLEX. Since CPLEX still has scalability issues when solving the larger synthetic problems, we restrict the comparison to problems with 2 and 4 scenarios and a horizon of 20 years.

Results on all of the 10 maps are very similar. As an example, Figure 8 shows the surrogate cost (PD-UB) and dual objective value (PD-LB) of the primal-dual solution on map 768, together with the optimal surrogate cost (CPLEX-MIP) and the lower bound computed for the LP by CPLEX. We see that compared to optimal, our algorithm can still produce solutions with similar costs, especially when the problem is easy (b is smaller). Also, we again see that the lower bound computed by CPLEX is better than the PD lower bound. However, the PD lower bound is still within a factor of 2.

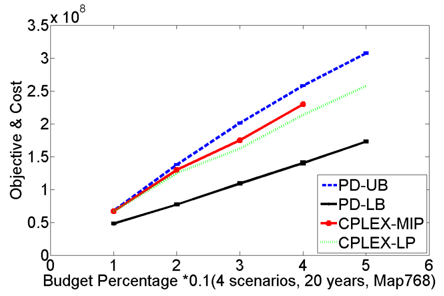

Figure 8: Cost and objective value of problems on map 768. The horizontal axis varies the amount of budget used to compute the upfront solution that is used as the conservation design given to the scheduling algorithms.

6.2.2 Quality of Conservation Schedules

We now consider larger problems based on 10 cascade scenarios and horizon $H = 40$, for which the MIP and LP are not practically solvable.

We first compare the average accumulated reward of the schedule returned by our primal-dual algorithm and the upfront schedule. Figure 9 shows results for one of our maps, indicating that the rewards achieved by our primal-dual schedules are always very close to those of the upfront schedules, as desired. The results for other maps are very similar.

We also study the cost curves of the schedules in order to illustrate their flexibility compared to upfront schedules. Figure 10 presents the average cost curves for our schedules across all the 10 maps. It is noted that when the budget is very limited, the purchase is not delayed very much. For example, when $b = 0.1$, most parcels are purchased before $t = 15$, long ahead of the time horizon. On further analysis this can be explained by the fact that

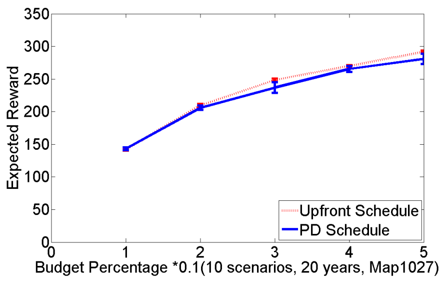

Figure 9: Rewards of primal-dual schedule and upfront schedule on map 1027.

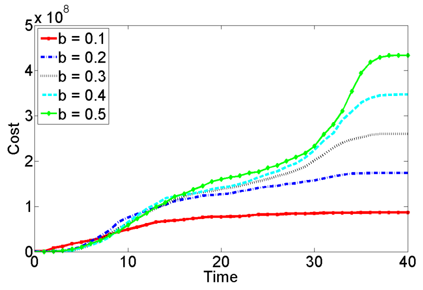

Figure 10: Average cost curves of PD schedules on 10 maps.

for many of the maps and such small budgets, the sets of affordable parcels are fairly spread out and loosely connected. This means that the population requires more time in order to reliably spread across such sets. Thus, the parcels must be purchased quite early in the horizon to support that spread. Rather for larger budgets, the sets of parcels that must be purchased are spread out but also more tightly coupled, which allows for easier, more reliable population spread. Thus, it is possible to delay purchases to a much larger extent as seen by the cost curves for the larger budgets. This shows that our algorithm is able to afford considerable flexibility when the initial conservation design supports reasonably reliable population spread.

6.3 Early-Stopping for Trading Off Flexibility and Reward

We now consider the early-stopping variant of our primal-dual algorithm, referred to as PD-ES, for producing schedules that trade off flexibility for reward using $\hat{\varepsilon} > 0$. The dataset we use here is the real conservation map. Figure 11 illustrates the cost curves of the early-stopping schedules with $\hat{\varepsilon} = 0.0, 0.05, 0.10, 0.15, 0.20$ and $H = 20, 40, 60, 80$, which demonstrates the possible budget saving over time if a corresponding fraction of reward loss is allowed. Figure 12 shows the average simulated rewards of these schedules.

First we notice that the average reward achieved by the early-stopping schedules is almost always within the specified error tolerance, which shows that the pruning step is generalizing effectively. We also see that the cost curves of the early-stopping schedules show significant improvement for even small values of $\hat{\varepsilon}$ compared to no early-stopping ($\hat{\varepsilon} = 0$). For example, when $H = 60$ and $\hat{\varepsilon} = 0.20$, there is almost no cost during the first several years, and for several decades the cost is approximately half of the cost required for $\hat{\varepsilon} = 0$. These results show that our approach is able to provide a set of schedules that spans a spectrum of trade-offs, which can then be considered by conservation managers.

7. Summary and Future Work

In this work, we addressed the problem of scheduling purchases of parcels in a conservation design. We formulated the problem as a network cascade optimization problem which was

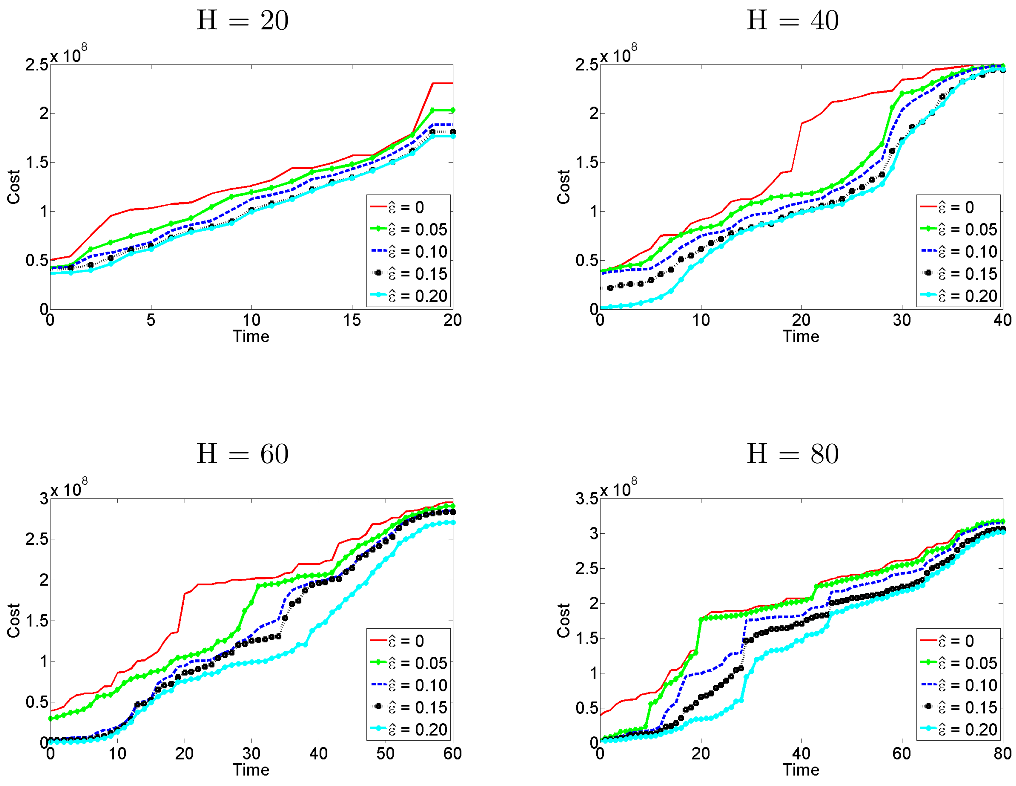

Figure 11: Cost curves of PD-ES schedules with pruning. The red line ($\hat{\varepsilon} = 0$) shows the cost curve of non-early-stopping PD schedule.

reduced to a novel variant of the classic directed Steiner tree problem. We showed that this problem is computationally hard and then developed a primal-dual algorithm for the problem. Our experiments showed that this algorithm produces close to optimal results and is much more scalable than a state-of-the-art MIP solver. We also showed that an early-stopping variant of the algorithm is able to explore the possible trade-offs between flexibility and reward, which is an important consideration in practice.

The scheduling problem considered in this work poses considerable challenges to generic off-the-shelf schedulers and planners. The complicating factors include: 1) highly-stochastic, exogenous dynamics that arise from the population spread model, 2) the need to reason about spatio-temporal processes, 3) the long horizons that must be considered, and 4) the combinatorial space of potential investment options at each point in time. The general solution schema we pursued in this work is likely to be applicable to other problems that pose similar challenges to existing techniques. In particular, this general schema suggests approximating the problem via the SAA and then studying the resulting deterministic optimization problem. Often the resulting deterministic problem will correspond to an existing well-studied problems, for which state-of-the-art approximation algorithms can be used. In other cases, such as in this work, the resulting problem will be related to an existing well-studied problem and a solution can be designed by extending existing solution frameworks. We expect that this generic SAA schema will be particularly useful

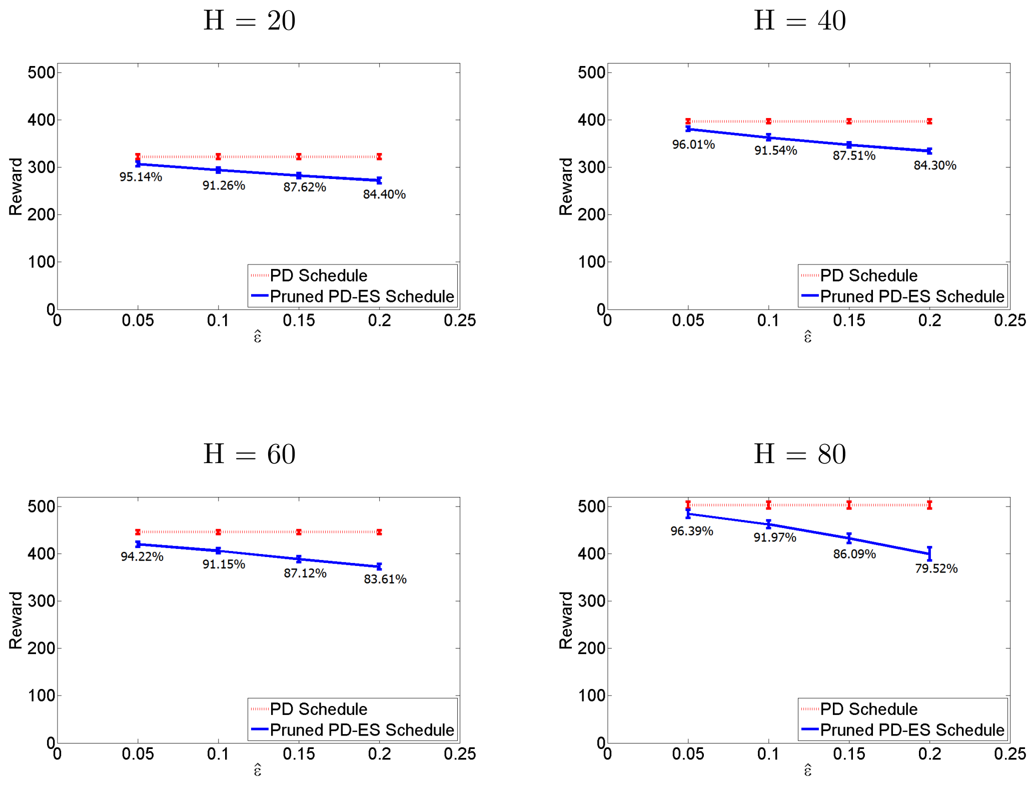

Figure 12: Rewards of primal-dual schedules with early-stopping. The number below each data point indicates the percentage of PD-ES reward over PD reward.

for problems involving stochastic spread of populations or information across networks, since the deterministic problems will typical map to graph-theoretic problems, for which there is a vast literature.

In future work, we plan to consider several improvements to the primal-dual algorithm. Currently, at each iteration, the algorithm randomly picks an unconnected terminal to grow a path from. It is likely that more intelligent selection mechanisms can improve the overall results. We are also interested in developing a primal-dual algorithm that directly incorporates the error tolerance constraint of our early-stopping approach. This would provide a more direct method for trading off reward for improved flexibility. Furthermore, we intend to pursue fully adaptive approaches to this and other conservation problems. One idea is to incorporate our scheduling approach into a replanning algorithm that selects purchases for the current decision epoch based on the most up-to-date information. In particular, at each decision epoch a schedule would be formed and those parcels scheduled to be purchased immediately (those with no flexibility) would be purchased or a subset of those in cases where the immediate budget would be exceeded. Considering more sophisticated approaches that take into account the immediate budget would be a natural and useful extension. It would also be interesting to consider the conservation problem with other variants of the reward function. For some species, rather than caring only the population in the end, the ecological goal may value the spread during the whole period the same or even more. Presumably such models would have different properties and complexities from the one we study in this paper.

Acknowledgements

Parts of the material in this paper appeared in an earlier work by Xue, Fern, and Sheldon (2012). This work was supported by NSF under grant IIS-0964705.

References

Ahmadizadeh, K., Dilkina, C., Gomes, C. P., & Sabharwal, A. (2010). An empirical study of optimization for maximizing diffusion in network. In *16th International Conference on Principles and Practice of Constraint Programming*.

Bent, R., & Van Hentenryck, P. (2004). Regret only! Online stochastic optimization under time constraints. In *Nineteenth AAAI Conference on Artificial Intelligence*.

Chang, H. S., Givan, R. L., & Chong, E. K. (2000). On-line scheduling via sampling. In *Artificial Intelligence Planning and Scheduling*.

Charikar, M., Chekuri, C., Cheung, T., Dai, A., Guha, S., & Li, M. (1998). Approximation algorithm for directed Steiner tree problems. In *The Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*.

Chong, E. K., Givan, R. L., & Chang, H. S. (2000). A framework for simulation-based network control via hindsight optimization. In *IEEE CDC conference*.

Crowley, M., & Poole, D. (2011). Policy gradient planning for environmental decision making with existing simulators. In *Twenty-fifth AAAI Conference on Artificial Intelligence*.

Drummond, L. M., & Santos, M. (2009). A distributed dual ascent algorithm for Steiner problems in multicast routing. *Networks*, *53*, 170–183.

Golovin, D., Krause, A., Gardner, B., Converse, S. J., & Morey, S. (2011). Dynamic resource allocation in conservation planning. In *Twenty-fifth AAAI Conference on Artificial Intelligence*.

Hanski, I., & Ovaskainen, O. (2000). The metapopulation capacity of a fragmented landscape. *Nature*, *404*(6779), 755–758.

Hwang, F. K., Richards, D. S., & Winter, P. (1992). *The Steiner Tree Problem*. Springer.

Kumar, A., Wu, X., & Zilberstein, S. (2012). Lagrangian relaxation techniques for scalable spatial conservation planning. In *Twenty-sixth AAAI Conference on Artificial Intelligence*.

Shapiro, A. (2003). Monte Carlo sampling methods. In *Stochastic Programming, Handbooks in Operations Research and Management Science*, Vol. 10, pp. 353–426.

Sheldon, D., Dilkina, B., Elmachtoub, A., Finseth, R., Sabharwal, A., Conrad, J., Gomes, C., Shmoys, D., Allen, W., Amundsen, O., & Vaughan, B. (2010). Maximizing the spread of cascades using network design. In *Uncertainty in Artificial Intelligence (UAI)*.

Vazirani, V. V. (2001). *Approximation Algorithms*. Springer, Berlin.

Williams, J., ReVelle, C., & Levin, S. (2005). Spatial attributes and reserve design models: a review. *Environmental Modeling and Assessment*, *10*(3), 163–181.

Wong, R. T. (1984). A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, *28*, 271–287.

Xue, S., Fern, A., & Sheldon, D. (2012). Scheduling conservation designs via network cascade optimization. In *Twenty-sixty AAAI Conference on Artificial Intelligence*.

Yoon, S., Fern, A., Givan, R. L., & Kambhampati, S. (2008). Probabilistic planning via determinization in hindsight. In *Twenty-third AAAI Conference on Artificial Intelligence*.