Fast Combinatorial Algorithm for Optimizing the Spread of Cascades

Xiaojian Wu **Daniel Sheldon** **Shlomo Zilberstein**

College of Information and Computer Sciences
University of Massachusetts, Amherst, MA 01003
{xiaojian,sheldon,shlomo}@cs.umass.edu

Abstract

We address a spatial conservation planning problem in which the planner purchases a budget-constrained set of land parcels in order to maximize the expected spread of a population of an endangered species. Existing techniques based on the sample average approximation scheme and standard integer programming methods have high complexity and limited scalability. We propose a fast combinatorial optimization algorithm using Lagrangian relaxation and primal-dual techniques to solve the problem approximately. The algorithm provides a new way to address a range of conservation planning and scheduling problems. On the Red-cockaded Woodpecker data, our algorithm produces near optimal solutions and runs significantly faster than a standard mixed integer program solver. Compared with a greedy baseline, the solution quality is comparable or better, but our algorithm is 10–30 times faster. On synthetic problems that do not exhibit submodularity, our algorithm significantly outperforms the greedy baseline.

1 Introduction

Spatial conservation planning problems have received significant attention from the AI community, resulting in a range of strategies for conserving land parcels in order to support the recovery of an endangered species or preserve biodiversity. Sheldon *et al.* [2010] studied a restricted version of this problem—*Red-cockaded Woodpecker* (RCW)—in which the planner selects a set of land parcels, subject to a budget constraint, to maximize the spread of a population over a geographical network of land patches within a specific time horizon. The underlying spreading process is modeled as a network diffusion process or cascade [Kempe *et al.*, 2003] using a *metapopulation model* developed by ecologists. Due to the stochasticity of process, the RCW problem can be written as a stochastic optimization problem. Similar propagation models have been studied in social networks, particularly the *influence maximization* problem where a planner strategically selects a certain number of sources to trigger the spread of influence over a social network [Domingos and Richardson, 2001; Kempe *et al.*, 2003]. Unlike that problem, a major challenge presented by RCW is that its objective function is not submodular. Consequently, greedy algorithms may produce solutions that are arbitrarily worse than an optimal solution.

Sheldon *et al.* [2010] developed an algorithm based on *sample average approximation* (SAA) and *mixed integer program* (MIP) to solve the problem approximately. In SAA, a certain number of random samples are drawn and their average is used to approximate the original objective function that is an expectation over all possible cascade scenarios. This way, the RCW problem is converted into a discrete optimization problem and solved optimally by a standard MIP solver. However, since the discrete optimization problem is NP-hard, the MIP solver can only handle limited network sizes and a small number of samples, undermining the approximation quality of SAA. For example, a standard MIP solver fails to work with more than 20 samples, either taking too long to finish or using too much memory, and an improved MIP based algorithm [Kumar *et al.*, 2012] can scale up to 40 samples but needs hours to finish. To address this challenge of scalability, we propose what we believe is the first combinatorial algorithm to solve the converted discrete optimization problem approximately and efficiently.

In fact, our algorithm is applicable to a more general problem called *Budget Set Weighted Directed Steiner Graph* (BSW-DSG) and, with minor modifications, to the *Quota Set Weighted Directed Steiner Graph* (QSW-DSG) and *Prize-Collecting Set Weighted Directed Steiner Graph* (PCSW-DSG) problems. We introduce each of these problem as a generalization of a classical network design problem to the setting where edges are purchased in sets instead of individually. Xue *et al.* [2015] first generalized the Steiner tree problem in this way; our work extends this generalization to the widely-used *quota*, *budget*, and *prize-collecting* Steiner tree variants [Johnson *et al.*, 2000]. All three problems share the same basic setup but have different constraints and objectives. In the basic setup, an initial graph and a set of candidate edge sets, each with a certain cost, are given. The planner can purchase a subset of edge sets and add the included edges into the initial graph. In the budget problem (BSW-DSG), the total cost of selected edge sets is constrained by a given budget and the objective is to maximize the number of *terminal vertices* connected to a unique *root* vertex in the augmented graph. In the quota problem (QSW-DSG), the number of terminal vertices connected to

the root is required to be greater than a given value and the objective is to minimize the total cost of achieving this requirement. In the prize-collecting problem (PCSW-DSG), both the number of terminal vertices connected to the root and cost are part of the objective and the tradeoff between them is controlled by a parameter. In this paper, we propose a fast primal-dual algorithm to solve the PCSW-DSG problem approximately. To address the BSW-DSG problem, we use Lagrangian relaxation [Jain and Vazirani, 2001; Ahuja *et al.*, 1988] to fold the budget constraint into the objective and convert the BSW-DSG problem into a PCSW-DSG problem that is parameterized by a Lagrangian multiplier β. We also derive a *bisection procedure* to find the value of β that gives a near optimal solution. At each iteration of the procedure, a PCSW-DSG problem parameterized by some β is solved near optimally. Similarly, the same algorithm can solve the QSW-DSG problem with a minor modification to the bisection procedure. For the sake of clarity, we focus the paper on the BSW-DSG problem.

When applied to the RCW dataset, our algorithm produces near optimal solutions and runs orders of magnitude faster than the MIP approach, for example, taking only 20 minutes for 300 samples. A greedy baseline also produces near-optimal results, however our algorithm still produces better quality in most cases and, most importantly, is 10-30 times faster. In experiments on synthetic problems with objective functions that are strongly non-submodular, our algorithm performs much better than the greedy baseline.

The rest of the paper is organized as follows. Section 2 describes the conservation planning problem and the basic solution based on SAA. Section 3 formalizes the general BSW-DSG problem. Section 4 presents our combinatorial algorithm. Section 5 describes how to apply our algorithm to a range of other practical applications. Experimental results are reported in Section 6.

2 Problem Statement

Consider a conservation area consisting of habitat patches that are the atomic units in the population dynamics model and can be either occupied or unoccupied by the species. These patches are grouped into non-overlapping *parcels* $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_L$, which are the smallest units available for purchase. A parcel $\mathcal{P}_l$ can be purchased with cost c_l used to restore and conserve its habitat patches so they are suitable for the species to occupy. A patch can be occupied only if the parcel that it belongs to is purchased. A conservation strategy is an L-dimensional vector $\mathbf{y}$ where $y_l = 1$ if that the l^{th} parcel is purchased and $y_l = 0$ otherwise.

Sheldon *et al.* [2010] use a *metapopulation model* from ecology [Hanski and Ovaskainen, 2000] to describe the stochastic occupancy dynamics of the species. In this model, a patch is either occupied or unoccupied at each time step $h \in \{0, 1, ..., H-1\}$. At time 0, the species occupy a set of patches called *sources*. At any time h, an occupied patch v triggers two stochastic events: (1) the population at u colonizes an unoccupied patch v with probability p_{uv} (typically decays with distance between the patches) that makes v occupied at time $h+1$; and (2) the population at v becomes extinct with probability $1 - p_{vv}$, making v unoccupied at time $h+1$.

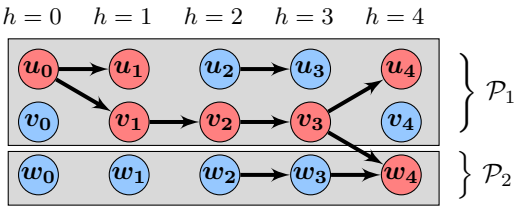


Figure 1: Example of the layered graph of a scenario with three patches u, v, w and two parcels $\mathcal{P}_1, \mathcal{P}_2$, showing parcels (grey boxes), occupied (red) and unoccupied (blue) patches.

The goal of conservation planning is to select a set of parcels to purchase, without exceeding a budget $\mathcal{B}$, such that the expected number of patches being occupied at time $H-1$ is maximized. More concretely, given a strategy y, define 0-1 random variable $X(y)_{v,h}$ for each patch v and each time h to indicate whether the patch v is occupied at time h ($X(y)_{v,h} = 1$) or not ($X(y)_{v,h} = 0$). The best strategy is obtained by solving the following optimization problem:

$$\arg\max_{y} \sum_{v} \mathbb{E}[X(y)_{v,H-1}] \quad s.t. \sum_{l:y_l=1} c_l \leq \mathcal{B}. \qquad (1)$$

Sample Average Approximation The stochastic optimization problem (1) is very hard to solve directly, but the SAA scheme can be used to solve it approximately. The idea is to sample N independent *cascade scenarios* from the stochastic spreading process, each of which represents a possible outcome of the stochastic process. For each scenario, given a strategy vector y, the occupancy status of any patch at any time step can be evaluated directly so that the number of occupied patches at the last time step can be determined. Hence, the average number of occupied patches over sampled scenarios is an approximation of the objective in (1).

More concretely, one scenario is a layered graph, as shown in Fig. 1, that contains a vertex v_h for patch v and time h. To construct the edges of the graph, for a pair of patches (u, v) $(u \neq v)$ and a $h \in \{0, \ldots, H-2\}$, a biased coin with probability p_{uv} being heads is flipped. A directed edge (u_h, v_{h+1}) is added to the graph *iff* the outcome is heads, meaning that v will be occupied at $h+1$ if u is occupied at h. Similarly, for a patch v and $h \in \{0, \ldots, H-2\}$, a coin with probability $1 - p_{vv}$ being heads is flipped. A directed edge (v_h, v_{h+1}) is added to the graph *iff* the outcome is heads, meaning that the population occupying v at h will survive to time $h+1$. With a conservation strategy, we can determine whether v is occupied at h by checking whether there exists a *valid path* from some u_0 to v_h such that (1) u is occupied at time 0 and (2) all patches on the path are purchased. By checking the layered graphs of N i.i.d. sampled cascade scenarios, we can obtain the average number of occupied patches at time $H-1$. Specifically, let $X(y)^n_{v,h}$ be a 0-1 variable indicating whether the patch v is occupied in the n^{th} scenario at time h ($X(y)^n_{v,h} = 1$) or not ($X(y)^n_{v,h} = 0$). Then, we have

$$\sum_{v} \mathbb{E}[X(y)_{v,H-1}] \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{v} X(y)^n_{v,H-1} \qquad (2)$$

When N goes to infinity, the right term converges to the left term and the vector y that maximizes the right term converges to the vector that maximizes the left term [Kleywegt *et al.*, 2002]. Thus, a good conservation strategy can be obtained by solving the following optimization problem:

$$\arg\max_{y} \frac{1}{N} \sum_{n,v} X(y)^n_{v,H-1} \quad s.t. \sum_{l:y_l=1} c_l \leq \mathcal{B} \qquad (3)$$

Instead of solving (3) as a MIP, we propose a significantly faster near-optimal algorithm.

3 Budget Set Weighted Directed Steiner Graph Problem

In this section, we propose a novel network design problem called *Budget Set Weighted Directed Steiner Graph* (BSW-DSG), of which (3) is a specific instance.

The input of a BSW-DSG problem is a directed graph $G = \{V, E\}$ with a unique *root* vertex $r \in V$, a set of terminal vertices $\mathcal{T} \subseteq V$ and a collection of M edge sets $\mathcal{E} = \{E_1, E_2, ..., E_M\}$ where $E_s \subseteq E$ and each E_s is associated with a nonnegative cost c_s. Let $\mathcal{A}$ denote a subset of $\mathcal{E}$. Each $\mathcal{A}$ corresponds to a subgraph $G^{\mathcal{A}} = \{V, E^{\mathcal{A}}\}$ where $E^{\mathcal{A}} = \bigcup_{E_s \in \mathcal{A}} E_s$. A vertex v is *connected* to r if there is a path from r to v. Given a budget $\mathcal{B}$, the goal is to find a set $\mathcal{A}$ with cost no greater than $\mathcal{B}$ such that the number of terminal vertices connected to r in $G^{\mathcal{A}}$ is maximized.

To write down its mathematical formulation, we start with a preprocessing step. Let $E^{\mathcal{T}}$ be an additional set of edges from the root *directly* to each terminal $k \in \mathcal{T}$. For any selected set $\mathcal{A}$, we can augment the subgraph $G^{\mathcal{A}}$ to be a *Steiner graph* (one where all terminals are connected to r) by adding edges $(r, k) \in E^{\mathcal{T}}$ for each terminal k that is not already connected to r in $G^{\mathcal{A}}$. Since $\mathcal{T}$ is fixed, the original goal of maximizing the number of connected terminals is equivalent to minimizing the number of edges needed to augment $G^{\mathcal{A}}$ to obtain a Steiner graph. Mathematically, BSW-DSG is formulated as follows:

$$\begin{aligned} &\arg\min_{\mathcal{A} \subseteq \mathcal{E}, \mathcal{A}^{\mathcal{T}} \subseteq E^{\mathcal{T}}} |\mathcal{A}^{\mathcal{T}}| \qquad (4) \\ &s.t. \sum_{s:E_s \in \mathcal{A}} c_s \leq \mathcal{B} \\ &\quad \mathcal{A}^{\mathcal{T}} \cup \Big(\bigcup_{E_s \in \mathcal{A}} E_s \Big) \text{ forms a Steiner graph} \end{aligned}$$

where $\mathcal{A}^{\mathcal{T}}$ is the set used for augmentation.

Problem (4) can be formulated as a MIP using the network-flow encoding shown in Fig. 2. The flow variable $x^k_{u,v}$ encodes the flow destined for terminal $k \in \mathcal{T}$ on the edge $(u, v) \in \mathbb{E}$, where $\mathbb{E}$ denotes $E \bigcup E^{\mathcal{T}}$. It is easy to show that, in the optimal solution of the MIP, $x^k_{u,v}$ is either 0 or 1. The 0-1 decision variable y_s indicates whether the edge set E_s is purchased ($y_s = 1$) or not ($y_s = 0$). The 0-1 variable z_{rk} indicates whether the edge $(r, k) \in E^{\mathcal{T}}$ is used ($z_{rk} = 1$) or not ($z_{rk} = 0$). Line (2) forces each terminal k to be connected to r by requiring one unit of flow on a path from r to terminal k. The objective is to minimize the number of used edges in $E^{\mathcal{T}}$. Line (3) says that the flow can pass an edge (u, v) only when (u, v) is in a purchased edge set E_s, (4) says that the flow can pass an edge $(r, k) \in E^{\mathcal{T}}$ only when r_{rk} is 1, and (5) is the budget constraint.

$$\min_{x,y,z} \sum_{(r,k') \in E^{\mathcal{T}}} z_{rk'} \qquad (1)$$

$$s.t. \sum_{(w,v) \in \mathbb{E}} x^k_{wv} - \sum_{(u,w) \in \mathbb{E}} x^k_{uw} = \begin{cases} 1 & \text{if } w = r \\ -1 & \text{if } w = k \\ 0 & \text{otherwise} \end{cases} \quad \forall w \in V \ \forall k \in \mathcal{T} \qquad (2)$$

$$x^k_{uv} \leq \sum_{s:(u,v) \in E_s} y_s \quad \forall k \in \mathcal{T}, \ \forall (u, v) \in E \qquad (3)$$

$$x^k_{rk'} \leq z_{rk'} \quad \forall k \in \mathcal{T}, \ \forall (r, k') \in E^{\mathcal{T}} \qquad (4)$$

$$\sum_{s=1}^{M} c_s y_s \leq \mathcal{B} \qquad (5)$$

$$x^k_{uv} \in [0, 1] \quad \forall k \in \mathcal{T}, \ \forall (u, v) \in \mathbb{E} \qquad (6)$$

$$y_s \in \{0, 1\} \quad \forall E_s \in \mathcal{E} \text{ and } z_{rk'} \in \{0, 1\} \quad \forall (r, k') \in E^{\mathcal{T}} \qquad (7)$$

Figure 2: MIP formulation of BSW-DSG problem

BSW-DSG Formulation of the RCW problem The optimization problem (3) can be formulated as a BSW-DSG problem in the following way. A new input graph of BSW-DSG is created by combining the layered graphs of N sampled scenarios in which a vertex v^n_h exists for each patch v, each scenario n and each time step h. Also, a new root vertex r is added to the graph along with directed edges from r to each v^n_0 with v being occupied at time 0. Then, v^n_h being connected to r implies the patch v is occupied at time h in the n^{th} scenario. The terminal set $\mathcal{T}$ is defined to contain all vertices with subscript $H-1$, that is, $\mathcal{T} = \{v^n_{H-1} : \forall v, n \in \{1, ..., N\}\}$ since the goal is to optimize occupancy of patches at time $H-1$. An edge set E_s is created for a parcel $\mathcal{P}_l$ in a way that if a patch v is in $\mathcal{P}_l$, all edges ending with v^n_h is in E_s. Formally, $E_s =$

$$\left\{ (u, v^n_h) : \begin{array}{l} \forall v \in \mathcal{P}_l, h = \{1, ..., H-1\}, n = \{1, ..., N\} \\ \forall (u, v^n_h) \text{ in the } n^{th} \text{ layered graph} \end{array} \right\}$$

The cost of E_s is the cost of $\mathcal{P}_l$. By now, we formulated the optimization problem (3) of the RCW problem as an instance of BSW-DSG.

4 Our Algorithm for BSW-DSG Problem

Lagrangian relaxation has been used to solve many constrained discrete optimization problems [Jain and Vazirani, 2001; Kumar *et al.*, 2012; Ahuja *et al.*, 1988]. The basic idea is to relax a complex constraint and bring it to the objective together with a Lagrangian multiplier β. The new optimization problem is called *relaxation problem* parameterized by β. In this paper, we take advantage of the problem structure to develop a bisection procedure to search for a β such that the near optimal solution can be computed by solving the relaxation problem with that β. For the BSW-DSG problem, the relaxation problem is a PCSW-DSG problem with tradeoff parameter β. To solve the PCSW-DSG problem, we propose a primal-dual algorithm motivated by [Wong, 1984].

PCSW-DSG Problem / Primal Problem:

$$\min_{(x,y,z)\in\mathcal{X}} L(x,y,z,\beta) = \sum_{(r,k')\in E^{\mathcal{T}}} z_{rk'} + \beta\left(\sum_{s=1}^{M} c_s y_s - \mathcal{B}\right) \quad (1)$$

$$\mathcal{X} = \{(x,y,z) : x, y, z \text{ satisfy constraints (2), (3), (4), (6), (7) in Fig. 2}\} \quad (2)$$

Dual Problem:

$$\max \sum_{k\in\mathcal{T}} \left(\mu_k^k - \mu_r^k\right) - \beta\mathcal{B} \quad (3)$$

$$s.\ t.\ \ \mu_v^k - \mu_u^k - \lambda_{uv}^k \le 0 \ \ \forall (u,v) \in \mathbb{E} \quad (4)$$

$$\sum_{(u,v)\in E_s} \sum_{k\in\mathcal{T}} \lambda_{uv}^k \le \beta c_s \ \ \forall E_s \in \mathcal{E},\ \ \forall k \in \mathcal{T} \quad (5)$$

$$\sum_{k\in\mathcal{T}} \lambda_{rk'}^k \le 1 \ \ \forall (r,k') \in E^{\mathcal{T}} \quad (6)$$

$$\mu_v^k \in \{-\infty, \infty\} \ \ \forall k \in \mathcal{T}, v \in V \quad (7)$$

$$\lambda_{uv}^k \in [0, \infty) \ \ \forall k \in \mathcal{T}, (u,v) \in \mathbb{E} \quad (8)$$

Figure 3: Primal and dual formulations of relaxation problem

In summary, our algorithm consists of three major steps. First, we relax the budget constraint (5) in Fig. 2 and move it to the objective with a parameter β to create the PCSW-DSG problem shown in Fig. 3. Second, we use the bisection procedure to find the β, using the primal-dual algorithm to solve the PCSW-DSG problem. Finally, we extract and refine the solution. The primal-dual algorithm and the bisection procedure will be explained next.

4.1 Bisection Procedure

Let's first analyze the function $L(x, y, z, \beta)$ in Fig. 3. Let $(x_\beta, y_\beta, z_\beta)$ denote the optimal solution of the PCSW-DSG problem for some β. Let, $Z(\beta) = \sum_{(r,k')\in E^{\mathcal{T}}} z_{rk'\beta}$, $C(\beta) = \sum_{s=1}^{M} c_s y_{s\beta}$, and $L(\beta) = Z(\beta) + \beta C(\beta)$. We have the following properties.

Proposition 1. *As β increases, $Z(\beta)$ is nondecreasing and $C(\beta)$ is nonincreasing.*

The proof is omitted due to space limitation. Intuitively, larger β puts larger penalty on the cost and thus results in a less costly y but larger $Z(\beta)$.

The bisection procedure starts with a suitably large interval bounding β and then narrows it iteratively. At each iteration, a new β is picked as the middle point of the current range and then the primal-dual algorithm is used to calculate $C(\beta)$ and produce a near optimal strategy for β. By Proposition 1, if $C(\beta) < \mathcal{B}$, all β' greater than β will give $Z(\beta') \ge Z(\beta)$ and therefore can be abandoned. If $C(\beta) > \mathcal{B}$, all β' smaller than β will give $C(\beta') \ge C(\beta)$ and therefore can be abandoned. The procedure terminates when the range is less than some threshold and then a β in the range along with the computed strategy is returned.

Proposition 1 is true when the PCSW-DSG problem is solved optimally for each β, but the bisection procedure is still valid with our approximate algorithm. Let $(\hat{x}_\beta, \hat{y}_\beta, \hat{z}_\beta)$ denote the near optimal solution computed by the primal-dual algorithm for some β. And let $\hat{Z}(\beta) = \sum_{(r,k')\in E^{\mathcal{T}}} \hat{z}_{rk'\beta}$ and $\hat{C}(\beta) = \sum_{s=1}^{M} c_s \hat{y}_{s\beta}$. Then, we have

Algorithm 1 Primal-Dual Algorithm for PCSW-DSG

```
1: function PRIMALDUAL(G, E, β)
2:     λ^k_uv ← 0 ∀k ∈ T ∀(u,v) ∈ E
3:     G' = {V, E'}, E' ← φ
4:     y ← 0, z ← 0
5:     while G' is not a steiner graph do
6:         Pick a k ∈ T not connected to r
7:         Find δ(k) : a set of cut edges between r and k
8:         S = {s|E_s ∩ δ(k) ≠ 0, E_s ∉ A}
9:         s* = arg min_{s∈S} Δ(s,k) where
10:        Δ(s,k) = βc_s − Σ_{k'∈T,(u,v)∈E_s} λ^{k'}_uv / |E_s ∩ δ(k)|
11:        if Δ(s*,k) ≥ (1 − λ^k_rk) then
12:            Δ(s*,k) ← 1 − λ^k_rk
13:            E' ← E' ∪ {(r,k)} and z_{r,k} ← 1
14:        else
15:            E' ← E' ∪ E_{s*} and y_{s*} ← 1
16:        end if
17:        λ^k_uv ← λ^k_uv + Δ(s*,k) ∀(u,v) ∈ δ(k)
18:    end while
19:    Set y_s ← 0 if removing E_s, G' is still Steiner graph.
20:    Return y
21: end function
```

Proposition 2. *Suppose terminal vertices in line 6 of Algorithm 1 are chosen in a predefined order, as β increases, $\hat{Z}(\beta)$ is nondecreasing and $\hat{C}(\beta)$ is nonincreasing.*

The proof is omitted. Intuitively, as β becomes larger, the equality in (5) in the dual problem becomes harder to satisfy. Since the terminal vertices are picked in the same order over different β, the set of edge sets purchased for a larger β is the subset of edge sets for a smaller β which gives us Proposition 2. Finally, if the primal-dual algorithm performs near optimally for the PCSW-DSG problem, the final solution computed by the bisection procedure will be near optimal for the BSW-DSG problem. Although this procedure is not accompanied by a formal approximation guarantee, we observe that it produces near-optimal results on the problem instances used in our experiments.

4.2 Primal-Dual Algorithm for PCSW-DSG

Algorithm 1 summarizes the PCSW-DSG solver. This algorithm builds technically on the primal-dual algorithm of Xue *et al.* [2015] for the SW-DSG problem (where *all* terminals must be connected). The dual formulation, shown in Fig. 3, is obtained based on the LP relaxation of the primal problem. The basic idea of the primal-dual algorithm is to repeatedly adjust dual variables to increase the dual objective and at the same time construct a feasible primal solution while maintaining the primal complementary slackness condition [Vazirani, 2001]. This condition is defined as follows for our problem.

Definition 1. *Primal Complementary Slackness Condition:*

1. $y_s \ne 0$ *implies* $\sum_{(u,v)\in E_s} \sum_{k\in\mathcal{T}} \lambda_{uv}^k = \beta c_s$
2. $z_{rk'} \ne 0$ *implies* $\sum_{k\in\mathcal{T}} \lambda_{rk'}^k = 1$

Without loss of generality, we assume that E doesn't contain edge (r, k) for any $k \in \mathcal{T}$. Initially, the primal variables

y, z are set to be 0 (line 4), meaning that no edge sets are purchased and no edges in $E^{\mathcal{T}}$ are used (line 3), which represents an infeasible solution for the primal problem. Also, the dual variables are set to be 0 (line 2), which is a feasible solution for the dual problem. Note that we omit the dual variables $\{\mu_v^k\}$ in the presentation of the algorithm because their values can be derived from the values of $\{\lambda_{uv}^k\}$ and our goal is to construct the feasible primal solution rather than explicitly calculate the actual value of the dual objective. The relationship of μ and λ is described by the following Proposition.

Proposition 3. *Consider the edge-weighted graph $G_k = \{V, \mathbb{E}\}$ where each edge (u, v) is assigned length equal to λ_{uv}^k. For any pair of vertices u and v where there is a path from u to v in G_k, the maximum value that $\mu_v^k - \mu_u^k$ can take subject to constraint (4) in the dual problem, is the shortest path distance from u to v in G_k.*

Thus, maximizing $\mu_k^k - \mu_r^k$ is equivalent to the problem of assigning a length λ_{uv}^k to each edge in $\mathbb{E}$, subject to constraints (5) and (6) of the dual problem, to maximize the shortest distance from r to k. To maximize the objective of the dual problem, the lengths of $\{\lambda_{uv}^k\}$ are assigned, subject to constraints (5) and (6), to maximize the sum of the shortest distances from r to k for all $k \in \mathcal{T}$.

In each iteration (line 5-18), our algorithm increases the dual objective by increasing the value of $\mu_k^k - \mu_r^k$ for one terminal k that is not connected to r in graph G', while maintaining feasibility. It first finds a set of edges $\delta(k)$ cutting k and r (line 7). Lines 8-16 calculate a maximum value $\Delta(s^*, k)$ that all λ_{uv}^k with $(u, v) \in \delta(k)$ can increase by without violating constraints (5) and (6), which therefore increases the value of $\mu_k^k - \mu_r^k$ by the same amount. In line 8-10, for each E_s, the maximum amount of value that all variables λ_{uv}^k with $(u, v) \in E_s \cap \delta(k)$ can increase is equal to the remaining value of constraint (5) divided by the number of edges in $E_s \cap \delta(k)$. For the edge $(r, k) \in E^{\mathcal{T}}$, since it is always in $\delta(k)$ and the value of $\lambda_{r,k}^k$ can at most increase to 1 by constraint (6), the maximum value $\Delta(s^*, k)$ is at most $(1 - \lambda_{r,k}^k)$ (line 11-12). If $\Delta(s^*, k)$ is equal to $(1 - \lambda_{r,k}^k)$ eventually, the constraint (6) will become tight for (r, k), so z_{rk} is set to be 1 due to the complementary slackness condition 2 and the edge (r, k) is added to G' (line 13). Otherwise, the constraint (5) will become tight for s^*, so y_{s^*} is set to be 1 due to the complementary slackness condition 1 and all edges in E_{s^*} are added to G' (lines 15). In line 17, dual variables are updated. The algorithm terminates when all terminals are connected to r. Finally, in line 19, we check whether there exists some purchased E_s that can be removed without disconnecting any terminal from r. If we find such E_s, we set $y_s = 0$.

4.3 Improvement Techniques

Several techniques can improve the results of Algorithm 1.

Expanding Direction In line 7, the edge sets cutting r and k can be found in two different ways:

- *Forward method*: The cutset $\delta(k)$ contains an edge (u, v) if u is connected to r in the constructed graph G' and k is connected to v in the original graph G. Namely, the path from r to k is created in a forward direction.
- *Backward method*: the cutset $\delta(k)$ contains an edge (u, v) where k is connected to v in G' and u is connected to r in G. Namely, the path from r to k is created in a backward direction.

We found that each method may perform better than other depending on the budget, so we use both methods in our implementation.

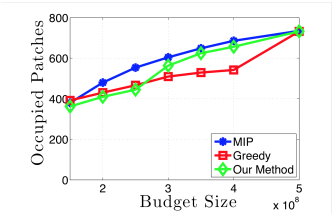
Local Search In the experiments, we found that different orders of picking terminals can give slightly different solution qualities, so we experimented with a local search method to further improve the solution. We start with the β value returned by the bisection procedure and run the PCSW-DSG solver using a different, randomly selected order. If $C(\beta) < \mathcal{B}$, we decrease β by a small amount (by Proposition 1). Otherwise, we increase β by a small amount. We repeat this process for 10-20 iterations and in each iteration, we re-select the order of picking terminals randomly. Finally, we return the best feasible solution found. We note, however, that the improvement is limited, indicating that the order of picking terminals doesn't affect the quality that much.

Greedy Padding Since we find a near-optimal β rather than an optimal one, the computed solution may not use all the available budget. Therefore, to improve the solution, a greedy procedure can be used to continue purchasing edge sets until the budget is exhausted. In our experiments, only a few additional edge sets can be purchased so the procedure is very fast, but the improvement is not significant, usually 1-2%.

5 Extensions to Other Problems

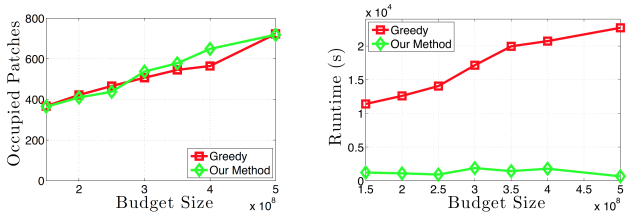
BSW-DSG is a very general problem that can capture a range of network optimization problems since the candidate edge sets $\mathcal{E}$ can be defined arbitrarily, unlike RCW that is limited to adding patches. For example, our framework is applicable when the underlying network is a road network or river network [Wu *et al.*, 2014a; 2014b] where the investment is used to improve certain edge segments rather than certain locations in the network.

The techniques we propose in this paper can also solve the quota problem (QSW-DSG) by using a slightly different bisection procedure: when narrowing the interval of β, we can check whether $Z(\beta)$ is greater than the required number of terminals being connected to r and then abandon one half of the interval accordingly. The QSW-DSG problem is an important problem in conservation planning and scheduling. For example, the problem studied by Xue *et al.* [2012] is a special instance of the QSW-DSG problem where all terminals are required to be connected to r. Xue *et al.* [2015] mention a useful extension to their problem that only requires a fixed fraction of all terminals to be connected to r, which is equivalent to the QSW-DSG problem. While their paper does not give a solution to that problem, our algorithm can solve it approximately. Furthermore, our algorithm can quickly produce a curve to visualize the tradeoff between cost and the fraction of terminals that need to be connected, by solving the PCSW-DSG problem with different parameters, which offers a useful new capability to end-users.



(a) Optimality w.r.t. budget (b) Runtime w.r.t. budget

Figure 4: Optimality and runtime versus budget with 10 samples



(a) Optimality w.r.t. budget (b) Runtime w.r.t. budget

Figure 5: Optimality and runtime versus budget with 300 samples

Optimal solutions to the prize-collecting problem are Pareto optimal with respect to cost and coverage; varying β explores different Pareto optimal solutions. Thus, our algorithm can explore different (approximate) Pareto optimal solutions, but it only partially explores the space and does not guarantee generating all Pareto optimal solutions. Finally, our algorithm can be potentially extended to solve the network topology optimization problem in social networks [Khalil *et al.*, 2014]. The applicability of our algorithm to this broad list of problems underscores its significance.

6 Experimental Results

6.1 Empirical Data of RCW Problem

We used the data set for the RCW problem introduced by Sheldon *et al.* [2010]. The study area includes 2537 patches and 443 parcels. The data set specifies the colonization probabilities between all pairs of patches, the grouping of patches into parcels, the prices of parcels and the initial occupancy status of patches. We used a planning horizon of 100 years and compared our algorithm with a MIP solver and with a greedy algorithm. In each iteration, the greedy algorithm chooses the edge set with the highest ratio of increase in objective value to cost, which gives better solutions than another greedy variant that chooses the edge set with the highest increase in objective value. We used the Gurobi Optimzer as the MIP solver [Gurobi, 2015]. All the experiments were run on a 2.2GHz Intel Core i7 CPU with 16GB of RAM.

Small Sample Size We compare three algorithms for 10 samples and the results are shown in Fig. 4. For our algorithm, we applied Algorithm 1 twice, once using the forward method and once using the backward method, and then applied the local search and greedy padding techniques to the better of those two solutions. In Fig. 4(b), our algorithm runs significantly faster than MIP and 10–20 times faster than the greedy algorithm. In Fig. 4(a), both our algorithm and the greedy algorithm produce near optimal solutions. Our algorithm outperforms the greedy algorithm on larger budget sizes and performs slightly worse on smaller budget sizes.

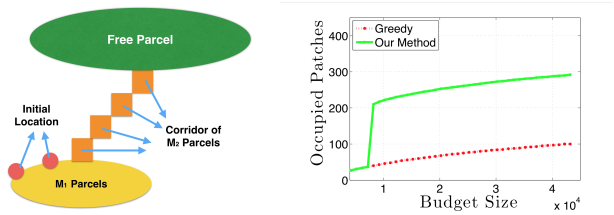
Large Sample Size We also tested our algorithm and the greedy baseline using 300 samples. The results are show in Fig. 5. We did not test Gurobi because it fails to solve MIPs with more than 20 samples by either using too much time or running out of memory. Our algorithm performs almost identically to the greedy algorithm when the budget is smaller than 3×10^8 and outperforms the greedy algorithm when the budget is greater than 3×10^8. Fig. 5(b) shows that our algorithm is 10–30 times faster than the greedy algorithm and its running time is almost constant with respect to budget, while the greedy algorithm takes longer for larger budgets. This implies that our algorithm can scale to larger network sizes or more samples, that latter of which leads to improved solution quality within the SAA methodology.

6.2 Synthetic Data

In the RCW problem, even though the discrete optimization problem is NP-hard, the greedy algorithm is still able to produce a solution within 80% of the optimal value, which suggests that the problem does not badly violate submodularity. In this section, we design a group of problems that are more challenging. These synthetic problems are motivated by the corridor design problem, which is another important conservation planning problem [Gomes, 2009] in which the goal is to purchase a subset of parcels to build a (long) corridor connecting distant habitat areas in a fragmented landscape. When formulated within our context, the objective function violates submodularity because purchasing individual parcels has no or little effect on the objective function until the entire corridor is purchased, at which point the endpoints become connected and there is a large jump in the objective value.

With this motivation, we construct a small problem instance as follows (see Fig. 6(a)). We first create M_1 parcels, each of which are directly connected to the source population and so carry an immediate reward of p_1 if purchased. We also create a free parcel which has reward of p_2 if it is reachable from the source. The free parcel is connected by a corridor of M_3 parcels, which have reward of p_3 if accessible. All parcels in the corridor must be purchased to realize the reward of the free parcel. If we set $p_2 \gg p_1 > p_3$, the optimal strategy should first purchase all parcels in the corridor whenever the budget allows, and first purchase all of the M_1 parcels if the budget is not enough to purchase the corridor.

The experimental results of our algorithm and the greedy algorithm on the synthetic data are shown in Fig. 6(b). Our algorithm gives the optimal solution as its curve overlaps with the curve of the MIP solver. When budget is not enough (less than 7500) to purchase all parcels in the corridor, the greedy algorithm performs optimally. But it performs poorly when budget is enough (greater than 7500) because it never purchases any parcels in the corridor before all M_1 parcels are purchased. From these results, we observed that when the objective function is not submodular as in our simple problem setting, the greedy algorithm may produce arbitrarily bad



(a) Corridor design problem (b) Optimality w.r.t. budget

Figure 6: Experiments on synthetic data

solutions, while our algorithm can still produce near-optimal solutions.

7 Conclusion

We propose a fast approximate algorithm to solve a spatial conservation planning problem in which a set of budget-constrained land parcels are purchased to maximize the expected spread of a population of a species. Our algorithm can approximately solve three different variants of a more general network design problem: the budget, quota, and prize-collecting extensions of the Set-Weighted Directed Steiner Graph Problem. It therefore has potential to apply to a variety of other applications, which we plan to explore in future work. On the Red-cockaded Woodpecker problem, our algorithm produces near optimal solutions and runs significantly faster than a standard mixed integer program solver. Also, the solution quality is comparable to or better than a greedy baseline (which is also near optimal on this problem), but our algorithm is 10–30 times faster. On synthetic problems that do not exhibit submodularity, our algorithm significantly outperforms the greedy baseline.

References

[Ahuja *et al.*, 1988] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows*. Alfred P. Sloan School of Management, MIT, 1988.

[Domingos and Richardson, 2001] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66. ACM, 2001.

[Gomes, 2009] Carla P. Gomes. Computational sustainability: Computational methods for a sustainable environment, economy, and society. *The Bridge*, 39(4):5–13, 2009.

[Gurobi, 2015] Gurobi. Optimization software. `http://www.gurobi.com/`, 2015.

[Hanski and Ovaskainen, 2000] Ilkka Hanski and Otso Ovaskainen. The metapopulation capacity of a fragmented landscape. *Nature*, 404(6779):755–758, 2000.

[Jain and Vazirani, 2001] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[Johnson *et al.*, 2000] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting Steiner tree problem: Theory and practice. In *Proc. of the 11th ACM SIAM Symposium on Discrete Algorithms (SODA)*, pages 760–769, 2000.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[Khalil *et al.*, 2014] Elias Boutros Khalil, Bistra Dilkina, and Le Song. Scalable diffusion-aware optimization of network topology. In *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1226–1235, 2014.

[Kleywegt *et al.*, 2002] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[Kumar *et al.*, 2012] Akshat Kumar, Xiaojian Wu, and Shlomo Zilberstein. Lagrangian relaxation techniques for scalable spatial conservation planning. In *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 309–315, 2012.

[Sheldon *et al.*, 2010] Daniel Sheldon, Bistra Dilkina, Adam Elmachtoub, Ryan Finseth, Ashish Sabharwal, Jon Conrad, Carla Gomes, David Shmoys, William Allen, Ole Amundsen, and William Vaughan. Maximizing the spread of cascades using network design. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 517–526, 2010.

[Vazirani, 2001] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag, 2001.

[Wong, 1984] Richard T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984.

[Wu *et al.*, 2014a] Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Rounded dynamic programming for tree-structured stochastic network design. *Proc. of the 28th Conference on Artificial Intelligence (AAAI)*, pages 479–485, 2014.

[Wu *et al.*, 2014b] Xiaojian Wu, Daniel Sheldon, and Shlomo Zilberstein. Stochastic network design in bidirected trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 882–890, 2014.

[Xue *et al.*, 2012] Shan Xue, Alan Fern, and Daniel Sheldon. Scheduling conservation designs via network cascade optimization. In *Proc. of the 26th Conference on Artificial Intelligence (AAAI)*, pages 391–397, 2012.

[Xue *et al.*, 2015] Shan Xue, Alan Fern, and Daniel Sheldon. Scheduling conservation designs for maximum flexibility via network cascade optimization. *Journal of Artificial Intelligence Research*, 52:331–360, 2015.